АВТОМАТНОЕ УПРАВЛЕНИЕ АСИНХРОННЫМИ ПРОЦЕССАМИ В ЭВМ И ДИСКРЕТНЫХ СИСТЕМАХ

АВТОМАТНОЕ УПРАВЛЕНИЕ АСИНХРОННЫМИ ПРОЦЕССАМИ В ЭВМ И ДИСКРЕТНЫХ СИСТЕМАХ

Под редакцией В. И. ВАРШАВСКОГО

МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1986

ББК 32.81
А22
УДК 62-50

В. И. ВАРШАВСКИЙ, М. А. КИШИНЕВСКИЙ,
В. Б. МАРАХОВСКИЙ, В. А. ПЕСЧАНСКИЙ,
Л. Я. РОЗЕНБЛЮМ, А. Р. ТАУБИН, Б. С. ЦИРЛИН

Автоматное управление асинхронными процессами в ЭВМ и дискретных системах/Под ред. В. И. Варшавского.— М.: Наука. Гл. ред. физ.-мат. лит., 1986.— 400 с.

Освещаются результаты поиска новых схемотехнических и архитектурных решений в области вычислительной техники и дискретной автоматики. Излагается теория апериодических (самосинхронизирующихся) устройств, асинхронное взаимодействие которых осуществляется по принципу «запрос — ответ», а поведение не зависит от величин задержек компонентов. От формальных моделей типа сетей Петри авторы переходят к вопросам синтеза апериодических устройств, устанавливают их самодиагностические свойства и возможность организации саморемонта.

Для специалистов в области вычислительной техники, автоматики, робототехники.

Табл. 19. Ил. 159. Библиогр. 315 назв.

Рецензенты:
член-корреспондент АН СССР *И. Я. Матюхин*,
доктор технических наук *Д. А. Поспелов*

А $\frac{1504000000-081}{053(02)-86}$ 144-86

Но стрелка, стрекоча, семенила дальше, пренебрегая цифрами, которых достигала, касалась, уходила, снова приближалась, снова достигала. Она была равнодушна ко всякой цели, отрезкам, делениям. А следовало бы ей, отмерив шестьдесят секунд, хоть на миг задержаться или подать хоть крошечный сигнал: тут, мол, что-то завершилось. Но по тому, как она торопливо переступала этот предел — она так же переступала и всякую иную, не обозначенную цифрой черточку,— становилось ясно, что всю эту систему цифр и делений ее пути ей лишь подсунули, а она только шла, шла...

Томас Манн

ОТ РЕДАКТОРА

Судьбы научных идей имеют много общего с человеческими судьбами. Не многим выпадает пережить драму. Жизнь большинства тонет в рутине повседневности. Однако практически у всех бывает свой «звездный час». Наступает момент, когда именно индивидуальность личности, ее неповторимые черты в совокупности с обстоятельствами, ее окружающими, оказываются решающими. Важно не пропустить этот момент. Сказанное во многом относится к идеям, лежащим в основе этой книги.

Более четверти века назад были сделаны попытки преодолеть несоответствие между понятием конечного автомата как формального объекта, представляющего последовательность дискретных событий, и понятием дискретного устройства, являющегося динамической системой, т. е. системой, развертывающей эти события в реальном физическом времени. Эта попытка была предпринята Д. Е. Маллером в рамках теории «схем, не зависящих от скорости» (точнее — не зависящих от задержек элементов). В течение многих лет эта теория оставалась «гадким утенком» в ряду многочисленных попыток модернизировать классический подход к проектированию дискретных вычислительных и управляющих схем. Усилиями многих исследователей теория неуклонно, но, к сожалению, медленно развивалась. Но она не имела ни технологической поддержки, ни «социального заказа». Большая часть утверждений ее адептов, казалось, не вызывала доверия. Действительно, стоило ли задумываться о природе внутрисистемного времени и механизмах, его порождающих, если все вопросы, свя-

занные с временны́м поведением схемы, легко снимались введением общесистемных часов? Стоило ли городить сложные теоретические построения, идти на существенное усложнение схем только для того, чтобы избавиться от такого тривиального устройства, как тактовый генератор? В общетеоретическом плане, наверное, стоило, но прикладной смысл таких исследований оставался туманным. Однако развитие техники с неизбежностью приближало тот час, когда рассматриваемая теория должна была оказаться необходимой. Судя по всему, этот час наступил. Этому способствовали две тенденции, связанные с развитием микроуровня (базовой схемотехники и технологии) и макроуровня (системной организации). Следствиями этих тенденций явилось то, что задержки в линиях связи становятся доминирующими по сравнению с задержками активных компонент, да и само временно́е поведение активных компонент становится все более и более неопределенным. Рост сложности устройств в свою очередь выдвинул весьма жесткие, а подчас невыполнимые требования к единой, централизованной системе синхронизации. Все это привело к тому, что на закрытии совещания по проблемам самосинхронизации в Массачусетском технологическом институте в 1979 г. профессор Дж. Деннис заявил: «Я думаю, что выражу общее мнение, если скажу, что проблема самосинхронизации играет важную, если не главную, роль в создании устройств сверхвысокой интеграции». Разработчики микросхем постепенно начинают убеждаться в справедливости этого утверждения.

В 1976 г. вышла книга «Апериодические автоматы» [52], первая и до выхода настоящей книги единственная монография, посвященная теории и практике построения схем, не зависящих от задержек. Изложенный в ней материал позволял проектировать и строить такие схемы. Возникает естественный вопрос, почему же до сих пор устройства такого типа практически не реализованы. Ответ на этот вопрос тривиален: потому, что у представителей промышленности до сих пор не было реальной потребности в переходе на новую схемотехнику.

В последние годы ситуация принципиально изменилась. Имеется несколько функционирующих макетных образцов устройств, основанных на апериодических принципах. Изготовлены первые образцы БИС, содержащие апериодические конвейерные регистры. Тем временем,

как сообщает мартовский 1984 г. номер журнала «Компьютер Дизайн», фирмы ТРВ и «Мостек» выпустили на рынок БИС указанного типа. Университет Карнеги — Меллон совместно с Вашингтонским университетом и фирмой ДЕК разработал проект самосинхронного интерфейса ТРИМОСБАС (правда, не являющегося полностью по зависящим от задержек). Вопросы построения самосинхронизирующихся схем включены в программы большинства американских университетов и вошли в основной американский учебник по микроэлектронике. По крайней мере в двух вузах СССР читаются основы апериодической схемотехники. Состоялась школа по апериодической схемотехнике для работников промышленности. Принцип самосинхронизации используется в ряде перспективных разработок.

Резкое повышение интереса разработчиков к апериодической схемотехнике объясняется, кроме всего прочего, установлением факта полной самопроверяемости апериодических схем относительно константных неисправностей логических элементов. На содержательном уровне на это обращалось внимание еще в книге «Апериодические автоматы», но точное доказательство было получено позже. Учитывая, что проблемы надежности и самопроверяемости становятся стержневыми в развитии вычислительной техники, установление тесной связи этих проблем резко повысило интерес разработчиков к тематике.

Предлагаемая вниманию читателя книга представляет собой «мгновенную фотографию» состояния работ в группе исследователей, которую представляют авторы книги, на конец 1983 г. Естественно, со времени написания книги появились новые результаты. Начиная с работ Д. Е. Маллера, во всех публикациях по теории апериодических автоматов (схем, не зависящих от скорости, самосинхронизирующихся схем) предполагалось, что задержка элемента приведена к его выходу, а задержки проводов пренебрежимо малы или в проводах, имеющих разветвления, сосредоточены до точки разветвления. Эта гипотеза является некорректной, поскольку с ростом степени интеграции величины задержек в проводах начинают существенно влиять на поведение схем. К тому же гипотеза об эквивалентности константных неисправностей элемента возникновению на его выходе бесконечно большой задержки лишь весьма приблизительно отражает реальные дефекты в СБИС. С этой точки зрения

возникает вопрос о возможности построения схем, поведение которых не зависит от задержек транзисторов и проводов как базовых схемотехнических элементов, а не вентилей, являющихся составными объектами. Сегодня на этот вопрос (для большого класса схем) получен положительный ответ, но этот материал не вошел в книгу.

Следующий вопрос развития тематики — это вопрос о декомпозиции апериодических схем. Проблема декомпозиции сильно связана с использованием содержательных, неформальных соображений, когда топология размещения подсхем навязывает и типы декомпозиции. Особенно это становится важным при использовании принципов объектно-модульного проектирования («планов размещения»), ориентированных на библиотеки масок. В этом случае декомпозиция сводится не к разбиению устройства на подсхемы, а к организации интерфейса между готовыми подсхемами для обеспечения заданного поведения. При этом критериями минимизации являются число проводов, соединяющих подсхемы, и их длина, а ограничением — независимость поведения от величин задержек в указанных пределах. Предварительные результаты говорят о том, что апериодический принцип открывает ряд заманчивых возможностей, но здесь требуются существенные теоретические исследования.

От имени авторов мне хочется выразить надежду, что эта книга окажется полезной не только для научных работников, но и для разработчиков ЭВМ и дискретных систем и стимулирует внедрение апериодической схемотехники в промышленные разработки.

Мне хотелось бы скрыть от читателей конкретный вклад каждого из авторов в создании книги: все первоначальные варианты глав неоднократно подвергались серьезной переработке, так что мы несем коллективную ответственность за возможные ошибки и огрехи.

Авторы выражают глубокую благодарность рецензентам Н. Я. Матюхину и Д. А. Поспелову и особенно редактору книги С. В. Петрову, приложившему немало усилий для того, чтобы сделать книгу более логичной и удобочитаемой, а также нашим коллегам, чьи фамилии не числятся в списке авторов, но чья работа во многом способствовала получению изложенных здесь результатов.

Ленинград, август 1984 г. *В. Варшавский*

Также и времени нет самого по
себе, но предметы
Сами ведут к ощущенью того,
что в веках совершилось,
Что происходит теперь и что воспоследует позже.

Лукреций Кар

Глава 1. ВВЕДЕНИЕ

Современные системы управления содержат весьма важный подкласс систем — управляющие устройства, осуществляющие координацию процессов с конечным множеством состояний. Понятие «состояние системы» должно быть интуитивно понятно читателю, при необходимости оно может и будет уточняться ниже в зависимости от синтаксиса языка описания модели системы. Типичный пример системы с конечным числом состояний — управление обменом информацией в сетях ЭВМ. При этом не исключается наличие непрерывных переменных. Например, при управлении металлорежущим станком учитываются координаты режущей кромки, скорости изменения этих координат, составляющие усилий на инструмент. Однако такие переменные, как включение и выключение двигателей подачи, состояние редуктора, данные о дальнейшем режиме и пр. являются дискретными. Ими определяется последовательность дискретных состояний системы, реализация которой и составляет управление процессом обработки. Ряд других примеров можно найти в самых различных областях: это управление пуском и остановкой мощных энергоустановок, управление локомоторной деятельностью роботов и т. д.

Существенными чертами таких систем управления являются:

а) наличие у процессов (и составляющих их подпроцессов) явно выраженных фаз, в течение которых происходят изменения состояний процессов, причем в каждой фазе системе управления выдается информация о ее завершении;

б) согласованность — переход к следующей фазе процесса инициируется системой управления только после получения сигнала об окончании предыдущего перехода;

в) параллельность — возможность одновременных переходов в нескольких подпроцессах;

г) асинхронность — отсутствие ограничений на относительную длительность осуществления переходов, зависящую от многочисленных неконтролируемых факторов.

Параллельность и *асинхронность* являются естественными чертами практически всех технических систем, проявляющимися как в системах управления, так и в управляемых объектах. Вариации длительностей переходов в элементах и блоках систем управления и каналах связи порождаются как разбросом технологических параметров, так и изменениями физических условий функционирования — температуры, давления и пр. Асинхронность выдвигает требование инвариантности поведения системы к изменениям длительностей переходов за исключением тех случаев, когда длительность перехода несет информацию, существенную для процесса управления. *Согласованность* — менее очевидное свойство поведения системы, предназначенной для реализации желаемого поведения объекта управления, отражающее причинно-следственные связи взаимодействия объекта и системы управления. Тот факт, что система управления является согласованной параллельной асинхронной системой, поведение которой определяется взаимодействием с управляемым объектом и внешней средой, дает возможность рассматривать совокупность «объект управления — внешняя среда» как «единое целое без боязни упустить из виду некоторые качественные эффекты взаимодействия. Поскольку речь идет о системе с конечным множеством состояний, то кажется естественным в качестве модели такой системы рассматривать конечный автомат. Однако сложность использования классической модели конечного автомата заключается в том, что эта модель игнорирует понятие физического времени. Именно поэтому каждый раз, когда мы сталкиваемся с проблемой технической реализации конечного автомата, возникает необходимость существенного изменения исходной модели: техническое устройство, представляющее конечный автомат, является динамической системой со всеми присущими такой системе особенностями.

Современная техника преодолевает возникающие здесь трудности, как правило, за счет использования систем внешней синхронизации.

Существуют *два типа систем синхронизации.* Если длительности переходов в элементах и блоках системы являются величинами одного порядка и их верхние границы известны, то влияние физического времени учитывается путем не зависящего от системы, основанного только на знании этих верхних границ определения моментов завершения переходов. Синхронизация такого типа привлекательна тем, что от формальной модели легко перейти к технической реализации. Однако неучет вариаций длительности переходов приводит к «недоиспользованию» скоростных возможностей системы. Кроме того, если длительность перехода превысит максимальную ожидаемую, это может привести к неправильной работе всей системы. Последнее явление, называемое *параметрическим отказом*, является одной из наиболее распространенных причин возникновения неисправностей. Этот факт хорошо известен инженерам, занимающимся эксплуатацией вычислительной техники: при возникновении сбоев они пытаются устранить их снижением частоты тактового генератора.

В тех случаях, когда длительности переходов в некоторых блоках существенно превышают длительности переходов в других блоках схемы, либо когда вариации длительностей столь велики, что делают неразумным использование первого типа синхронизации, выявление нового состояния осуществляется путем периодического зондирования асинхронных сигналов тактовыми импульсами. Типичным примером такой синхронизации является система обработки прерываний. Казалось бы, повышение частоты зондирующих (тактовых) сигналов и быстродействия логических элементов позволяет получить сколь угодно точную синхронизацию. Однако было обнаружено и исследовано *явление арбитража* — аномального поведения схемы, возникающего, когда тактовый сигнал подается в момент изменения зондируемого асинхронного сигнала. В такой конфликтной ситуации арбитр должен вынести одно из двух решений: считать тактовый сигнал поданным до или после изменения зондируемого. Сейчас можно считать (см. гл. 9) установленным фактом невозможность построения абсолютно надежных арбитров, синхронизаторов и инерциальных задержек в базисах, не содержащих аналоговых схем (например, компараторов). С ростом частоты зондирования растет вероятность возникновения конфликтных ситуаций, а с ростом быстро-

действия логических элементов возрастает относительная длительность пребывания схем с памятью в аномальном состоянии.

Таким образом, несмотря на широкое использование синхронных систем в современной технике, на пути повышения их быстродействия имеются очевидные трудности. Последние существенно усугубляются в связи с переходом на субмикронную микроэлектронную технологию, в которой основные задержки в распространение сигналов вносят не активные элементы, а соединительные линии, разброс задержек в которых может в сотни раз превышать величины задержек в транзисторном переходе. По мнению ведущих исследователей, *использование принципа внешней синхронизации в следующих поколениях сверхбольших и сверхбыстродействующих интегральных схем весьма проблематично*[1]). В субмикронных интегральных схемах размеры транзисторного перехода приближаются к тысячам атомных радиусов, зоны взаимодействия становятся сравнимыми с длинами волн сигналов взаимодействия, в связи с чем можно говорить об *эквихронных зонах*, т. е. таких зонах, компоненты которых функционируют в «одном времени», и о проблемах согласования (синхронизации) функционирования объектов, расположенных в различных эквихронных зонах кристалла, хотя сегодня постановка таких проблем может рассматриваться как спекуляция. С ростом сложности систем проблемы, возникающие на микроуровне, начинают проявляться и на системном уровне.

Учитывая, что термин «синхронный» обычно связывается с рядом вполне определенных технических решений, далее имеет смысл рассматривать лишь вопросы организации временно́го поведения сложных систем или, как говорилось выше, управления параллельными асинхронными процессами. При этом нас будут интересовать различные уровни детализации описания — от транзистора и провода до устройства высокой сложности,— моделью поведения которых может являться процесс с конечным множеством состояний. Было бы удобно рассматривать все уровни детализации с единых методологических позиций.

Прежде всего следует уточнить, что физическое понятие «время» «выражает порядок смены явлений»

[1]) См., например, [207, 291].

(«Физический энциклопедический словарь», 1983, с. 592). Отсюда следует, что нет другого способа выразить время, кроме как через отношения между наблюдаемыми событиями, состояниями, явлениями, через их последовательности и длительности. Внешняя синхронизация есть процесс координации событий в системе с состояниями «часов», не принадлежащих самой системе. События, происходящие в таких «внешних часах», в принципе, не имеют причинно-следственных связей с событиями в системе и во взаимодействующей с ней внешней среде: «После этого не значит из-за этого»[1]). По-видимому, именно этот факт является источником методических трудностей, возникающих при построении систем внешней синхронизации. С другой стороны, смена состояний системы, происходящая в реальном физическом времени, порождает систему отношений между событиями в системе и событиями во внешней среде, и эти отношения являются системным временем для данной конкретной системы и внешней среды. Квантование времени событиями, происходящими в системе и во внешней среде, а не во внешних часах, приводит нас к понятию *самосинхронизации*. Привлекательность этой развиваемой в книге концепции существенно зависит от возможности ее использования в технических приложениях, в частности, при построении согласованных параллельных асинхронных систем.

Как уже отмечалось выше, согласованная параллельная асинхронная система представляет собой совокупность взаимосвязанных элементов (блоков, подпроцессов). Событием в такой системе является *переход* — смена состояния системы. Для единообразного описания поведения таких систем на различных уровнях абстракции необходимы изобразительные средства, позволяющие задавать систему отношений между событиями, определяющую системное время.

Попытки ввести определяющую время систему отношений предпринимались еще Аристотелем, но первая строгая система была предложена, по-видимому, Дж. Н. Финдлеем в 1941 г. (см.: Кондаков Н. И. Логический словарь-справочник.— М.: Наука, 1975, с. 95). Од-

[1]) В принципе «часы» могут быть отнесены к внешней для системы среде. Однако именно разделение «часов» и внешней среды отличает в теории автоматов синхронный автомат от асинхронного.

нако потребовалось еще почти 20 лет, чтобы появились работы, формализующие отношения между событиями в системах. В 1959 г. Д. Е. Маллер и У. С. Бартки предложили модель, которую сейчас называют *диаграммами переходов* или *диаграммами Маллера*, для описания поведения асинхронных схем (схем Маллера). В 1962 г. К. А. Петри предложил модель (сейчас называемую *сетями Петри*), позволяющую описывать потоки дискретных событий инвариантно к абсолютной длительности этих событий. Каждая из указанных моделей обладает своими преимуществами и недостатками, однако по мало понятным причинам сети Петри являют собой пример достаточно модной и бурно развивающейся области знаний, а язык диаграмм Маллера пользуется гораздо меньшей популярностью, чем заслуживает.

Для рассмотрения поведения системы в гл. 2 предлагается некоторая метамодель, допускающая возможность различных интерпретаций. Эта модель, в которой отношения между состояниями системы отражают лишь множество причинно-следственных отношений между событиями в системе, названа *асинхронным процессом*. Реализация тех или иных логических возможностей здесь определяется как семантикой системы, так и внесистемными факторами, источником которых является внешняя среда. Вначале учет внесистемных факторов не нужен и невозможен. Существенно, однако, что внешнюю среду мы также можем рассматривать как асинхронный процесс. С точки зрения системы все другие взаимодействующие с ней системы могут считаться компонентами внешней среды. Язык асинхронных процессов позволяет уже на этом уровне абстракции решать принципиальные вопросы согласования и композиции процессов, а, значит, и систем. При согласовании процессов возникает необходимость построения дополнительного (согласующего) процесса, который может рассматриваться как формальная модель протокола взаимодействия системы с внешней средой.

Первый шаг, связанный с учетом семантики протекающих в системе процессов, сводится к интерпретации асинхронного процесса на языке более низкого уровня абстракции. В качестве таких языков в работе (гл. 2) используются следующие: сети Петри, диаграммы Маллера, параллельные асинхронные блок-схемы и конечные автоматы. Каждый из этих языков отражает некоторые

специфические особенности поведения систем, и поэтому все они применяются на практике. Это обстоятельство делает желательным рассмотрение всех четырех указанных языков.

Для интерпретации, равно как и для решения задач согласования и композиции, используется *структурирование состояний* исходного асинхронного процесса. Термин «структурирование» происходит от слова «структура» — «(лат. structura — строение, расположение) — определённая взаимосвязь, взаиморасположение составных частей, характеризующее строение, устройство чего-либо» (БСЭ, 3-е изд., т. 24, кн. 1, с. 598). В данном контексте структурирование состояний системы предполагает уточнение понятия состояния, выражающееся, например, в том, что состояния представляются векторами с двоичными или целочисленными компонентами или же в состояниях выделяются некоторые характеризующие их упорядоченные компоненты — скажем, входные, выходные и внутренние.

Нашей конечной целью является построение композиции управляемых подпроцессов и системы управления. Однако вначале система управления как таковая игнорируется, внимание обращается только на координацию переходов в подпроцессах. В структурированных подпроцессах можно выделить два типа переменных: *информационные и фазовые* (управляющие) *переменные*, инициирующие и завершающие очередной переход. Для определенности будем рассматривать двоичные фазовые переменные. Именно они в первую очередь выделяются при структурировании состояний асинхронного процесса на этапе описания системы управления. Более того, предполагается, что переход, инициируемый изменением входной фазовой переменной, завершается тем, что выходная фазовая переменная повторяет значение входной. Тогда относительно фазовой переменной подпроцесс может быть представлен как задержка, величина которой не определена и подвержена неконтролируемым изменениям. Если задержки, вносимые подпроцессами, конечны, то построенное таким образом описание системы управления отражает взаимную координацию фаз подпроцессов независимо от длительности этих подпроцессов. Следовательно, такое описание содержит достаточную информацию для построения системы управления, оставляя открытым вопрос о возможности и способах ее построения.

При переходе от этапа задания процесса управления к описанию системы управления прежде всего необходимо решить вопрос о кодировании фазовых переменных подпроцессов и состояний системы в целом. Коды, отвечающие необходимым для организации управления требованиям, изучаются в гл. 3 работы и названы *самосинхронизирующимися кодами*. Коль скоро введено двоичное кодирование переменных, то в качестве элементов системы управления будем рассматривать функциональные элементы, вход-выходное поведение которых описывается булевыми функциями или конечно-автоматными уравнениями. Будем говорить, что *схема* из функциональных элементов *моделирует систему управления* параллельными асинхронными подпроцессами, заданными на одном из вышеназванных языков, если а) такое описание гомоморфно исходной диаграмме переходов схемы в том смысле, что упорядочения общих для исходной и расширенной диаграмм переходов переменных совпадают; б) каждому фазовому сигналу соответствует в точности один выход функционального элемента схемы.

Поскольку диаграмма переходов схемы по определению описывает упорядоченности переходов, возможные при любом соотношении длительностей переходов, она не изменится, если между выходом каждого элемента моделирующей схемы и проводами, передающими значение этого выхода на входы других элементов схемы, подключить произвольную по величине конечную задержку. Как уже было сказано, по отношению к фазовым сигналам (входному и выходному) подпроцесс представляет собой задержку, следовательно, моделирующая схема с подключенными к выходам элементов задержками обеспечивает требуемую координацию фаз подпроцессов, моделируемых этими задержками.

Далее основное внимание уделяется теории *схем, поведение которых* в указанном смысле *не зависит от задержек элементов (апериодических схем, схем Маллера, схем, не зависящих от скорости)*. Классическая теория таких схем, основы которой были заложены Маллером и Бартки, фактически оставляла открытыми вопросы возможности синтеза на элементах ограниченного базиса и вопросы взаимодействия таких схем с внешней средой. Эти ограничения были преодолены в теории апериодических автоматов, являющейся существенным расширением теории схем Маллера.

Диаграммы Маллера были введены с целью анализа схем на независимость их поведения от задержек элементов. С другой стороны, по полумодулярной диаграмме Маллера можно составить логические уравнения элементов схемы. Однако эта процедура не решает задачу синтеза, так как она может привести к необходимости использования элементов, для которых по существующей технологии не выполняются исходные предпосылки теории Маллера (например, оказывается неверным предположение, что задержка элемента сосредоточена на его выходе). Для некоторых элементов может просто не существовать технологической реализации, свободной от состязаний. В гл. 5 показано, что для любой диаграммы Маллера от k переменных существует полумодулярная расширенная диаграмма Маллера от $2k$ переменных такая, что исходная диаграмма является проекцией расширенной диаграммы на множество исходных переменных, причем расширенная диаграмма может быть реализована на элементах типа И-ИЛИ-НЕ.

Такой подход открывает возможность построения по произвольной диаграмме Маллера *моделирующей ее схемы* в базисе И-ИЛИ-НЕ. Из соображений двойственности этот результат распространяется и на базис ИЛИ-И-НЕ. Заметим, что речь идет об элементах с неограниченным числом входов. Технологические требования ограничивают возможности использования этого метода, так что возникает вопрос о минимальных схемных базисах. Такими базисами являются: для дистрибутивных диаграмм Маллера — элемент 2И-НЕ (или, в силу двойственности, 2ИЛИ-НЕ), а для произвольных полумодулярных диаграмм Маллера — 2И-НЕ, 2ИЛИ-НЕ[1]). Доказательства *функциональной полноты* конструктивны в том смысле, что они обеспечиваются методами синтеза.

Класс *безопасных устойчивых сетей Петри* (см. гл. 2) также допускает непосредственный переход к моделирующим их схемам, не зависящим от скорости. Для этих целей в гл. 5 используются схемы из стандартных фрагментов, в которых каждой позиции сети Петри соответствует триггер, а каждому событию (переходу) — провод. Состояние (*маркировка*) сети моделируется состояниями указанных триггеров, а факт наступления события — циклом 1-0-1 или 0-1-0 изменений состояния соответству-

[1]) Цифра в обозначении есть число входов.

ющего провода. Схема избыточна, однако (что важно для наших целей) она гомоморфна исходной сети Петри. Схемным базисом для построения таких схем являются элементы И-ИЛИ-НЕ или ИЛИ-И-НЕ, а минимальным базисом — 2И-НЕ или 2ИЛИ-НЕ (как и в случае реализации по дистрибутивным диаграммам Маллера). Этот факт является косвенным указанием на то, что класс полумодулярных диаграмм Маллера шире класса безопасных устойчивых сетей Петри. Во всяком случае, не удалось найти способ сведения недистрибутивных полумодулярных диаграмм Маллера к безопасным устойчивым сетям Петри.

Язык *параллельных асинхронных блок-схем* (гл. 5) описывает совместное функционирование подпроцессов, представляемых посредством операторов. Фактически для построения не зависящей от скорости схемы, моделирующей блок-схему, достаточно реализовать набор подсхем, моделирующих фрагменты параллельной асинхронной блок-схемы. Еще Дж. Деннисом был введен следующий набор фрагментов: бифуркатор, сборка, синхронизатор, условный переход и цикл. Деннис предполагал, что каждый оператор выполняется отдельным блоком. Если же использовать для выполнения всех однотипных операторов один блок (например, все операторы сложения выполнять на одном сумматоре), то набор подсхем, реализующих эти фратменты, оказывается недостаточным. Необходима специальная схема, управляющая многократным использованием одного и того же блока.

В предлагаемой системе (также гл. 5) каждый управляющий блок имеет две фазы работы. Одну из них можно назвать рабочей, во второй происходит возврат блока в исходное состояние. Предполагаются два варианта организации управления. В первом варианте блоки в порядке, предусмотренном блок-схемой, переходят в рабочую фазу, а затем в том же порядке возвращаются в исходное состояние. Во втором варианте каждый блок реализует обе фазы, прежде чем передать управление другим блокам. В общем случае по затратам оборудования предпочтение нельзя отдать ни одному из вариантов. Выбор зависит от конкретных требований к системе управления.

Альтернативой распределенной схеме управления, получающейся при составлении ее из стандартных блоков, соответствующих фрагментам блок-схем, является *асинхронный автомат*. Его применение выгодно для реализации сложных фрагментов алгоритма, в которых отсутствуют параллельные участки. Будем считать, что переход автомата инициируется установкой входного фазового сигнала, а после окончания перехода изменяется выходной фазовый сигнал. Если, кроме того, правильность работы автомата не зависит от задержек элементов реализующей его схемы, будем называть автомат *апериодическим*. По любому конечному автомату можно построить соответствующий апериодический автомат, для чего предлагаются два метода: а) процедура синтеза автомата в виде стандартной схемы с памятью на D-триггерах или T-триггерах с использованием индикации момента окончания переходных процессов (гл. 4); б) процедура представления апериодического автомата как композиции схем Маллера (гл. 5), построенных по диаграммам переходов для фиксированных значений входных переменных; взаимодействие с внешней средой обеспечивается при этом замыканием некоторых связей через внешнюю среду. Организация взаимодействия с внешней средой имеет много общего с соединением не зависящих от скорости схем по теореме Маллера — Бартки.

В ряде случаев природа взаимодействия подпроцессов такова, что возникают конфликтные ситуации, разрешение которых может привести к существенному замедлению работы устройств. Так, например, любая предварительно установленная процедура предоставления общего ресурса, не учитывающая вариаций длительности подпроцессов, либо не исключает арбитража, либо приводит к длительной приостановке одного подпроцесса, когда согласно процедуре очередь на предоставление ресурса подошла для другого подпроцесса, а ресурс ему не нужен. Наличие конфликтных ситуаций соответствует нарушению устойчивости сетей Петри или полумодулярности диаграмм Маллера и приводит к аномальному поведению соответствующих схем. Исследование явления *арбитража* (гл. 9) с учетом динамических характеристик схемотехнических элементов показывает, что на логических элементах невозможно реализовать абсолютно надежную схему арбитра, свободную от метастабильной или колебательной аномалий. С другой стороны, условием выхода схемы арбитра (триггера) из аномального состояния является выход за пороговое значение разности напряжений на его плечах, что позволяет индициро-

вать завершение аномалии аналоговыми компараторами. Поэтому могут быть построены не зависящие от скорости схемы управления процессами, содержащими конфликтные ситуации. Таким образом, класс моделируемых схемами управления ситуаций и процессов может быть существенно расширен, однако возникает весьма существенная задача выявления таких ситуаций. Кроме того, проблема анализа связана не только с выявлением конфликтных ситуаций. Дело в том, что хотя упомянутые выше стандартные методы синтеза и обеспечивают построение не зависящих от скорости схем, но, как всякие общие методы, они превосходны для всех случаев, но могут оказаться весьма неэкономичными для каждого конкретного случая. Это подтверждается опытом развития техники. Конкретные схемы приходится изобретать. Однако изобретенные схемы обязательно требуют проверки на независимость правильности их работы от задержек элементов и соответствия их поведения исходному заданию. Хотя общие признаки независимости от скорости известны еще из работ Маллера, их проверка непосредственно по диаграмме переходов достаточно сложна: ее сложность растет экспоненциально с ростом числа элементов схемы. В работе (гл. 8) рассматриваются методы анализа, позволяющие за приемлемое время (с использованием ЭВМ) проверять схемы, содержащие до сотен элементов.

Схемы, не зависящие от скорости, обладают одним весьма полезным свойством — *самопроверяемостью* относительно константных неисправностей. *Константная неисправность* элемента эквивалентна возникновению в нем бесконечно большой задержки. А это означает, что если по логике работы схемы элемент должен переключиться, то при его константной неисправности дальнейшее функционирование схемы прекратится. Этот факт важен с двух точек зрения. Во-первых, схемы, не зависящие от скорости, останавливаются и, следовательно, не работают «нештатно» при константных неисправностях, что важно для многих систем управления. Во-вторых, удается строго показать (гл. 10), что такие схемы являются полностью самопроверяемыми относительно константных неисправностей элементов. Наличие константной неисправности может быть выявлено за счет временной избыточности схемы — на основе дополнительных внемодельных факторов можно указать критическое время, в течение

которого в схеме должно произойти изменение данного сигнала. Отсутствие такого изменения в течение критического интервала говорит о наличии неисправности в схеме, что позволяет локализовать неисправность с точностью до технологического элемента замены без применения специальных тестов, используя простую схему встроенного контроля. Возможность получения сигнала локальной неисправности позволяет организовать саморемонт схемы. Он особенно эффективен для схем с высокой степенью однородности — таких, например, как память, где резервными могут быть одна резервная ячейка и одна числовая (разрядная) линейна. В качестве иллюстративного примера приводится схема счетчика с одним резервным разрядом, парирующего один отказ. Здесь, однако, следует отметить, что сказанное не относится к возможности возникновения неисправности в момент переключения элемента, так как такая неисправность может привести к возникновению конфликтной ситуации или перевести схему в состояние, не предусмотренное условиями ее функционирования. Вопросы парирования таких отказов требуют дополнительного исследования.

Сформулированные здесь и достаточно полно освещенные в работе вопросы имеют более широкую область применения, чем только построение систем управления. Сами апериодические автоматы также могут интерпретироваться как асинхронные процессы, а это открывает широкие возможности для блочного синтеза самосинхронизирующихся вычислительных и управляющих схем высокой сложности, где в качестве блоков могут выступать как стандартные узлы вычислительной техники (гл. 11), так и комплексированные системы. В этом случае можно говорить об иерархии блочных описаний.

Весьма важным классом устройств, для которых изложенные методы и подходы оказываются эффективными, являются интерфейсы (гл. 7). Имеется в виду широкий класс интерфейсов — от интерфейсов между эквихронными зонами в сверхбольших интегральных схемах до межкристалльных межмодульных и межмашинных интерфейсов и интерфейсов в мультипроцессорных системах. Характерным для таких систем является: 1) необходимость управления каналами связи с неконтролируемыми задержками, 2) «перекосы» задержек в параллельных пучках проводов, 3) взаимодействие с абонентами, время реакции которых непредсказуемо. Кроме того, с

точки зрения организации саморемонта интерфейсы привлекательны однородностью каналов связи и приемопередающих устройств. Примеры самосинхронизирующихся интерфейсов, обеспечивающих правильный обмен независимо от задержек в каналах связи и элементах системы управления, являются хорошей иллюстрацией эффективности предлагаемых подходов.

Как для интерфейсов, так и для ряда систем обработки существенным резервом повышения производительности систем является конвейеризация процессов, причем наиболее эффективной является организация асинхронного конвейера. Построение асинхронных конвейерных структур (§ 5.5 и гл. 11) также связано с синтезом согласованных параллельных асинхронных систем и управляющих схем, не зависящих от скорости. Соответствующие моделирующие схемы иллюстрируют применение полученных в работе результатов.

Мы знаем: время растяжимо.
Оно зависит от того,
Какого рода содержимым
Вы наполняете его.

С. Маршак

Глава 2. АСИНХРОННЫЕ ПРОЦЕССЫ И ИХ ИНТЕРПРЕТАЦИЯ

В настоящее время на вооружении проектировщиков дискретных вычислительных, информационных и управляющих систем имеется ряд *моделей*, позволяющих описать динамику функционирования этих систем с учетом возможного параллелизма и асинхронности взаимодействия их подсистем. Под *математической моделью* здесь, как обычно, понимается «приближенное описание какого-либо класса явлений внешнего мира, выраженное с помощью математической символики» (Математическая энциклопедия, т. 3, с. 574). Каждая из этих моделей отражает те или иные аспекты поведения систем и в силу этого имеет ограниченное применение. Тем не менее наличие общих свойств у таких моделей позволяет предложить некоторую удобную в методологическом отношении *метамодель*, которая порождает частные, объектные модели. Термин «метамодель» понимается как модель, применяемая для исследования и описания некоторого класса моделей, по аналогии с другими понятиями, образованными посредством приставки «мета». Механизм такого порождения определяется способами *интерпретации* — «задания значения (смысла) математических выражений (символов, формул и т. д.)» (там же, т. 2, с. 635) — основных понятий метамодели, в качестве которой здесь будет использована некоторая формализация, носящая название «асинхронный процесс».

§ 2.1. Асинхронный процесс

2.1.1. Определение. Само понятие «асинхронный процесс» будет далее использоваться как для задания наиболее компактного описания поведения системы, так и для решения задач анализа и реализации системы. Чтобы

выделить тот или иной аспект моделирования, будем различать термины «дескриптивный асинхронный процесс» и «асинхронный процесс».

Определение 2.1. Назовем *дескриптивным асинхронным процессом DP* четверку $\langle S, \mathcal{F}, \mathcal{I}, \mathcal{R}\rangle$, в которой: S — непустое множество *ситуаций*, $\mathcal{F}$ — бинарное *отношение*, определенное на множестве $\mathcal{P}(S)$ подмножеств ситуаций, которое некоторому подмножеству ситуаций $\alpha \in \mathcal{P}(S)$ ставит в соответствие подмножество ситуаций $\beta \in \mathcal{P}(S)$, $\alpha \neq \beta$, $\mathcal{I}$ — множество *инициаторов*, $\mathcal{I} \subseteq \mathcal{P}(S)$, таких, что: 1) для любого $i \in \mathcal{I}$ существует такое α, что $i\mathcal{F}\alpha$; 2) если $i\mathcal{F}\alpha$, $\alpha \notin \mathcal{I}$, то из $\alpha\mathcal{F}\beta$ следует, что $\beta \notin \mathcal{I}$; 3) для любого $i \in \mathcal{I}$ не существует α, $\alpha \notin \mathcal{I}$, такого, что $\alpha\mathcal{F}i$; $\mathcal{R}$ — множество результантов, $\mathcal{R} \subseteq \mathcal{P}(S)$, таких, что: 1) для любого $r \in \mathcal{R}$, если $r\mathcal{F}\alpha$, то $\alpha \in \mathcal{R}$; 2) для любого $r \in \mathcal{R}$ существует такое α, что $\alpha\mathcal{F}r$.

Если для некоторого процесса $\mathcal{I} = \mathcal{R} = \emptyset$, то такой процесс называется *автономным*.

Прокомментируем определение 2.1.

Слова «ситуация» и «процесс» нужно понимать в общеупотребительном смысле. «Ситуация (позднелат. situatio — положение) — сочетание условий и обстоятельств, создающих определенную обстановку, положение» («Советский энциклопедический словарь», 1980, с. 1226). Конкретизация термина «ситуация» зависит от интерпретации понятия асинхронный процесс и будет дана ниже. К этому следует добавить, что в любой момент наблюдения асинхронный процесс может находиться в одной и только одной ситуации.

«Процесс (от лат. processus — продолжение) — 1) последовательная смена состояний...; 2) совокупность последовательных действий для достижения какого-либо результата...» (БСЭ, 3-е изд., т. 21, с. 161). В данном случае процесс описывает динамику смены ситуаций. Термин «процесс» уточняется прилагательными *«дескриптивный»* и *«асинхронный»*. Первое уточнение подчеркивает, что эта модель предназначена главным образом для компактного описания моделируемой системы — в отличие от другой модели, которая будет описана ниже; эта другая модель, носящая название «асинхронный процесс» (она не содержит в своем названии слова «дескриптивный»), ориентирована в основном на анализ свойств системы и решение вопросов ее аппаратной реализации. Уточнение «асинхронный» используется для того, чтобы

показать, что категория времени формально не фигурирует в определении; привязка к сущности «время» осуществляется с помощью отношения $\mathcal{F}$: если $\alpha\mathcal{F}\beta$, то время перехода из подмножества ситуаций α в подмножество ситуаций β не оговаривается — оно может быть произвольным, но конечным.

Введенное отношение $\mathcal{F}$ является отношением следования и может пониматься как *логическая необходимость* (термин заимствован из модальной логики). Иначе: запись $\alpha\mathcal{F}\beta$ означает, что α является совокупностью причин, а β — возможными следствиями. Отношение $\mathcal{F}$, определенное на множестве подмножеств ситуаций, подчеркивает недетерминированность (дескриптивного) асинхронного процесса, в частности «приблизительность» его задания, вызванную отсутствием точных знаний о том, в какой ситуации из множества α находится процесс и в какую ситуацию из множества β он должен перейти.

Инициаторы — подмножество ситуаций, активизирующих процесс. Назначение инициаторов делается на основе семантики процесса. Введенное в определении 2.1 ограничение на их выбор говорит лишь о том, что инициатор не может быть только следствием некоторого подмножества ситуаций, он обязательно должен быть причиной.

Результанты — подмножества, состоящие из финальных ситуаций. Их выбор также делается на основе семантики процесса. Ограничения в определении 2.1 означают, что результант не может быть только причиной, он обязательно должен быть следствием какой-то причины. По существу, при определении результантов подчеркивается тот факт, что если какая-либо ситуация из S объявлена результантом, то любая другая «следующая за ней» ситуация, если она, конечно, существует, также является результантом.

Пример 2.1. Пусть асинхронный процесс задан в следующем виде:

$$S = \{s_1, s_2, s_3, s_4, s_5\};\ \{s_1, s_2\}\mathcal{F}\{s_3, s_4, s_5\},$$
$$\{s_1, s_2, s_3, s_4\}\mathcal{F}\{s_5\};\ \mathcal{I} = \{i_1\},\ i_1 = \{s_1, s_2\};$$
$$\mathcal{R} = \{r_1\},\ r_1 = \{s_5\}.$$

В этом примере в соответствии с определением 2.1 отношение $\mathcal{F}$ задается на множестве подмножеств ситуаций. Поскольку процесс в моменты наблюдений находится в одной из ситуаций, возникает вопрос об установле-

нии отношения между ситуациями, т. е. об уточнении отношения $\mathcal{F}$. Конкретизация описания процесса может быть достигнута, если использовать отношение M, определенное на $S \times S$. Запись $s_j M s_k$ интерпретируется как *логическая возможность* (опять-таки термин из модальной логики) перехода из s_j в s_k, а $s_j \neg M s_p$ — как «логическая невозможность» перехода из s_j в s_p (s_j не является причиной s_p). Может одновременно выполняться $s_j M s_k$ и $s_j M s_t$. Отношение M сохраняет недетерминированный характер асинхронного процесса. Если для некоторой s_j существует единственная s_k такая, что $s_j M s_k$, то логическая возможность означает также и логическую необходимость перехода из s_j в s_k.

Если процесс задается с помощью отношения M, множества I инициаторов и R результатов естественно задать как подмножества $S (I \subset S, R \subset S)$.

Отношение M можно изобразить орграфом. Вершины его соответствуют ситуациям процесса. Дуга исходит из s_j и входит в s_k, если $s_j M s_k$. Ситуации s_l и s_m, для которых $s_l \neg M s_m$ и $s_m \neg M s_l$, дугами не связываются.

Пример 2.1 (продолжение). На рис. 2.1 изображены два возможных уточнения дескриптивного асинхронного процесса примера 2.1. Рис. 2.1, *а* соответствует отношению M_1: $s_1 M_1 s_3$, $s_1 M_1 s_4$, $s_1 M_1 s_5$, $s_3 M_1 s_5$, $s_4 M_1 s_5$, рис. 2.1, *б* — отношению M_2: $s_1 M_2 s_3$, $s_1 M_2 s_5$, $s_2 M_2 s_4$, $s_2 M_2 s_5$, $s_3 M_2 s_5$, $s_4 M_2 s_5$.

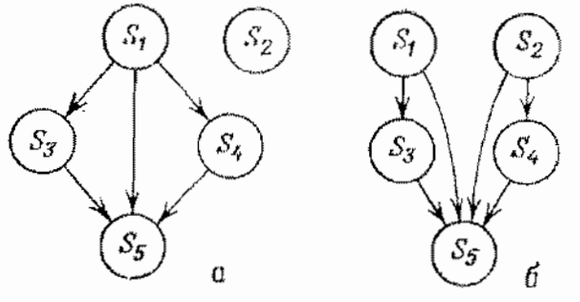


Рис. 2.1. Два асинхронных процесса, интерпретированных ориентированными графами

Отметим, что если задано $\alpha \mathcal{F} \beta$, то для некоторой пары s_j, $s_k \in \alpha \cup \beta$, для которой $s_j M s_k$, может выполняться:

1) $s_j, s_k \in \alpha$,
2) $s_j \in \alpha$, $s_k \in \beta$,
3) $s_j, s_k \in \beta$.

Отношение M *транзитивно*, но *не рефлексивно*, поскольку

в определении 2.1 не допускалось $\alpha \mathcal{F} \alpha$, и, вообще говоря, *не симметрично*. Так, для рис. 2.1, *а* $s_1 M_1 s_5$, но $s_5 \neg M_1 s_1$. Заметим, что $s_j M s_k$ не обязательно означает, что из ситуации s_j процесс сразу же попадает в s_k. Пусть, например, $s_j M s_k$, $s_j M s_l$, $s_l M s_k$. Тогда отношение M для данного фрагмента процесса можно изобразить так, как показано на рис. 2.2, *а*. Изучение семантики процесса может показать, что из ситуации s_j в ситуацию s_k процесс попадает только через ситуацию s_l, хотя s_j является причиной и s_k, и s_l. Поэтому может представлять интерес еще одно уточнение описания процесса, заданного на фиксированном множестве ситуаций. Оно достигается введением отношения F непосредственного следования ситуаций. При указанном выше ограничении для фрагмента рис. 2.2, *а* отношение F определяется следующим образом: $s_j F s_l$, $s_l F s_k$, а граф этого отношения имеет вид рис. 2.2, *б*. Если же за s_j может непосредственно следовать s_k, то граф отношения F совпадает с графом отношения M (рис. 2.2, *а*).

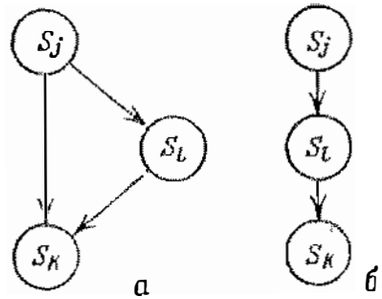


Рис. 2.2. Два фрагмента асинхронных процессов

Отметим, что отношение M можно выразить через степени отношения F. Запись $s_i F^n s_k$ означает, что существуют $n-1$ промежуточных ситуаций $s_a, s_b, \ldots, s_w$, для которых имеет место $s_i F s_a$, $s_a F s_b$, $\ldots$, $s_w F s_k$, т. е. существует траектория, включающая $n+1$ вершину и n дуг и ведущая из s_i в s_k.

Итак, от определения дескриптивного асинхронного процесса имеет смысл перейти к другому определению, которое в основном и будет использоваться далее, если специально не дается ссылка на определение 2.1.

Определение 2.2. Назовем *асинхронным процессом* (АП) четверку $\langle S, F, I, R \rangle$, в которой: S — непустое множество *ситуаций*, F — *отношение непосредственного следования ситуаций*, определенное на множестве $S \times S$ ($F \subset S \times S$), I — множество *инициаторов* ($I \subset S$), т. е. таких ситуаций из S, для которых, если iFs_k, $i \in I$, $s_k \notin I$, то из $s_k F s_l$ следует, что $s_l \notin I$, R — множество результатов ($R \subset S$) т. е. таких ситуаций из S, для которых, если rFs, $r \in R$, то $s \in R$.

(В отличие от определения 2.1, здесь, во избежание путаницы, используются прямые буквы для обозначения отношения F, инициаторов I и результантов R.)

Комментарии к определению 2.1 с некоторыми оговорками могут быть отнесены и к определению 2.2.

2.1.2. Некоторые подклассы. Пусть имеется последовательность (возможно, бесконечная) ситуаций $s(1)\ldots s(i)s(i+1)\ldots$, где $s(i)$ — элемент, стоящий в последовательности на i-м месте. Для АП можно составить множество последовательностей ситуаций, в которых для любого места i, кроме последнего, справедливо $s(i)Fs(i+1)$ и таких, что ни одна из последовательностей не является частью другой. Все такие последовательности назовем допустимыми. Каждая допустимая последовательность ситуаций описывает возможный ход процесса смены ситуаций (траекторию АП) и соответствует реализации АП.

Из сказанного выше ясно, что всегда можно построить отношение M, соответствующее отношению F. Действительно, если для пары ситуаций s_i, s_j существует траектория из s_i в s_j, то s_iMs_j.

Пусть задан АП, у которого:

1) для любого $s\in S\backslash R$ найдется $r\in R$ такой, что sMr;

2) для любого $s\in S\backslash I$ найдется $i\in I$ такой, что iMs;

3) не найдется ситуаций s_i и s_j таких, что $(s_i\notin R)\&(s_j\notin R)\&(s_iMs_j)\&(s_jMs_i)$.

Определение 2.3. Асинхронный процесс, удовлетворяющий свойствам 1) — 3), будем называть *эффективным*.

Таким образом, из инициаторов эффективного процесса все траектории ведут в результанты (свойства 1), 3)) и каждая из траекторий, приводящих к результанту, начинается в каком-либо инициаторе (свойства 1), 2)). Эффективность АП оставляет место недетерминированности, т. е. возможно, что из некоторого инициатора процесс попадает в разные результанты, но он не содержит ориентированных циклов вне ситуаций, принадлежащих I и R.

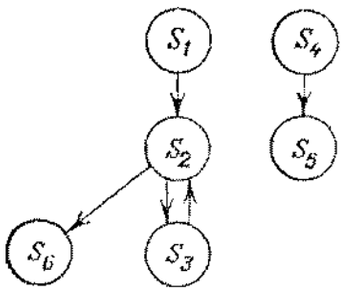


Рис. 2.3. Пример асинхронного процесса

Пример 2.2. Рассмотрим АП $P=\langle S, F, I, R\rangle$, $S=\{s_1, \ldots, s_6\}$, для которого отношение F задается рис. 2.3. Если для этого АП $I=\{s_1, s_4\}$, $R=\{s_5\}$, то АП не является эффективным, так как не выполняется свойство 1). Если $I=\{s_1\}$, $R=\{s_2, s_3, s_5, s_6\}$, то не выполняется свойство 2). Если $I=\{s_1, s_4\}$, $R=\{s_5, s_6\}$, то не выполняется свойство 3). Заметим, что если $I=\{s_1, s_4\}$, $R=\{s_3, s_5, s_6\}$, то не выполняется ограничение определения 2.2. Действительно, выполняется s_3Fs_2, но $s_3\in R$, а $s_2\notin R$. Следующий вариант назначения инициаторов и результантов дает эффективный АП: $I=\{s_1, s_4\}$, $R=\{s_2, s_3, s_5, s_6\}$.

Для некоторых подмножеств множества ситуаций S можно определить отношение E:

1) для s_i, $s_j\in S$ s_iEs_j, если s_iMs_j и s_jMs_i,

2) для любого $s\in S$ sEs.

Условие 1) обеспечивает симметричность и транзитивность отношения E (транзитивность E вытекает из транзитивности M), а условие 2) — рефлексивность E. Следовательно, E является *отношением эквивалентности*.

Отношение E позволяет построить фактормножество множества S, т. е. разбиение $\pi=(S_1, \ldots, S_p)$ множества S на классы эквивалентности. Для классов эквивалентности определим отношение F непосредственного следования классов. Все допустимые последовательности классов эквивалентности, определяемые отношением F, конечны (в отличие от допустимых последовательностей для F, определяемого на множестве $S\times S$, которые могут быть и бесконечными). В допустимых последовательностях классов можно выделить начальные и конечные элементы, которые будем называть начальными и заключительными классами эквивалентности.

На основании введенных понятий можно сформулировать следующие утверждения.

1. Если некоторая ситуация s является инициатором и $s\in S(j)$, где $S(j)$ — класс эквивалентности, стоящий на j-м месте в некоторой допустимой последовательности, то все ситуации классов $S(1)$, ..., $S(j)$ в этой последовательности являются инициаторами.

2. Если некоторая ситуация s является результантом и $s\in S(j)$, то все ситуации из классов $S(j)$, $S(j+1)$, ..., $S(q)$, где $S(q)$ — заключительный класс, являются результантами.

3. Для эффективного АП любой начальный класс состоит только из инициаторов, а любой заключительный класс — только из результантов.

4. Для эффективного АП любой класс эквивалентности ситуаций, не принадлежащих к результантам, состоит из одной ситуации.

Определение 2.4. Если в эффективном асинхронном процессе каждая допустимая последовательность классов ведет из начального класса в один и только один заключительный класс, то такой процесс назовем *управляемым*.

В управляемом АП, таким образом, вводится ограничение на степень недетерминизма: все траектории из любого инициатора ведут в один заключительный класс.

Пример 2.3. Рассмотрим АП с множеством ситуаций $S = \{s_1, \ldots, s_{10}\}$ и отношением F, заданным рис. 2.4, *а*. На рис. 2.4, *б* показано отношение F для классов эквивалентности множества ситуаций этого АП.

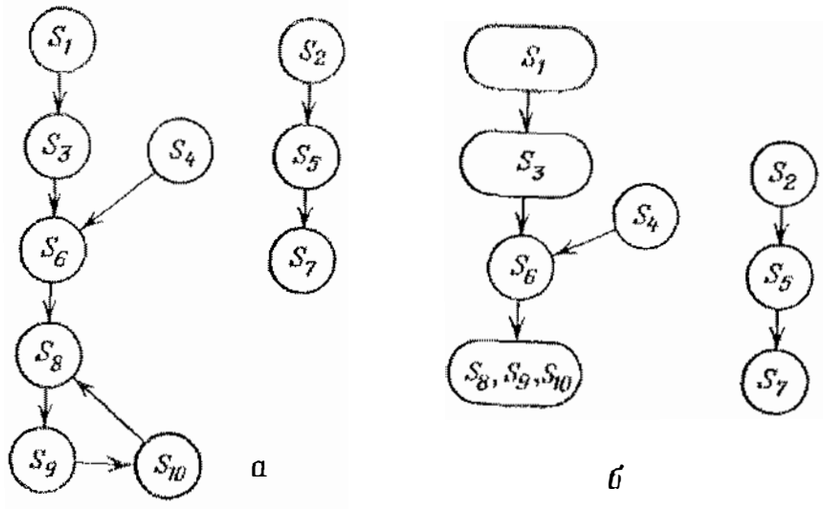


Рис. 2.4. Пример асинхронного процесса (*а*) и разбиения на классы эквивалентных ситуаций (*б*)

Если $I = \{s_1, s_2, s_3, s_4\}$, $R = \{s_7, s_8, s_9, s_{10}\}$, легко убедиться, что этот процесс является управляемым. Если отношение F дополнить парой $s_2 F s_4$, то полученный АП становится неуправляемым (но остается эффективным).

Введем понятие, которое окажется полезным при рассмотрении некоторых АП.

Определение 2.5. Если ситуации s_i и s_k некоторого асинхронного процесса связаны отношением $s_i M s_k (s_i F^n s_k)$, то фрагмент процесса, содержащий все части траекторий, ведущие из s_i в s_k, назовем *переходом* $s_i - s_k$.

В дальнейшем будет использоваться также следующий класс АП. Пусть в эффективном АП:

1) для любых $i \in I$ и $s \in S$ из iFs следует $s \notin I$;
2) для любых $s \in S$ и $r \in R$ из sFr следует $s \notin R$.

Другими словами, из инициатора (результанта) нельзя попасть в другой инициатор (результант), т. е. каждая траектория содержит в точности по одному инициатору и результанту.

Определение 2.6. Асинхронный процесс, удовлетворяющий свойствам 1), 2), будем называть *простым*.

Если рис. 2.1, *б* задает отношение F для АП, у которого $I = \{s_1, s_2\}$, $R = \{s_5\}$, то такой АП является простым. Пример 2.3 описывает не простой АП.

Определение 2.7. *Протоколом* простого *асинхронного процесса* будем называть отношение $Q \subseteq I \times R$.

Протокол простого АП можно рассматривать как простой асинхронный процесс, в котором за каждым инициатором непосредственно следует результант. Поэтому в протоколе простого АП множество ситуаций состоит лишь из инициаторов и результантов: $S = I \cup R$. При переходе от простого АП к его протоколу для каждой пары (i, r) пучки траекторий, ведущих из i в r, возможно через промежуточные ситуации, заменяются одной дугой. Протокол является удобной формой «вход-выходного» описания асинхронного процесса.

2.1.3. Репозиция. В рамках определений 2.1—2.4 не уточнялся механизм перехода от результантов к инициаторам. Между тем, описание такого механизма необходимо для получения эффекта возобновления АП, его повторных активизаций. Такой механизм задается репозицией асинхронного процесса.

Определение 2.8. *Репозицией асинхронного процесса* $P = \langle S, F, I, R\rangle$ назовем эффективный асинхронный процесс $P' = \langle S', F', I', R'\rangle$ такой, что $S' \subseteq I \cup R \cup S^{\text{д}}$, $I' \subseteq R$, $R' \subseteq I$.

Ситуации S' репозиции могут содержать лишь те ситуации из исходного процесса, которые являются лишь инициаторами или результантами, и, кроме того, некоторые дополнительные ситуации из $S^{\text{д}}$, отсутствовавшие в описании исходного АП ($S^{\text{д}} \cap S = \phi$). Отношение F' задает траектории переходов от элементов из $I' \subseteq R$ к элементам из $R' \subseteq I$, возможно, через дополнительные ситуации из $S^{\text{д}}$.

Если $I' = R$, $R' = I$, то репозицию назовем *полной*, если $F' = \phi$, то репозиции не существует, в остальных случаях она называется *частичной*.

Объединение АП и его полной репозиции образует автономный процесс $P^{\text{а}} = \langle S^{\text{а}}, F^{\text{а}}\rangle$, $S^{\text{а}} = S \cup S^{\text{д}}$, $F^{\text{а}} = F \cup F'$.

Для формулировки понятия конвейерного процесса рассмотрим некоторые классы репозиций АП.

Определение 2.9. Полные репозиции простого и не простого АП будем называть соответственно *тривиальной* и *нетривиальной*.

Как следует из определения 2.9 и 2.6, при нетривиальной репозиции в качестве инициаторов АП репозиции могут служить результанты исходного АП, не обязательно принадлежащие заключительному классу эквивалентности.

Обозначим через R^3 объединение заключительных классов эквивалентности ситуаций некоторого АП, а через S^t — множество ситуаций АП, принадлежащих траектории t.

Пусть задан управляемый АП $P=\langle S, F, I, R\rangle$ такой, что:

1) каждый его заключительный класс эквивалентности ситуаций состоит из одного результанта r_j $(r_j \in R^3)$;

2) вместе с нетривиальной репозицией АП образует автономный процесс $P^a=\langle S^a, F^a\rangle$, на множестве S^a ситуаций которого определена функция g: $S^a \to R^3$ такая, что если в исходном управляемом АП iMr, $i \in I$, $r \in R^3$, то в соответствующем автономном процессе из $i \in S^t$ следует, что существует $s \in S^t$, $iM^a s$ и $g(s)=r$.

Определение 2.10. Автономный асинхронный процесс, удовлетворяющий свойствам 1) и 2) и имеющий фрагменты траекторий вида $\ldots i_1 \ldots i_2 \ldots s_1 \ldots s_2 \ldots$, где $g(s_1)=r_1$, $g(s_2)=r_2$, называется *конвейерным* АП.

Введение понятия конвейерного АП позволяет изучать конвейерный принцип обработки информации, в основе которого лежит идея такой повторной инициации АП, при которой АП еще не достиг результанта из заключительного класса эквивалентности ситуаций. Для иллюстрации конвейерного принципа воспользуемся следующей аналогией. Представим себе наклонный желоб, по которому скатываются шарики. Когда пущен первый шарик, на верхний конец желоба можно поместить другой, потом третий и т. д. Шарик может догнать предыдущий либо отстать от него, но не может обогнать предыдущий: догнав его, он может катиться дальше не быстрее предыдущего. Примеры конвейерных процессов будут приведены ниже.

2.1.4. Структурирование. Для описания процессов и задания их взаимодействия часто требуется структурировать ситуации. *Структурирование* (см. с. 13) в зависимости от решаемых задач может выполняться различными способами. Одним из них является разбиение ситуаций на компоненты (события); при этом каждой компоненте ставится в соответствие некоторый предикат p_i, принимающий значение 1, если значение соответствующего логического условия истинно, и 0 в противном случае. Ситуация при таком структурировании представляется двоичным вектором, размерность которого равна числу семантически задаваемых компонент, а число единиц вектора соответствует числу истинных (в этой ситуации) предикатов.

Подчеркнем, что попытки применить АП в приложениях обязательно связаны с семантической интерпретацией и структурированием ситуаций. Рассмотрим следующий пример.

Пример 2.4. Имеется горизонтальный ленточный конвейер (рис. 2.5), на который подаются детали двух сортов — тяжелые и легкие. В зоне А конвейера расположены весы, сигнализатор

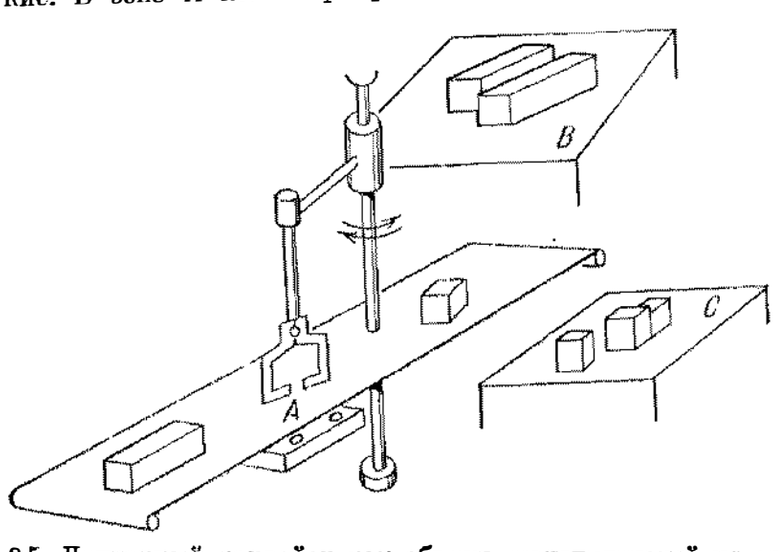


Рис. 2.5. Ленточный конвейер как объект, моделируемый на языке асинхронного процесса

которых срабатывает от тяжелой детали и нечувствителен к прохождению легкой. Легкие детали транспортируются без обработки к концу конвейера, где сваливаются в бункер C. При появлении тяжелой детали в зоне А конвейер останавливается, и рука манипулятора, находящаяся в исходном положении, производит захват детали. Затем привод манипулятора переносит тяжелую деталь в зону B; одновременно включается лента конвейера. В зо-

не B после срабатывания соответствующего сигнализатора отпускается захват, после чего осуществляется запуск привода манипулятора в направлении начального состояния. Далее выявляется новая тяжелая деталь, и процесс повторяется.

В данном случае можно выделить 9 компонент (с точки зрения организации управления процессом, некоторые из них могут оказаться избыточными), которым соответствуют следующие истинные значения предикатов:

$p_1 = 1$ — лента конвейера в движении,
$p_2 = 1$ — тяжелая деталь в зоне A;
$p_3 = 1$ — лента конвейера остановлена,
$p_4 = 1$ — рука манипулятора в исходном положении,
$p_5 = 1$ — рука манипулятора держит тяжелую деталь,
$p_6 = 1$ — работает привод руки в сторону зоны B,
$p_7 = 1$ — рука манипулятора в зоне B,
$p_8 = 1$ — захват отпущен,
$p_9 = 1$ — работает привод руки в исходное положение.

Ситуации в этом примере представляют собой двоичные векторы $(p_1, \ldots, p_9)$. Из 2^9 возможных векторов ситуациями являются лишь следующие 7 (не указанные компоненты равны 0):

$$s_1 : p_1 = p_4 = p_8 = 1, \qquad s_5 : p_3 = p_6 = p_7 = 1,$$
$$s_2 : p_1 = p_2 = p_4 = p_8 = 1, \qquad s_6 : p_1 = p_7 = p_8 = 1,$$
$$s_3 : p_2 = p_3 = p_4 = p_5 = 1, \qquad s_7 : p_1 = p_8 = p_9 = 1.$$
$$s_4 : p_3 = p_5 = p_6 = 1,$$

Описание процесса обнаружения, захвата и препровождения одной тяжелой детали из зоны A в зону B включает ситуации $s_1 - s_6$, отношение F задает последовательность $s_1 \ldots s_6$, инициатором естественно считать ситуацию s_1 ($I = \{s_1\}$), а результатом — s_6 ($R = \{s_6\}$). Последовательность $s_6 s_7 s_1$ можно считать репозицией этого процесса ($I' = \{s_6\}$, $R' = \{s_1\}$, $S^{\text{д}} = \{s_7\}$).

В некоторых случаях структурирование ситуаций предполагает выделение входной и выходной компонент или только одной из них. Эти компоненты соответствуют тем событиям моделируемой системы, которые связаны с изменениями значений сигналов на ее входах и выходах. При выделении как входной, так и выходной компоненты ситуация s_i представима упорядоченной тройкой $s_i = (x_j, y_k, z_l)$, где x_j — значение (символ) входной компоненты, $x_j \in X$, $1 \leqslant j \leqslant p$, y_k — значение (символ) выходной компоненты, $y_k \in Y$, $1 \leqslant k \leqslant q$, z_l — значение компоненты, не являющейся ни входной, ни выходной, $z_l \in Z$, $1 \leqslant l \leqslant n$.

2.1.5. Асинхронный процесс как метамодель. Асинхронный процесс по существу является общей моделью описания динамики поведения параллельно функционирующих асинхронных систем. Эта модель задает допустимые последовательности действий над некоторыми объектами систем, каждой из которых соответствует некоторая траектория АП. В этом смысле можно говорить о том, что АП есть модель управляющей структуры системы. АП может служить инструментом исследования весьма общих свойств и закономерностей параллельных систем, абстрагирующимся от семантики их работы. Поэтому АП можно считать неинтерпретированной моделью. С другой стороны, АП можно понимать как метамодель, порождающую различные широко используемые динамические модели, в том числе рассматриваемые далее. Порождение частных моделей предусматривает использование механизма интерпретации асинхронных процессов, который может быть двухступенчатым. Первый этап (назовем его *модельной интерпретацией*) состоит в том, что основным понятиям АП (ситуации, отношению следования, инициаторам и результантам) ставятся в соответствие понятия частных моделей, причем эта система соответствий базируется на введении дополнительных ограничений на АП.

Примеры модельных интерпретаций АП будут рассмотрены ниже в этой главе.

Модельные интерпретации сами по себе могут оставаться неинтерпретированными моделями. Однако если АП или его модельным интерпретациям при решении конкретных задач поставить в соответствие некоторые метки, обозначающие символы, операторы, атрибуты или предикаты из заданного множества (пользуясь семантическими соображениями), то такие модели становятся интерпретированными. Это вторая, *предметная, интерпретация* АП, которая должна быть согласована с приложениями и зависит от специфики решаемых задач. Конкретный случай предметной интерпретации, когда компонентам структурированных ситуаций ставились в соответствие некоторые предикаты, рассматривался в примере 2.4.

§ 2.2. Сети Пётри

2.2.1. Описание модели. В этом пункте описана предложенная К. А. Петри модель описания потоков событий. Понятие «событие» в данном случае может быть интерпретировано как изменение значения какой-либо компо-

ненты ситуации АП. Связь между событиями выражается с помощью некоторого числа условий, каждое из которых имеет два значения: «условие выполнено» или «условие не выполнено». Согласно Петри, событие может наступить, если выполнены все условия, от которых зависит его наступление, а если событие наступило, то изменяются значения некоторого числа условий. Связь между условиями и событиями в этой модели изображается в виде двудольного графа, называемого сетью Петри, с двумя типами вершин: кружочками и полочками. Кружки (иначе — *позиции*) соответствуют условиям, а полочки (*переходы*) — событиям. Кружки и полочки соединяются дугами, причем дуги от кружков могут быть направлены только к полочкам, а от полочек — только к кружкам. Дугой, идущей от кружка к полочке, обозначается связь между условием наступления некоторого события (входным условием) и самим этим событием. Если же дуга идет от полочки к кружку, то она показывает, какое условие (назовем его выходным) выполняется в результате наступления события. Условие может быть одновременно входным и выходным для некоторого события. Если условие выполнено, то внутри кружка помещаются *точки* (фишки, маркеры). В этом случае говорят, что условие маркировано (размечено).

Формально сеть Петри можно определить следующим образом.

Определение 2.11. *Сетью Петри* называется пятерка вида $N = \langle P, T, M_0, H, F\rangle$, где

$P = \{p_1, \ldots, p_n\}$ — конечное непустое множество *условий*,

$T = \{t_1, \ldots, t_m\}$ — конечное непустое множество *событий*,

$$\left.\begin{array}{l} F : P \times T \to \{0, 1\} \\ H : T \times P \to \{0, 1\} \end{array}\right\} \text{— } \textit{функции инцидентности},$$

$M_0 : P \to \{0, 1, 2, \ldots\}$ — *начальная разметка.*

Разметкой M сети N называется функция M: $P \to \{0, 1, 2, \ldots\}$.

На графе кружок, соответствующий условию p_i, содержит $M(p_i)$ меток (точек).

Обозначим:

$$I(p) = \{t \in T \mid H(t, p) = 1\}, \quad O(p) = \{t \in T \mid F(p, t) = 1\},$$
$$I(t) = \{p \in P \mid F(p, t) = 1\}, \quad O(t) = \{p \in P \mid H(t, p) = 1\},$$

Определение 2.12. *Событие* t_i *возможно* при разметке M, если для любого $p_j \in I(t_i)$, $M(p_j) \geqslant 1$. *Множество событий* $T' \subseteq T$ *совместно возможно* при разметке M, если для любого $p_j \in \bigcup\limits_{t_i \in T'} I(t_i)$, $M(p_j) \geqslant 1$ и

$$M(p_j) - \sum_{t_i \in T'} (F(p_j, t_i) - H(t_i, p_j)) \geqslant 0.$$

Реализация (*срабатывание*) *события* t_i есть смена разметки M, при которой она возможна, разметкой M' по следующему правилу:

$$M'(p_j) = M(p_j) - F(p_j, t_i) + H(t_i, p_j), \quad p_j \in P.$$

Будем говорить, что *разметка* M' *следует за разметкой* M и записывать этот факт $M \xrightarrow{t_i} M'$. *Реализацией множества событий* T' будем называть смену разметки M, при которой T' совместно возможно, разметкой M' по следующему правилу:

$$M'(p_j) = M(p_j) - \sum_{t_i \in T'} F(p_j, t_i) + \sum_{t_i \in T'} H(t_i, p_j), \quad p_j \in P.$$

Будем говорить, что разметка M' следует за разметкой M по множеству событий T', и использовать для этого обозначение $M \xrightarrow{T'} M'$.

Сеть Петри функционирует, переходя от разметки к разметке путем реализации возможных событий. Поясним введенное определение.

Если входные условия некоторого события выполнены (во всех кружках входных событий есть точки), то событие является *возможным*, в противном случае — невозможным. Факт реализации возможного события обозначается на сети тем, что из всех кружков его входных условий изымается по одной точке и во все кружки его выходных условий добавляется по одной точке. Невозможное событие наступить не может. Последовательность маркировок (разметок) в сети отражает динамику процесса, поток событий распространяется вдоль направления дуг.

Пример 2.5. На рис. 2.6 показан пример сети Петри. При заданной исходной разметке в сети возможными являются со

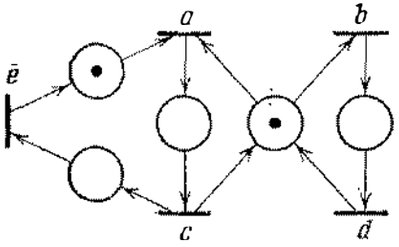


Рис. 2.6. Пример сети Петри

бытия a и b, но после реализации одного из них (например, b) другое (a) становится невозможным.

Правила функционирования сети Петри не указывают, как быстро реализуется возможное событие. Таким образом, этот язык адекватен ранее введенным требованиям: он позволяет учесть реальную асинхронность и непредсказуемость времени наступления событий.

Определение 2.13. Разметку M, при которой ни одно из событий сети N невозможно, будем называть *тупиковой*.

Определение 2.14. Разметка M^* *достижима* в сети N от разметки M, если существует последовательность следующих друг за другом маркировок (разметок) от M до M^*.

Обозначим через $R(N)$ множество всех разметок сети N, достижимых от разметки M_0 (включая M_0).

Событие t сети N будем называть *живым*, если от любой разметки M из множества $R(N)$ достижима хотя бы одна разметка M', при которой событие t возможно. Сеть будем называть *живой*, если каждое ее событие живо.

2.2.2. Некоторые классы. Следующие определения позволяют выделить некоторые классы сетей Петри.

Определение 2.15. Разметку M сети N будем называть *конфликтной*, если найдутся множества совместно возможных при M событий T_1 и T_2 ($T_1 \subseteq T$ и $T_2 \subseteq T$, $T_1 \neq T_2$) и T_2 не является множеством совместно возможных событий при M', $M \xrightarrow{T_1} M'$. Сеть Петри будем называть *устойчивой*, если $R(N)$ не содержит конфликтных разметок.

Таким образом, сеть Петри устойчива относительно некоторой исходной разметки, если в процессе функционирования каждое ее возможное событие может стать невозможным лишь в результате своей реализации.

Определение 2.16. Сеть Петри будем называть *безопасной*, если $R(N)$ содержит только такие разметки M, что $M(p_i) \leqslant 1$ для всех $p_i \in P$.

Очевидно, безопасные сети Петри допускают наиболее простую интерпретацию условий сети («выполнено», «не выполнено»).

2.2.3. Интерпретация. Сеть Петри есть модельная интерпретация АП. *Ситуациями* в сети являются начальная разметка M_0 и все разметки, достижимые от M_0, т. е. $M \in R(N)$. *Отношение* F для любой возможной разметки M задает все разметки, которые могут непосредственно следовать за M. Очевидно, что на множестве $R(N)$ можно, как в п. 2.1.2, определить отношение эквивалентности разметок и задать отношение F непосредственного следования для классов эквивалентности. Если для некоторой сети N существует единственный класс эквивалентности, то соответствующий сети асинхронный процесс является автономным. Если классов разметок несколько (больше 1), то можно выделить подмножество *разметок — инициаторов* (в которое всегда входит начальный класс эквивалентности) и множество *разметок-результантов* (в которое обязательно входит хотя бы один заключительный класс).

Пример 2.6. Рассмотрим сеть Петри рис. 2.7. Если обозначить разметку перечислением условий, содержащих точки, то соответствующий сети процесс P можно определить следующим образом:

$$P\colon\ S = \{AB,\ C\},\quad F = \{(AB,\ C)\},\quad I = \{AB\},\quad R = \{C\}.$$

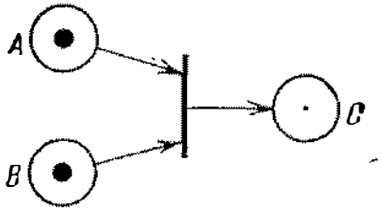


Рис. 2.7. Пример сети Петри

Замечания.

1. Сеть Петри с конечным числом условий $|P|$ и событий $|T|$ может иметь бесконечное множество разметок, если $M(p_j)$ для некоторых p_j может стать неограни-

ченно большим. Такие (неограниченные) сети здесь не рассматриваются.

2. Работа реальных объектов часто описывается не единственной сетью Петри, а некоторым множеством сетей Петри $\{N_i\}$ таким, что все сети из множества $\{N_i\}$ имеют одинаковые множества P и T и одинаковые функции инцидентности F и H, но отличаются начальными разметками. Соответствующая интерпретация дана в примере 2.7, являющемся продолжением примера 2.6.

Пример 2.7. Сеть Петри рис. 2.8, *а* представляет линейное упорядочение трех событий. Она безопасна относительно тех маркировок, при которых общее число точек в условиях не превышает 1. Этот факт свидетельствует о том, что такая сеть моделирует

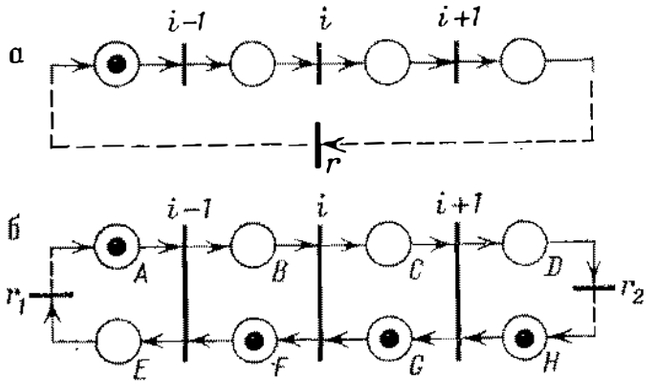


Рис. 2.8. Сеть Петри, моделирующая процесс последовательного решения задачи (*а*) и ее конвейерный аналог (*б*)

процесс решения одной задачи. Решение следующей задачи может быть инициировано процессом тривиальной репозиции (через событие r). Объединение процесса и его репозиции соответствует живой сети Петри.

В соответствующей *конвейеризированной сети* (рис. 2.8, *б*), т. е. сети, реализующей конвейерный (в смысле определения 2.10) процесс, i-е событие возможно, если наступило $(i-1)$-е (как и в исходной сети) и $(i+1)$-е (свидетельство того, что в выходных условиях i-го события нет точек, и наступление i-го события не нарушит безопасности сети). Обратим внимание на то, что во всех разметках из множества M каждая пара расположенных друг под другом условий не нарушает безопасность сети рис. 2.8, *б*. Если все точки находятся в нижних условиях, маркировка является тупиковой. Любая из допустимых маркировок с наличием точки в верхнем левом условии может считаться начальной. Процесс нетривиальной репозиции конвейерной сети осуществляется с помощью событий r_1 и r_2. В полученной автономной сети все допустимые маркировки могут быть получены друг из друга.

В сети на рис. 2.8, *б* может одновременно наступать не более двух событий (не считая r_1 и r_2). Если для событий $(i-1)$, i, $(i+1)$ считать присутствие точки во входном условии слева от события наличием задания на обработку, а наступление самого события — обработкой задания в соответствующем блоке, то в сети может одновременно находиться не более трех заданий и обрабатываться — не более двух.

В общем случае в линейной конвейерной сети, содержащей n событий, может одновременно наступать не более $]n/2[$ событий, где $]a[$ — ближайшее к a целое число, но меньшее a, и равное a, если a — целое.

§ 2.3. Сигнальные графы

Продолжая классификацию сетей Петри (п. 2.2.2), введем следующее определение.

Определение 2.17. Сеть Петри, в которой любое условие может иметь только по одной входной и выходной дуге, называется *маркированным графом.*

В силу традиции маркированный граф изображается не в виде сети Петри, а как обычный ориентированный граф, дуги которого маркируются точками. Соответствие фрагментов сети Петри фрагментам маркированного графа представлено на рис. 2.9. Наличие точек на всех дугах, входящих в некоторую вершину маркированного графа, означает ее возбуждение. Через непредсказуемый, но конечный промежуток времени возбужденная вершина срабатывает. При этом появляются точки на всех ее выходных дугах и снимается по одной точке со всех входных.

Все вершины, в которые входит более одной дуги (фрагменты *4* и *6* на рис. 2.9), называются *синхронизаторами*, а вершины, из которых выходит более одной дуги (фрагменты *5* и *6*),— *бифуркаторами.*

Динамика функционирования маркированного графа описывается сменой его маркировок, порождаемых описанным выше правилом и заданной исходной маркировкой.

Маркированный граф называется *живым*, если каждая его вершина либо возбуждена, либо может быть возбуждена посредством некоторой последовательности срабатываний возбужденных вершин. По аналогии с определением 2.11 можно ввести понятие безопасного маркированного графа.

Определение 2.18. Маркированный граф называется *безопасным* относительно заданной маркировки, если: 1) его маркировка живая, 2) ни одна дуга не содержит

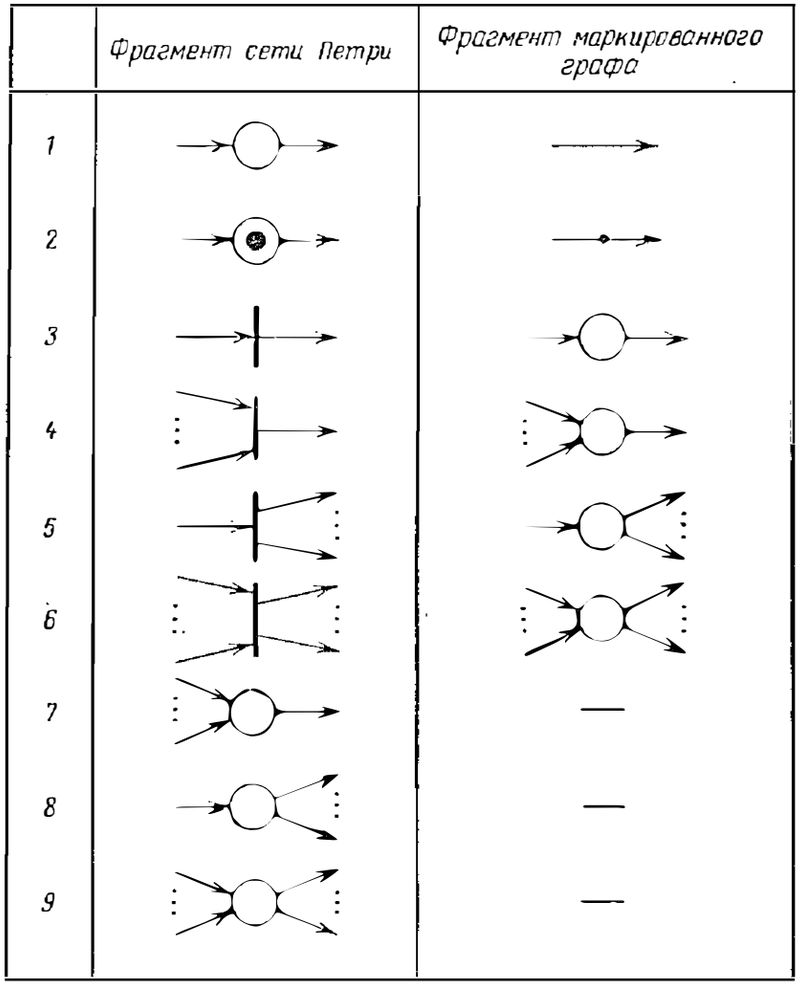


Рис. 2.9. Соответствие между фрагментами сетей Петри и фрагментами маркированных графов. Наличие фрагментов 7—9 в сети Петри свидетельствует о том, что она не может быть представлена в виде маркированного графа

более одной точки, 3) нет последовательности возбуждений, приводящей к появлению двух или более точек хотя бы на одной дуге.

Ясно, что маркированный граф как подкласс сетей Петри является одной из модельных интерпретаций АП.

Ситуация в графе — это текущая маркировка, отношение R задается правилом ее смены, инициаторы и результанты назначаются в соответствии с определением 2.1.

Маркированные графы являются подклассом сетей Петри; доказано, что функциональные возможности маркированных графов ниже, чем у сетей Петри. Маркированные графы двойственны так называемым *автоматным сетям Петри*, в которых каждое событие (переход) имеет только по одному входному и выходному условию. Автоматные сети Петри могут моделировать конфликты (в смысле определения 2.15) посредством условий с несколькими выходами, но не могут моделировать параллельные процессы (из-за отсутствия возможности синхронизировать входные условия событий). Напротив, маркированные графы могут моделировать параллельность (есть синхронизация), но не могут моделировать конфликты (нет вершин с альтернативными выходами).

Предметная интерпретация маркированных графов может быть введена, например, следующим образом. Вершинам графа типа, изображенного на рис. 2.10, *а*,

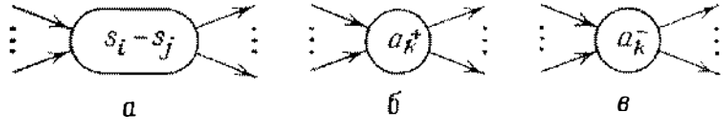


Рис. 2.10. Три типа вершин в сигнальном графе

поставим в соответствие переходы (см. определение 2.5). При возбуждении вершин этого типа (наличии точек на всех входных дугах) инициируется переход $s_i - s_j$. При достижении s_j точки со всех входных дуг снимаются и устанавливаются точки на всех входных дугах. Любая промежуточная ситуация перехода не вызывает изменения маркировки. Ситуации переходов можно структурировать, поставив им в соответствие некоторые двоичные наборы. При возбуждении вершины типа, изображенного на рис. 2.10, *б* (2.10, *в*), осуществляется изменение значения некоторой компоненты ситуации из 0 в 1 (из 1 в 0). При завершении перехода маркировка изменяется описанным выше способом.

Определение 2.19. *Сигнальным графом* будем называть предметную интерпретацию маркированного графа

такую, что его вершины помечены обозначениями переходов видов $s_i - s_j, a_k^+$, a_k^-, а смена маркировок подчиняется описанным выше правилам.

Таким образом, сигнальный граф позволяет сократить описание некоторого класса АП: полное описание перехода можно заменить единственным переходом от инициатора к результанту. Классификация сигнальных графов соответствует принятой для маркированных графов. Примеры использования сигнальных графов для задания поведения схем и устройств можно найти ниже.

§ 2.4. Модель Маллера

Рассмотрим АП, обладающий следующими особенностями.

1. Ситуации из S структурированы: представлены в виде наборов $(a_1, \ldots, a_n)$ значений двоичных переменных z_i, $1 \leqslant i \leqslant n$.

2. Отношение F изображается орграфом, вершины которого соответствуют ситуациям, и если $s_i F s_j$, то вершины s_i и s_j соединены дугой, направленной из s_i в s_j, причем если в s_i компонента $z_k = a_i$, а в s_j $z_k = a_j$ и $a_i \neq a_j$, $a_i, a_i \in \{0, 1\}$, то в наборе s_i значение a_i компоненты z_k помечается звездочкой. Компоненты, значения которых помечены звездочкой, называются возбужденными, остальные — устойчивыми.

3. Инициаторы и результанты назначаются в соответствии с определением 2.2.

Определение 2.20. Модельную и предметную интерпретации асинхронного процесса, удовлетворяющие условиям 1—3, будем называть *диаграммой переходов*.

Определение 2.21. Ситуацию s_i диаграммы переходов будем называть *конфликтной*, если:

1) в ситуации s_i существует компонента z_k, значение которой помечено звездочкой;

2) существует ситуация s_j такая, что $s_i F s_j$, значения компоненты z_k в s_i и s_j совпадают и в ситуации s_j значение компоненты z_k звездочкой не помечено.

Определение 2.22. Диаграмму переходов будем называть *полумодулярной*, если она не содержит конфликтных ситуаций.

Заметим, что определение 2.21 аналогично условию определения 2.15 устойчивости сети Петри.

Пример 2.8. Диаграмма переходов, соответствующая некоторому автономному АП, представленная на рис. 2.11, *а*, не содержит конфликтных переходов и поэтому является полумодулярной.

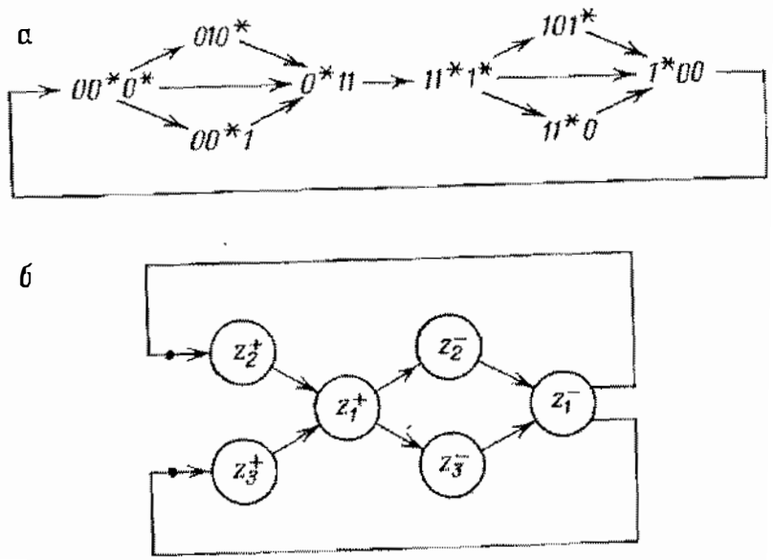


Рис. 2.11. Пример диаграммы переходов (*а*) и ее изображения в виде сигнального графа (*б*)

Заметим, кстати, что в данном случае диаграмма переходов может быть изображена в виде сигнального графа рис. 2.11, *б*.

Определение 2.23. Диаграмму переходов будем называть *управляемой*, если она соответствует управляемому асинхронному процессу.

Подклассами управляемых диаграмм переходов являются полумодулярные, *дистрибутивные* и *последовательные диаграммы*, обозначаемые ниже через U, D и K соответственно. Класс управляемых диаграмм переходов обозначим через W.

Определение полумодулярных диаграмм переходов было только что дано, однако можно предложить другое определение, поясняющее причину использования понятия «полумодулярный». Для этого нужно ввести термин *кумулятивное состояние*, под которым понимается вектор, i-я компонента которого представляет число изменений значения компоненты z_i на траектории, ведущей от некоторого исходного состояния. Кумулятивные состояния могут быть частично упорядочены, что позволяет задать названные выше подклассы диаграмм следующим образом.

Определение 2.24. *Полумодулярными U (дистрибутивными D, последовательными K) диаграммами переходов* назовем такие диаграммы, множество кумулятивных состояний которых образует полумодулярную (дистрибутивную, полностью упорядоченную) структуру.

Имеет место $K \subset D \subset U \subset W$.

Определение 2.25. *Моделью Маллера* асинхронного процесса будем называть систему булевых уравнений $z_i = f_i(z_1, \ldots, z_i, \ldots, z_n)$, $i = 1, 2, \ldots, n$.

В некоторых случаях модель Маллера дополняют указанием исходных значений переменных $z_1, \ldots, z_n$. Если модель соответствует неавтономному АП, то эти исходные значения соответствуют инициаторам АП.

Установим соответствие между диаграммой переходов и системой уравнений. Для нахождения последних можно воспользоваться таблицей истинности, составленной в соответствии со следующими правилами:

1) в левой части таблицы выписываются все возможные наборы переменных диаграммы (например, в порядке возрастания их десятичных эквивалентов); в тех наборах, которые имеются в диаграмме переходов, возбужденные переменные помечаются звездочками;

2) в правой части таблицы против каждого имеющегося в диаграмме набора записывается набор, в котором все переменные, не помеченные звездочками, сохраняют свои значения, а помеченные принимают противоположные;

3) не встречающиеся в диаграмме переходов наборы в правой части таблицы могут быть доопределены произвольно.

Систему уравнений можно получить из таблицы путем использования любой из известных процедур минимизации булевых функций.

Пример 2.8 (продолжение). По приведенной на рис. 2.11, *а*

Таблица 2.1

z_1	z_2	z_3	z_1	z_2	z_3	z_1	z_2	z_3	z_1	z_2	z_3
0	0*	0*	0	1	1	0	0*	1	0	1	1
1*	0	0	0	0	0	1	0	1*	1	0	0
0	1	0*	0	1	1	0*	1	1	1	1	1
1	1*	0	1	0	0	1	1*	1*	1	0	0

диаграмме переходов можно получить табл. 2.1, которой соответствует следующая система уравнений:

$$z_1 = z_2 z_3 \vee (z_2 \vee z_3) z_1, \quad z_2 = \bar{z}_1, \; z_3 = \bar{z}_1.$$

Модель Маллера позволяет описывать АП весьма подробно, с точностью до собственных функций элементов соответствующей схемы.

Определение 2.26. В модели Маллера вектор $\alpha = \alpha_1, \ldots, \alpha_n$, $\alpha_i \in \{0, 1\}$ значений всех переменных $z_1, \ldots, z_n$ называется *состоянием схемы*. Будем говорить, что *переменная z_i возбуждена* в состоянии α, если $\alpha_i \neq f_i(\alpha)$, и *устойчива* в противном случае.

Будем считать, что каждая переменная модели Маллера соответствует выходу элемента моделируемой схемы и элемент работает следующим образом. Будучи возбужден, элемент через некоторое время *в результате срабатывания* переходит в устойчивое состояние, изменив значение выхода. Если же в возбужденном состоянии набор значений на его входах изменится так, что условия возбуждения будут сняты, то элемент оказывается в устойчивом состоянии, не изменяя значения выхода (не срабатывая).

Из состояния α модель может перейти во всякое состояние β, которое отличается от α только значениями каких-либо переменных, возбужденных в α. При этом говорят, что α *непосредственно предшествует* состоянию β, а β *непосредственно следует* за α (непосредственно достижимо из α). Отношение непосредственного следования в модели Маллера далее будет обозначаться через $\alpha \to \beta$. Если $\alpha \to \beta$ и α отличается от β только значением переменной z_i (α и β соседние по z_i), то это обозначается $\alpha \xrightarrow[z_i]{1} \beta$ или $\alpha \xrightarrow{1} \beta$.

§ 2.5. Параллельные асинхронные блок-схемы

По сравнению с широко используемыми граф-схемами алгоритмов граф-схемы, задающие асинхронные процессы, должны обладать дополнительными средствами, отображающими параллельность и асинхронность протекания отдельных частей процесса. Введем понятие *параллельной асинхронной блок-схемы* (ПАБС).

ПАБС — это такие граф-схемы алгоритмов, которые могут быть представлены в виде орграфа с пятью типами вершин: оператор, условный переход, сборка, бифур-

катор и синхронизатор. Условное изображение вершин вместе с инцидентными им дугами приведено на рис. 2.12.

Вершине типа *оператор* (рис. 2.12, *а*) соответствует выполнение оператора A_i. Вершины этого типа имеют по одной входной и одной выходной дуге.

Вершина типа *условный переход* (рис. 2.12, *б*) имеет одну входную и две выходные дуги. Направление движения из этой вершины зависит от результата проверки

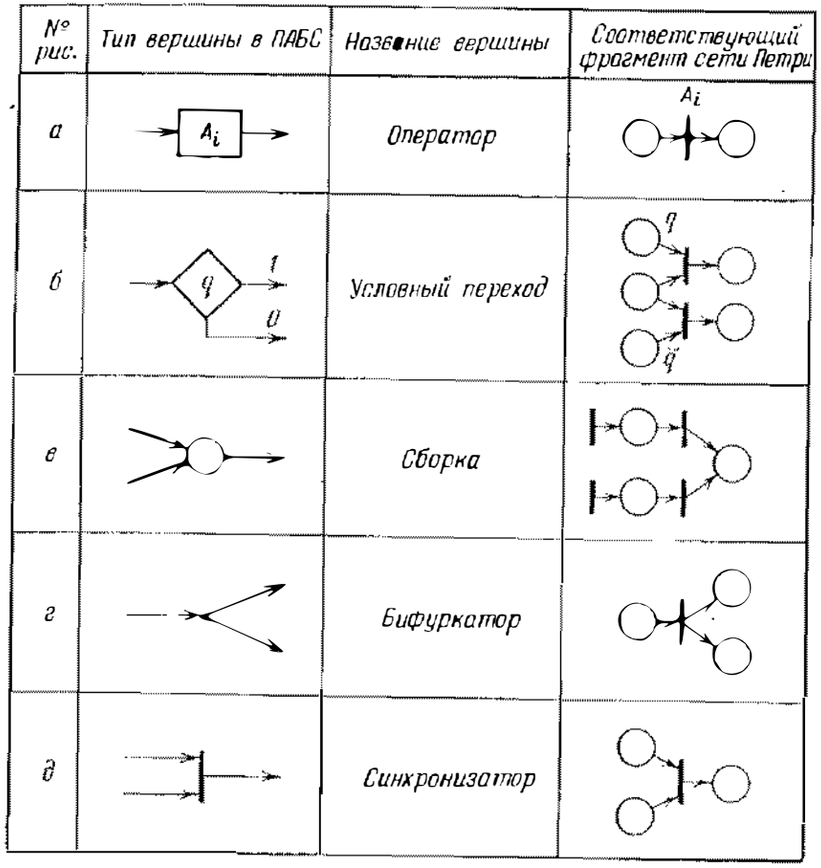


Рис. 2.12. Типы вершин параллельных асинхронных блок-схем и соответствующие им фрагменты сетей Петри

некоторого предиката q, соответствующего вершине. Одной выходной дуге соответствует $q = 1$, другой — $\bar{q} = 1$.

Через вершины типа *сборка* (рис. 2.12, *в*) осуществляется переход к одной и той же вершине ПАБС из различных ее вершин. Иными словами, они служат для графического изображения возможности перехода к некоторому фрагменту алгоритма из различных мест. Вершины этого типа имеют две или более входных дуг и одну выходную. ПАБС должна быть такой, чтобы исключалась возможность одновременного перехода в вершину типа сборка по двум или более дугам. ПАБС, удовлетворяющая этому условию, по аналогии с сетью Петри называется *безопасной*.

Вершине типа *бифуркатор* (рис. 2.12, *г*) соответствует выход на участок алгоритма, представимый в виде ряда параллельных траекторий. Вершины этого типа имеют одну входную и две или более выходных дуг. Бифуркатору соответствуют операторные скобки типа **parbegin** в языке алгол 68 или синхропримитив **fork**, широко используемый в параллельном программировании.

Вершине типа *синхронизатор* (рис. 2.12, *д*) соответствует слияние двух или более параллельных участков (траекторий) и переход к последовательному участку. Синхронизация возможна только после завершения вычислений на всех этих траекториях. Эти вершины имеют две или более входных и одну выходную дугу. Синхронизатору соответствует синхропримитив типа **join**.

Обычно допускается совмещение некоторых типов вершин, например бифуркатора и синхронизатора.

Наличие вершин типов оператора и условного перехода предполагает наличие предметной интерпретации описываемого АП. Для модельной интерпретации необходимо ввести механизм маркировки входных и выходных дуг каждой вершины точками, как в маркированном графе. Тогда текущая ситуация АП, изображаемого ПАБС, будет соответствовать ее маркировке. Преобразование маркировки ПАБС осуществляется в соответствии со следующими правилами:

1) если точкой помечена входная дуга оператора и условного перехода, то данный оператор или условный переход может выполняться. В результате выполнения (реализации) оператора (или условного перехода) пометка с его входной дуги снимается, а его выходная дуга (одна из двух дуг) помечается точкой;

2) точки переходят со входных дуг сборки, бифуркатора и синхронизатора на выходные дуги, причем для появления точки на выходной дуге сборки достаточно точки на одной из ее входных дуг, синхронизатора — наличие точек на всех его входных дугах, а точка на

входной дуге бифуркатора вызывает появление точек на всех его выходных дугах;

3) появление точек на выходных дугах вершин и изъятие точек с их входных дуг являются неделимой операцией.

Таким образом, течение АП изображается сменой маркировок на дугах ПАБС. Очевидно, что ПАБС может быть описана сетью Петри, составленной из фрагментов, соответствующих вершинам ПАБС (рис. 2.12), которая имеет ту же предметную интерпретацию, что и ПАБС.

Обычно в ПАБС выделяют одну вершину, не имеющую входных дуг,— начальный оператор, и вершину, не имеющую выходных дуг,— оператор конца. Тогда инициатором процесса, изображенного ПАБС, является маркировка, при которой единственная точка имеется на выходной дуге начального оператора, а результантом — маркировка, при которой все точки поглощены конечным оператором. Соединив начальный и конечный операторы дугой, направленной от конечного оператора к начальному, получим автономный АП. Примеры ПАБС читатель найдет ниже.

§ 2.6. Асинхронные автоматы

Определение 2.27. *Конечный* (детерминированный) *автомат* — это пятерка $\langle Y, X, Z, \lambda, \delta\rangle$, где $Y = \{y_1, \ldots, y_m\}$ — конечное множество *внутренних состояний*, $X = \{x_1, \ldots, x_n\}$ — конечное множество *входных символов* (входной алфавит), $Z = \{z_1, \ldots, z_r\}$ — конечное множество *выходных символов* (выходной алфавит), λ — *функция переходов*, $\lambda\colon X \times Y \to Y$, δ — *функция выходов*, $\delta\colon X \times Y \to Z$.

Одним из признаков, по которому классифицируются конечные автоматы, является способ введения времени (тактов) в приведенную выше модель. У асинхронного автомата такт формируется внешней средой, его длительность определяется временем, в течение которого состояние входа остается неизменным. Более точным является

Определение 2.28. *Асинхронным* называется такой конечный (детерминированный) автомат, который при каждом изменении входного символа достигает устойчивого состояния, т. е. для каждой пары $x \in X$, $y \in Y$ имеет место $\lambda(x, \lambda(x, y)) = \lambda(x, y)$.

Это определение кажется не совсем верным в силу того, что устойчивое состояние асинхронного автомата может достигаться по цепочке неустойчивых, что не соответствует приведенной выше формуле. Однако в задании асинхронного автомата (в таблице переходов или алфавите внутренних состояний) неустойчивые состояния не фигурируют. Они могут появиться лишь на этапах синтеза, следующих за составлением задания. Расширение исходного алфавита Y за счет введения неустойчивых состояний видоизменяет исходное состояние автомата и делает неприменимым определение 2.28. В этом смысле, очевидно, следует различать понятия автомата и его реализации.

Поскольку здесь не преследуется цель сколь-нибудь подробно затронуть проблемы, связанные с этапами задания, абстрактного и структурного синтеза асинхронных автоматов (эти проблемы можно считать достаточно хорошо изученными, хотя далеко не исчерпанными), отметим лишь следующее.

При переходе от абстрактной модели асинхронного автомата к его структурным моделям, как известно из теории автоматов, требуется определенная конкретизация, связанная с заданием механизма взаимодействия с внешней средой, требованиями к виду допустимых последовательностей входных символов, способами кодирования символов из входного и выходного алфавитов и внутренних состояний автомата, исключающими возможность возникновения функциональных состязаний и гонок, а также с принятыми гипотезами о характере задержек элементов и соединительных линий.

Интерпретация асинхронного автомата в терминах АП состоит в следующем. Ситуация — полное состояние автомата $s_i = (x_j^a, y_k^a, z_l^a)$, где x_j^a, y_k^a и z_l^a — входной символ, выходной символ и внутреннее состояние соответственно. Заметим, что вход, выход и внутреннее состояние не обязательно полностью совпадают с соответствующими компонентами ситуаций автомата как процесса: в АП реализуются отображения $X \times Y \times Z \to Z$ и $X \times Y \times Z \to Y$, а не λ и δ из определения 2.27. Пусть $x_j^a = (x_j, x_j')$, $y_k^a = (y_k, y_k')$. Тогда входной компонентой автомата как процесса может быть, например, x_j, выходной — y_k, а компонента z_l запишется в виде $z_l = (z_l^a, x_j, y_k')$. Таким образом, входная и выходная

компоненты могут соответствовать не всем, а лишь части входов и выходов. Интерпретация инициатора и результанта для асинхронного автомата совпадает с интерпретацией их для АП общего вида. Отношение F задается функциями переходов и выходов, а также *правилами, задающими репозицию*. Последние обычно выражаются фразами типа: «Очередной входной символ можно подавать только после того, как автомат оказался в устойчивом состоянии» или «Допускается смена входного набора только на соседний».

Важно заметить, что предложенная конструкция АП позволяет описывать поведение асинхронного автомата с различной степенью подробности. На уровне абстрактного синтеза как нормальная, так и не нормальная таблицы переходов и выходов по существу являются протоколами простых АП. Промежуточные ситуации возникают на этапах кодирования входных и выходных символов и размещения внутренних состояний.

§ 2.7. Замечания по библиографии

Концепция асинхронного процесса предложена в [9]. Наиболее близкие построения (тяготеющие скорее к проблематике теоретического программирования) описаны в [48, 75, 113, 253, 259].

Понятие классов эквивалентности, по существу, заимствовано из [120] (более подробно в гл. 5).

Модель, позднее названная сетями Петри, впервые была предложена К. А. Петри [285]. Она стала объектом глубоких и многосторонних исследований и, по существу, оформилась в самостоятельное направление. Подробный обзор основных результатов теории сетей Петри выполнен в [284], а затем в [144]; достоинствами этих обзоров, содержащих обширную библиографию, является возможность ознакомиться с тематикой на неформальном уровне. Ряд статей, дающих достаточно полное представление о проблематике работ в области сетей Петри, сведен в [275, 292]. Значительные исследования по сетям Петри были проведены в США в рамках программы *MAC* в Массачусетском технологическом институте [83, 224, 225, 238, 239, 243, 260, 263, 264, 269, 278, 279, 280], где они были связаны с развитием теории систем и теории автоматов, а также в Бонне (ФРГ), где они развивались в направлении создания более общей и абстрактной теории сетей Петри; более подробно они проведены в [284]. География этих исследований значительно расширилась, выйдя за пределы упомянутых центров. Обзор работ французских ученых по сетям Петри и смежным вопросам приведен в [299]. Из отечественных работ следует отметить [99, 101, 132, 164]. К настоящему времени по сетям Петри опубликованы три монографии [160, 283, 295].

Маркированные графы были исследованы в [260]. Понятие сигнального графа, введенное здесь, отличается от трактовки, принятой в [247], но концептуально восходит к уже давно предложенной М. А. Гавриловым таблице включений [71].

Модельный подход, описанный в § 2.4, принадлежит Д. Е. Маллеру и его сотрудникам [199, 229, 270—273], он также освещен в [120] и базируется на теории решеток [47, 81].

Параллельные асинхронные блок-схемы в принятой здесь редакции описаны в [53, 56, 62, 307] и по существу представляют собой модификацию известного языка граф-схем алгоритмов [45, 46, 108], ориентированную на адекватное описание асинхронных процессов.

Различным аспектам теории автоматов посвящены книги [7, 48, 50, 54, 71, 72, 74, 78, 80, 84, 86—91, 100, 106, 108, 109, 117, 120, 121, 137, 143, 166, 169, 170, 173, 187, 208], проблематика асинхронных автоматов полнее всего освещена в [5, 6, 10, 82, 107, 149, 153, 191], см. также [105].

Читателям, интересующимся другими возможными модельными и предметными интерпретациями асинхронных процессов, можно посоветовать обратиться к обзорам [76, 102, 103, 125, 198] и публикациям [83, 96, 104, 167, 189, 224, 246, 288].

Вы славно время провели?
Но это ложь:
Его никак не проведешь!

Илья Варшавский

Глава 3. САМОСИНХРОНИЗИРУЮЩИЕСЯ КОДЫ

Здесь будут рассмотрены некоторые вопросы кодирования сигналов при построении схем, а также систем передачи информации, поведение которых инвариантно к временным параметрам приемнопередающих устройств и каналов связи.

В предыдущей главе на основе понятия асинхронного процесса были рассмотрены динамические модели, которые далее используются в книге для задания (описания) поведения асинхронных параллельно функционирующих устройств, характерных для ЭВМ и дискретных управляющих систем.

Как следует из введения, целью книги является изложение одного подхода к синтезу средств аппаратной поддержки асинхронных параллельных вычислительных процессов. Базовой концепцией, принятой авторами, является концепция самосинхронизации, независимости поведения устройств от значений временных задержек, вносимых компонентами системы. Хотя изучаемые вопросы обладают определенной спецификой, материал книги далее излагается в порядке, типичном для книг по теории автоматов и ее приложениям, который можно охарактеризовать последовательностью крупных этапов: модель (задание закона функционирования) — абстрактный синтез (кодирование состояний и сигналов) — структурный синтез (построение реализаций в заданном элементном базисе) и т. д. В рассматриваемой проблематике вопросы кодирования состояний (ситуаций) и сигналов носят весьма важный характер. Им и посвящена третья глава, претендующая на систематизацию способов кодирования параллельно передаваемой информации в асинхронных системах.

§ 3.1. Предварительные определения

Определим некоторые понятия, которые будут нужны для дальнейшего изложения.

Рассмотрим переход $s_i - s_h$ (см. определение 2.5) в АП, ситуации которого закодированы наборами значений двоичных переменных $z_1, \ldots, z_n$ из множества Z.

Пусть s_i соответствует набор $a = a_1 \ldots a_n$, а s_h — набор $b = b_1 \ldots b_n$.

Обозначим через $\omega(a)$ *конституенту набора* a, т. е. конъюнкцию вида

$$\omega(a) = \mathop{\&}_{i=1}^{n} z_i^{a_i},$$

где $z_i^{a_i} = z_i$, если $a_i = 1$, и $z_i^{a_i} = \bar{z}_i$, если $a_i = 0$. Образуем n-ку вида $\alpha_1 \ldots \alpha_n$ такую, что

$$\alpha_j = \begin{cases} 0, & \text{если } a_j = b_j = 0, \\ 1, & \text{если } a_j = b_j = 1, \\ -, & \text{если } a_j \neq b_j, \end{cases} \quad 1 \leqslant j \leqslant n.$$

Пусть эта n-ка содержит k прочерков (при $k = 0$ $a = b$). Доопределим прочерки всеми возможными комбинациями из нулей и единиц (2^k способами). Полученное таким образом множество наборов назовем *подкубом* (a, b) *перехода* $a - b$. Название это обязано геометрической интерпретации перехода. Подкуб перехода включает все ситуации перехода, в частности наборы a и b. *Внутренним подкубом* будем называть подкуб без наборов a и b и обозначать его $[a, b]$.

Подкубу (a, b) перехода $a - b$ можно поставить в соответствие *терм констант перехода* $\omega(a, b) = \mathop{\&}_{j=1}^{n} \beta_j$, где

$$\beta_j = \begin{cases} \bar{z}_j, & \text{если } \alpha_j = 0, \\ z_j, & \text{если } \alpha_j = 1, \\ 1, & \text{если } \alpha_j = -, \end{cases} \quad 1 \leqslant j \leqslant n.$$

Содержательно: в терм (конъюнкцию) констант перехода входят те переменные или инверсии переменных, значения которых в данном переходе неизменны.

Кроме того, можно образовать n-ку вида $\gamma_1 \ldots \gamma_n$ такую, что

$$\gamma_j = \begin{cases} 0, & \text{если } a_j = 1, \quad b_j = 0, \\ 1, & \text{если } a_j = 0, \quad b_j = 1, \\ -, & \text{если } a_j = b_j, \quad 1 \leqslant j \leqslant n \end{cases}$$

и поставить ей в соответствие *терм вариаций перехода* $\varepsilon(a, b) = \mathop{\&}\limits_{j=1}^{n} \rho_j$, где

$$\rho_j = \begin{cases} \bar{z}_j, & \text{если } \gamma_j = 0, \\ z_j, & \text{если } \gamma_j = 1, \\ 1, & \text{если } \gamma_j = -, \quad 1 \leqslant j \leqslant n. \end{cases}$$

Содержательно: в терм вариаций перехода входят те переменные и инверсии переменных, которые изменяются в данном переходе.

Определение 3.1. Переход $a - b$ назовем *правильным*, если в процессе перехода каждая из переменных z_j, $1 \leqslant j \leqslant n$, изменяет свое значение не более одного раза.

Для любого набора t_k, $t_k \neq a$, $t_k \neq b$, встретившегося в правильном переходе, справедливо $t_k \in [a, b]$.

Определение 3.2. Наборы a и b называются *соседними* (по z_j или $\bar{z}_j$), если для перехода $a - b$ выполняется $\varepsilon(a, b) = z_j$ или $\varepsilon(a, b) = \bar{z}_j$. Переход по соседним наборам будем для краткости называть *соседним переходом*.

Определение 3.3. Наборы a и b называются *сравнимыми*, если в терм вариаций перехода $\varepsilon(a, b)$ переменные входят только без отрицания $(a < b)$ или если терм вариаций содержит только переменные с отрицанием $(a > b)$. Переход по сравнимым наборам будем для краткости называть *сравнимым переходом*.

Обозначим через $O^-(b)$ $(O^+(b))$ множество наборов, соседних с набором b, встречающихся при переходе $a - b$ (т. е. принадлежащих $[a, b]$) и таких, что если $c \in O^-(b)$ $(c \in O^+(b))$, то $c < b$ $(c > b)$. Очевидно, что мощность $O^-(b)$ при переходе $a - b$ равна числу переменных, входящих в терм вариаций $\varepsilon(a, b)$ без инверсий, а мощность $O^+(b)$ при переходе $a - b$ — числу переменных, входящих в $\varepsilon(a, b)$ с инверсией.

Определение 3.4. *Спейсером* назовем набор, сравнимый со всеми наборами из некоторого множества.

Если множество задается посредством 2^n наборов, то спейсерами являются два набора — i и e, состоящие только из нулей и единиц соответственно.

Далее потребуется также

Определение 3.5. Функция $f(z_1, \ldots, z_n)$ называется

а) *изотонной (антитонной) по переменной* z_j, если $f(z_j = 1) \geqslant f(z_j = 0)$ $(f(z_j = 1) \leqslant f(z_j = 0))$;

б) *изотонной (антитонной)*, если она изотонна (антитонна) по всем переменным z_j, $1 \leqslant j \leqslant n$;

в) *монотонной*, если она изотонна или антитонна по каждой из переменных;

г) *изотонной (антитонной) по переходу* $a - b$, если она изотонна (антитонна) по всем переменным, входящим в терм $\varepsilon(a, b)$ вариаций перехода.

Понятие «изотонная функция» синонимично понятию «неубывающая функция», а «антитонная функция» — «невозрастающая функция». Термин «монотонная функция» здесь используется, таким образом, для обозначения класса булевых функций, которые либо изотонны, либо антитонны по всем своим переменным. В литературе по дискретной математике термин «монотонная булева функция» обычно используется в ограничительном смысле, как функция, изотонная по всем своим переменным.

Попутно заметим, что если функция $f(z_1, \ldots, z_n)$ изотонна по переменной z_j, то она представима в виде [1])

$$f(z_1, \ldots, z_n) = z_j f(z_j = 1) \vee f(z_j = 0),$$

где $f(z_j = c) = f(z_1, \ldots, z_{j-1}, c, z_{j+1}, \ldots, z_n)$.

Отсюда следует, что если f изотонна, то для любого сравнимого перехода $a = b$ $(a < b)$ выполняется $f(a) \leqslant f(b)$. Аналогичное представление может быть получено для антитонных функций.

Пример 3.1. Наборам $i = 00000$ (спейсеру) и $r = 10011$ соответствует конституенты единицы $\omega(i) = \bar{z}_1\bar{z}_2\bar{z}_3\bar{z}_4\bar{z}_5$ и $\omega(r) = z_1\bar{z}_2\bar{z}_3z_4z_5$, подкубу (i, r) перехода $i - r$ — терм констант перехода $\omega(i, r) = \bar{z}_2\bar{z}_3$ и терм вариаций перехода $\varepsilon(i, r) = z_1z_4z_5$. Наборы i и r являются сравнимыми $(i < r)$, но не соседними. Множество $O^-(r)$ соседних с r наборов при переходе $i - r$ составляют наборы 00011, 10001 и 10010.

Пример 3.2. Функция $f = z_1\bar{z}_2 \vee z_2\bar{z}_3$ изотонна по переменной z_1, антитонна по z_3 и не монотонна по z_2.

---

[1]) Здесь и далее в булевых формулах вместо знака конъюнкции используется слитное написание символов.

Далее перейдем к определению самосинхронизирующихся кодов. Это понятие широко используется ниже для обозначения классов кодовых систем, используемых для кодирования состояний апериодических автоматов и представления сигналов в асинхронных системах передачи информации по параллельным каналам.

Пусть задан алфавит X, каждому символу ξ которого поставлено в соответствие подмножество символов $X(\xi) \subseteq X$, которые могут появляться после символа ξ. В общем случае подмножество $X(\xi)$ может содержать все символы алфавита ($X(\xi) = X$), минимально возможное число символов в $X(\xi)$ — один.

Предметом рассмотрения будет некоторая *кодовая система* Z, мощность которой (число кодовых наборов системы) не меньше мощности алфавита X (числа символов в нем), а каждому символу ставится в соответствие по крайней мере один уникальный (не совпадающий с кодом другого символа) кодовый набор. Способ кодирования предполагает, что при заданном ξ каждому символу α, появление которого возможно вслед за $\xi (\alpha \in X(\xi))$, ставится в соответствие один кодовый набор $a \in Z$. Такое кодирование по определению обладает следующими свойствами:

1) множеству символов $X(\xi)$ соответствует множество кодирующих наборов $Z(\xi) \subseteq Z$ той же мощности, что и мощность $X(\xi)$;

2) одному символу в общем случае соответствует не один, а несколько кодирующих наборов, причем расшифровка набора однозначна только в том случае, когда известен набор, появившийся перед расшифровываемым.

Будем обозначать через $Z(a)$ множество наборов кодовой системы Z, которые могут появиться вслед за набором a. Если набор a кодирует символ α, то $Z(a) = Z(\alpha)$.

Определение 3.6. Переход $a - b$ в кодовой системе Z назовем *допустимым*, если выполняются следующие условия:

1) $a \in Z$, $b \in Z(a)$;
2) $a \neq b$;
3) $a - b$ — правильный переход;
4) для любого $t \in [a, b]$ $t \notin Z(a)$.

Определение 3.7. Кодовая система, в которой каждый возможный переход является допустимым, называется *самосинхронизирующимся кодом* (ССК).

Как будет показано ниже, момент завершения допустимого перехода $a - b$ можно зафиксировать независимо от времени реализации перехода по самому факту установки его результата — набора b. Это свойство дало название рассматриваемым кодам — самосинхронизирующиеся. Существование ССК доказывается ниже конструктивно.

§ 3.2. Коды с прямыми переходами

Рассмотрим случай, когда

а) для каждого символа ξ в явном виде задается множество $X(\xi)$ символов, которые могут появиться после символа ξ;

2) один и тот же символ не может появляться в два последовательных момента времени, т. е. $\xi \notin X(\xi)$;

3) каждому символу ставится в соответствие единственный кодовый набор.

Определение 3.8. Кодовая система, удовлетворяющая названным выше трем условиям, называется *кодом* с *прямыми переходами*.

Одна из возможных процедур построения таких кодовых систем, которая далее будет предложена, базируется на учете следующих обстоятельств.

1. В соответствии с условием (4) определения 3.6 допустимости перехода $a - b$ для любого набора $x \in Z(a) \setminus b$ должна существовать кодирующая переменная, которая в подкубе (a, b), а значит на наборах a и b, принимает одно значение (скажем, 1), а на наборе x — противоположное (0), т. е. переменная, про которую в теории автоматов говорят, что она *порождает* π-*разбиение* вида

$$\pi = \{\overline{a.b}, \overline{x}\}. \tag{3.1}$$

(Напомним, что π-разбиением на подмножестве S_π множества S называют совокупность подмножеств (блоков) из S_π таких, что их попарные пересечения пусты, а объединение всех блоков составляет S_π. Если для некоторого двублочного π-разбиения состояниям из одного блока присваивается значение 0 (1) кодирующей переменной y_k, а состояниям другого блока — значение 1 (0) той же переменной, то говорят, что переменная y_k порождает это π-разбиение.)

2. Если после набора a может следовать единственный набор b, то подкуб (a, b) может быть любым, т. е. реализация перехода $a - b$ не накладывает никаких ограничений на кодирование.

3. Все символы алфавита X должны быть закодированы различными наборами. Это означает, что для любой пары символов α и β, $\alpha, \beta \in X$, должна найтись *кодирующая переменная*, которая на наборах a и b, соответствующих этим символам, принимает различные значения, т. е. переменная, порождающая π-разбиение вида

$$\pi = \{\bar{a}, \bar{b}\}. \qquad (3.2)$$

Пример 3.3. Пусть $X = \{\alpha, \beta, \sigma, \delta, \varepsilon, \varphi\}$, $X(\alpha) = \{\beta, \sigma, \delta\}$, $X(\beta) = \{\delta, \varepsilon\}$, $X(\sigma) = \{\delta, \varphi\}$, $X(\delta) = \{\varphi\}$, $X(\varepsilon) = \{\alpha\}$, $X(\varphi) = \{\alpha, \varepsilon\}$.

Результатом кодирования будут наборы a, b, c, d, e, f, составляющие кодовую систему Z и соответствующие символам $\alpha, \beta, \sigma, \delta, \varepsilon, \varphi$. Тогда $Z(a) = \{b, c, d\}$, $Z(b) = \{d, e\}$, $Z(c) = \{d, f\}$, $Z(d) = \{f\}$, $Z(e) = \{a\}$, $Z(f) = \{a, e\}$.

Составим таблицу π-разбиений, соответствующих условию (3.1). Для рассматриваемого примера это первые

Таблица 3.1

π_i	α	β	σ	δ	ε	φ
$\pi_1 = \{\overline{a.b}, \bar{c}\}$	1	1	0	—	—	—
$\pi_2 = \{\overline{a.b}, \bar{d}\}$	1	1	—	0	—	—
$\pi_3 = \{\overline{a.c}, \bar{b}\}$	1	0	1	—	—	—
$\pi_4 = \{\overline{a.c}, \bar{d}\}$	1	—	1	0	—	—
$\pi_5 = \{\overline{a.d}, \bar{b}\}$	1	0	—	1	—	—
$\pi_6 = \{\overline{a.d}, \bar{c}\}$	1	—	0	1	—	—
$\pi_7 = \{\overline{b.e}, \bar{d}\}$	—	1	—	0	1	—
$\pi_8 = \{\overline{b.d}, \bar{e}\}$	—	1	—	1	0	—
$\pi_9 = \{\overline{c.d}, \bar{f}\}$	—	—	1	1	—	0
$\pi_{10} = \{\overline{c.f}, \bar{d}\}$	—	—	1	0	—	1
$\pi_{11} = \{\overline{a.f}, \bar{e}\}$	1	—	—	—	0	1
$\pi_{12} = \{\overline{e.f}, \bar{a}\}$	0	—	—	—	1	1
$\pi_{13} = \{\bar{b}, \bar{f}\}$	—	1	—	—	—	0
$\pi_{14} = \{\bar{c}, \bar{e}\}$	—	—	1	—	0	—

12 строк табл. 3.1. В этой таблице столбцы обозначены символами алфавита X, а строки — π-разбиениями. В клетку таблицы записывается 1, если символ, обозначающий столбец, входит в первый блок разбиения, 0, если символ входит во второй блок разбиения, и прочерк, если символ не присутствует ни в одном из блоков. В таблицу π-разбиений необходимо еще включить разбиения типа (3.2). Однако из существования разбиения $\{\overline{a.b}, \bar{c}\}$ следует, что та же переменная порождает разбиения $\{\bar{a}, \bar{c}\}$ и $\{\bar{b}, \bar{c}\}$. Легко проверить, что в табл. 3.1 следует добавить только два разбиения типа (3.2) — π_{13} и π_{14}.

Далее поиск решения задачи кодирования наборами минимальной длины кода с прямыми переходами ведется методом, основанным на поиске так называемых максимальных групп пересечений.

Определение 3.9. Два набора $a = a_1 \ldots a_n$ и $b = b_1 \ldots b_n$, элементы a_i и b_i $(1 \leqslant i \leqslant n)$ которых принимают значения из $\{0, 1, -\}$, *образуют пересечение*, если выполняется одно из двух условий:

либо $a_i = b_i$ для всех i, на которых $a_i, b_i \in \{0, 1\}$,

либо $a_i \neq b_i$ для всех таких i.

Будем говорить, что строка из нулей и единиц размерности, совпадающей с мощность алфавита, порождает разбиение π_i, если она образует пересечение со строкой π_i таблицы π-разбиений.

Поскольку строки, соответствующие π-разбиениям, определены не на всех позициях, одна строка из нулей и единиц может порождать несколько разбиений. Доопределим строки таблицы нулями и единицами всеми возможными способами. В полученном множестве строк могут встретиться пары строк, образующих пересечение. Из каждой такой пары в списке оставим одну строку, а вторую вычеркнем. Продолжим процедуру вычеркивания до тех пор, пока в списке не останется строк, образующих пересечения. Возьмем каждую из оставшихся строк и сравним ее со строками таблицы разбиений. Зафиксируем, какие разбиения порождает данная строка.

Далее необходимо решить *задачу покрытия*: выбрать минимальное число строк, которые порождают все разбиения.

Для примера 3.3 описанным способом по табл. 3.1 построена табл. 3.2. В этой таблице около каждой строки из нулей и единиц выписаны номера разбиений, которые порождает данная строка. При поиске минимального числа строк, порождающих все разбиения, можно воспользоваться любым из известных методов решения задачи покрытия [1]). В данном случае минимальное число строк равно 3. Это, например, строки, отмеченные в табл. 3.2 птичками. Выбору этих строк соответствует следующее кодирование:

[1]) Вопросы оценки вычислительной сложности задачи покрытия не являются предметом данной книги.

$a = 111$, b 110, $c = 011$, $d = 101$, $e = 000$, $f = 010$. Каждый код представляет собой соответствующий столбец «усеченной» табл. 3.2, содержащей лишь три отмеченные строки.

Таблица 3.2

	α	β	σ	δ	ε	φ	π_i
	1	1	0	0	0	0	1, 2, 12, 13
V	1	1	0	1	0	0	1, 6, 8, 10, 12, 13
	1	1	0	0	1	0	1, 2, 7, 13, 14
	1	1	0	1	1	0	1, 6, 10, 13, 14
	1	1	0	0	0	1	1, 2, 9, 11
	1	1	0	1	0	1	1, 6, 8, 11, 14
	1	1	0	0	1	1	1, 2, 7, 9, 14
	1	1	0	1	1	1	1, 6, 14
	1	1	1	0	0	0	2, 4, 12, 13, 14
	1	1	1	0	1	0	2, 4, 7, 13
V	1	1	1	0	0	1	2, 4, 10, 11, 14
	1	1	1	0	1	1	2, 4, 7, 10
	1	0	1	0	0	0	3, 4, 12
V	1	0	1	1	0	0	3, 5, 7, 9, 12, 14
	1	0	1	0	1	0	3, 4, 8, 14
	1	0	1	1	1	0	3, 5, 9
	1	0	1	0	0	1	3, 4, 10, 11, 13
	1	0	1	1	0	1	3, 5, 7, 11, 13, 14
	1	0	1	0	1	1	3, 4, 8, 10, 13, 14
	1	0	1	1	1	1	3, 5, 13
	1	0	0	1	0	0	3, 6, 7, 10, 12
	1	0	0	1	1	0	5, 6, 10, 13
	1	0	0	1	0	1	5, 6, 7, 11, 13
	1	0	0	1	1	1	5, 6, 13, 14
	1	0	0	0	1	1	8, 9, 13, 14
	1	1	1	1	0	0	8, 9, 12, 13, 14
	1	0	0	0	1	0	8, 14
	1	1	1	1	0	1	8, 11, 14
	1	0	0	0	0	1	9, 11, 13
	1	1	1	1	1	0	9, 13

В заключение отметим, что для рассматриваемого кода любой возможный переход осуществляется за одну фазу — без фиксации промежуточных (пустых) наборов. Это свойство характерно не для всех ССК.

§ 3.3. Двухфазные коды

Рассматриваемые в этом и последующих разделах кодовые системы применяются, если множество кодируемых символов разбивается на два подмножества: S и L. В подмножество S входит один — *пустой* — символ, а в подмножество L — все остальные (будем называть их *рабочими*). Для любого $\lambda \in L$ $X(\lambda) = S = \{\sigma\}$ и $X(\sigma) = X \backslash S$, т. е. после пустого символа может появляться любой из рабочих, а после любого рабочего — только пустой символ. Такое ограничение, не являющееся принципиальным для кодов с прямыми переходами, принято из соображений удобства интерпретации сообщений: рабочие символы несут информацию о передаваемых данных, в то время как пустой символ играет роль разделителя, паузы между сообщениями. Прием рабочего кодового набора инициирует активную фазу работы соответствующего устройства, прием разделителя (спейсера) — пассивную фазу (подготовку устройства к приему очередного рабочего набора). Поэтому рассматриваемые далее самосинхронизирующиеся коды (ССК) и называют *двухфазными*.

В рассматриваемых ниже трех кодовых системах каждому символу ставится в соответствие один кодовый набор, причем пустой символ кодируется спейсером, а рабочие символы — несравнимыми кодовыми наборами.

§ 3.4. Парафазные коды

В этой кодовой системе используются $2^n + 1$ кодовых набора длин $2n$, 2^n из которых попарно несравнимы. Рабочим символам алфавита X мощности $|X|$ присваиваются номера от 0 до $|X| - 2$. Двоичные представления номеров символов затем перекодируются в наборы парафазного кода так, что i-му разряду двоичного номера соответствуют два разряда парафазного кода: z_{2i-1} и z_{2i}. Если i-й разряд двоичного номера имеет значение a_i, то в парафазном коде $z_{2i-1} = a_i$, $z_{2i} = \bar{a}_i$. Пустому символу соответствует один из спейсеров.

Пример кодирования алфавита $X = \{\alpha, \beta, \gamma, \delta, \sigma\}$ с пустым символом σ приведен в табл. 3.3, символу σ соответствует спейсер i.

Таблица 3.3

Символ	Двоичный номер символа	Парафазный код			
α	00	0	1	0	1
β	01	0	1	1	0
γ	10	1	0	0	1
δ	11	1	0	1	0
σ	—	0	0	0	0

Теорема 3.1. *Парафазный код является самосинхронизирующимся кодом, если каждый из возможных переходов правильный.*

Доказательство. Нужно показать, что каждый из возможных переходов в парафазном коде является допустимым, а именно, что выполняются условия (1)—(4) допустимости перехода. Выполнение условий (1) и (2) включено в определение парафазного кода. Условие (3) выполняется по условию теоремы. Поскольку рабочие наборы парафазного кода попарно несравнимы, то для любой пары наборов a и b найдется переменная, порождающая разбиение $\pi = \{\overline{a.i}, \overline{b}\}$, следовательно $b \notin (a, i)$. Таким образом, условие (4) допустимости переходов выполняется, и теорема доказана.

Для парафазного кода примем аббревиатуру ПК. Кроме того, для удобства дальнейшего изложения условимся следовать такому соглашению. Если переход ПК в спейсер или обратно осуществляется через единичный спейсер e, то j-й разряд ПК будем обозначать парой $(x_j, \breve{x}_j)$, если переход осуществляется через нулевой спейсер i — парой $(x_j, \hat{x}_j)$. Очевидно, что пара $(x_j, \breve{x}_j)$ задается на множестве значений {10, 01, 11}, а пара $(x_j, \hat{x}_j)$ — на множестве {10, 01, 00}. При этом комбинации 10 и 01 соответствуют значениям 1 и 0 закодированного парафазным кодом разряда символа, а комбинация 11 либо 00 — транзитному (промежуточному) значению разряда.

§ 3.5. Код с идентификатором

В этой кодовой системе используются наборы, содержащие n *основных* и k *дополнительных* разрядов, $k =]\log_2(n+1)[$; такого рода скобками будем пользоваться здесь и далее как знаком округления до ближайшего сверху целого.

Основные разряды используются для кодирования *рабочих символов*, а дополнительные — *как идентификатор*. Пустой символ кодируется одним из спейсеров (для определенности в качестве спейсера возьмем набор i из нулей во всех, основных и дополнительных, разрядах).

Разобьем кодовые наборы, представленные основными разрядами, на $(n+1)$ класс так, чтобы в один класс (*w-класс*) попали коды с одинаковым числом w единиц, $0 \leqslant w \leqslant n$. Таким же способом разобьем кодовые наборы (идентификаторы), представленные дополнительными разрядами, на *v-классы*, $0 \leqslant v \leqslant k$.

Для наборов, кодирующих рабочие символы, потребуем одновременного выполнения трех условий.

1. Одному кодовому набору v-класса соответствует единственный w-класс.

2. Если одному w-классу соответствует несколько идентификаторов, то эти идентификаторы должны быть несравнимы.

3. Если два кодовых набора c и $\boldsymbol{d}$ принадлежат разным w-классам, имеют сравнимые идентификаторы и $w(c) > w(\boldsymbol{d})$, то $v(c) < v(\boldsymbol{d})$, где $w(c)$ и $w(\boldsymbol{d})$ — числа единиц в основных разрядах, а $v(c)$ и $v(\boldsymbol{d})$ — числа единиц в идентификаторах наборов c и $\boldsymbol{d}$ соответственно.

Кодовую систему, удовлетворяющую перечисленным требованиям, назовем *кодом с идентификатором* (КИ).

Докажем существование КИ для случая, когда одному w-классу ставится в соответствие в точности один идентификатор.

За выполнением условия 2 в этом случае следить не нужно. Условие 1 всегда выполнимо, так как число различных w-классов равно $n+1$, а число различных идентификаторов $2^k \geqslant (n+1)$ для $k =]\log_2(n+1)[$.

Условие 3 может быть выполнено, например, при следующем способе назначения идентификаторов. Классу с $w = 0$, которому соответствует единственный код, состоящий из n нулей, поставим в соответствие идентификатор с $v = k$. Каждому из следующих $C_k^{k-1} = k$ w-классов с $w = 1, 2, \ldots, k$ поставим в соответствие один из идентификаторов класса $v = k-1$. Каждому из следующих C_k^{k-2} w-классов с $w = k+1, \ldots, k + C_k^{k-2}$ поставим в соответствие один из идентификаторов класса $v = k-2$ и т. д.

Если $\log_2(n+1)$ — целое число, то классу с $w = n$ соответствует идентификатор с $v = 0$, и все кодовые комбинации в дополнительных разрядах оказываются занятыми. В противном случае кодирование заканчивается идентификатором с $v > 0$ и остаются незанятые идентификаторы. Тогда одному w-классу можно назначить несколько идентификаторов, и оказывается, что КИ можно использовать для кодирования более чем 2^n символов.

Теорема 3.2. *Код с идентификатором является самосинхронизирующимся кодом, если каждый из возможных переходов правильный.*

Доказательство. Как и в теореме 3.1, необходимо показать, что для КИ выполняется условие (4) допустимости переходов. Доказательство проводится от противного. Допустим, что в КИ существует рабочий набор c, который принадлежит подкубу (i, d) перехода i-d.

Рассмотрим следующие случаи.

1. $w(c) > w(d)$ или $v(c) > v(d)$. Очевидно, что в этом случае $c \notin (i, d)$, так как в любой части промежуточного набора должно быть не больше единиц, чем в этой же части рабочего набора, к которому осуществляется переход.

2. $w(c) \leqslant w(d)$, $v(c) < v(d)$. Этот случай может иметь место по условию 3, только если идентификаторы наборов c и d несравнимы, а значит, несравнимы наборы c и d. Следовательно, набор c не может быть промежуточным при переходе i-d.

3. $w(c) = w(d)$, $v(c) = v(d)$. В этом случае либо $c = d$, а значит $c \notin [i, d]$, либо $c \neq d$, и значения в основных разрядах c и d принадлежат одному w-классу. Наборы из одного w-класса несравнимы, и поэтому $c \notin (i, d)$.

4. $w(c) < w(d)$, $v(c) = v(d)$. В этом случае значения в дополнительных разрядах c и d различны по условию 2 и принадлежат одному v-классу, а наборы из одного v-класса несравнимы. Поэтому $c \notin (i, d)$.

Поскольку других случаев нет, принятое допущение всегда приводит к противоречию. Теорема доказана.

Пример 3.4. Рассмотрим вариант КИ для случая $n = 8$, $k =]\log_2(8+1)[= 4$. КИ в данном случае содержит 12 разрядов $x_1 - x_{12}$; $x_1 - x_8$ — основные разряды, $x_9 - x_{12}$ — дополнительные. Вариант КИ, удовлетворяющий условиям 1—3, представлен в девяти строках левой части табл. 3.4. В левом столбце этой таблицы

Таблица 3.4

w	x_9	x_{10}	x_{11}	x_{12}	w	x_9	x_{10}	x_{11}	x_{12}
0	1	1	1	1	1	1	1	1	0
1	0	1	1	1	2	0	1	1	0
2	1	0	1	1	3	1	0	1	0
3	0	0	1	1	4	0	0	1	0
4	1	1	0	1	5	1	1	0	0
5	0	1	0	1	6	0	1	0	0
6	1	0	0	1	7	1	0	0	0
7	0	0	0	1					
8	0	0	0	0					

перечислены различные w-классы, а в четырех оставшихся — значения соответствующих идентификаторов. Эти девять строк позволяют закодировать $2^8 = 256$ рабочих символов, которые в основных разрядах представлены в позиционном двоичном коде.

Поскольку $\log_2(n+1)$ в данном случае не является целым числом, не все значения идентификаторов заняты (заняты 9 из 16). Использование остальных семи идентификаторов дает возможность увеличить число возможных рабочих наборов КИ. Однако добавочные наборы в основных разрядах представлены уже не в позиционном двоичном коде [1]).

При построении КИ возникают две задачи. Первая — найти вариант кодирования, гарантирующий получение максимального числа добавочных кодовых наборов. Вторая — найти кодирование, обеспечивающее (в ущерб информационной емкости) простоту преобразования добавочных кодовых наборов КИ в позиционный двоичный код, если такое преобразование необходимо.

Для решения первой задачи существует достаточно простой алгоритм кодирования, однако рассматриваться он не будет, поскольку приводит к слишком сложной функции декодирования. Для 12-разрядного КИ максимальное число добавочных наборов равно 462, а общее

Таблица 3.5

w	x_9	x_{10}	x_{11}	x_{12}	w	x_9	x_{10}	x_{11}	x_{12}
0	1	1	1	1	3	1	1	1	0
1	0	1	1	1	4	1	0	1	0
2	1	0	1	1	4	1	0	0	1
3	1	1	0	1	4	0	1	1	0
4	1	1	0	0	4	0	1	0	1
5	0	0	1	0	4	0	0	1	1
6	0	1	0	0	5	0	0	0	1
7	1	0	0	0					
8	0	0	0	0					

число рабочих наборов — $256 + 462 = 718$. Соответствующий вариант кодирования представлен в табл. 3.5, в которой девять строк левой части соответствуют обязательным наборам, а семь строк правой части — добавочным.

Вариант кодирования с простой арифметической функцией преобразования в позиционный двоичный код иллюстрируется табл. 3.4, в семи нижних строках которой представлены добавочные наборы (их 254). Всего в этой таблице содержится 510 рабочих наборов КИ. Для представления 510 двоичных чисел с десятичными эквивалентами от 0 до 509 требуется 9 разрядов позиционного кода $a_1 - a_9$. Преобразование этого кода в наборы КИ осуществляется следующим образом:

[1]) *Позиционным* называют двоичный код с естественным распределением весов разрядов: $2^0, 2^1, \ldots, 2^{n-1}$, где n — число разрядов.

для чисел от 0 до 255, для которых $a_9 = 0$, $r_i = a_i$, $1 \leqslant i \leqslant 8$, идентификаторы соответствуют 9 верхним строкам табл. 3.4 (обязательные наборы);

для чисел от 256 до 509, для которых $a_9 = 1$, значения $r_1 - r_8$ равны значениям соответствующих разрядов числа

$$N = a_1 2^0 + a_2 2^1 + \ldots + a_8 2^7 + 1,$$

а идентификаторы соответствуют семи нижним строкам табл. 3.4 (добавочные наборы).

Обратное преобразование наборов КИ в позиционный двоичный код очевидно.

§ 3.6. Оптимальный равновесный код

В этом разделе будет рассмотрен еще один самосинхронизирующийся код, обладающий, как будет показано ниже, наименьшей избыточностью среди всех ССК.

Равновесными называются кодовые системы, в которых для кодирования рабочих символов используются наборы, содержащие фиксированное число единиц (*вес*).

Из множества s-разрядных двоичных наборов можно выделить максимум C_s^k наборов, каждый из которых содержит k единиц, $0 \leqslant k \leqslant s$. Пусть требуется закодировать N различных рабочих символов. Тогда длина s кодового слова, необходимая для их кодирования равновесным кодом с фиксированным весом k, должна быть такой, чтобы

$$C_s^k \geqslant N.$$

Напомним, что $C_s^{]s/2[} = C_s^{[s/2]}$. Поэтому будем использовать обозначение

$$C_s^{s/2} = C_s^{]s/2[} = C_s^{[s/2]}.$$

Отметим, что при фиксированном s

$$C_s^{s/2} = \max_k C_s^k, \qquad 0 \leqslant k \leqslant s. \tag{3.3}$$

Выберем для кодирования N символов такую длину m кодового набора, чтобы выполнялись неравенства

$$C_{m-1}^{(m-1)/2} < N \leqslant C_m^{m/2}. \tag{3.4}$$

Равновесный код с весом $k = m/2$, где m удовлетворяет (3.4), назовем *оптимальным равновесным кодом* (ОРК).

В соответствии с (3.4) для $N = 2^8 = 256$ (представление байта информации) требуется ОРК с $m = 11$, в котором кодовые наборы содержат 5 (или 6) единиц, поскольку

$$C_{11}^5 = 462 > 256 > C_{10}^5 = 252.$$

Таким ОРК можно кодировать от 253 до 462 рабочих символов. Своим названием ОРК обязан следующей теореме.

Теорема 3.3. *Оптимальный равновесный код имеет минимальную длину по всем кодовым системам, использующим для кодирования рабочих символов несравнимые между собой наборы.*

Доказательство. 1. Для равновесных кодов справедливость теоремы следует из (3.3).

2. Рассмотрим случай неравновесных кодов, т. е. множество кодовых наборов, содержащих различное число единиц. Пусть в качестве кодовых используются x наборов с весом $[m/2] - p$, $0 < p \leqslant [m/2]$. Обозначим $u = [m/2]$, $v =]m/2[$. Каждый из этих наборов имеет $v + p$ соседних наборов с весом $u - p + 1$ (будем говорить: имеет $v + p$ связей с наборами весом $u - p + 1$); x наборов с весом $u - p$ имеют $x(v + p)$ таких связей. С другой стороны, каждый из наборов с весом $u - p + 1$ имеет $u - p + 1$ соседних наборов с весом $u - p$, а y наборов с весом $u - p + 1$ имеют $y(u - p + 1)$ связей с наборами весом $u - p$; x наборов с весом $u - p$ связаны не менее чем с x^* наборами весом $u - p + 1$, причем x^* определяется из соотношения

$$x(v + p) = x^*(u - p + 1),$$

откуда

$$x^* = x\frac{v + p}{u - p + 1}.$$

Принимая во внимание, что $v \geqslant u$, для допустимых значений p $x^* > x$. Это означает, что вместо x наборов с весом $u - p$ можно использовать в качестве кодовых большее число x^* наборов с весом $u - p + 1$. Поскольку вместо x наборов с весом $u - p$ можно использовать в качестве кодовых $x^* > x$ наборов с весом $u - p + 1$, а вместо них $x^{**} > x^*$ наборов с весом $u - p + 2$ и т. д., то очевидно, что наибольшее число кодовых наборов получится, если их вес равен $[m/2]$.

Пусть теперь в качестве кодовых используются x наборов с весом $]m/2[+ p$, $0 \leqslant p \leqslant]m/2[$. Каждый из этих наборов имеет $v + p$ соседних наборов с весом $v + p + 1$, x наборов с весом $v + p$ имеют $x(v + p)$ связей с наборами весом $v + p - 1$. С другой стороны, y наборов с весом $v + p - 1$ имеют $y(u - p + 1)$ связей с наборами весом $v + p$. Таким образом, x наборов с весом $v + p$ связаны не менее чем с x^* наборами весом $v + p - 1$, причем x^* определяется из соотношения

$$x(v + p) = x^*(u - p + 1),$$

откуда

$$x^* = x\frac{v + p}{u - p + 1}.$$

Поскольку $v \geqslant u$, для допустимых значений p $x^* > x$.

Из последнего неравенства также следует вывод, что число кодовых наборов будет наибольшим, если все они будут с весом $]m/2[$. Поскольку для четных m $]m/2[= [m/2] = m/2$, для этого случая минимальным является равновесное кодирование с $k = m/2$.

Для нечетных m неравновесное кодирование имеет место и в случае, когда совместно используются наборы с весом u и v. Однако из подсчета числа связей между наборами следует, что вместо x наборов с весом u можно использовать не меньшее число наборов с весом v, и наоборот. Поэтому равновесное кодирование, когда используются наборы либо с весом u, либо с весом v, по крайней мере не хуже случая неравновесного кодирования наборами весом u и v. Теорема доказана.

Теорема 3.4. *Равновесный код (в том числе ОРК) является самосинхронизирующимся кодом, если каждый из возможных переходов правильный.*

Справедливость теоремы следует из несравнимости любой пары рабочих наборов равновесного кода.

§ 3.7. Об избыточности кодов

Сравним избыточности парафазного кода (ПК), КИ и ОРК для случая, когда система передачи информации ориентирована на передачу любого позиционного двоичного кода длины n. Пусть для этого требуются самосинхронизирующиеся коды длин t (для ПК), q (для КИ) и r (для ОРК).

Известно, что *избыточность кодовой системы* оценивается по формуле

$$R = 1 - \frac{\log_2 N_0}{\log_2 N},$$

где N_0 — число наборов, используемых в кодовой системе, N — общее число возможных наборов.

Для ПК $t = 2n$,

$$R_{\text{ПК}} = 1 - \frac{\log_2 2^n}{\log_2 2^{2n}} = \frac{2n-n}{2n} = 0{,}5.$$

Значения t и $R_{\text{ПК}}$ для $2 \leqslant n \leqslant 16$ приведены в столбцах 2 и 3 табл. 3.6.

Для КИ $g = n +]\log_2(n+1)[$,

$$R_{\text{КИ}} = 1 - \frac{\log_2 2^n}{\log_2 2^{n+]\log_2(n+1)[}} = \frac{]\log_2(n+1)[}{n+]\log_2(n+1)[} = \frac{q-n}{q}.$$

Значения q и $R_{\text{КИ}}$ для $2 \leqslant n \leqslant 16$ приведены в столбцах 4 и 5 табл. 3.6.

Из (3.4) следует, что длина r при использовании ОРК для фиксированного n должна удовлетворять

Таблица 3.6

n	t	$R_{\text{ПК}}$	q	$R_{\text{КИ}}$	r	$R_{\text{ОРК}}$
1	2	3	4	5	6	7
2	4		4	0,500	4	0,500
3	6		5	0,400	5	0,400
4	8		7	0,429	6	0,333
5	10		8	0,375	7	0,286
6	12		9	0,333	8	0,250
7	14		10	0,300	10	0,300
8	16		12	0,333	11	0,273
9	18	0,5	13	0,308	12	0,250
10	20		14	0,286	13	0,231
11	22		15	0,267	14	0,214
12	24		16	0,250	15	0,200
13	26		17	0,235	16	0,188
14	28		18	0,222	17	0,176
15	30		19	0,211	18	0,167
16	32		21	0,238	19	0,158

неравенству

$$C_{r-1}^{(r-1)/2} < 2^n \leqslant C_r^{r/2}$$

или

$$\log_2 C_{r-1}^{(r-1)/2} < n \leqslant \log_2 C_r^{r/2}. \qquad (3.5)$$

Для $2 \leqslant n \leqslant 16$ значения r, удовлетворяющие неравенству (3.5), указаны в столбце 6 табл. 3.6.

Очевидно, что для ОРК

$$R_{\text{ОРК}} = 1 - \frac{\log_2 2^n}{\log_2 2^r} = \frac{r-n}{r}. \qquad (3.6)$$

Соответствующие значения $R_{\text{ОРК}}$ приведены в столбце 7 табл. 3.6.

Для больших n при приближенных вычислениях можно воспользоваться формулой Стирлинга $r! = \sqrt{2\pi r}\, r^{r+0,5} e^{-r}$, откуда

$$C_{r-1}^{(r-1)/2} = 2^r / \sqrt{2\pi(r-1)},$$
$$C_r^{r/2} = 2^{r+1} / \sqrt{2\pi r}. \qquad (3.7)$$

Из (3.6) с учетом (3.5) получаем

$$\frac{\log_2 \sqrt{2\pi (r-1)}}{r} < R_{\text{ОРК}} \leqslant \frac{\log_2 \sqrt{2\pi r} - 1}{r}.$$

Подстановка (3.7) в (3.5) дает

$$r - \log_2 \sqrt{2\pi (r-1)} < n \leqslant r - \log_2 \sqrt{2\pi r} - 1. \qquad (3.8)$$

Из табл. 3.6 видно, что для $2 \leqslant n \leqslant 16$ $r \leqslant q \leqslant t$ (причем $r = q = t$ только при $n = 2$, а $r = q$ также при $n = 3, 7$) и $R_{\text{ОРК}} \leqslant R_{\text{КИ}} \leqslant R_{\text{ПК}}$.

Из теоремы 3.3 следует, что $r \leqslant q$, $R_{\text{ОРК}} \leqslant R_{\text{КИ}}$ для любых n. Неравенства (3.8) задают зависимость r от n неявно. Из табл. 3.6 видно, что для $2 \leqslant n \leqslant 16$

$$r \leqslant n + [\log_2 n],$$

Таким образом, «дополнительная плата» за использование ССК для представления наборов длины n не превосходит (в случае ОРК) $\log_2 n$.

§ 3.8. Коды в изменениях

Недостатком кодовых систем ПК, КИ и ОРК является то, что появление рабочих наборов перемежается появлением спейсера. В случае использования ССК для передачи информации по каналам связи, где задержки в линиях значительны, желательно исключить передачу наборов, соответствующих пустым символам. Исключение пустого символа оказывается возможным, если использовать *передачу* наборов *в изменениях*. В этом случае в качестве носителя информации используются не сами значения сигналов, а их изменения. Этот прием часто применяют при последовательной передаче информации по каналу связи. Отсутствие (наличие) изменения значения сигнала в канале интерпретируется как передача бита информации со значением 0 (1).

Предлагается применить эту идею для параллельной передачи данных по группе шин, причем при переходе от одного набора к другому допускается независимое изменение значений сигналов на шинах. Рассмотрим передачу в изменениях s-разрядного кодового набора по s шинам. Пусть в некоторый момент времени на шинах $1 - s$ установлен набор $a_1 = a_{11} \ldots a_{s1}$ и требуется передать рабочий набор $r = r_1 \ldots r_s$. Тогда в следующий момент времени на шинах устанавливается набор $a_2 = a_{12} \ldots a_{s2}$ такой, что $a_{i2} = a_{i1} \oplus r_i$, $1 \leqslant i \leqslant s$, где $\oplus$ — знак суммирования по модулю два. Восстановление кодового набора r в приемнике, который содержит специальный блок, фигурирующий далее под названием преобразователя приемника, осуществляется по формуле $r_i = a_{i1} \oplus a_{i2}$, $1 \leqslant i \leqslant s$, из которой следует, что приемник должен помнить предыдущий набор a_1. После восстановления кодового набора r на выходах преобразователя приемника, обозначенных через y_i, $1 \leqslant i \leqslant s$, $y_i = r_2$. Использовав его и подготавливаясь к приему очередного набора, приемник осуществляет в своей памяти замену набора a_1 набором a_2, после которой на выходах y_i устанавливается спейсер (скажем, e). Таким образом, любой переход $a_k - a_l$ на выходных шинах источника обусловливает последовательность двух подпереходов $e - r - e$ (двухфазную дисциплину со спейсером) на выходах y_i преобразователя приемника.

Сделаем два замечания.

1. В качестве рабочих наборов r, преобразуемых в наборы кода в изменениях, можно использовать рабочие наборы ПК, КИ или ОРК.

2. Одному подлежащему передаче набору r (который можно рассматривать как рабочий символ) соответствует множество наборов кода в изменениях, поэтому для восстановления рабочего символа необходимо помнить предыдущий принятый набор.

3. Вообще говоря, коды в изменениях не являются ССК. Однако они могут быть приведены к ССК посредством увеличения длины кодовых наборов (включения в число кодирующих переменных, соответствующих памяти). Именно это и позволяет использовать коды в изменениях для построения апериодических схем.

Центральным в этой главе является понятие самосинхронизирующегося кода (ССК) — определение 3.7. Вообще говоря, при передаче информации в процессе смены одного кодового набора другим (перехода) возможные изменения значений нескольких кодовых переменных происходят с разной скоростью. Из всех кодовых систем только ССК позволяют распознавать факт завершения перехода безотносительно к его длительности. Таким образом, только ССК дают возможность организовывать не зависящий от временных параметров процесс переработки и передачи информации. В частности, в последующих главах 4, 5, 7 ССК различного типа (коды с прямыми

переходами, парафазные коды, коды с идентификаторами, оптимальные равновесные коды) используются для кодирования сигналов в апериодических схемах и интерфейсах.

§ 3.9. Замечания по библиографии

На возможность использования равновесных кодов при построении схем с неограниченными задержками впервые указано, по-видимому, в [195]. К сожалению, авторы этой работы не догадались, что простейшим подклассом этих кодов, удобным для реализации, являются парафазные коды. Парафазные представления впервые исследовались в [193] и использовались в [5], а также в [173], где соответствующие коды назывались автосинхронными. Похожие кодовые системы применялись в работах по технической диагностике (см. гл. 10). Коды с идентификаторами называют также кодами Бергера [200].

Результат теоремы 3.3, а точнее — факт, что максимальное число пар несравнимых наборов содержит оптимальные равновесные коды, получен впервые, очевидно, в [122]; доказательство опирается на работу [73]. На это внимание авторов книги обратил Ю. Л. Сагалович.

В [94, 95] развивается концепция парафазной логики, однако использование обоих транзитных состояний (вместо одного) для диагностических целей не позволяет применить эту концепцию для аппаратной реализации и диагностирования неисправностей в асинхронных схемах.

Приведенное определение избыточности кодовой системы традиционно — см., например, [73].

При кодировании с прямыми переходами (разд. 3.6) используется модификация метода Трейси [304], описанного также в [123]; по поводу построения полностью разделяющих кодовых систем см. также [150].

Изложенный в гл. 3 материал опубликован в [8, 123, 129, 130].

Какое расстояние между Халёбом и Дамаском? — спросили одного. Он ответил:
— Двенадцать дней пути — шесть дней туда и шесть обратно.

Абуль-Фарадж

Глава 4. АПЕРИОДИЧЕСКИЕ СХЕМЫ

В этой главе будут описаны подходы к построению схем, правильность функционирования которых не зависит от величин задержек элементов. Такие схемы (комбинационные и с памятью) принято в силу сложившейся традиции называть *апериодическими* или, иначе, *самосинхронизирующимися*.

В этой главе речь будет идти только о реализации модели конечного автомата; другие модели самосинхронизирующихся схем будут рассмотрены в последующих главах.

Взаимодействие автомата с внешней средой осуществляется по принципу «запрос — ответ», а сама реализация автомата (схема) обладает тем свойством, что правильность ее поведения не зависит от величин задержек элементов. Последнее требование не позволяет использовать результаты классической структурной теории автоматов, ибо если верхняя граница величин задержек неизвестна, то использование структурных моделей синхронного и асинхронного автоматов бессмысленно. Апериодическая схема, заданная моделью автомата, должна правильно функционировать при любых соотношениях величин задержек элементов. В частности, нет необходимости задавать верхнюю границу этих величин; достаточно считать, что она конечна. В этом заключается одно из основных преимуществ апериодических схем перед традиционными. Принцип самосинхронизации позволяет строить схемы, работающие по реальным задержкам элементов, т. е. максимально использовать их возможности по быстродействию, чего никакой другой подход не позволяет.

Иногда вместо термина «апериодическая реализация

нее точным термином *апериодический автомат*, определение которого будет дано в конце главы. В нем отражены два требования к модели конечного автомата: 1) запрос-ответное взаимодействие со средой и 2) независимость поведения реализующей его схемы от задержек ее элементов. Эти требования связаны скорее со структурной, чем с абстрактной моделью автомата и родственны с основным требованием модели Хаффмена асинхронного автомата: инициация нового перехода автомата может иметь место лишь после того, как автомат окажется в устойчивом состоянии. Предполагается, что апериодический автомат может каким-то образом фиксировать («индицировать») реальные моменты окончания переходных процессов и уведомлять об этом среду.

§ 4.1. Двухфазная реализация конечного автомата

Установим следующие ограничения.

1. Работа автомата осуществляется в две фазы, условно называемые *рабочей* и *нерабочей*.

2. Символы входного алфавита X и выходного алфавита Y автомата кодируются рабочими (см. разд. 3.3) кодовыми наборами X^i, $1 \leqslant i \leqslant n$, и Y^j, $1 \leqslant j \leqslant m$, ССК, принадлежащими множествам X' и Y' соответственно. Считается, что ССК, помимо наборов из $X'(Y')$, включает также еще один кодовый набор — спейсер $s_X(s_Y)$, соответствующий пустому символу входного (выходного) алфавита.

3. *Репозиция* соответствующей фазы автомата осуществляется заменой внешней средой входного спейсера s_X кодовым набором из X' (инициация рабочей фазы) или кодового набора из X' входным спейсером s_X (инициация нерабочей фазы). Таким образом, допустимые последовательности входных наборов имеют вид $s_X X^1 s_X X^2 \ldots$

4. Множество внутренних состояний Z состоит из двух подмножеств — Z_1 (*рабочие состояния*) и Z_2 (*нерабочие состояния*), $Z_1 \cup Z_2 = Z$.

5. *Функция* λ *переходов* имеет вид

$$\lambda\colon \{s_X\} \times Z_1 \to Z_2, \quad X' \times Z_2 \to Z_1.$$

6. *Функция* δ *выходов* задается в виде

$$\delta\colon \{s_Y\} \times Z_2 \to \{s_Y\}, \quad X' \times Z_2 \to Y',$$

Из пп. 1—6 следует, что допустимые последовательности выходных наборов имеют вид $s_Y Y^1 s_Y Y^2 \ldots$

7. *Внешняя среда*, совместно с которой функционирует автомат, организована так же, как и сам автомат, т. е. удовлетворяет требованиям, аналогичным пп. 1—6, но ее входным и выходным алфавитом являются Y и X соответственно, причем появление на входе спейсера s_Y влечет за собой выдачу кодового набора из X', а появление рабочего кодового набора из Y' — выдачу спейсера s_X.

8. *Гипотеза о характере задержек*:

а) величины задержек элементов непредсказуемы, на них накладывается единственное ограничение: они конечны;

б) задержки элементов могут быть как инерциальными, так и чистыми;

в) задержки в неразветвляющихся проводах приведены к задержкам элементов, к выходу которых они подключены (т. е. если задержка элементов равна D, а задержка провода равна d, то в результате приведения принимается, что задержка элемента составит $D+d$, в то время как задержка провода будет равна нулю), и могут быть по величине любыми; задержки в проводах после разветвления, напротив, приведены ко входам элементов, с которыми они соединены, поэтому разброс задержек в проводах должен быть меньше минимальной задержки элементов, к которым они приведены;

г) расширители (конъюнкторы) И элементов И-ИЛИ-НЕ безынерционны (все задержки этих элементов приведены к задержкам выходного инвертора).

9. *Дополнительное требование на реализацию*: *применение специальных* (*встроенных*) *элементов задержки недопустимо.*

Определение 4.1. *Реализацию конечного автомата, удовлетворяющую ограничениям типа* 1—9, *будем называть двухфазной.*

Динамику совместной работы двухфазной реализации автомата и внешней среды можно описать следующим образом. Пусть первоначально автомат находился в нерабочем состоянии из Z_2 и выдал на выход спейсер s_Y. При подаче на автомат рабочего входного набора из X' (запрос) он переходит в рабочее состояние из Z_1 и выдает на выход набор из Y' (ответ). Это вызывает переход в среде, заканчивающийся выдачей автомату спейсера s_X (снятие запроса). Замена набора из X' спейсером

s_X инициирует переход автомата в нерабочее состояние из Z_2 и заканчивается выдачей спейсера s_Y (снятие ответа). Замена набора из Y' спейсером s_Y инициирует переход в среде, который заканчивается выдачей автомату набора из X' (новый запрос), и процесс повторяется.

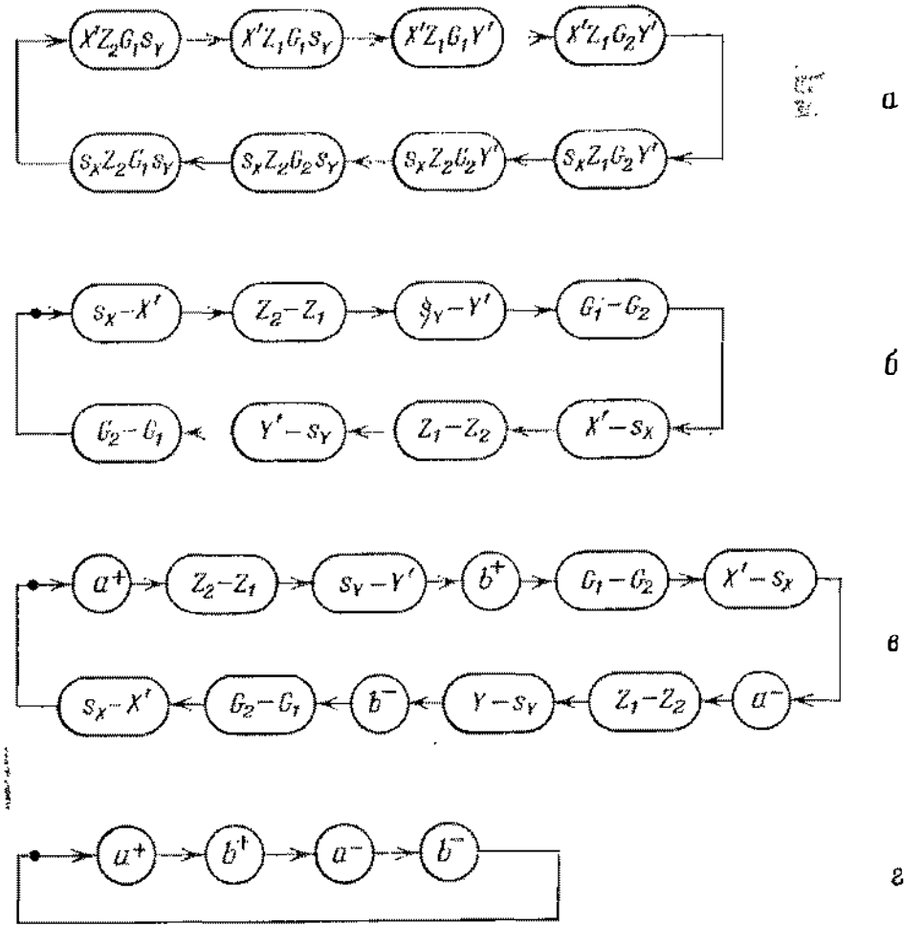


Рис. 4.1. Описание взаимодействия автомата и внешней среды: (*а*) структурированным асинхронным процессом; (*б*) сигнальным графом (случай двухфазной реализации); *в*) сигнальным графом; *г*) сигнальным графом на уровне управляющих сигналов (случай согласованной реализации)

Обозначим множество рабочих (нерабочих) состояний автомата, представляющего внешнюю среду, через $G_1(G_2)$. Положим, что ситуации АП совместного функционирования автомата и внешней среды состоят из упорядоченных четверок компонент: 1) входного набора автомата (выходного набора среды) из $\{s_X\} \cup X'$; 2) состояния автомата из $Z_1 \cup Z_2$; 3) состояния среды из $G_1 \cup G_2$; 4) выходного набора автомата (входного набора среды) из $\{s_Y\} \cup Y'$. Тогда соответствующий автономный (последовательный) АП системы автомат — среда запишется в виде рис. 4.1, *а*, а на языке сигнальных графов — в виде рис. 4.1, *б*, откуда видно, что изменение класса выходных наборов среды влечет за собой изменение класса выходных наборов автомата и обратно.

Требуется внести еще одно уточнение, касающееся сохранения кода предыдущего состояния в нерабочей фазе. Оно сводится к тому, что память автомата должна быть двухступенчатой. В рабочей фазе в одном из двух регистров (ступеней) памяти вырабатывается код внутреннего состояния автомата, для его выработки в другой ступени (буфере) сохраняется код предыдущего состояния автомата. Комбинация состояний регистров и образует код рабочего состояния автомата, принадлежащего множеству Z_1. В нерабочей фазе код состояния первого регистра переписывается в буфер, а в сам регистр заносится либо спейсер, либо он сохраняет то же состояние, что и в рабочей фазе. Описанная комбинация состояний регистров образует код нерабочего состояния, принадлежащего Z_2.

Кодовые слова, принадлежащие состояниям из $Z_1 \cup Z_2$, должны принадлежать самосинхронизирующимся кодам. С примерами читатель встретится в разд. 4.5.

4.1.1. Согласованная реализация. Двухфазную модель можно конкретизировать следующим образом. Естественно иметь специальный двоичный сигнал, называемый обычно *фазовым* (обозначим его буквой a), отмечающий моменты инициации фаз работы автомата, т. е. моменты смены спейсера s_X набором из X' и обратно. Устройства, фиксирующие факт смены классов наборов, назовем *индикаторами*. Индикатор естественно поместить во внешней среде. Для инициации перехода среды индикатор должен быть и в самом автомате. Цель этого индикатора — выработать сигнал b, отмечающий моменты смены спейсера s_Y символом из Y' и обратно.

Определение 4.2. Двухфазную реализацию конечного автомата, в которой выделены два двоичных управляющих сигнала («запроса» и «ответа»), будем называть *согласованной*.

По $a = 1$ (запрос) начинается рабочая фаза. Конец фазы (окончание переходного процесса фазы) фиксируется индикатором автомата. Сигнал $b = 1$ (ответ) инициирует переход среды; по окончании перехода $a = 0$. Этот сигнал в свою очередь инициирует переход автомата в

нерабочую фазу (снятие запроса). После окончания переходного процесса фазы $b=0$ (снятие ответа). Затем начинается новый цикл совместной работы. На языке сигнальных графов совместное функционирование согласованной реализации с внешней средой может быть изображено так, как показано на рис. 4.1, *в*. Заметим, что изменение значений управляющих сигналов происходит всегда после того, как изменились внутренние состояния автомата и среды и вслед за ними — выходные наборы (если это имело место). Учет этого обстоятельства позволяет перейти к более экономному заданию сигнального графа лишь на уровне управляющих сигналов (рис. 4.1, *г*).

Напомним, что в соответствии с принятой для сигнальных графов символикой на рис. 4.1, *в* и 4.1, *г* a^+ означает переход 0-1 сигнала a (запрос), a^- — переход 1-0 того же сигнала (снятие запроса), b^+ — переход 0-1 сигнала b (ответ), b^- — переход 1-0 того же сигнала (снятие ответа).

§ 4.2. Индикаторы и тесторы

В дальнейшем для апериодических реализаций будут использоваться различные типы индикаторов, о которых было упомянуто в предыдущем разделе. *Индикатор* — обязательная принадлежность согласованных реализаций; изменение сигнала на его выходе свидетельствует об окончании соответствующей фазы переходного процесса. По существу, весь материал гл. 4 посвящен методике построения апериодических устройств и, в частности, индикаторов.

К индикаторам принадлежат и так называемые тесторы, назначение которых состоит в том, чтобы фиксировать принадлежность кодового набора ССК или спейсеру. Таким образом, *тестор* — это устройство, имеющее, скажем, s входов $x_1, \ldots, x_s$ и один выход y. При подаче рабочего кодового набора, принадлежащего ССК, выход y должен принимать одно значение (скажем, 0), при подаче спейсера — другое (1).

Простейшим является тестор для случая, когда каждый из двух классов состоит из одного набора, а именно одним классом наборов является спейсер i — код, состоящий только из нулей, а другим — спейсер e — код, состоящий только из единиц (тестор спейсеров). Тестор спейсеров функционирует в соответствии с сигнальным графом, изображенным на рис. 4.2, *а*. Переход 0-1 (1-0) сигнала y происходит по окончании установки спейсера $e(i)$, но сигнал y не изменяет своего значения на любом из

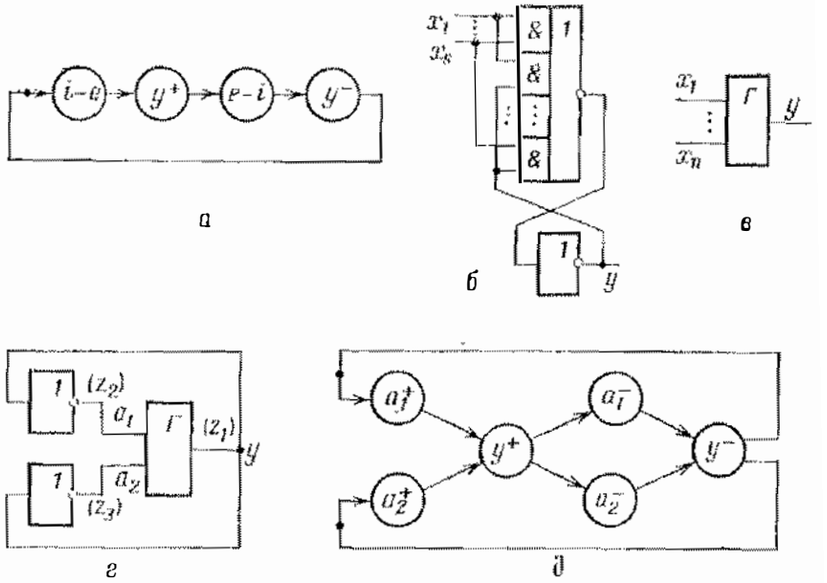


Рис. 4.2. Тестор однофазных сигналов: сигнальный граф (*а*), реализация (*б*), называемая Г-триггером, условное обозначение последнего (*в*); функционирование Г-триггера с внешней средой (*г*) и его задание сигнальным графом (*д*)

промежуточных наборов. Поведение такого тестора должно удовлетворять автоматному уравнению

$$y = \mathop{\&}_{j=1}^{s} x_j \vee y \left(\bigvee_{j=1}^{s} x_j \right) \qquad (4.1)$$

(слитное написание второго дизъюнктивного члена формулы, как было условлено выше, означает конъюнкцию буквы y и члена, заключенного в скобки). Уравнение (4.1) реализуется так называемым *гистерезисным триггером* (*Г-триггером*), построенным на элементе И-ИЛИ-НЕ и инверторе так, как показано на рис. 4.2, *б*. Условное обозначение Г-триггера приведено на рис. 4.2, *в*, а совместное функционирование с внешней средой, моделью которой являются s инверторов (рис. 4.2, *г* для случая $s=2$), может быть также изображено рис. 4.2, *д*, который эквивалентен рис. 4.2, *а* в общем случае.

Пусть кодовые наборы представлены парафазным кодом (ПК) $x_1\tilde{x}_1 \ldots x_s\tilde{x}_s$ (обозначение x_j соответствует, как об этом было условлено в разд. 3.4, ПК с нулевым спейсером i). Тогда поведение *тестора ПК*, очевидно, должно задаваться автоматным уравнением

$$y = \mathop{\&}_{j=1}^{s} x_j\tilde{x}_j \vee y\left(\bigvee_{j=1}^{s} x_j\tilde{x}_j\right). \qquad (4.2)$$

Реализация тестора ПК (рис. 4.3, *а*) является модификацией Г-триггера; условное обозначение показано на рис. 4.3, *б*. В случае $s = 1$ (один парафазный сигнал) уравнение (4.2) вырождается: оно имеет вид $y = x_1\tilde{x}_1$ и

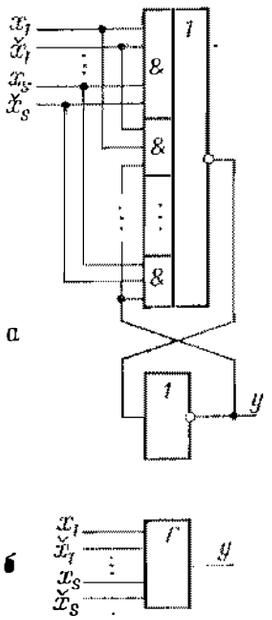


Рис. 4.3. Реализация тестора парафазных кодов (*а*) и его обозначение (*б*)

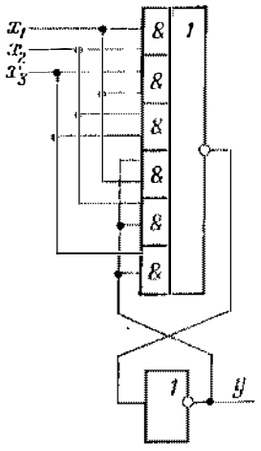


Рис. 4.4. Тестор оптимального равновесного кода для $s = 3$

потому реализуется одним элементом И (а инверсия — одним элементом И-НЕ).

Тестор оптимального равновесного кода (ОРК) описывается автоматным уравнением

$$y = \mathrm{maj}(x_1, \ldots, x_s) \vee y\left(\bigvee_{j=1}^{s} x_j\right), \qquad (4.3)$$

где $\mathrm{maj}(x_1, \ldots, x_s)$ — мажоритарная функция, содержащая $C_s^{s/2}$ термов ранга $]s/2[$. Реализация для случая $s = 3$ показана на рис. 4.4.

Тестор кода с идентификатором (КИ) наиболее сложен в реализации. Для определения принадлежности набора КИ требуется: 1) определить w-класс набора, 2) проверить соответствие этому классу идентификатора, 3) зафиксировать принадлежность набора КИ обязательным или дополнительным наборам и затем сформировать выходной сигнал тестора.

Схема определения w-класса, входящая в состав тестора, служит для подсчета числа k нулей в основных разрядах набора $x_1, \ldots, x_n$. По значению k в рабочей фазе определяется соответствующий w-класс: $w = n - k$. Для случая $n = 8$ может быть использована двухъярусная схема, которая в первом ярусе содержит две одинаковые подсхемы $\mathscr{E}$ и $\mathscr{H}$ со входами $x_1 - x_4$ и $x_5 - x_8$ и выходами $e_1 - e_4$ и $h_1 - h_4$ соответственно.

Подсхема $\mathscr{E}$ подсчитывает число нулей в разрядах $x_1 - x_4$ набора КИ. При подаче на ее входы первых четырех разрядов спейсера e $e_1 = \ldots = e_4 = 0$. Если при подаче КИ один из входов $x_1 - x_4$ равен 0, то $e_1 = 1$, если два входа равны 0, то $e_1 = e_2 = 1$, если три — то $e_1 = e_2 = e_3 = 1$, и, наконец, если четыре — $e_1 = e_2 = e_3 = e_4 = 1$. Аналогично работает подсхема $\mathscr{H}$. Система собственных функций подсхемы $\mathscr{E}$ имеет вид

$$\begin{aligned}
e_1 &= \overline{x_1x_2x_3x_4},\\
e_2 &= \overline{x_1x_2x_3 \vee x_1x_2x_4 \vee x_1x_3x_4 \vee x_2x_3x_4},\\
e_3 &= \overline{x_1x_2 \vee x_1x_3 \vee x_1x_4 \vee x_2x_3 \vee x_2x_4 \vee x_3x_4},\\
e_4 &= \overline{x_1 \vee x_2 \vee x_3 \vee x_4}.
\end{aligned}$$

Соответствующую систему функций $h_1 - h_4$ для подсхемы $\mathscr{H}$ можно выписать, заменив в приведенной выше системе x_j, $1 \leqslant j \leqslant 4$, на x_{j+4}.

Во втором ярусе реализуется система функций $u_1 - u_8$:

$$\begin{aligned}
u_1 &= \overline{e_1 \vee h_1}, \quad u_2 = \overline{e_2 \vee h_2 \vee e_1h_1},\\
u_3 &= \overline{e_3 \vee h_3 \vee e_1h_1h_2 \vee e_1e_2h_1},\\
u_4 &= \overline{e_4 \vee h_4 \vee e_1e_2e_3h_1 \vee e_1e_2h_1h_2 \vee e_1h_1h_2h_3},\\
u_5 &= e_1e_2e_3e_4h_1 \vee e_1e_2e_3h_1h_2 \vee e_1e_2h_1h_2h_3 \vee e_1h_1h_2h_3h_4,\\
u_6 &= \overline{e_2e_3e_4h_2 \vee e_2e_3h_2h_3 \vee e_2h_2h_3h_4},\\
u_7 &= \overline{e_3e_4h_3 \vee e_3h_3h_4}, \quad u_8 = \overline{e_4h_4}.
\end{aligned} \qquad (4.4)$$

Система содержит избыточные буквы, что связано с необходимостью корректной реализации апериодических комбинационных схем (см. разд. 4.3).

Схема проверки соответствия идентификаторов w-классам (опять же для случая $n=8$) содержит четыре яруса. В первом реализуется система функций

$$g_1 = \overline{x_9 \vee x_{10} \vee x_{11} \vee x_{12}},$$
$$g_2 = \overline{u_2x_9 \vee u_3x_{10} \vee u_4x_9x_{10} \vee u_5x_{11} \vee u_6x_9x_{11} \vee u_7x_{10}x_{11} \vee u_8x_9x_{10}x_{11}}, \quad (4.5)$$
$$g_3 = \overline{u_1 \vee u_2x_{10} \vee u_2x_{11} \vee u_3x_{11} \vee u_4x_{11} \vee u_6x_{10}x_{11}},$$
$$g_4 = \overline{u_3x_9x_{11} \vee u_5x_9x_{11} \vee u_5x_{10}x_{11} \vee u_7x_9x_{10} \vee x_{12}},$$
$$g_5 = u_7u_8x_9x_{10}x_{11}, \quad g_6 = u_1u_2u_3u_4u_5u_6,$$

во втором — сигналы a_1, a_2 и a_3 в виде

$$\bar{a}_1 = \overline{g_1g_2g_4g_5 \vee g_2g_3g_5g_6},$$
$$\bar{a}_2 = g_3g_4g_5g_6, \quad a_3 = g_1 \vee g_2 \vee g_3 \vee g_5 \vee g_6;$$

следующие два яруса занимает тестор спейсеров, который реализует автоматное уравнение

$$\bar{y} = \bar{a}_1\bar{a}_2\bar{a}_3 \vee \bar{y}(\bar{a}_1\bar{a}_2 \vee \bar{a}_3)$$

(получается из (4.1) с учетом того, что $a_1a_2 = 0$).

После подачи спейсера $e(x_1 = \ldots = x_{12} = 1)$ $e_1 = \ldots e_4 = h_1 = \ldots = h_4 = 0$, $u_1 = \ldots = u_8 = 1$, $g_1 = \ldots = g_6 = 0$, $\bar{a}_1 = \bar{a}_2 = \bar{a}_3 = 1$. При подаче обязательного (дополнительного) набора по окончании переходного процесса в тесторе $\bar{a}_1 = 0$ ($\bar{a}_2 = 0$), $\bar{a}_3 = 0$ и затем $\bar{y} = 0$.

При построении многовходовых тесторов необходимо учесть, что в промышленных сериях интегральных микросхем (ИМС) числа входов элементов И и ИЛИ (И-ИЛИ) ограничены. Практическим приемом, позволяющим обойти эти ограничения, является построение пирамидальной схемы из тесторов с ограниченным числом входов.

Утверждение 4.1. *Любая пирамидальная схема из тесторов спейсеров является тестором спейсеров.*

Доказательство тривиально: пока не изменятся состояния всех тесторов предыдущего яруса, с которыми связаны входы тестора следующего яруса, его выход не может измениться.

Это утверждение дает основу для построения тесторов из элементов с ограниченным числом входов в тех случаях, когда выполняются следующие условия: 1) множество тестируемых выходов можно разбить на подмножества, для каждого из которых можно построить тестор из элементов с заданным числом входов; 2) тестор выходов построенных тесторов является тестором однофазных сигналов.

Лучшие практические результаты дает использование другого приема — так называемого *параллельного сжатия*, описанного ниже в примере 4.3.

Приведенные конструкции тесторов являются базовыми при построении индикаторов. Рекомендации по реализации индикаторов вытекают из нижеследующих разделов этой главы.

§ 4.3. Синтез комбинационных схем

Одной из необходимых компонент структурного этапа синтеза двухфазных (и, в частности, согласованных) реализаций является построение комбинационных схем, под которыми далее будут попиматься (возможно, содержащие обратные связи) схемы, реализующие системы булевых функций. Пусть схемы удовлетворяют определению 4.1 с теми ограничениями, что множество Z внутренних состояний (п. 4) пусто, функция переходов (п. 5) не задается, а функция выходов (п. 6) есть $\{s_X\} \to \{s_Y\}$, $X' \to Y'$.

Такие комбинационные схемы будем относить к классу апериодических, поскольку их поведение инвариантно к величинам задержек элементов. Далее в гл. 4 будут рассматриваться только *апериодические комбинационные схемы, соответствующие двухфазной реализации*, если это не оговорено особо, т. е. речь будет идти об апериодических двухфазных реализациях комбинационных схем.

Задача синтеза апериодической комбинационной схемы ставится следующим образом. По заданной системе булевых функций

$$\varphi_q = \varphi_q(x_1, \ldots, x_h), \quad 1 \leqslant q \leqslant g,$$

необходимо построить апериодическую комбинационную схему. Очевидно, последняя должна содержать некоторую избыточность хотя бы в силу принятых в гл. 3 соглашений о необходимости использования избыточного кодирования входных и выходных переменных и представления

наборов в ССК. Будем считать, что апериодическая схема будет иметь n входов $X=\{x_1, \ldots, x_n\}$, $n \geqslant h$, и m выходов $Y=\{y_1, \ldots, y_m\}$, $m \geqslant g$, причем ей будет соответствовать *система собственных функций* (ССФ) $y_j=f_j(x_1, \ldots, x_m)$, $1 \leqslant j \leqslant m$, или, короче, $Y=F(X)$. Под ССФ, как обычно, понимается система, записанная как суперпозиция функций, реализуемых элементами асинхронной схемы (далее для краткости — просто *схемы*).

Пусть кодирование наборов из множеств $\tilde{X}$ и $\tilde{Y}$ (множества всех двоичных наборов на входах и выходах схемы соответственно) выполнено посредством ССК (гл. 3) так, что в каждом из них можно выделить по два класса наборов — A и B и K и L соответственно, причем $A \cup B \subset \subset \tilde{X}$, $K \cup L \subset \tilde{Y}$. Напомним, что в определении 3.6 было введено понятие допустимого перехода. Допустимый переход $a-b$ $(b-a)$, $a \in A$, $b \in B$, входных наборов инициирует в схеме переходный процесс, завершающийся переходом $k-l$ $(l-k)$, $k \in K$, $l \in L$, выходных наборов. Естественно потребовать, чтобы из соображений возможности стыковки схем переходы выходных наборов были также допустимыми. Введенное требование допустимости переходов входных и выходных наборов, задающих систему реализуемых схемой функций, гарантирует отсутствие недопустимых переходов наборов и по существу обобщает широко известное в автоматной проблематике понятие *функциональных состязаний* на случай систем булевых функций, реализуемых многовыходной схемой. Кроме того, апериодические схемы должны удовлетворять требованию отсутствия *логических состязаний* (всплесков и дребезгов сигналов на выходах схемы), которое связано с принятой гипотезой о характере задержек элементов и соединительных линий (проводов).

Заметим, что в апериодической схеме, каждый элемент которой реализует изотонную (антитонную) по допустимым переходам своих входных наборов функцию, логические состязания возникнуть не могут, если на входах каждого элемента осуществляется допустимый переход.

При синтезе апериодических схем возникают определенные трудности, которые будут продемонстрированы на примере.

Пример 4.1. Пусть требуется построить апериодическую схему, реализующую функцию переноса одноразрядного сумматора

$$P = xy \vee yp \vee xp, \qquad (4.6)$$

где x, y и p — первое слагаемое, второе слагаемое и перенос из предыдущего разряда соответственно. Пусть входные и выходные переменные представляются ПК и используется двухфазная дисциплина со спейсером i. Пусть также схема должна быть построена в базисе И-НЕ. Одно из возможных решений при принятых

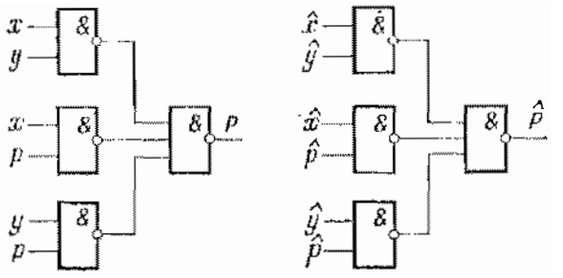


Рис. 4.5. Некорректная реализация функции переноса сумматора

ограничениях, казалось бы, представляется рис. 4,5, откуда имеем (см. соглашение об обозначениях ПК в разд. 3.4)

$$P = \overline{\overline{xy} \cdot \overline{xp} \cdot \overline{yp}}, \quad \hat{P} = \overline{\overline{\hat{x}\hat{y}} \cdot \overline{\hat{x}\hat{p}} \cdot \overline{\hat{y}\hat{p}}}. \qquad (4.7)$$

При подаче нулевого спейсера $i(x=\hat{x}=y=\hat{y}=p=\hat{p}=0)$ выходы всех элементов первого (от входов) яруса принимают значение 1, а элементов второго яруса — значение 0. Переход спейсер-любой рабочий набор вызывает переход 0-1 одного из выходных сигналов P или $\hat{P}$. Однако схема рис. 4.5 неработоспособна. Действительно, пусть подан рабочий набор 101010, т. е. $x=y=p=1$, $\hat{x}=\hat{y}=\hat{p}=0$, влекущий за собой переход 0-1 сигнала P (выход $\hat{P}$ сохраняет значение 0), и задержка цепи, включающей элементы $\overline{xy}$, P и цепей внешней среды, много меньше задержки хотя бы одного из элементов $\overline{xp}$ или $\overline{yp}$. Последнее предположение естественно, поскольку никаких ограничений на соотношение величин задержек в апериодической реализации нет. После того как $\overline{xy}=0$, $P=1$, и если судить только по состоянию выходов, переход закончен, можно подавать спейсер i. Пусть при переходе к спейсеру первым изменится значение на входе x: $x=\hat{x}=0$ (напомним, что $y=p=1$, но на выходе $\overline{yp}$ значение 0 еще не появилось). Тогда $\overline{xy}=1$, $P=0$, и, судя по выходам, закончилась нерабочая фаза. Но поскольку $y=p=1$, на выходе элемента $\overline{yp}$ может появиться значение 0, и P может стать равным 1 до подачи очередного рабочего входного набора, что недопустимо. Для правильного функционирования схема должна регистрировать изменение состояний не только выходов, но и входов, а также промежуточных элементов.

Ясно, что в общем случае выявление некорректностей типа указанных в примере — весьма трудоемкое дело.

Поэтому хотелось бы иметь аппарат, позволяющий определять принадлежность произвольной комбинационной схемы классу апериодических. Такой аппарат разработан в следующем разделе.

4.3.1. Индицируемость. Ограничения определения 4.1 недостаточны для корректного построения логических схем, реализующих системы булевых функций, поэтому дополнительно вводится понятие индицируемости, которое формулируется следующим образом.

Определение 4.3. Пусть на входах схемы с ССФ $Y = F(X)$ осуществляется допустимый переход $a - b$, $a \in A$, $b \in B$, инициирующий на выходах допустимый переход k-l, $k \in K$, $l \in L$. Если для любого $t \in [a, b]$ выполняется $F(t) \neq l$, то будем говорить, что *в схеме входы X индицируются на выходах Y при переходе a-b* или, короче, что *схема индицирует переход a-b*. В случае, когда в схеме входы X индицируются на выходах Y для всех допустимых переходов, будем говорить, что в схеме входы X индицируются на выходах Y (*схема является индицируемой*).

Смысл понятия *индицируемость* состоит в том, что результат переходного процесса в схеме — появление выходного набора из L — может иметь место только лишь после того, как на входе схемы установится набор из B. Определение индицируемости, к сожалению, не задает в явном виде связи между ССФ схемы и этим свойством. Для исследования этой связи рассмотрим сначала простейший частный случай индицируемости.

Определение 4.4. Пусть на входах схемы осуществляется соседний переход a-b такой, что $a \in N^-(b)$, $\varepsilon(a, b) = x_l$, $1 \leqslant l \leqslant n$ (или $a \in N^+(b)$, $\varepsilon(a, b) = \bar{x}_l$), инициирующий на выходах схемы допустимый переход k-l такой, что терм вариаций $\varepsilon(k, l)$ содержит y_r или $\bar{y}_r$, $1 \leqslant r \leqslant m$. Будем говорить, что в этом случае y_r *транслирует* x_l, и обозначать этот факт через $y_r = T(x_l/a\text{-}b)$.

Смысл понятия *транслируемость* состоит в том, что изменение значения некоторого сигнала на входе необходимо влечет за собой изменение значения какого-либо сигнала на выходе схемы.

Определение 4.5. Если для любого допустимого соседнего перехода a^i-b (либо $a^i \in N^-(b)$, $\varepsilon(a, b) = x_j$, $1 \leqslant j \leqslant n$, либо $a^i \in N^+(b)$, $\varepsilon(a, b) = \bar{x}_j$) для всех допустимых a^i-b найдется переменная y_r, $1 \leqslant r \leqslant m$, такая, что $y_r = T(x_j/a^i\text{-}b)$, то будем говорить, что в схеме *входы X транслируются на выходах Y на шаг от набора b*, и обозначать это через $Y = T_s(X/b)$.

Изучение свойства индицируемости позволяет доказать ряд утверждений.

Напомним, что в терм вариаций $\varepsilon(a, b)$ перехода a-b переменная входит в прямом виде, если ее значение 0 сменяется единицей, и в инверсном виде, если ее изменение есть 1-0. Будем называть первые переменные *возрастающими*, а вторые — *убывающими* при переходе a-b.

Лемма 4.1. *Для того чтобы схема была индицируемой, необходимо, чтобы в ее ССФ все функции были изотонны (антитонны) по возрастающим переменным и антитонны (изотонны) по убывающим переменным при допустимых* входных переходах.

Доказательство. Предположим противное, т. е. что найдется допустимый переход a-b и функция $f_i(X) \in F(X)$, которая: либо 1) не является монотонной по переходу a-b, либо 2) антитонна (изотонна) как по возрастающим, так и по убывающим по переходу a-b переменным, либо 3) для части возрастающих (убывающих) по переходу изотонна, а для части — антитонна. Рассмотрим последовательно эти три случая.

1) Немонотонность $f_m(X)$ в соответствии с определением 3.5 означает, что найдется переменная x_n, по которой $f_m(X)$ не является ни изотонной, ни антитонной по переходу a-b, т. е., согласно п. г) определения 3.5, для функции $f_m^{\omega}(X)$, полученной из $f_m(X)$ соответствующей фиксацией переменных, входящих в $\omega(a, b)$, не выполняется ни $f_m^{\omega}(x_n = 0) \geqslant f_m^{\omega}(x_n = 1)$, ни $f_m^{\omega}(x_n = 0) \leqslant f_m^{\omega}(x_n = 1)$. Напомним, что $f_m^{\omega}(x_n = 0)$ и $f_m^{\omega}(x_n = 1)$ зависят от остальных переменных, входящих в $\varepsilon(a, b)$. Сказанное означает, что при переходе a-b найдется такая последовательность промежуточных наборов, что функция $f_m(X)$ изменит свое значение более одного раза. Тогда переход k-l на выходах схемы не будет правильным (определение 3.1).

2. В этом случае должна найтись пара таких переменных x_j, x_k, входящих в $\varepsilon(a, b)$, что: а) x_j — возрастающая, а x_k — убывающая по переходу a-b переменная; б) $f_m^{\omega}(x_j = 0) \geqslant f_m^{\omega}(x_j = 1)$ и $f_m^{\omega}(x_k = 0) \geqslant f_m^{\omega}(x_k = 1)$. Переход a-b для пары x_j, x_k является переходом 01-10, и найдется такая последовательность наборов из $[a, b]$, на которой функция $f_m(X)$ меняет свое значение более одного раза, т. е. переход k-l вновь не будет правильным.

3. Найдется пара возрастающих (убывающих) по переходу a-b переменных x_j, x_k таких, что $f_m^{\omega}(x_j = 0) \geqslant f_m^{\omega}(x_j = 1)$ и $f_m^{\omega}(x_k = 0) \leqslant f_m^{\omega}(x_k = 1)$.

Как и в предыдущем случае, опять найдется такая последовательность промежуточных наборов из $[a, b]$, что $f_m(X)$ изменит свое значение более одного раза и переход k-l не будет правильным.

Таким образом, во всех трех случаях не выполняются условия определения 3.6 допустимого перехода (п. 3) и определения 4.2 индицируемости, что противоречит условиям леммы.

Следствие. Если любой допустимый переход a-b входных переменных схемы происходит по сравнимым наборам, вызывает на выходах переход k-l и $a, k \in \{i, e\}$, то для индицируемости схемы необходимо, чтобы: 1) все функции ССФ были изотонны (антитонны) при допустимых входных переходах; 2) переходы на выходах схемы, вызванные допустимыми переходами на входах, также происходили по сравнимым наборам.

Лемма 4.2. *Для того чтобы переходы* s-b $(b$-$s)$, $b \in B$, *двухфазной дисциплины со спейсером* s *были допустимыми, необходимо, чтобы класс* B *состоял из попарно несравнимых наборов.*

Доказательство. При двухфазной дисциплине со спейсером s после него может быть подан любой набор из B, например b^k или b^l. Пусть $s - i$ (i — нулевой спейсер). Тогда в силу определения 3.3 переходы i-b^k и i-b^l должны быть сравнимыми: $i < b^k$ и $i < b^l$ (i — наименьший набор). Предположим, что b^k и b^l сравнимы; для определенности пусть $b^k < b^l$. Тогда имеет место $i < b^k < b^l$, т. е. $b^k \in [i, b^l]$, что противоречит п. 4) определения 3.6. Аналогичные рассуждения справедливы для случаев обратного перехода b-i, $s - e$ и общего случая $s \neq i, e$.

Введем также

Определение 4.6. *Частной производной* $\partial f(X)/\partial x_i$ или *булевой разностью* функции $f(X) = f(x_1, \ldots, x_n)$ по ее переменной x_i называется выражение

$$\partial f(X)/\partial x_i = f(x_i = 1) \oplus f(x_i = 0),$$

где $\oplus$ — знак суммы по модулю два.

Если функция $f(X)$ изотонна по x_i, то булеву производную можно записать в виде

$$\partial f(X)/\partial x_i = \bar{f}(x_i = 0) f(x_i = 1). \qquad (4.8)$$

Соответствующее выражение может быть получено для случая, когда $f(X)$ антитонна по x_i.

Для нахождения необходимых и достаточных условий индицируемости схемы перейдем к анализу условий транслируемости.

Для сокращения текстов последующих утверждений и их доказательств примем обозначение $\tilde{x}_j$, $\tilde{x}_j \in \{x_j, \bar{x}_j\}$, и при выполнении $\varepsilon(a, b) = x_j$ или $\varepsilon(a, b) = \bar{x}_j$ будем писать $\varepsilon(a, b) = \tilde{x}_j$. Тогда из справедливости какого-либо утверждения для $\varepsilon(a, b) = x_j$ его доказательство для

$\varepsilon(a, b) = \bar{x}_j$ следует из того факта, что $\partial f_r(X)/\partial \bar{x}_j = \partial f_r(X)/\partial x_j$.

Утверждение 4.2. *Выход* y_r, $1 \leqslant r \leqslant m$, *транслирует вход* x_j, $1 \leqslant j \leqslant n$, *на соседнем переходе* a-b, $\varepsilon(a, b) = \tilde{x}_j$, *тогда и только тогда, когда набор значений переменных, задаваемых термом* $\omega(a, b)$, *является корнем уравнения* $\partial f_r(X)/\partial x_j = 1$.

Доказательство. Пусть выполняются условия определения 4.4 — например, при $\varepsilon(a, b) = x_l$ терм вариаций $\varepsilon(k, l)$ содержит y_r. Это значит, что найдется такая переменная y_r, что $y_r(a) \oplus y_r(b) = 1$ при $a_l \neq b_l$, т. е. $a_l = \bar{b}_l$; тогда $y_r(x_l = 0) \oplus y_r(x_l = 1) = 1$. Из определения 4.6 булевой производной следует, что это равенство справедливо тогда и только тогда, когда фиксация X, задаваемая термом $\omega(a, b)$, является корнем уравнения $\partial f_r(X)/\partial x_l = 1$.

Из этого утверждения и определения 4.5 непосредственно следует утверждение 4.3.

Утверждение 4.3. *Входы* X *транслируются на выходах* Y *на шаг от набора* b *тогда и только тогда, когда для любого допустимого соседнего перехода* a^i-b *набор значений переменных, задаваемых термом* $\omega(a^i, b)$, *является корнем соответствующего* (j-*го*) *уравнения вида*

$$\bigvee_{r=1}^{m} \partial f_r(X)/\partial x_j = 1, \qquad (4.9)$$

где $\tilde{x}_j$ — *переменная, составляющая терм вариаций* $\varepsilon(a^i, b)$, m — *число выходов схемы.*

Если все функции $f_r(X)$ изотонны по переменной x_j, уравнение (4.9) в соответствии с (4.8) примет вид

$$\bigvee_{r=1}^{m} \bar{f}_r(x_j = 0) f_r(x_j = 1) = 1, \qquad (4.10)$$

что упрощает использование утверждений 4.1 и 4.2. Легко построить соответствующее выражение для случая, когда функция $f(X)$ антитонна по переменной x_j.

Пример 4.2. Пусть ССФ схемы есть $f_1(X) = x_1x_2$, $f_2(X) = x_1 \vee x_2$. Для набора $b^1 = 11$ множество $N^-(b^1)$ образуют наборы $a^1 = 01$ и $a^2 = 10$, множество $N^+(b^1) = \varnothing$. Для набора $b^2 = 00$ множество $N^+(b^2)$ образуют наборы a^1 и a^2, $N^-(b^2) = \varnothing$. Производные функций $f_1(X)$ и $f_2(X)$ по x_1 равны соответственно

$$\partial f_1(X)/\partial x_1 = f_1(x_1 = 0) \oplus f_1(x_1 = 1) = x_2, \quad \partial f_2(X)/\partial x_1 = \bar{x}_2.$$

Аналогично можно вычислить производные по x_2. Выполняется $y_1 = T_e(X/b^1)$, если допустимы переходы из $N^-(b^1)$ в b^1, и $y_2 = T_e(X/b^2)$, если допустимы переходы из $N^+(b^2)$ в $\{a^1, a^2\}$.

Следующая теорема позволяет рассматривать транслируемость как частный случай индицируемости.

Теорема 4.1. *Пусть в схеме используется двухфазная дисциплина со спейсером $s \in \{i, e\}$, в которой все переходы соседние (для всех $b \in B$ $\varepsilon(s, b) = \tilde{x}_j$, $1 \leqslant j \leqslant n$. Тогда, для того чтобы схема была индицируемой, необходимо и достаточно, чтобы выполнялись следующие три условия: 1) все функции ССФ схемы были изотонны или антитонны по допустимым переходам; 2) все наборы из класса L были попарно несравнимыми; 3) для любого допустимого перехода нашелся бы выход y_r, $1 \leqslant r \leqslant m$, транслирующий x_j либо $\bar{x}_j$.*

Доказательство. Достаточно рассмотреть случай, когда в ССФ схемы все функции изотонны по допустимым переходам и $\varepsilon(s, b) = x_j$.

а) *Необходимость.* Необходимость условия 1) следует из леммы 4.1, условия 2) — из того, что выходной набор $k = F(s)$ играет роль спейсера в двухфазной дисциплине (выходной), и из леммы 4.2. Необходимость условия 3) следует из определения 4.3: для любого допустимого перехода s-b (b-s), $F(s) \neq F(b)$, следовательно, существует по крайней мере одна переменная y_r такая, что $y_r = T(x_j/s\text{-}b)$ при $\varepsilon(s, b) = \tilde{x}_j$.

б) *Достаточность.* Допустимость входной дисциплины задастся условием теоремы. Требование $F(t) \neq l$ выполняется, поскольку $\{t\} = \varnothing$: наборы s и b — соседние, и подкуб $[s, b]$ не содержит других наборов. Допустимость выходных переходов следует из таких соображений. Если для любого допустимого перехода s-b (b-s) найдется $y_r = T(x_j/s\text{-}b)$, то $k \neq l$ ($F(s) \neq F(b)$). Правильность и сравнимость переходов k-l вытекает из изотонности ССФ схемы. Остается показать, что для любого $h \in [k, l]$ имеет место $h \in Y \setminus (\{k\} \cup \{l\})$. Из изотонности ССФ следует, что $k < h < l$, а из несравнимости любой пары наборов из L, что $h \notin L$. Таким образом, все условия определения 4.3 выполняются.

Обобщение теоремы 4.1 на случай произвольных допустимых переходов дает

Теорема 4.2. *Пусть в схеме используется двухфазная дисциплина со спейсером $s \in \{i, e\}$. Тогда, для того чтобы схема была индицируемой, необходимо и достаточно, чтобы выполнялись следующие четыре условия: 1) в ССФ схемы все функции были изотонны или антитонны по допустимым переходам; 2) все наборы из класса B были попарно несравнимыми; 3) все наборы из класса L были попарно несравнимыми; 4) входы X транслировались на выходах Y на шаг от наборов b и s.*

Доказательство. а) *Необходимость.* Необходимость условия 1) следует из леммы 4.1, условий 2) и 3) — из леммы 4.2 (см. доказательство теоремы 4.1). Необходимость условия 4) следует из определения 4.3: для всех $t \in (s, b)$, $t \neq b$, s, $F(t) \neq l$, в том числе для всех t, соседних с b, т. е. $F(t) \neq F(l)$. По определению 4.5 имеет место $Y = T_s(X/b)$.

б) *Достаточность.* Допустимость входной дисциплины задается условием теоремы. Требование $F(t) \neq l$ выполняется, поскольку для любого набора $t \in [s, b]$, $t \neq b$, найдется соседний с b набор d такой, что $d \in [t, b]$. По определению 4.5 $F(d) \neq F(b)$, и поскольку $F(t) \leqslant F(d) \leqslant F(b)$ (функции изотонны), имеет место $F(t) \neq F(b)$, $F(t) \neq l$. Допустимость выходных переходов доказывается так же, как в теореме 4.1. Таким образом, все условия определения 4.3 выполнены.

Обратим внимание на то, что индицируемость на переходе s-b не гарантирует индицируемости на переходе b-s. Из определения 4.3 и теоремы 4.2 непосредственно следует, что последовательное соединение схем, входы которых индицируются на выходах, также обладает свойством индицируемости.

Пример 4.1 (продолжение). Схема, реализующая функцию переноса (4.7) в виде

$$\begin{aligned} d &= \overbrace{\tilde{x}\tilde{y}\tilde{p}\tilde{x}\tilde{y}\tilde{p}}, \\ P &= \overbrace{\tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p}}, \\ \hat{P} &= \overbrace{\tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p} \vee \tilde{x}\tilde{y}\tilde{p}}, \end{aligned} \tag{4.11}$$

имеющая три парафазных входа и три выхода, является индицируемой. При подаче спейсера e $P = \hat{P} = d = 0$, при подаче любого рабочего набора ($x \neq \hat{x}$, $y \neq \hat{y}$, $p \neq \hat{p}$) $P \neq \hat{P}$ и $d = 1$. Входы этой схемы индицируются на ее выходах при всех допустимых при двухфазной дисциплине со спейсером i переходах. Действительно, все функции в системе (4.11) антитонны, входные и выходные наборы представлены ПК и ОРК соответственно и поэтому попарно несравнимы. Наконец, входы схемы транслируются на шаг от рабочих наборов. Так, для перехода 010110-000000 — переменные перечислены в порядке $\tilde{x}\tilde{x}\tilde{y}\tilde{y}\tilde{p}\tilde{p}$ — дизъюнкция трех производных $\partial d/\partial x \vee \partial P/\partial x \vee \partial \hat{P}/\partial x$ на наборе $\tilde{x}\tilde{y}\tilde{y}\tilde{p}\tilde{p}$ — 10110 обращается в 1, как и дизъюнкция производных по остальным пяти переменным на соответствующих наборах. Аналогично можно убедиться, что входы схемы транслируются и на шаг от спейсера e. Таким образом, все четыре условия теоремы 4.2 выполнены, так что схема с ССФ (4.11) является индицируемой.

Ниже показан один из базовых способов построения тесторов.

Пример 4.3. В схеме, состоящей из одного элемента И (ИЛИ) с собственной изотонной функцией $y_1 = x_1 x_2 \ldots x_n$ ($y_2 = x_1 \vee x_2 \vee \ldots \vee x_n$) входы индицируются на выходе при переходе $i - e$ ($e - i$), но не индицируются при переходе $e - i$ ($i - e$). Эта схема не является индицируемой, ибо не выполняется условие 4) теоремы 4.2. Однако в схеме рис. 4.6, состоящей из двух

элементов — И и ИЛИ (так называемая схема параллельного сжатия), входы $x_1, \ldots, x_n$ индицируются на выходах y_1, y_2 при дисциплине вида $i-c-i-\ldots$, а потому эта схема является индицируемой. Действительно,

$$\partial y_2/\partial x_i = \bar{x}_1\bar{x}_2\ldots\bar{x}_{i-1}\bar{x}_{i+1}\ldots\bar{x}_n,$$
$$\partial y_2/\partial x_i = \bar{x}_1\bar{x}_2\ldots\bar{x}_{i-1}\bar{x}_{i+1}\ldots\bar{x}_n,$$

так что $y_1 = T_s(X/e)$, а $y_2 = T_s(X/i)$, и все условия теоремы 4.2 выполняются. Очевидно, что многовходовые элементы И и ИЛИ (или И-НЕ и ИЛИ-НЕ) в схеме параллельного сжатия можно заменить многоярусными из тех же элементов с меньшим числом входов (в том числе и двухвходовыми). Читателю предоставляется возможность доказать, что подсхемы тестора КИ, описываемые ССФ (4.4) и (4.5), принадлежат классу инициируемых. *Указание.* Следует воспользоваться теоремой 4.2, как и в примере 4.1 (продолжение).

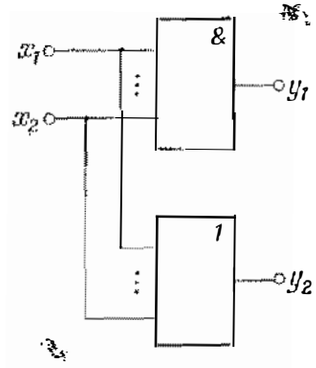


Рис. 4.6. Схема параллельного сжатия

Утверждение 4.4. *Двухфазная комбинационная схема является апериодической тогда и только тогда, когда она индицируема.*

Доказательство. Необходимость докажем от противного. Предположение, что двухфазная реализация комбинационной схемы не обладает свойством индицируемости, сводится к тому, что найдется такой набор $t \in [a, b]$, что $F(t) = l$. Но тогда переход в l произойдет до того, как переключится некоторый ее элемент, т. е. до того, как входной набор $a \in A$ сменится набором $b \in B$, что для двухфазной реализации невозможно. *Достаточность* также показывается от противного. Пусть входы X индицируются на выходах Y схемы и нарушается двухфазная дисциплина: изменение выходных сигналов происходит до того, как завершится переходный процесс в каком-то элементе схемы. Тогда на входах схемы имеет место некоторый набор $t \notin B$, а на ее выходах уже установился набор $l \in L$ или какой-то из наборов $h \in [k, l]$, $h \notin L$. Это противоречит определению 4.3 индицируемости.

Утверждение 4.4 является центральным в § 4.3. Оно позволяет использовать развитые в п. 4.2.1 методы анализа комбинационных схем на принадлежность классу апериодических.

4.3.2. Стандартные реализации. Поскольку процедура анализа схем на индицируемость достаточно трудоемка, имеет смысл остановиться на *канонических приемах синтеза* схем, позволяющих перейти от системы функций непосредственно к реализации, называющейся в этом случае *стандартной*. Методически процесс синтеза при

этом весьма несложен, но ценой за это является увеличение избыточности стандартной реализации.

Реализация по минимальным формам. Если ССФ задана в минимальных формах (ДНФ или КНФ), то этим формам можно поставить в соответствие минимальные представления в ПК. Они получаются заменой каждой инверсии переменной $\bar{x}_j$ на $\hat{x}_j$ или $\tilde{x}_j$ в функции и ее инверсии. Для сохранения индицируемости можно потребовать, чтобы выходы всех элементов схемы также представлялись в виде ПК. Это требование приводит к *двухканальной* так называемой *перекрестной реализации*, в которой каждому элементу с выходом α соответствует элемент с двойственной собственной функцией $\hat{\alpha}$ ($\tilde{\alpha}$). Тогда пара (α, $\tilde{\alpha}$), где $\tilde{\alpha}$ есть либо $\hat{\alpha}$, либо α, представляет собой парафазную переменную. Таким образом, в данном случае стандартная реализация имеет удвоенное число входов и выходов.

Пример 4.1 (продолжение). Для иллюстрации рассмотренного способа представим функцию (4.6) в виде

$$P = xy \vee yp \vee xp, \quad \hat{P} = \hat{x}\hat{y} \vee \hat{y}\hat{p} \vee \hat{x}\hat{p} \qquad (4.12)$$

и реализуем эту пару в виде

$$P = \overline{\overline{xy}\cdot\overline{yp}\cdot\overline{xp}}, \quad \hat{P} = \overline{\overline{\hat{x}\vee\hat{y}}\vee\overline{\hat{y}\vee\hat{p}}\vee\overline{\hat{x}\vee\hat{p}}}. \qquad (4.13)$$

Полученная схема является апериодической, если в качестве ее шести входов используются x, $\hat{x}$, y, $\hat{y}$, p, $\hat{p}$, а в качестве выходов — P, $\hat{P}$ и выходы всех шести элементов И-НЕ и ИЛИ-НЕ, собственные функции которых записаны под знаком черты в выражениях (4.13) Общее число входов и выходов схемы равно 14, число элементов — 8. Согласованная схема должна к тому же содержать индикатор на 14 входов.

Реализация по ортогональным формам. Отказавшись от минимальных форм представления ССФ, можно перейти к *ортогональному представлению*, при котором для любых двух термов β_i и β_j при $i \neq j$ выполняется условие $\beta_i\beta_j = 0$. Очевидно, что такое представление всегда существует: в предельном случае это совершенная ДНФ.

Пример 4.1 (продолжение). Уравнения (4.12) перепишем в виде

$$P = xy \vee \hat{x}\hat{y}p \vee \hat{x}\hat{y}p, \quad \hat{P} = \hat{x}\hat{y} \vee x\hat{y}\hat{p} \vee \hat{x}\hat{y}\hat{p}. \qquad (4.14)$$

В двухъярусной реализации (4.14) на элементах И-НЕ выходы элементов $\overline{xy}$, $\overline{\hat{x}\hat{y}p}$, $\overline{\hat{x}\hat{y}p}$, $\overline{\hat{x}\hat{y}}$, $\overline{x\hat{y}\hat{p}}$, $\overline{\hat{x}\hat{y}\hat{p}}$ первого яруса индицируются

на выходах $P, \widehat{P}$, поскольку вследствие ортогональности в любой фазе переключается только один из элементов первого яруса в обоих каналах. Однако входы схемы не индицируются на ее выходах P, $\widehat{P}$. Поэтому в качестве выходов схемы следует рассматривать также все ее шесть входов (помимо выходов элементов P и $\widehat{P}$). Согласованная схема должна содержать индикатор на восемь входов.

Реализация на элементах типа гистерезисных триггеров. Этот подход основан на идее совмещения реализации ССФ и использования индикаторов, встроенных в базисные элементы. Такое совмещение требует применения ИМС типов И-ИЛИ-НЕ и инверторов. Оно осуществляется на специальных триггерах, в некотором смысле похожих на гистерезисные.

Если булевы функции x_1x_2, $x_1 \vee x_2$, $x_1 \oplus x_2$ представить в виде автоматных уравнений

$$\begin{aligned} f_{\mathscr{C}} &= x_1\breve{x}_1x_2\breve{x}_2 \vee f_{\mathscr{C}}(x_1x_2 \vee x_1\breve{x}_1 \vee x_2\breve{x}_2), \\ f_{\vee} &= x_1\breve{x}_1x_2\breve{x}_2 \vee f_{\wedge}(x_1 \vee x_2), \\ f_{\oplus} &= x_1\breve{x}_1x_2\breve{x}_2 \vee f_{\oplus}(x_1\breve{x}_2 \vee \breve{x}_1x_2) \end{aligned} \qquad (4.15)$$

соответственно, то легко видит, что при подаче спейсера e каждая из функций в (4.15) примет значение 1, а при подаче ПК либо останется прежней (1), либо изменится на 0.

По реализации в базисе $\vee$, $\&$, $\oplus$ можно перейти к двухканальной перекрестной реализации на элементах типа Г-триггеров с собственными функциями (4.15) в одном канале и двойственными им функциями в другом канале. В такой реализации входы индицируются на выходах; в согласованной реализации индикатор имеет входами лишь выходы перекрестной реализации.

Реализация «с коллективной ответственностью». Рассмотрим еще один способ построения схем. Схема может не обладать свойством индицируемости из-за того, что заранее неизвестно, какое число элементов в данном ярусе может изменить состояния своих выходов в рабочей фазе. Идея, положенная в основу этого способа, состоит в следующем. Реализуется возможность принудительного перевода всех элементов яруса в одном канале в одно состояние. Этот перевод осуществляется сигналом обратной связи с элемента последующего яруса.

Рассмотрим рис. 4.7. При подаче на вход спейсера i и фазового сигнала $\bar{a} = 1$ на выходах всех элементов первого яруса устанавливается спейсер e, а на выходах всех элементов второго яруса — спейсер i. При предъявлении рабочего набора ПК и установке фазового сигнала $\bar{a} = 0$ на выходе одного или нескольких элементов одного из каналов первого яруса появится сигнал 0. Тогда на выходе элемента второго яруса того же канала появится сигнал 1, из-за чего выходы всех элементов первого яруса этого канала

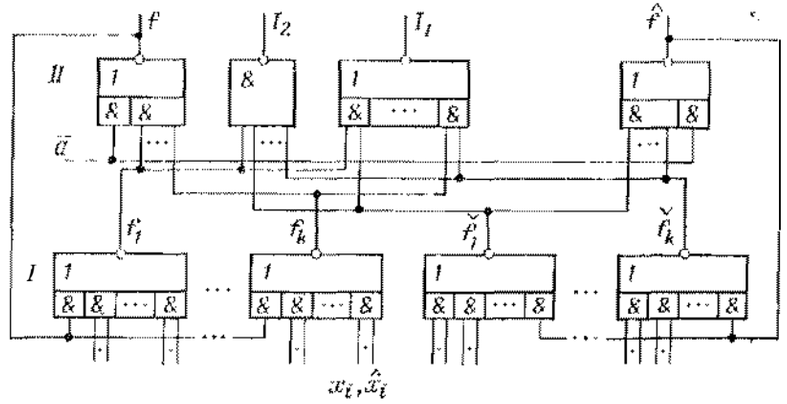


Рис. 4.7. Идея реализации методом коллективной ответственности

примут значение 0. Выходы элементов первого яруса индицируются на выходах схемы с ССФ

$$J_1 = \overline{f_1\breve{f}_1 \vee \ldots \vee f_k\breve{f}_k}, \quad J_2 = \overline{f_1\breve{f}_1 \ldots f_k\breve{f}_k}.$$

В конце рабочей фазы (фазы гашения) $J_1 = J_2 = 1(0)$. В согласованной схеме выходы f, $\breve{f}$, J_1, J_2 наряду со всеми входными переменными ПК являются входами индикатора.

Рассмотренные в этом разделе способы стандартной реализации булевых функций в классе апериодических комбинационных схем, естественно, не исчерпывают всех возможных подходов. В заключение раздела можно отметить, что, исходя из соображений, связанных с анализом избыточности самосинхронизирующихся кодов, должен существовать такой апериодический аналог синхронной схемы с h входами и q выходами, который имеет порядка $h + \log_2 h$ входов и $q + \log_2 q$ выходов.

§ 4.4. Апериодические триггеры

Прежде чем перейти к стандартным апериодическим реализациям схем с памятью, следует рассмотреть конструкции устройств памяти автоматов, в которых, вследствие допущения о невозможности использования встроенных элементов задержки, обычно применяют триггеры различных типов (как правило, RS-, D- и T-типов).

Две идеи организации индикации моментов окончания переходных процессов в *RS*-триггерах иллюстрируются рис. 4.8. На рис. 4.8, *а* изображен обычный асинхронный *RS-триггер на элементах И-НЕ*, снабженный

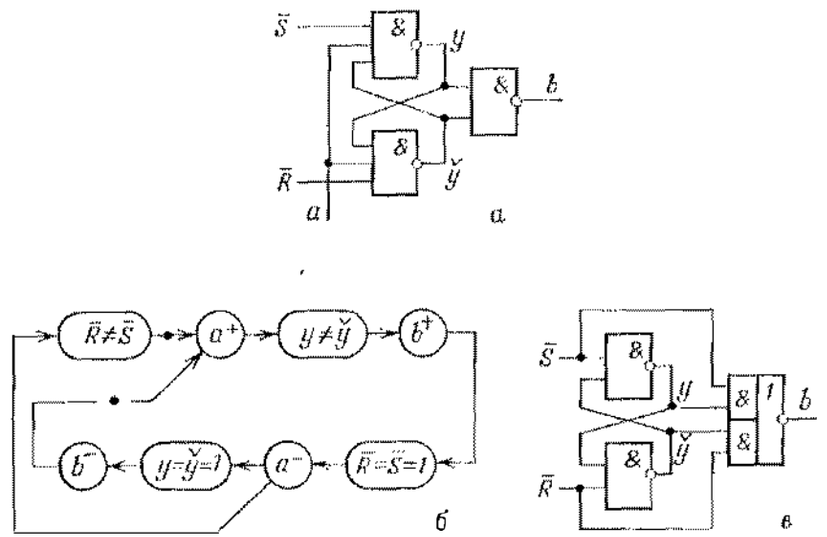


Рис. 4.8. Реализация идеи индикации моментов окончания переходных процессов в триггерах: с фазовым сигналом (*а*), организация взаимодействия с внешней средой (*б*), индикация по Маллеру (*в*)

дополнительным *фазовым сигналом a и индикатором b*, представляющим собой элемент 2И-НЕ.

В исходном состоянии $a = 0$, $y = \check{y} = 1$, $b = 0$. Если на входах $R \neq S$, то по $a = 1$ начинается рабочая фаза. Первым переключается в 0 одно из плеч триггера $\check{y}$ [1]), а затем — индикатор b (в 1). В нерабочей фазе по $a = 0$ может изменяться значение сигналов на входах $\bar{R}$ и $\bar{S}$, и одно из плеч триггера устанавливается в 1 ($y = \check{y} = 1$), после чего $b = 0$. Новая рабочая фаза ($a = 1$) может начаться только по $R \neq S$ и $b = 0$. Описанная динамика работы триггера изображается сигнальным графом (рис. 4.8, *б*). Обозначение типа $y \neq \check{y} - y = \check{y}$ в данном случае заменяет обозначения переходов 01-11 или 10-11 сигналов y и $\check{y}$. Параллельность

[1]) Здесь и далее триггеры, образованные парами элементов $(y, \check{y})$ или $(y, \hat{y})$, будем для экономии обозначать одной из букв — y или $\hat{y}$. Такое обозначение позволяет сразу же определить тип транзитного состояния триггера — единичный в первом случае и нулевой во втором.

в сигнальном графе введена в данном случае для того, чтобы показать, что немедленно после $a = 0$ может быть начато формирование новых значений сигналов на установочных входах. В чисто последовательном варианте (когда после b^- непосредственно следуют $R \neq S$, а после этой вершины следует a^+) возможное время на формирование новых значений R и S внешней средой увеличилось бы на задержку срабатывания одного из плеч триггера и индикатора b.

Длительности каждой фазы переходного процесса $2T$ [1]).

Аналогично схеме рис. 4.8, *а* может быть построен триггер $\hat{y}$ на элементах ИЛИ-НЕ, собственные функции которых двойственны собственным функциям элементов И-НЕ, а обозначения входных сигналов заменены на инверсные.

Индикация моментов окончания фаз переходного процесса здесь оказалась возможной благодаря тому, что схема рис. 4.8, *а* — пример согласованной реализации. Этот факт доказывает сравнение рис. 4.8, *б* и 4.1, *в*: порядок следования в этих сигнальных графах фрагментов, относящихся к управляющим сигналам, одинаков. К сожалению, в нерабочей фазе пару сигналов $\bar{R} = \bar{S} = 1$ можно рассматривать как спейсер, так что информация, записанная в триггер y, теряется ($y = \check{y}$). Поэтому приведенную конструкцию можно рассматривать только как *пример вырожденного автомата*. Однако на базе этой конструкции, как показано ниже, можно реализовать апериодические *RS*-триггеры.

Если в схеме рис. 4.8, *а* индикация моментов окончания переходных процессов в каждой фазе осуществлялась благодаря использованию простейшего индикатора — элемента И-НЕ (частный случай Г-триггера с одним парафазным входом), то вторая идея индикации осуществляется *сравнением состояния собственно триггера со значениями его установочных входов*.

Обратимся к рис. 4.8, *в*. Здесь используется индикатор с собственной функцией $b = \overline{\bar{S}y \vee \bar{R}\check{y}}$. В рабочей фазе (фазе записи), если значения установочных входов соответствуют состоянию триггера $(\bar{S} = \check{y}, \bar{R} = y)$, то переключения триггера не происходит и $b = 1$. В противном случае $(\bar{S} = y, \bar{R} = \check{y})$ $b = 1$ после переключения триггера, $b = 0$ во время нахождения триггера в единичном

[1]) Здесь и далее через T будет обозначаться средняя задержка одного элемента схемы.

транзитном состоянии. В нерабочей фазе (фазе хранения) $\bar{S} = \bar{R} = 1$, предыдущее состояние триггера не изменяется, но переключается индикатор: в конце фазы $b = 0$.

Способ построения индикаторов, основанных на идее, использованной в триггере рис. 4.8, *в*, весьма широко используется в апериодической схемотехнике.

***RS*-триггеры.** От «заготовок» (рис. 4.8, *а* и *в*) можно перейти к апериодическим *RS*-триггерам, изображенным

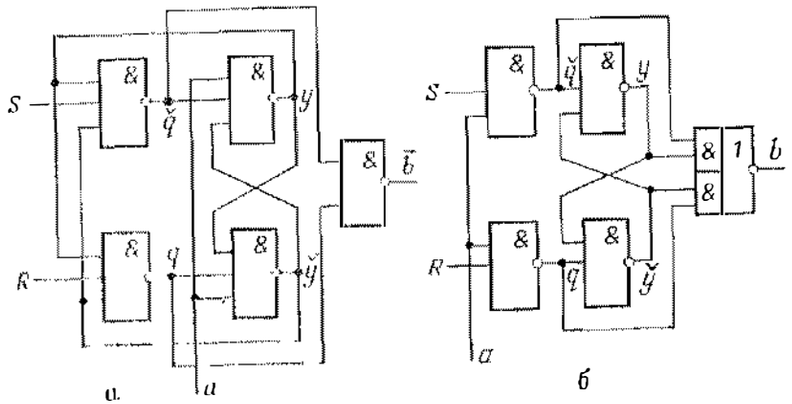


Рис. 4.9. Два апериодических триггера с раздельными входами на элементах И-НЕ

на рис. 4.9. В отличие от схем рис. 4.8 здесь происходит автоматическая *отсечка* от входных сигналов тотчас после завершения рабочей фазы, и значения R и S фиксируются только на время переходного процесса в этой фазе, пока (в схеме на рис. 4.9, *а*) не установится $\check{b} = 0$. После этого изменения R и S могут быть любыми.

В фазе гашения ($a = 0$) для этой схемы не должно выполняться условие $RS = 1$, т. е. спейсер должен быть нулевым. Поскольку переключение входных вентилей происходит после переключения триггера $\check{y}$, окончание переходного процесса должно индицироваться по вентилям: последние переходят в состояния, соответствующие инверсиям входных сигналов, только после гашения триггера $\check{y}$, и в состояние $q = \check{q} = 1$ — после завершения перехода триггера в рабочее состояние.

Цикл работы схемы рис. 4.9, *а* — $6T$ (*по* $3T$ *в обеих фазах*).

В фазе гашения информация, записанная в триггер, теряется, так как в этой фазе $y = \check{y}$. Если нужно хранить информацию в триггере и в нерабочей фазе (за исключением, естественно, времени перехода самого триггера) — в этом случае пользуются термином «фаза хранения», используют триггер рис. 4.9, *б*. При $a = 0$ сигналы R и S не влияют на схему, $q = \check{q} = 1$, и триггер находится в фазе хранения ($b = 0$). При $a = 1$ вентили инвертируют входные сигналы, т. е. $\check{q} = \bar{S}$, $q = \bar{R}$; после окончания фазы $b = 1$. Сигнал на выходе индикатора $\check{b} = 0$ сохраняется, пока не выполнены оба условия $\check{y} = q$, $y = \check{q}$. Таким образом, $b = (\check{y} \oplus q)(y \oplus \check{q})$. Если учесть, что транзитное состояние здесь единичное, собственную функцию индикатора можно записать в виде $b = \overbrace{q\,y \vee \check{q}\check{y}}$. Как уже говорилось, в фазе хранения $a = 0$, $q = \check{q} = 1$, $y \neq \check{y}$ и $b \neq 0$. После $a = 1$ изменяются значения q и $\check{q}$. Если информация, записанная в триггере прежде, соответствует значениям входных сигналов, то изменения состояния триггера не происходит, а изменяется лишь значение сигнала на выходе индикатора, т. е. непосредственно вслед за изменением состояния одного из вентилей $b = 1$. Переходный процесс в рабочей фазе (фазе записи) на этом завершается. Если же значения входных сигналов не соответствуют состоянию триггера, то либо $q\check{y} = 1$, либо $\check{q}y = 1$ и, поскольку триггер имеет единичное транзитное состояние, то сигнал на выходе индикатора изменится лишь после окончания переходного процесса в триггере. В рассмотренной схеме недопустимо изменение входных сигналов в течение всего переходного процесса фазы записи, а затем до конца фазы записи S и R могут либо сохранять свои значения, либо оба стать равными 0. Информация из *RS*-триггера может быть считана в любое время, кроме времени перехода из фазы хранения в фазу записи. Переход из фазы записи в фазу хранения длится $2T$, а обратный переход — либо $4T$, либо $2T$ (в зависимости от того, изменяется или не изменяется состояние триггера). *Общий цикл работы схемы либо* $4T$, *либо* $6T$ и в среднем короче, чем цикл работы схемы рис. 4.9, *а*.

Триггеры рис. 4.10 на элементах И-ИЛИ-НЕ аналогичны схеме рис. 4.9, однако имеют большее быстродействие. При $\bar{a} = 1$ (фаза хранения) схема рис. 4.10, *а* невосприимчива к входным сигналам R и S, $b = 0$. После $\bar{a} = 0$ (фаза записи) поведение триггера определяется

его состоянием и входными сигналами: если состояние соответствует значениям входных сигналов, то триггер сохраняет его, переход 0-1 сигнала b завершает переходный процесс фазы записи; в противном случае $b = 0$ до тех пор, пока триггер не изменит своего состояния. Заметим, что при переходе через единичное транзитное

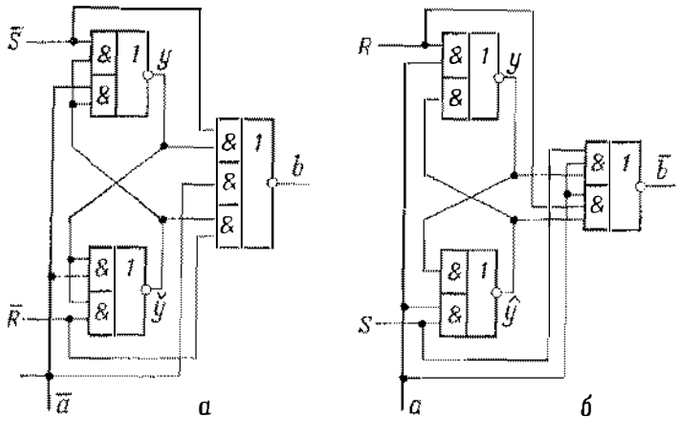


Рис. 4.10. Два апериодических триггера с раздельными входами на элементах И-ИЛИ-НЕ

состояние значение выхода $b = 0$ индикатора не изменяется, $b = 1$ лишь после завершения фазы. Длительность фазы записи равна либо $3T$, либо T. После перехода 0-1 фазового сигнала состояние триггера сохраняется, но замыкаются цепи обратной связи, исключающие влияние входных сигналов на триггер, и одновременно происходит изменение состояния индикатора (задержка равна T).

Цикл работы триггера рис. 4.10, *а* — $2T$ *или* $4T$.

Триггер рис. 4.10, *б* в фазе хранения ($a = 0$) не реагирует на входные сигналы, $\bar{b} = 1$. Если состояние триггера соответствует значениям входных сигналов, то после перехода 0-1 фазового сигнала изменяетя значение сигнала на выходе индикатора: $\bar{b} = 0$. В противном случае транзитным состоянием триггера является нулевое, и $\bar{b} = 1$ сохраняется до тех пор, пока не выполнится одно из двух условий: либо $Sy = 1$, либо $R\hat{y} = 1$. Установка $\bar{b} = 0$ завершает фазу. *Длительность перехода из фазы хранения в фазу записи равна* T *или* $3T$. Переход 1-0 фазового сигнала приводит к «отсечке» от входных сигналов

и смене состояния индикатора, чем и завершается *фаза хранения длительностью* T.

***D*-триггеры.** *D-триггер*, осуществляющий однотактную задержку входного сигнала, при использовании триггеров с гашением, очевидно, должен иметь основной и вспомогательный триггеры: при гашении основного триггера

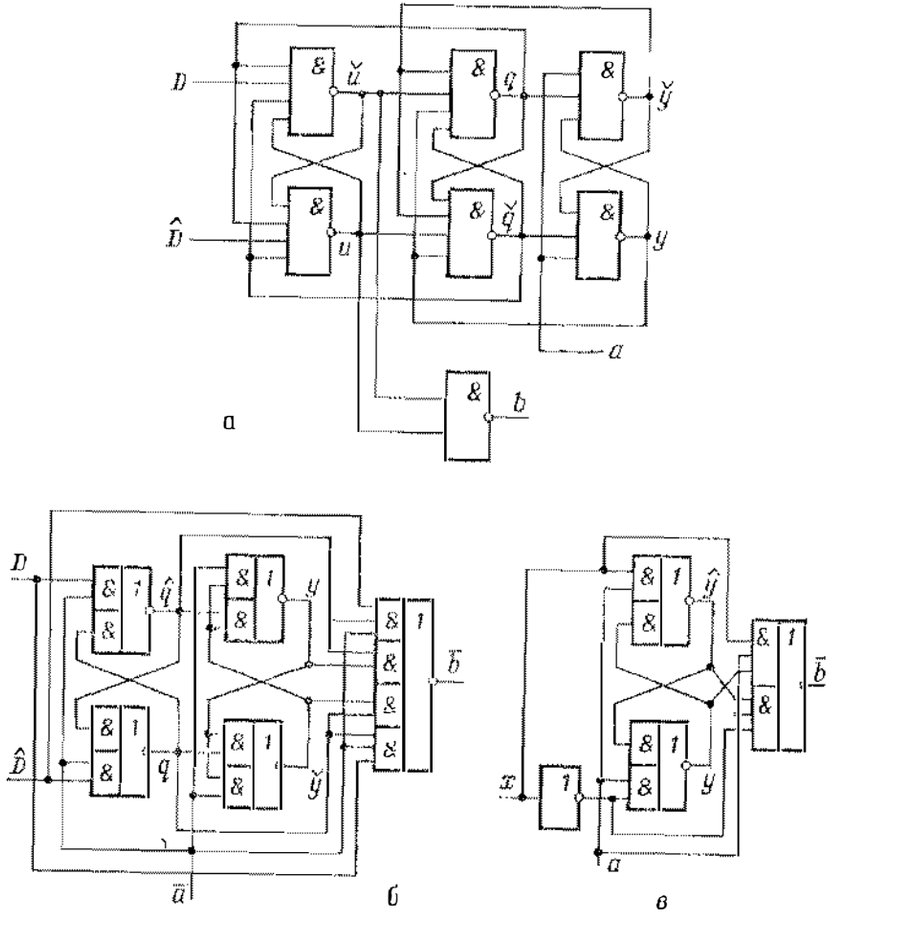


Рис. 4.11. Апериодические триггеры — задержки: с парафазным входом на элементах И-НЕ (*а*) и И-ИЛИ-НЕ (*б*) и с однофазным входом (*в*)

информацию, записанную в предыдущей рабочей фазе, надо хранить во вспомогательном. В схеме рис. 4.11, *а*, построенной по трехтриггерной схеме, в фазе гашения ($a = 0$) основной триггер y погашен, триггер q находится

в рабочем состоянии, а триггер $\check{u}$ — в гашении. Входы не влияют на поведение схемы, выход индикатора $b=0$. По $a=1$ триггер $\check{y}$ переходит в рабочее состояние, триггер $\check{q}$ гасится и тем самым разрешает запись в триггер u информации со входов. После этого $b=1$. Таким образом, рабочая фаза длится $4T$, как и фаза гашения. Действительно, после $a=0$ сначала гасится триггер y, затем триггер $\check{q}$ переходит в рабочее состояние, а потом гасится триггер u, после чего $b=0$. *Цикл работы схемы рис.* 4.11, *а равен* $8T$. Заметим, что пока погашен триггер $\check{u}$, допустимы любые изменения парафазного сигнала $\hat{D}$. На длительность пребывания этого сигнала в транзитном состоянии никаких ограничений не накладывается. Более того, в этом случае обратные связи в триггере $\check{u}$ могут быть исключены.

Если использовать триггеры без гашения, то можно построить *D-триггер на двух запоминающих элементах.* Такая схема приведена на рис. 4.11, *б*. В фазе записи ($\bar{a}=0$) вспомогательный триггер $\hat{q}$ отсечен от парафазного входа $\hat{D}$, и в основной триггер y записывается информация из вспомогательного. Основной триггер и та часть индикатора, на которую поступают его выходы, построен по схеме рис. 4.10, *а*. По окончании записи в основной триггер $b=1$. Длительность фазы записи — T или $3T$. В фазе хранения основной триггер «отсекается» от вспомогательного, и запись информации в последний с внешних входов разрешается. Запись во вспомогательный триггер (построенный по схеме рис. 4.10, *б*) завершается переходом 1-0 выхода индикатора. Время переходного процесса в этой фазе также равно T или $3T$, так что *цикл работы лежит в пределах от* $2T$ *до* $6T$ (против $8T$ для триггера рис. 4.11, *а*).

D-триггеры рис. 4.11, *а* и *б* имеют парафазные входные сигналы. В ряде случаев полезно иметь конструкцию *с однофазным информационным сигналом.* Одна из таких конструкций приведена на рис. 4.11, *в*. Она представляет собой композицию триггера типа рис. 4.10, *б* (в котором для формирования второго установочного входа используется инвертор) и индикатора, собственная функция которого есть сумма по модулю два: $\overline{b}=\overline{xy\vee\bar{x}\check{y}}$. Поэтому такая конструкция может быть использована для фиксации момента совпадения состояния триггера $\hat{y}$ со значением входного однофазного сигнала x в рабочей фазе: при $x=\hat{y}$ $\overline{d}=0$, при $x=\hat{y}$ $\overline{d}=1$. В нерабочей фазе ($a=0$) информация хранится в триггере, можно менять входные сигналы. При использовании в качестве триггера $\hat{y}$ схемы рис. 4.10, *а* собственная функция индикатора имеет вид $\overline{b}=x\hat{y}\vee\bar{x}\hat{y}$.

***T*-триггеры.** В *T*-триггере, принимающем в рабочей фазе состояние, противоположное тому, в котором он находился в предыдущей рабочей фазе, фазовый сигнал играет роль счетного входа. Вообще говоря, *T-триггер* может быть построен непосредственно на основе *D*-триггера, например типа рис. 4.11, *б*, если выходные сигналы основного триггера использовать в качестве входов схемы, а именно $D=\check{y}$ и $\hat{D}=y$. Однако в силу того, что выходные сигналы триггера используются в качестве входных, реализация *T*-триггера может быть упрощена по сравнению с его *D*-прототипом. Так, в схеме рис. 4.12, *а*, полученной на основе *D*-триггера рис. 4.11, *б*, уравнение для индикатора упрощается следующим образом:

$$b=\overline{\bar{a}Dq\vee\bar{a}\hat{D}\hat{q}\vee\check{y}q\vee y\hat{q}}=\overline{\check{y}q\vee y\hat{q}}.$$

Длительность обеих фаз одинакова по $3T$, *цикл* $6T$.

Рассмотрим схему *T*-триггера, приведенную на рис. 4.12, *б*. Пусть фазовый сигнал $a=0$, тогда триггер $\check{y}$ погашен, и его выходы не влияют ни на состояния вентилей p, $\bar{p}$, ни на состояние триггера q: $p=\bar{q}=q$, $\bar{p}=\bar{q}=q$ и $\bar{b}=1$. При переходе 0-1 фазового сигнала в триггер $\check{y}$ производится запись информации с вентилей, и после ее окончания $\check{y}=p$ и $\bar{\check{y}}=\bar{p}$, что в свою очередь приводит к изменению состояния триггера $\check{q}$. После перехода триггера q в такое состояние, что $q=y$ и $\bar{q}=\bar{y}$, изменится состояние того из вентилей, выход которого был равен 0. В результате $p=\bar{p}=1$ и переход 1-0 выхода индикатора завершает переходный процесс. Таким образом, при переходе в рабочую фазу триггер $\check{y}$ попадает в состояние, противоположное тому, в котором он был в предыдущей рабочей фазе. Длительность рабочей фазы составляет $5T$.

При $a=0$ гасится триггер y, что приводится к изменению состояния одного из вентилей. В результате $b=1$, чем и завершается фаза. Длительность ее равна $3T$, а *полный цикл работы триггера* $8T$.

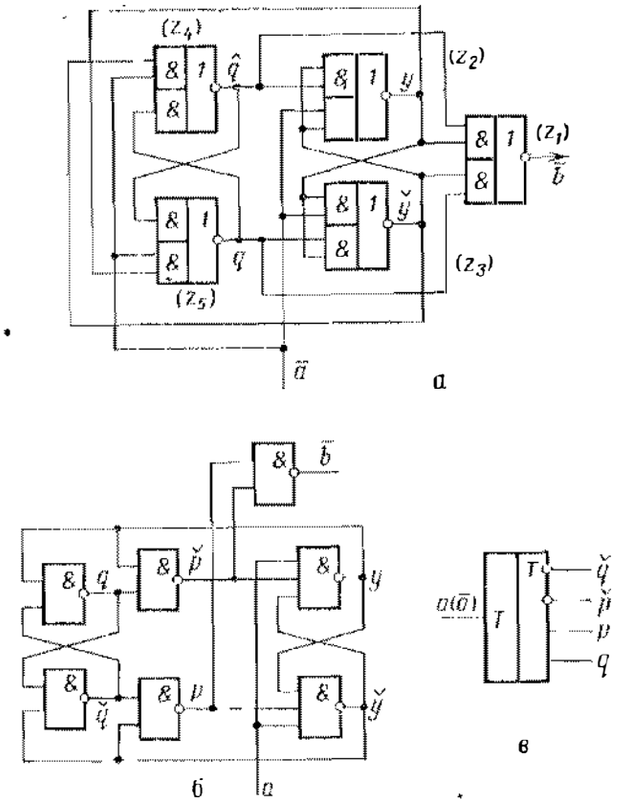


Рис. 4.12. Апериодические счетные триггеры: на элементах И-ИЛИ-НЕ (*а*) и элементах И-НЕ (гарвардский) (*б*); условное обозначение (*в*)

В заключение параграфа следует отметить, что описанные конструкции апериодических триггеров относятся к классу линейных циклических схем и могут быть формально синтезированы методами, изложенными в § 5.5. В примере 5.8 будет продемонстрирована процедура синтеза триггера рис. 4.12, *а*.

§ 4.5. Стандартные апериодические реализации конечных автоматов

В этом параграфе будут описаны некоторые универсальные, канонические подходы к построению апериодических реализаций конечных автоматов. Такие подходы позволяют значительно упростить процедуру синтеза, не исследуя особенностей данного автомата, но платой за это упрощение являются бо́льшие аппаратурные затраты. В результате синтеза получаются схемы, которые обычно называют *стандартными реализациями*. В описанных конструкциях, как правило, используется неизбыточное кодирование состояний автомата. Для автомата с s внутренними состояниями требуется $s_0=]\log_2 s[$ элементов памяти (апериодических). Тип элемента памяти в основном и определяет вид стандартной реализации. Структурно все они представляют собой композицию комбинационной схемы, регистра из s_0 элементов памяти с поразрядными индикаторами и общего индикатора автомата. Построение комбинационных схем и общего индикатора осуществляется по материалам п. 4.3.2 и § 4.2 соответственно, но требует особого внимания из-за необходимости учета типов транзитных состояний логики и триггеров автомата.

4.5.1. Реализация на базе триггеров-задержек. При использовании в качестве элементов памяти с разрядными индикаторами D-триггеров структура реализации имеет вид, показанный на рис. 4.13. Она представляет собой

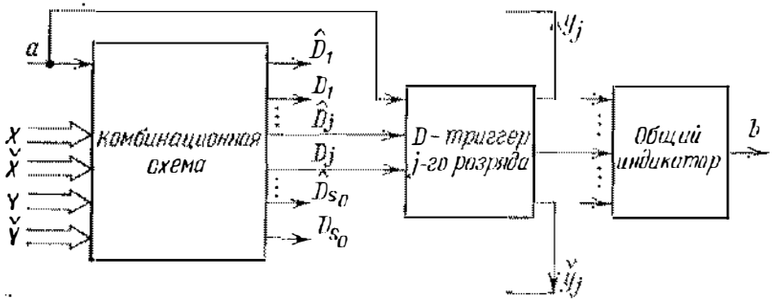


Рис. 4.13. Структура апериодической реализации конечного автомата на базе триггеров-задержек

композицию комбинационной схемы с парафазными входами и выходами, регистра из D-триггеров и общего индикатора. Фазовый сигнал в общем случае подается как

на комбинационную схему, так и на регистр. Схема рис. 4.13 может быть уточнена в зависимости от способа реализации логики и триггеров. Одно из таких уточнений дается в нижеследующем примере.

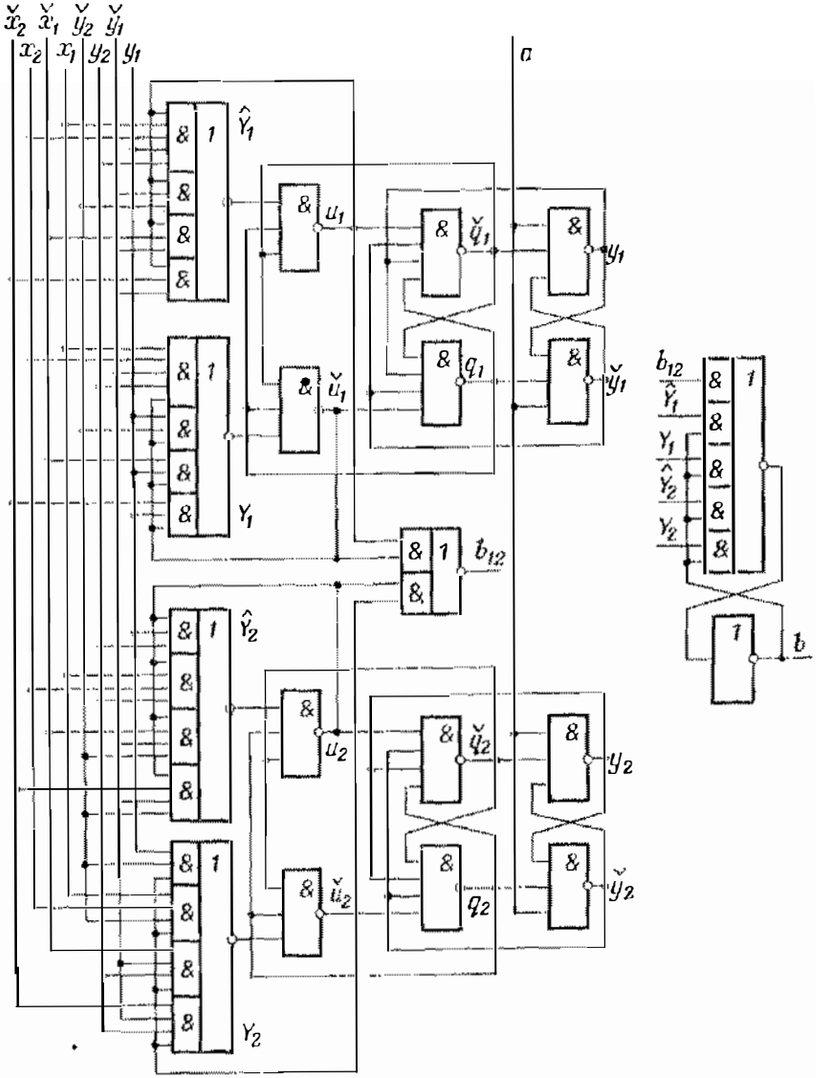


Рис. 4.14. Пример реализации автомата на базе D-триггеров

Пример 4.2. Пусть конечный автомат задан системой уравнений

$$Y_1 = x_1x_2y_1y_2 \vee \bar{y}_1\bar{y}_2 \vee \bar{x}_1\bar{y}_1 \vee \bar{x}_2\bar{y}_1,$$
$$Y_2 = y_1y_2 \vee x_1x_2y_2 \vee \bar{x}_1\bar{y}_1\bar{y}_2 \vee \bar{x}_2\bar{y}_1\bar{y}_2. \qquad (4.16)$$

В соответствии с обозначениями рис. 4.13 $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$. Реализация комбинационной схемы требует также нахождения инверсий этих выражений, которые имеют вид

$$\bar{Y}_1 = x_1x_2\bar{y}_1y_2 \vee y_1\bar{y}_2 \vee \bar{x}_1y_1 \vee \bar{x}_2y_1,$$
$$\bar{Y}_2 = y_1\bar{y}_2 \vee x_1x_2\bar{y}_2 \vee \bar{x}_1\bar{y}_1y_2 \vee \bar{x}_2\bar{y}_1y_2. \qquad (4.17)$$

Пусть используются D-триггеры типа, показанного рис. 4.11, *а*. Тогда стандартная реализация может быть построена по типу рис. 4.13 так, как показано на рис. 4.14. Логика реализована по принципу коллективной ответственности.

Исходно $a = 0$, код старого состояния хранится в триггерах $\check{q}_1$ и $\check{q}_2$. Переходом 0-1 начинается рабочая фаза, и в результате переключений триггеров (протекающих в том порядке, о котором говорилось при анализе триггера на рис. 4.11, *а*) на выходах Y_1, $\hat{Y}_1$, Y_2, $\hat{Y}_2$ устанавливается код нового состояния; в конце фазы $b_{12} = 1$, $b = 1$.

При переходе 1-0 сигнала a начинается фаза гашения, в ходе которой код нового состояния переписывается в триггеры $\check{q}_1$, $\check{q}_2$, выходы Y_1, $\hat{Y}_1$, Y_2, $\hat{Y}_2$ устанавливаются в состояние гашения (нулевое!). В конце фазы $b_{12} = 0$, $b = 0$.

Внутреннее устойчивое состояние автомата при реализации типа, показанного рис. 4.14, представляется на-

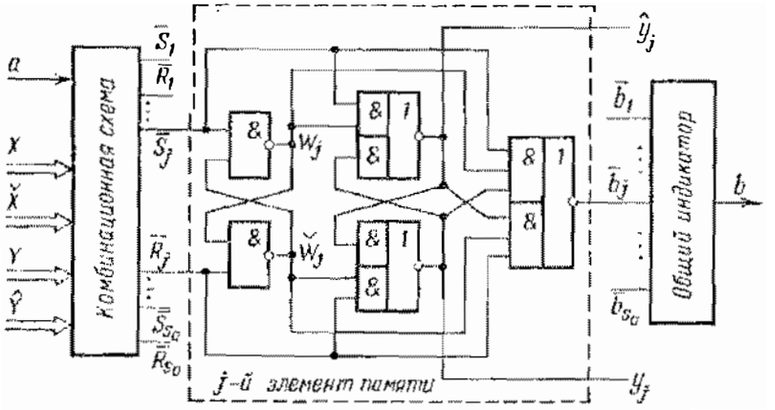


Рис. 4.15. Структура апериодической реализации конечного автомата на базе триггеров с раздельными входами

бором равновесного кода с $4s_0$ компонентами: на j-й разряд, $1 \leqslant j \leqslant s_0$, приходится четыре компоненты — y_j, $\check{y}_j$, q_j, $\check{q}_j$. Рабочим состояниям (из множества Z_1) соответствуют те из наборов, в которых при $a = 1$ $y_j \neq \check{y}_j$, $q_j = \check{q}_j = 1$, нерабочим (из множества Z_2) — те, в которых

при $a=0$ $y_j=\breve{y}_j=1$, $q_j\neq\breve{q}_j$. Заметим, что в данном случае $Z_1\cap Z_2=\varnothing$.

4.5.2. Реализация на базе триггеров с раздельными входами. Если в качестве элемента памяти (с индикатором) использовать некоторую модификацию схемы рис. 4.10, *б*, в которой наряду с основным триггером $\hat{y}_j$ используется вспомогательный w_j, то стандартная реализация имеет вид, показанный на рис. 4.15.

Работа элемента памяти, как и всего автомата, состоит из двух фаз. В рабочей фазе, когда комбинационная схема при $a=1$ выработала значения функций возбуждения S_j и R_j, соответствующие новому состоянию автомата, триггер w_j либо не переключается, если значения функций возбуждения соответствуют его состоянию, либо переключается в противном случае. В обоих случаях срабатывание индикатора ($\bar{b}_j=0$) завершает фазу.

В нерабочей фазе, когда $a=\bar{S}_j=\bar{R}_j=1$, код состояния переписывается из триггера w_j в триггер y_j, так что в конце фазы $y_j=w_j$, $\hat{y}_j=\breve{w}_j$, $\bar{b}_j=1$.

Внутреннее устойчивое состояние автомата представляется здесь набором парафазного кода с $4s_0$ компонентами: на j-й разряд, $1\leqslant j\leqslant s_0$, приходится четыре компоненты — w_j, $\breve{w}_j$, y_j, $\hat{y}_j$. Рабочим состояниям (из множества Z_1) соответствуют те из наборов, в которых при $S_j\neq R_j$ имеют место $w_j\neq\breve{w}_j$ и $y_j\neq\hat{y}_j$, нерабочим (из множества Z_2) — те, в которых при $\bar{S}_j=\bar{R}_j=1$ $y_j=w_j$, $\hat{y}_j=\breve{w}_j$. Все остальные состояния являются промежуточными. Заметим, что в данном случае $Z_1\cap Z_2\neq\varnothing$.

Таблица 4.1

y_j	Y_j	S_j	R_j
0	0	0	*
1	0	0	1
0	1	1	0
1	1	*	0

Сделаем следующее замечание. Для нахождения выражений для функций возбуждения S_j и R_j по автоматным уравнениям можно воспользоваться тем, что поведение RS-триггеров описывается системой уравнений $Y_j=S_j\vee\bar{R}_jy_j$, $S_jR_j=0$, где Y_j и y_j — новое и старое состояние триггера соответственно. Непосредственно отсюда получаем табл. 4.1, которая указывает путь упрощения функций возбуждения за счет доопределений.

Пример 4.2 (продолжение). Минимизируя таблицы истинности для Y_1 и Y_2 из (4.16) с учетом табл. 4.1, получим

$$\bar{S}_1=y_1\vee x_1y_2,\quad \bar{R}_1=\bar{y}_1\vee\bar{x}_2\bar{y}_2\vee x_1x_2y_2,$$
$$\bar{S}_2=y_1\vee y_2\vee x_1x_2,\quad \bar{R}_2=y_1\vee\bar{y}_2\vee x_1x_2.$$

4.5.3. Реализация на базе счетных триггеров. Если в качестве элементов памяти в апериодической реализации конечного автомата использовать T-триггеры, то функции перехода $Y_j=Y_j(x_1,\ \ldots,\ x_n,\ y_1,\ \ldots,\ y_{s_0})$ необходимо представить в виде

$$Y_j=y_j\oplus F_j(x_1,\ \ldots,\ x_n,\ y_1,\ \ldots,\ y_{j-1},\ y_{j+1},\ \ldots,\ y_{s_0}).$$

Переход от функции перехода Y_j к *функции переключения* F_j позволяет использовать структуру типа рис. 4.16. В качестве элементов памяти используются схемы типа,

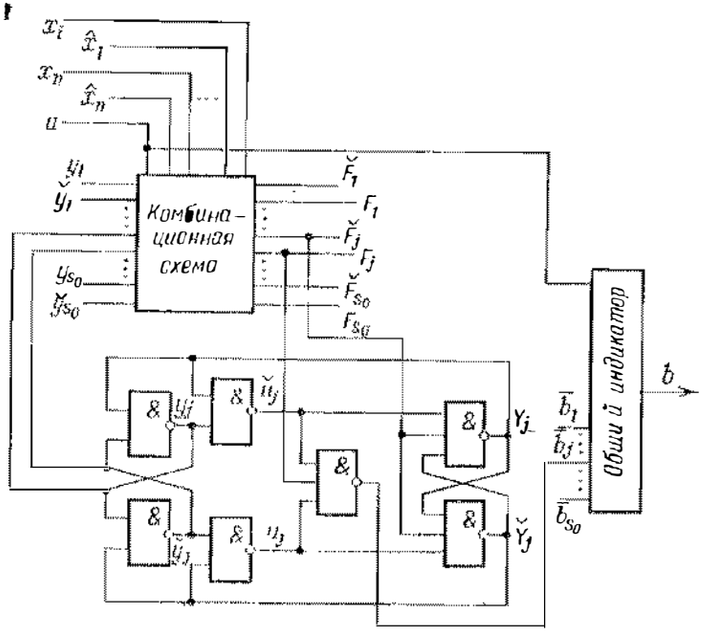


Рис. 4.16. Структура апериодической реализации конечного автомата на базе счетных триггеров

показанного на рис. 4.12, *б*, индикаторы которых имеют по одному дополнительному входу, которые подключены к выходам F_j комбинационной схемы. Счетный вход триггера j-го разряда соединяется с ее выходом $\breve{F}_j$. Таким образом, собственная функция индикатора $\bar{b}_j$ имеет вид $b_j=u_i\breve{u}_jF_j$. Действительно, в конце нерабочей фазы (см. описание работы схемы рис. 4.12, *б* в § 4.4) один из вентилей u_j или $\breve{u}_j$ устанавливается в 0, так что $\bar{b}=1$, и в конце рабочей фазы $u_j=\breve{u}_j=\breve{F}_j=1$ и $\bar{b}=0$.

Комбинационная схема, входами которой являются фазовый сигнал автомата a, парафазные входные переменные x_j, $\tilde{x}_j$, $1 \leqslant j \leqslant n$, парафазные внутренние переменные y_j, $\breve{y}_j$, $1 \leqslant j \leqslant s_0$, снимаемые с триггеров y_j, в которых хранятся коды внутренних переменных в фазе гашения, а выходами — парафазные выходы функций переключения F_j, $\breve{F}_j$, реализуется любым из способов, описанных в п. 4.3.2.

Общий индикатор со входами a, $\tilde{b}_j$, $1 \leqslant j \leqslant s_0$, и выходом b строится обычными методами (§ 4.2).

Пример 4.3. Разряд преобразователя парафазного кода в код в изменениях строится в точном соответствии со стандартной реализацией типа, показанного на рис. 4.16, но имеет вырожденную комбинационную часть. Прямой вход подается непосредственно на счетный вход T-триггера, а инверсный вход $\hat{x}_j$ — на разрядный индикатор $\tilde{b}_j$, который в данном случае имеет собственную функцию $\tilde{b}_j = \hat{x}_j \vee u_i \breve{u}_j$. Код в изменениях образуют в преобразователе сигналы y_j. В фазе гашения $x_j = \hat{x}_j = 0$, $u_j = \breve{u}_j$. Если в рабочей фазе $x_j = 1$, то изменяется состояние T-триггера и выходного сигнала y_j. Если же $x_j = 0$, то состояние T-триггера не изменяется, но $\hat{x}_j = 1$.

В m-разрядном преобразователе собственная функция общего индикатора имеет вид $\tilde{b} = \mathop{\&}\limits_{j=1}^{m} \tilde{b}_j \vee \tilde{b}\left(\bigvee\limits_{j=1}^{m} \tilde{b}\right)$.

Пример 4.2 (продолжение). Представив функции переходов (4.16) в виде $Y_1 = y_1 \oplus \overline{x_1 x_2 y_2}$, $Y_2 = y_2 \oplus \overline{y_1 \vee x_1 x_2}$, можно перейти к стандартной реализации типа рис. 4.16 на T-триггерах. В этом случае один из вариантов комбинационной схемы имеет следующие собственные функции:

$$F_1 = \overline{x_1 x_2 y_2},\ \breve{F}_1 = \overline{\breve{x}_1 \vee \breve{x}_2 \vee \breve{y}_2},\quad F_2 = \overline{y_1 \vee x_1 x_2},$$
$$\breve{F}_2 = \overline{\breve{y}_1 \breve{x}_1 \vee \breve{y}_1 \breve{x}_2},$$

а общий индикатор — $\tilde{b} = \overline{\tilde{b}_1 \tilde{b}_2 a \vee (\tilde{b}_1 \vee \tilde{b}_2 \vee a)\tilde{b}}$.

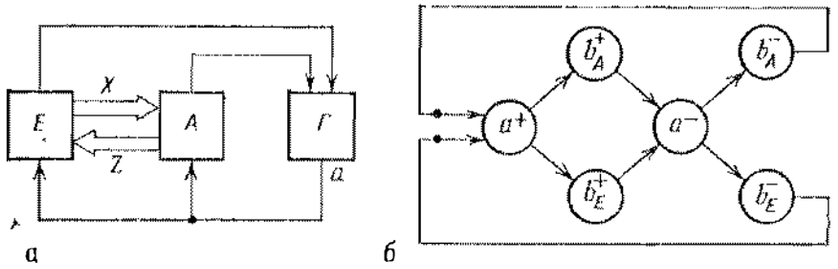
При желании можно уклониться от стандартной реализации, совместив функции счетных триггеров и комбинационной схемы.

Общий индикатор для данного примера строится в соответствии с уравнением

$$b = u_1 \breve{u}_1 u_2 \breve{u}_2 a \vee b(\breve{y}_2 u_1 \breve{u}_1 \vee \breve{x}_1 u_1 \breve{u}_1 \vee \breve{x}_2 u_1 \breve{u}_1 \vee \breve{x}_1 \breve{y}_1 u_2 \breve{u}_2 \vee \breve{x}_2 \breve{y}_1 u_2 \breve{u}_2), \quad (4.18)$$

что требует некоторых пояснений. При переходе 1-0 фазового сигнала a условия завершения переходного процесса в нерабочей фазе зависят от значения функции переключения. Если 1) $x_1 x_2 y_2 = y_1 \vee x_1 x_2 = 0$, то этим условием является $u_1 \breve{u}_1 \vee u_2 \breve{u}_2 = 0$, если 2) $x_1 x_2 y_2 = 0$, $y_1 \vee x_1 x_2 = 1$, то $u_1 \breve{u}_1 = 0$, если 3) $x_1 x_2 y = y_1 \vee x_1 x_2 = 1$, то $a = 0$. Это объясняется тем, что в первом случае изменяются значения сигналов на вентилях обоих триггеров, во втором случае — только в первом триггере, в третьем — состояние автомата сохраняется. Отсюда при $a = 0$ условие завершения фазы есть $\overline{x_1 x_2 y_2} u_1 \breve{u}_1 \vee \overline{y_1 \vee x_1 x_2} u_2 \breve{u}_2$, что эквивалентно заключенному в скобки выражению в (4.18). В рабочей фазе при $a = 1$ условие завершения $u_1 \breve{u}_1 u_2 \breve{u}_2 = 1$. Отсюда и получается (4.18).

Конструкции типа показанной на рис. 4.16 наталкивают на мысль о том, что взаимодействие автомата и



а б

Рис. 4.17. Возможная организация взаимодействия автомата и внешней среды: структура (*а*) и сигнальный граф (*б*)

внешней среды может быть организовано так, как показано на рис. 4.17.

Методика синтеза рассмотренных стандартных реализаций состоит из следующих этапов:

— получение уравнений функций переходов и их инверсий для каждой из переменных, кодирующих внутренние состояния автомата;

— получение уравнений функций возбуждения или переключения;

— выбор на основе анализа полученных уравнений наиболее экономичного способа реализации функций возбуждения (или переключения) с учетом согласования спейсеров внешних входов, спейсеров входов элементов памяти и транзитных состояний триггеров элементов памяти;

— построение общего индикатора.

§ 4.6. Реализация с несколькими управляющими сигналами

Рассмотренная в § 4.2 модель системы автомат — среда конкретизировалась таким образом, что переходные процессы в автомате и среде завершались изменениями значения единственного сигнала; в автомате — сигнала индикации, в среде — фазового сигнала. В этой модели, помимо информационных шин, автомат и среда связываются специально выделенной парой управляющих шин — шиной запроса и шиной ответа (согласованная реализация).

Здесь рассматривается *более общая модель*, в которой, помимо информационных шин, имеется более одной пары специально выделенных управляющих шин. Принята следующая *дисциплина изменения управляющих сигналов*: наборы значений сигналов на управляющих шинах могут изменяться только на соседние; переходный процесс в автомате инициируется изменением значения сигнала на одной из его входных управляющих шин; сигналом окончания переходного процесса в автомате является изменение значения сигнала на одной из его выходных управляющих шин. Принятые допущения определяют следующую динамику работы системы автомат — среда. Выставив на информационных шинах фиксированный набор значений сигналов, среда изменяет значение одного из управляющих сигналов на выходе. Это вызывает переходный процесс в автомате, который состоит из двух последовательных этапов. Первый этап заканчивается установкой значений информационных выходов автомата, второй — изменением значения сигнала на одной из управляющих выходных шин. Далее среда предъявляет автомату новый набор на информационных шинах, затем изменяет значение одного из своих управляющих выходов, инициируя следующий переход автомата, и т. д.

Структурная реализация описанной модели базируется на идее декомпозиции автомата на *управляющий* и *операционный*. Однако эта идея реализована не совсем по В. М. Глушкову. Операционый автомат здесь — это устройство, осуществляющее переработку последовательности входных символов в последовательность выходных символов, но имеющее пару управляющих сигналов и, следовательно, могущее быть реализовано как обычный согласованный автомат (см. п. 4.1.1). Цель управляющего автомата — преобразовать заданную дисциплину смены значений входных управляющих сигналов в необходимую для операционного автомата и выработать заданную дисциплину смены значений выходных управляющих сигналов. В модели согласованной реализации значение управляющего входа соответствовало одной фазе работы операционного автомата. В описываемой модели одному изменению значений сигналов на управляющих шинах должны соответствовать обе фазы его работы. Для индикации перехода входных управляющих сигналов в случае соседней дисциплины может быть использован подход, состоящий в применении сумматора по модулю два выходных переменных. При соседнем переходе входного набора на выходе сумматора осуществляется один из переходов 0-1 или 1-0. В данном случае нужно, чтобы фазовый сигнал операционного автомата осуществлял два перехода 0-1-0 при каждой смене входного набора. Такой переход может быть получен на выходе схемы,

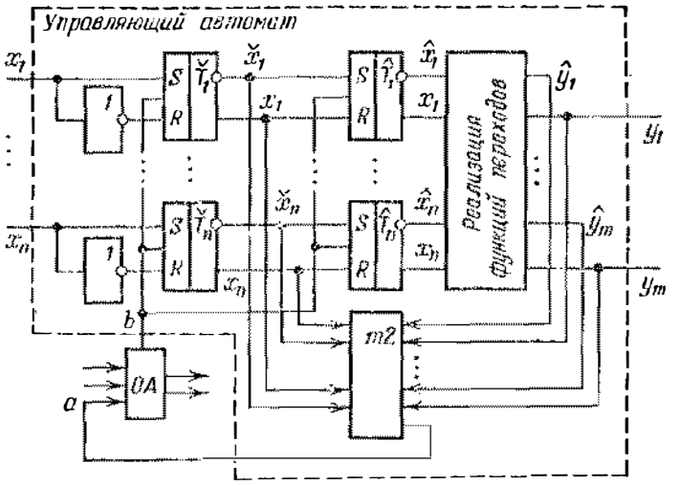


Рис. 4.18. Структура управляющего автомата с несколькими управляющими автоматами. ОА — операционный автомат

реализующей сумму по модулю два значений сигналов на управляющих входах и выходах автомата. Действительно, пусть в устойчивом состоянии автомата сумма по модулю два этих сигналов равна 0. После смены входного набора соседним она станет равной 1, а после очередного перехода (также соседнего) — вновь 0.

Схемная реализация управляющего автомата, естественно, должна предусматривать индикацию окончания обеих фаз операционного автомата и перехода управляющего автомата в следующее состояние. Пример структуры управляющего автомата, удовлетворяющего сформулированным требованиям, приведен на рис. 4.18. На этом рисунке $x_1, x_2, \ldots, x_n$ — управляющие входные шины, $y_1, y_2, \ldots, y_m$ — управляющие выходные шины. Триггеры $\check{T}_i$, $1 \leqslant i \leqslant n$, имеют единичные транзитные состояния, а триггеры $\hat{T}_i$ и триггеры, из которых построена схема реализации функций переходов — нулевые.

Пусть в исходном состоянии $a = x_1 \oplus \ldots \oplus x_n \oplus y_1 \oplus \ldots \oplus y_m = 0$, $a = b = 0$, триггеры $\check{T}_i$ находятся в состояниях, соответствующих входам x_i, $1 \leqslant i \leqslant n$, и операционный автомат находится в фазе гашения. Изменение входного набора (например, переменной x_j) вызывает изменение состояния триггера $\check{T}_j$.

Схема сумматора по модулю два с выходом a может быть построена таким образом, что изменение значения на ее выходе происходит после того, как $\check{T}_j$ придет в устойчивое состояние.

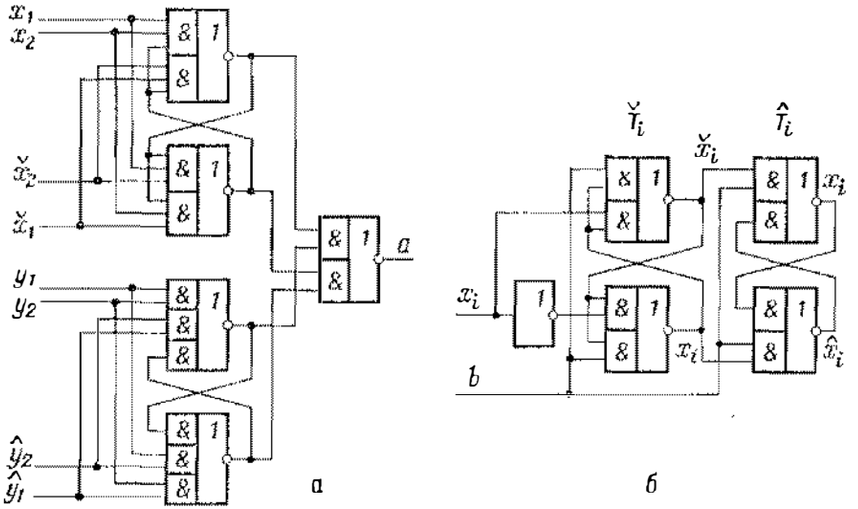


Рис. 4.19. Сумматор по модулю два (*а*) и триггеры (*б*) в реализации с двумя управляющими сигналами

После того как установится значение $a = 1$, операционный автомат переходит в рабочую фазу, и после установки значений информационных выходов $b = 1$. Разрешается перезапись из $\check{T}_j$ в $\hat{T}_j$, окончание перезаписи инициирует переход управляющего автомата в новое состояние. Процесс перехода состоит в изменении

состояния одного триггера (например, y_k). При переходе 0-1 значения выхода y_k индицируется окончание переходного процесса в этом триггере. При переходе 1-0 появление 0 на выходе y_k не означает завершения переходного процесса в этом триггере (последний может находиться в транзитном состоянии).

Схема сумматора по модулю два выполнена таким образом, что на выходе a происходит переход 1-0 только после установки триггера в устойчивое состояние. Далее возможно изменение информационных и управляющих входов автомата. Однако запись новых значений управляющих входов в триггеры $\hat{T}_i$ произойдет только после того, как закончится фаза гашения в операционном автомате ($b = 0$). После этого может начаться следующий переход автомата.

Реализация сумматора по модулю два для случая $n = 2$, $m = 2$ и триггеров $\hat{T}_i$ и $\check{T}_i$, удовлетворяющая поставленным требованиям, показана на рис. 4.19, *а* и *б* соответственно. Подход к реализации функций переходов управляющего автомата будет продемонстрирован ниже, в п. 7.4.3.

§ 4.7. Реализация с прямыми переходами

Здесь будет описана конструкция для случая, когда *входные и выходные сигналы кодируются самосинхронизирующимися кодами с прямыми переходами* (§ 3.6). Соответствующая структура представлена на рис. 4.20, она является композицией регистра входных переменных X с индикатором a и согласованной реализации с фазовым сигналом a и индикатором b. На этом рисунке двойными стрелками изображены информационные, одиночными — управляющие связи.

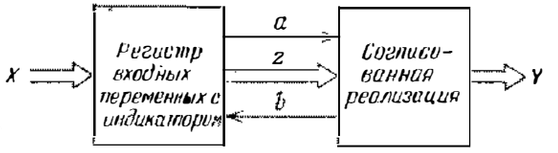


Рис. 4.20. Структура реализации с прямыми переходами

Перед подачей входного набора согласованная реализация находится в рабочей фазе ($a = 1$, $b = 1$), а регистр — в фазе хранения входного набора. После изменения входного набора $a = 0$ реализация переходит в фазу хранения, в которой она сохраняет выходной набор Y. По $b = 0$ разрешается изменение информационных входов Z согласованной реализации и осуществляется запись нового входного набора в регистр, после окончания кото-

рой $a = 1$. Далее осуществляется переход в рабочую фазу: изменяется внутреннее состояние и вырабатывается новый выходной набор. После окончания фазы $b = 1$. Схема пришла в исходное состояние.

Автомат может быть задан произвольным образом, например таблицей переходов (не обязательно нормальной) или автоматными уравнениями. Для простоты будем считать, что заданный автомат является автоматом Мура, т. е. его внутренние состояния одновременно являются выходами. Согласованная реализация осуществляется любым из способов, описанным в § 4.5.

В том случае, когда необходимо стыковать синтезируемую согласованную реализацию с другой, также имеющей структуру типа рис. 4.20, необходимо кодировать состояния автомата кодами с прямыми переходами (см. § 3.6). Непосредственно по таблице переходов для каждого состояния $q_j \in Q$, $1 \leqslant j \leqslant m$, выписываются подмножества $Q(q_j)$, в которые автомат может прийти из состояния q_j. Имея множество состояний Q и подмножества $Q(q_j)$, можно выполнить размещение состояний по методу, изложенному в § 3.6. После этого согласованная реализация строится по схеме на рис. 4.20.

Разряды регистра можно построить, например, по схеме на рис. 4.11, *в*. В этом случае j-й разряд регистра состоит из триггера $\hat{z}_j$ и индикатора $\overline{d}_j$. Записью в регистр управляет фазовый сигнал, обозначенный на рис. 4.11, *в* через a, но в данном случае это выход индикатора b согласованной реализации. При записи $b = 1$, при хранении $b = 0$.

Для описания индикатора окончания перехода $a - b$ образуем сначала n-ку вида $\delta_1\delta_2 \ldots \delta_n$ такую, что

$$\delta_j = \begin{cases} d_j, & \text{если } a_j \neq b_j, \\ 1, & \text{если } a_j = b_j, \end{cases}$$

а затем терм $\delta(a, b) = \bigwedge_{j=1}^{n} \delta_j$. В этот терм, таким образом, входят те d_j, которые соответствуют переменным, значения которых изменяются при переходе (переменным, фигурирующим в терме вариаций (3.3)). Пусть d_j реализуется сумматором по модулю два. Далее определим функцию $f(a, b)$ такую, что $f(a, b) = \omega(a)\delta(a, b)$; $\omega(a)$ определено в (3.1). Легко видеть, что после окончания перехода a-b $f(a, b) = 1$.

Пример 4.4. Пусть $a = 010$, $b = 100$. Тогда $\omega(a) = \bar{x}_1 x_2 \bar{x}_3$, $\delta(a, b) = d_1 d_2$, $f(a, b) = \bar{x}_1 x_2 \bar{x}_3 d_1 d_2$.

Переход 1-0 выхода индикатора a регистра возможен, когда одна из функций $f(a, x)$ равна 1 и $b = 0$, т. е. когда

$$u = \Big(\bigvee_{a \in Z} \bigvee_{x \in Z(a)} f(a, x)\Big)\,\overline{b} = 1,$$

а переход 0-1 — когда $b = 1$ и закончилась запись в регистр (все функции $f(a, x)$ и все d_j равны 0), т. е. когда

$$w = \Big(\bigvee_{a \in Z} \bigvee_{x \in Z(a)} f(a, x) \vee \bar{b} \vee \bigvee_{j=1}^{n} d_j\Big) = 0.$$

Эти два условия позволяют записать уравнение индикатора в виде $a = \bar{w} \vee \bar{w}a$.

§ 4.8. Об определении апериодического автомата

До сих пор речь шла об апериодических реализациях конечного автомата, но понятие апериодического автомата не определялось. При желании получить такое определение сделать это можно, например, следующим образом.

Определение 4.7. Пусть на входах схемы, описываемой моделью автомата Мура, осуществляется допустимый переход a-b, $a \in A$, $b \in B$, инициирующий также допустимый переход c-d внутренних состояний автомата, $c \in C$, $d \in D$. Если для любых $t \in (a, b)$ и $q \in (c, d]$ система функций переходов такова, что $\lambda(t, q) \neq d$, то будем говорить, что *автомат индицирует переход* a-b, а в случае, когда это свойство выполняется для всех допустимых переходов,— что *автомат является индицируемым*.

Смысл определения индицируемости состоит в том, что кодирование входных символов и внутренних состояний автомата Мура должно быть выполнено таким образом (с помощью самосинхронизирующихся кодов), чтобы только по завершении установки нового входного набора было достигнуто новое внутреннее устойчивое состояние d, причем в ходе переходного процесса ни одно из промежуточных внутренних состояний не совпадало ни с одним из возможных устойчивых состояний его.

Автомат Мили можно представить как композицию автомата Мура и выходного преобразователя (комбинационная схема). Поэтому обобщение понятия индицируемого автомата Мили является тривиальным.

В некотором смысле концепция индицируемости является обобщением требований отсутствия функциональных и критических состязаний входных наборов и кодов внутренних состояний. Если к этим требованиям присовокупить еще одно — отсутствие логических состязаний при любых задержках элементов (точнее гипотеза о характере задержек сформулирована в п. 8 § 4.1), то получим

Определение 4.8. Конечный автомат будем называть *апериодическим* (*самосинхронизирующимся*), если: 1) он принадлежит классу индицируемых автоматов и 2) найдется такая его реализация, поведение которой инвариантно к величинам задержек элементов.

Читатель может убедиться в том, что приведенные в этой главе конструкции описываются моделью апериодического автомата. В частности, двухфазная реализация конечного автомата адекватна этой модели.

Итак, в данной главе был представлен материал, позволяющий (как надеются авторы) проектировщику, знакомому с классической теорией автоматов и интегральной схемотехникой, выполнить структурный синтез конечного автомата по его заданию (таблице переходов или автоматным уравнениям), т. е. построить реализующую его схему на интегральных микросхемах. Возможно, читателю покажется, что он и так это умеет делать и авторам нечего было ломать копья. Но эта точка зрения поверхностна: смысл книги в целом и этой главы в частности в том, чтобы научиться синтезировать не синхронные, а асинхронные автоматы, да еще такие, которые исправно функционируют при произвольных задержках элементов. Это ограничение является весьма существенным, и оно определило необходимость ревизии классических методов и разработки качественно новых подходов. Самосинхронизирующаяся (апериодическая) схема, реализующая конечный автомат, строится как композиция комбинационной схемы, триггеров и индикаторов. Выполнение условия индицируемости обеспечивает правильное функционирование автомата с внешней средой, которая должна быть согласована с автоматом (существенно, что в классических постановках вопрос о согласовании среды с автоматом фактически игнорируется). Для этого, в частности, требуются специфические способы кодирования информации и состояний, ранее изложенные в гл. 3.

§ 4.9. Замечания по библиографии

Неформальное определение апериодического автомата и некоторые его структурные модели были предложены в [6, 52, 68], согласованного — в [145]; уточнения сделаны в [144].

Понятие тестора широко используется в технической диагностике, хотя там оно чаще отождествляется с понятием схемы встроенного контроля — см., например, [111].

Модель Г-триггера, носившего в основополагающих работах Д. Маллера название С-элемента [272], используется в ряде работ [120, 175, 274]; реализация его в базисе И-ИЛИ-НЕ предложена в [6].

Понятие индицируемости и подход к анализу асинхронных схем на индицируемость описаны в [60], используется понятие производной булевой функции (см. [58, 137, 300]).

Концепция стандартной реализации принадлежит школе М. А. Гаврилова [3, 70]. Материал о стандартных реализациях комбинационных апериодических схем и апериодических автоматов на *D*- и *T*-триггерах, а также схемы триггеров заимствованы из [6], где также приведен анализ функциональных возможностей реализаций в зависимости от параметров элементов. Один из способов реализации комбинационных схем на элементах типа Г-триггеров подтвержден авторским свидетельством [19].

Способы индикации моментов окончания переходных процессов в схемах базируются на авторском свидетельстве [13] и идее Д. Маллера — см. [120].

Реализация на базе триггеров с раздельными входами описана в [144], а автомата с прямыми переходами — в [123].

Идея использования сумматора по модулю два для случая соседних переходов высказана в [205].

Предварительные соображения по организации структур апериодических автоматов при использовании равновесных и парафазных кодов были изложены в [5, 6, 66, 92, 173, 195, 236]. К сожалению, эти структурные модели не отличаются изяществом. Модель из [195] обладает к тому же существенным недостатком: из-за неудачного кодирования внешняя среда не может быть реализована без использования хотя бы одного встроенного элемента задержки. Переход от асинхронных схем к апериодическим позволяет преодолеть ряд концептуальных недостатков асинхронных автоматов, указанных в [231, 240, 257]; подробнее см. в [144].

Апериодические схемы свободны от состязаний (см. по этому поводу [6, 57, 183, 184, 276]).

Я отлично представляю себе, что такое время, пока не просят объяснить, что это такое, и совершенно перестаю понимать, как только пытаюсь объяснить.

Блаженный Августин

Глава 5. МОДЕЛИРОВАНИЕ УПРАВЛЕНИЯ

В предыдущей главе был выяснен вопрос о том, как можно построить самосинхронизирующуюся схему, отвечающую закону функционирования, заданному моделью конечного автомата. Очевидно, что соответствующие способы задания не являются единственно возможными: ничто не мешает задать алгоритм работы синтезируемого устройства, например, граф-схемой или каким-либо другим способом (системой секвенций, циклограммой, временной диаграммой и пр.). Базовой моделью динамического типа, принятой в этой книге, является модель асинхронного процесса (АП) и порождаемые ею частные модели. Поэтому в этой главе будет показано, как строить самосинхронизирующиеся схемы на основе рассмотренных в гл. 2 языков.

В § 2.3 рассматривалась модель Маллера как одна из интерпретаций АП. Поставим в соответствие уравнениям системы, задающей модель, схему из элементов, собственные функции которых суть эти уравнения. С учетом инициального состояния и принятой гипотезы о характере задержек такая схема моделирует некоторый АП, вследствие чего ее можно назвать *моделирующей схемой*.

Использование моделирующих схем открывает широкие возможности для применения соответствующих методов анализа АП. С другой стороны, поскольку моделирование АП является не самоцелью, а лишь этапом решения более общей задачи — построения устройства, реализующего данный АП, моделирующую схему можно рассматривать как основу реализации такого устройства (точнее, его *управляющей части*).

Положим, что моделируется поведение некоторой системы, представляющей собой композицию из n блоков

(модулей) B_i, причем каждый из этих блоков реализован как согласованный апериодический автомат (см. гл. 4), т. е., помимо информационных входов и выходов, он снабжен входным управляющим сигналом a_i (запрос) и выходным управляющим сигналом b_i (ответ), являющимся выходом индикатора автомата. *При функционировании такой блок фактически реализует функцию элемента задержки со входом a_i и выходом b_i* (значение величины задержки заранее неизвестно и непредсказуемо, известно лишь, что она конечна).

Пусть теперь задан АП, координирующий совместную работу блоков моделируемой системы, причем этот процесс относится к классу управляемых (см. гл. 2), а его ситуации представляют собой двоичные наборы длины n, так что i-я компонента каждой ситуации принимает два значения — 0 или 1. Функционирование АП состоит в смене ситуаций, т. е. в изменении значений компонент в наборах. Условимся i-й компоненте ситуации (набора) ставить в соответствие i-й блок системы, причем будем считать, что если i-я компонента системы находится в состоянии 1, то блок B_i находится в рабочей фазе, а если в состоянии 0, то этот блок находится в нерабочей (пассивной) фазе. При такой интерпретации модели Маллера диаграмма переходов (§ 2.4) моделирует последовательность переключений блоков системы, как моделирует работу системы и схема, реализующая соответствующую систему уравнений. Если моделирующая схема является полумодулярной, т. е. ей соответствует полумодулярная диаграмма переходов, то подключение последовательно с ее i-м элементом блока B_i (с входом a_i и выходом b_i) *не изменяет порядка срабатывания элементов*: функционирование полумодулярной схемы не зависит от величин задержек элементов, к которым можно отнести и задержку, вносимую блоком B_i. Следовательно, композиция моделирующей схемы с блоками, реализующими соответствующие подпроцессы, будет реализовать заданный АП. В ряде случаев такая композиция требует учета семантики асинхронного процесса, например наличия информационных потоков через блок B_i. Однако идея соединения операционных блоков и соответствующих элементов моделирующей схемы сохраняется. В случае, когда АП задан в виде диаграммы переходов, сведение последней к моделирующей схеме может быть осуществлено так, как указывалось в § 2.4.

Далее будут рассмотрены методы построения моделирующих схем применительно к другим двум способам задания АП — сетями Петри и граф-схемами алгоритмов.

§ 5.1. Моделирование сетей Петри

Будем рассматривать АП, задаваемый *безопасной и устойчивой сетью Петри*. Такое ограничение, во-первых, позволяет наиболее просто задавать маркировку сети, а во-вторых, получать полумодулярные моделирующие схемы.

5.1.1. Моделирование по событиям. Пусть *осуществление события сети Петри интерпретируется как срабатывание определенного элемента моделирующей схемы*, т. е. любое изменение значения сигнала на его выходе. Тогда значение входа возбужденного элемента не соответствует его выходу, а устойчивого элемента — соответствует. При этом событие с одним входным условием моделируется элементом задержки, а событие с несколькими входными условиями, не общими с другими событиями — Г-триггером с соответствующим числом входов (рис. 5.1, *г*, *д*, *е*).

Наиболее сложную реализацию имеют фрагменты сети, содержащие события с общими входными условиями. Для уменьшения числа различных типов фрагментов ограничимся моделированием так называемых *простых сетей Петри*, т. е. таких устойчивых и безопасных сетей, в которых каждое событие не может иметь более одного входного условия, общего с другими событиями.

Типовой фрагмент с общим входным условием для таких сетей имеет вид рис. 5.1, *ж*. С учетом всех допустимых для этого фрагмента маркировок, сохраняющих его устойчивость и безопасность, диаграмма переходов реализации этого фрагмента показана на рис. 5.2. Первые три позиции кодов этой диаграммы соответствуют входам реализации, а остальные две — выходам. Хотя на некоторых кодах звездочками помечены первая и третья позиции, переходов, соответствующих их одновременному изменению, нет, поскольку такое изменение противоречит условию устойчивости сети Петри рис. 5.1, *ж*. Обозначив входы схемы через x_1, x_2 и x_3, а выходы — через z_1 и z_2, из этой диаграммы известным методом (§ 2.4), получим

$$z_1 = z_1(x_1 \vee z_2\bar{x}_2 \vee \bar{z}_2x_2) \vee \bar{z}_2x_1x_2 \vee z_2x_1\bar{x}_2 = z_1(x_1 \vee (z_2 \oplus x_2)) \vee (z_2 \oplus x_2)x_1,$$

$$z_2 = z_2(x_3 \vee (z_1 \oplus x_2)) \vee (z_1 \oplus x_2)x_3.$$

Поскольку эти функции не монотонны относительно своих переменных, их реализация сопряжена с некоторыми трудностями: в частности, она требует применения *безынерционных сумматоров по модулю два* (рис. 5.1, *ж*).

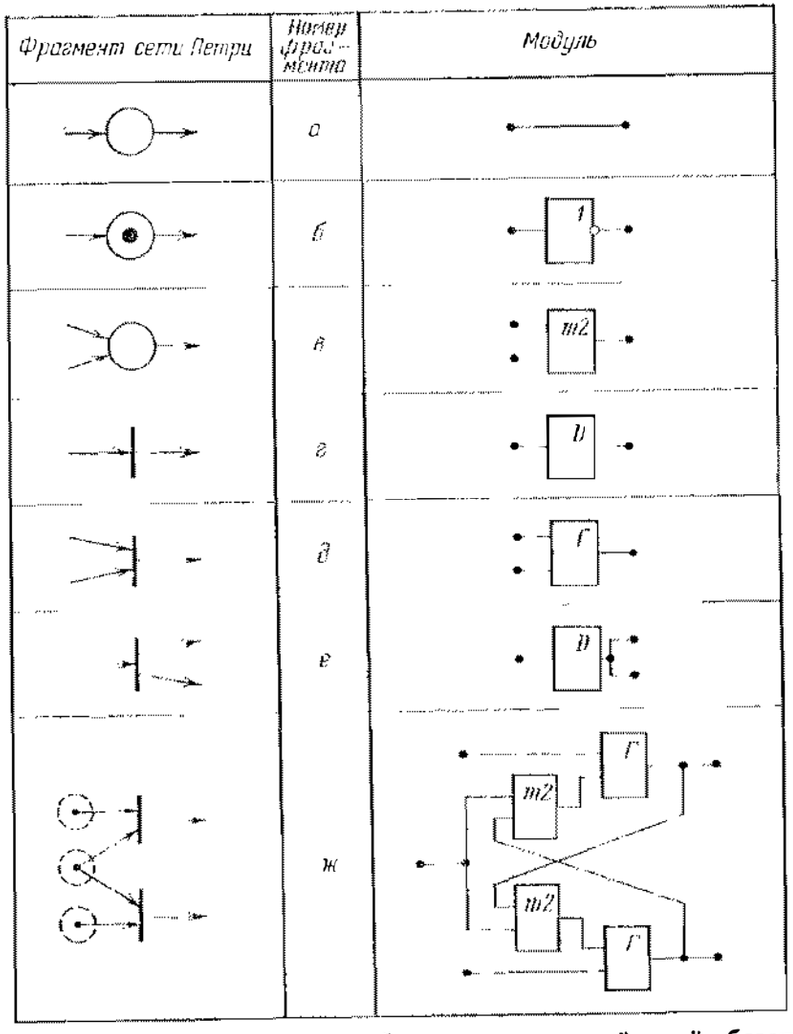


Рис. 5.1. Соответствие между фрагментами устойчивой безопасной сети Петри и реализующими их схемными элементами

Перейдем к реализации условий сетей Петри. Если какое-либо выходное условие некоторого события является входным условием другого события, то в моделиру-

ющей схеме должны быть соединены выход и вход соответствующей этим событиям пары элементов: либо непосредственно, если условие не выполнено, либо через инвертор — в противном случае (см. рис. 5.1, *а* и *б*).

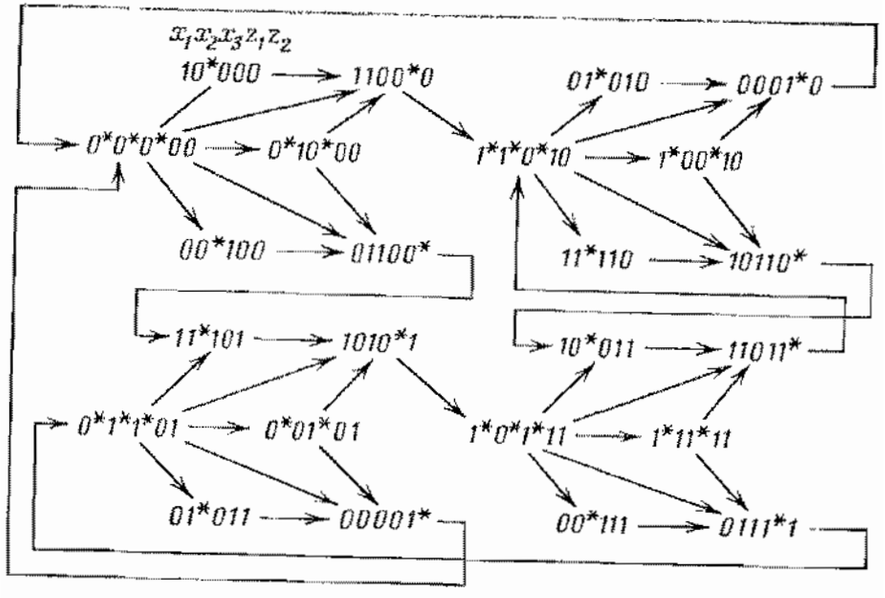


Рис. 5.2. Диаграмма переходов для фрагмента рис. 5.1, *ж*

Если выходное условие является общим для нескольких событий, то оно выполняется в результате наступления любого из них.

Безопасность сети при этом гарантирует наступление только одного из этих событий. В схеме срабатывание любого из соответствующих элементов должно вызвать изменение значения сигнала на входе элемента, реализующего событие, для которого указанное условие является входным. В этом случае условие реализуется сумматором по модулю два (рис. 5.1, *в*).

Поскольку реализация условий рис. 5.1, *а* — *в* играет роль линий связи между элементами, задержки в них должны удовлетворять тем же требованиям, что и задержки в соединительных проводах (§ 4.1). Поэтому в модуле рис. 5.1, *в*, как и в модуле рис. 5.1, *ж*, сумматоры по модулю два должны быть безынерционными, т. е. обладать нулевыми задержками.

Предложенный набор модулей, реализующих фрагменты простых сетей Петри, рассчитан на произвольную, но не нарушающую устойчивость и безопасность дисциплину изменения маркировки входных условий фрагмента. Это позволяет выделять описанные фрагменты в любой простой сети Петри, не анализируя ее работу. Между тем анализ в ряде случаев показывает, что для данного фрагмента сети соблюдается вполне определенная дисциплина смены маркировок. Это приводит к тому, что в соответствующей моделирующей схеме не встречаются некоторые в принципе допустимые для модуля комбинации значений его входов и выходов. Учет этого обстоятельства позволяет упростить реализацию моделирующей схемы. Например, модуль рис. 5.1, *в* иногда можно заменить схемой И, а иногда — ИЛИ; вместо модуля рис. 5.1, *ж* можно использовать либо два Г-триггера с общим входом, либо апериодический счетный триггер.

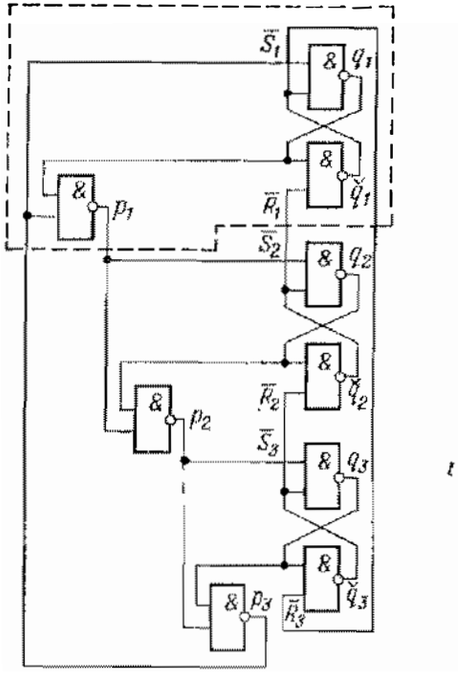


Рис. 5.3. Схема сдвигателя, построенного из элементов Давида (триггер с вентилем), обведенных на рисунке штриховой линией, которая моделирует сеть Петри на рис. 28, *а*

5.1.2. Моделирование по условиям. Рассмотрим построение моделирующих схем, в которых *событию сети Петри соответствует только определенный тип срабатывания модуля, например переход 0-1 значения его выходного сигнала.* При этом в схеме должен обеспечиваться обратный переход 1-0 модуля до того, как снова станет возможным очередной прямой переход, т. е. до того, как событие станет потенциально осуществимым. Метод основан на том, что *каждому условию сети ставится в соответствие триггер моделирующей схемы.* Значению «условие выполнено» соответствует единичное состояние триггера, значение «условие не выполнено» нулевое. (Используя вместо триггера регистр, можно моделировать и не

безопасные ограниченные сети Петри.) Таким образом, наступлению события сети Петри соответствует в схеме установка триггеров выходных условий этого события и сброс триггеров его входных условий.

Описание этого метода начнем с рассмотрения моделирующей схемы (рис. 5.3) для сети Петри рис. 2.8, *а* (репозиция через событие r отсутствует). Эта схема состоит из трех RS-триггеров q_j, $j = 1, 2, 3$, выполненных на элементах И-НЕ, и трех вентилей p_j, причем $p_j = p_{j-1}q_j$, $S_j = p_{j-1}$, $\overline{R}_j = q_{j+1}$, где R_j и S_j — функции возбуждения j-го триггера (суммирование и вычитание индексов производится по модулю три).

Пусть в исходном состоянии в первый триггер записана 1 ($q_1 = 1$, $\overline{q}_1 = 0$), а в два других 0 ($q_2 = q_3 = 0$, $\overline{q}_2 = \overline{q}_3 = 1$). Покажем, что при таком начальном состоянии схема рис. 5.3 является полумодулярной (точнее, последовательной). Очевидно, при заданном начальном состоянии $p_2 = p_3 = 1$, $p_1 = 0$, в схеме возбужден только один элемент — элемент с выходом q_2. Поскольку все остальные элементы находятся в устойчивом состоянии, q_2 не перейдет в устойчивое состояние, пока не изменит значения выхода. После того как это произойдет, возбужденным окажется элемент $\overline{q}_2$ и только он. Следовательно, q_2 (так же, как и q_2) окажется в устойчивом состоянии только после изменения значения выхода. Таким образом, в триггер q_2 будет записана 1 ($q_2 = 1$, $\overline{q}_2 = 0$). При этом возбужденным окажется сначала элемент q_1, а после изменения его выхода — q_1. После того как изменится значение выхода q_1 (т. е. после записи в триггер $\overline{q}_1 = 0$), возбудится элемент p_1, а после установки на его выходе 1 — элемент p_2. Наконец, при $p_3 = 0$ схема окажется в состоянии, аналогичном начальному, с той лишь разницей, что 1 «продвинулась» из триггера q_1 в триггер q_2. Вследствие однородности схемы процесс переписи 1 из q_2 и q_3 и из q_3 в q_1 аналогичен рассмотренному, а следовательно, в течение всей работы схемы каждый ее элемент, будучи возбужден, не переходит в устойчивое состояние, не изменив своего выхода, т. е. схема является полумодулярной. Как уже говорилось, события сети Петри соответствуют в рассматриваемой схеме переписи 1 из триггера входного условия в триггер выходного. Событие считается наступившим, когда триггер, соответствующий выходному условию, установлен в 1, а триггер, соответствующий входному условию, сброшен в 0.

Вообще говоря, при определении функционирования сети Петри предполагалось, что наступление события проявляется в одновременном изменении входных и выходных условий. В приведенной же моделирующей схеме сначала устанавливается триггер выходного условия, потом сбрасывается триггер входного условия, и, таким образом, введенная система соответствий не вполне корректна. Тем не менее порядок наступления событий в сети Петри и «событий» в моделирующей схеме одинаков.

Схема рис. 5.3 реализует чисто последовательный процесс, которому в сети Петри соответствует простейшая связь между условиями: после выполнения единственного входного условия становится потенциально осуществимым событие, в результате наступления которого выполняется единственное выходное условие, а входное условие перестает быть выполненным. При наличии нескольких входных и (или) выходных условий этим условиям ставятся в соответствие RS-триггеры с несколькими установочными входами. Функции их возбуждения зависят от вида сети.

На рис. 5.4, *а — е* приведены типовые связи между условиями, возможные в устойчивых безопасных сетях Петри. Фрагмент рис. 5.4, *а* соответствует только что рассмотренной схеме и пояснений не требует. На рис. 5.4, *б* два события имеют общее входное условие. Из требования безопасности очевидно, что возможными оба эти события одновременно быть не могут. Учет этого обстоятельства позволил упростить функцию установки триггера выходного условия: он устанавливается в 1, если в любой из триггеров входных условий записана 1. После этой установки сигнал сброса подается на оба триггера входных условий, хотя в принципе этот сигнал требуется одному из них. На рис. 5.4, *в* событие имеет два входных условия. Реализация такого события требует установки триггера выходного условия лишь после того, как 1 будет записана в оба триггера входных условий. Функция сброса триггеров входных условий не отличается от предыдущего случая. На рис. 5.4, *г* событие имеет два выходных условия. Триггеры этих условий имеют ту же функцию установки, что и в простейшем случае — рис. 5.4, *а*. Сброс триггера входного условия осуществляется лишь после того, как 1 будет записана в оба триггера выходных условий.

Рис. 5.4, *д* можно считать объединением двух фрагментов — рис. 5.4, *б* и *г*. Соответственно функции установки триггеров выходных условий совпадают с таковыми на первом из них, а функции сброса триггеров входных условий — со вторым. Фрагмент рис. 5.4, *е* совпадает с фрагментом рис. 5.1, *ж*. Функции установки и

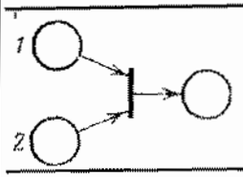
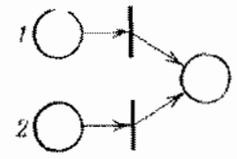
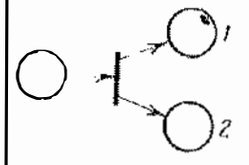
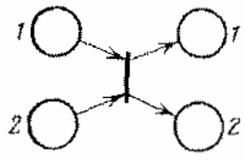
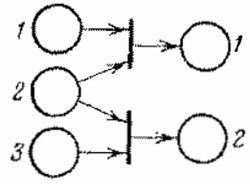
	*а*	$\overline{R}_{вх}=\overline{Q}_{вых}$, $\overline{S}_{вых}=P_{вх}$, $P_{вых}=\overline{P_{вх}Q_{вых}}$
	*б*	$\overline{R}_{вх1}=\overline{R}_{вх2}=\overline{Q}_{вых}$, $\overline{S}_{вых}=P_{вх1}\vee P_{вх2}$, $P_{вых}=\overline{P_{вх1}P_{вх2}Q_{вых}}$
	*в*	$\overline{R}_{вх1}=\overline{R}_{вх2}=\overline{Q}_{вых}$, $\overline{S}_{вых}=P_{вх1}P_{вх2}$, $P_{вых}=\overline{P_{вх1}P_{вх2}Q_{вых}}$
	*г*	$\overline{R}_{вх}=\overline{Q}_{вых1}\vee\overline{Q}_{вых2}$, $\overline{S}_{вых1}=\overline{S}_{вых2}=P_{вх}$, $P_{вых1}=\overline{P_{вх}Q_{вых1}}$, $P_{вых2}=\overline{P_{вх}Q_{вых2}}$
	*д*	$\overline{R}_{вх1}=\overline{R}_{вх2}=\overline{Q}_{вых1}\vee\overline{Q}_{вых2}$, $\overline{S}_{вых1}=\overline{S}_{вых2}=P_{вх1}\vee P_{вх2}$, $P_{вых1}=\overline{P_{вх1}P_{вх2}Q_{вых1}}$, $P_{вых2}=\overline{P_{вх1}P_{вх2}Q_{вых2}}$
	*е*	$\overline{R}_{вх1}=\overline{Q}_{вых1}$, $\overline{R}_{вх2}=\overline{Q}_{вых1}\overline{Q}_{вых2}$, $R_{вх3}=\overline{Q}_{вых2}$, $\overline{S}_{вых1}=P_{вх1}\vee P_{вх2}$, $\overline{S}_{вых2}=P_{вх2}\vee P_{вх3}$, $P_{вых1}=\overline{P_{вх1}P_{вх2}Q_{вых1}}$, $P_{вых2}=\overline{P_{вх2}P_{вх3}Q_{вых}}$

Рис. 5.4. Соответствие между фрагментами устойчивой безопасной сети Петри и их реализации на элементах Давида

сброса триггеров выходных и входных условий для этого фрагмента также легко выписываются.

Рассмотрим *ограничения*, которые накладывает предлагаемый метод синтеза моделирующих схем на вид сетей Петри, являющихся исходным заданием для синтеза.

1. Очевидно, схемы этого типа работоспособны, если не имеют петель, содержащих менее трех триггеров. Действительно, если в петле содержатся только два триггера и 1 переписывается из одного в другой, наступает момент, когда оба триггера находятся в состоянии 1, и невозможно определить, какой из них нужно сбросить. Необходимость последовательного включения не менее трех триггеров означает, что в соответствующей сети Петри не должно быть петель, содержащих менее трех условий (событий).

2. В сети Петри не должно быть событий, имеющих общее с другими событиями выходное условие, и более одного входного условия. Поясним причины этого ограничения на примере фрагмента сети Петри, изображенного на рис. 5.5. Относительно приведенной исходной маркировки (условия *1* и *2* выполнены, а *3* и *4* — не выполнены) этот фрагмент устойчив и безопасен. Возможным является событие, отмеченное на рисунке звездочкой — в результате его наступления сеть переходит к маркировке, при которой выполнены условия *2* и *4*,

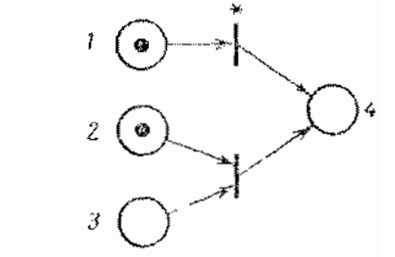


Рис. 5.5. Иллюстрация ограничений на применимость моделирования по условиям

а не выполнены — *1* и *3*. При реализации этого фрагмента по аналогии с фрагментом рис. 5.4, *б* установка в 1 триггера выходного условия вызовет сброс в 0 всех триггеров входных условий тех событий, которым соответствует это выходное условие, в частности и триггера входного условия *2*. Таким образом, схема будет неверно отражать связь условий в сети Петри.

Оба эти ограничения всегда можно обойти, «вставляя» в сеть Петри фиктивные условия и события, не на-

рушающие порядка наступления событий (выполнения условий) исходной сети.

В заключение этого раздела отметим, что хотя рассмотренные здесь методы позволяют синтезировать моделирующие схемы практически по любой устойчивой и безопасной сети Петри, не лишними являются методы, ориентированные на традиционные языки задания поведения схем. Следующий раздел посвящен синтезу моделирующих схем по их заданию с помощью языка граф-схем алгоритмов, который имеет широкое распространение.

§ 5.2. Моделирование параллельных асинхронных блок-схем

Понятие параллельной асинхронной блок-схемы (ПАБС) было введено в § 2.5 при рассмотрении интерпретаций АП. Однако при построении моделирующих схем, представляющих ПАБС, необходима некоторая дополнительная детализация. ПАБС представляют собой формальное задание АП, не учитывающее его семантику. Последняя существенно влияет на характер организации управления и должна быть учтена при построении моделирующей схемы. Если ПАБС описывает, например, систему промышленной автоматики, то изменение информационных переменных вызывает лишь смену состояний объекта управления, и задание процесса может быть осуществлено на уровне управляющих сигналов. Если же ПАБС представляет собой описание вычислительного процесса, то необходимо учитывать как управляющие, так и информационные сигналы. Конкретный пример этого — многократное использование одного и того же операционного блока в различных местах ПАБС. В этом случае операционный блок не может быть инициирован повторно до тех пор, пока выработанную им после очередного срабатывания информацию не воспримут все нуждающиеся в нем блоки. Необходимость учета семантики выражается также и в том, что ПАБС отражает общий характер взаимодействия блоков в системе, но не отражает дисциплину управления блоками. Апериодические блоки, как уже отмечалось выше, имеют двухфазную дисциплину управления: каждый блок может находиться относительно управляющей информации в двух состояниях (фазах) — рабочем и нерабочем. Кроме того, получив сигнал о смене состояния, блок оказывается в переходном режиме. Завершив переходный процесс, блок выдает сигналы исполнения и до следующего сигнала на смену состояния находится в режиме ожидания. При построении моделирующей схемы, очевидно, должна быть реализована именно такая двухфазная дисциплина.

5.2.1. Реализация управления по стандартным фрагментам. Каждый АП, интерпретированный на языке ПАБС, можно свести к композиции типовых ПАБС.

На рис. 5.6 изображены четыре *элементарных ПАБС*. Все они имеют по одной входной дуге α (начало соответствующего алгоритма) и по одной выходной дуге β (конец алгоритма). Для ПАБС можно определить операцию композиции: ПАБС B называется композицией ПАБС C и D, если B может быть получена из C заменой одной из ее вершин типа оператор (см. § 2.5) на ПАБС D. Для однозначности результата композиции необходимо указать, какая вершина заменяется ПАБС. К полученным таким образом ПАБС также можно применить операцию композиции и т. д. Назовем ПАБС, которые могут быть получены из элементарных рекурсивным применением операции композиции, *правильными*. Множество всех правильных ПАБС замкнуто относительно операции композиции, т. е. композиция двух правильных ПАБС всегда является правильной ПАБС.

Внимание к этому классу ПАБС объясняется тем, что значительная часть встречающихся на практике ПАБС принадлежит правильным ПАБС. В тех случаях, когда заданная ПАБС не является правильной (см., например, рис. 5.6, *а*), их всегда можно привести к алгоритмически эквивалентным правильным ПАБС (рис. 5.6, *б*). Приведение осуществляется введением фиктивных вершин типа оператор, дополнительных переменных и условных переходов по ним, т. е. так, как это вынуждены делать в подобных ситуациях программисты, работающие, скажем, на языке фортран, который не допускает использования циклов с двумя входами, зацепленных циклов и т. д.

Однако не следует делать вывод, что вопросы реализации моделирующих схем для неправильных ПАБС не представляют интереса. В тех случаях, когда такая реализация возможна, она позволяет получить более компактные схемы, чем полученные из неправильных ПАБС,

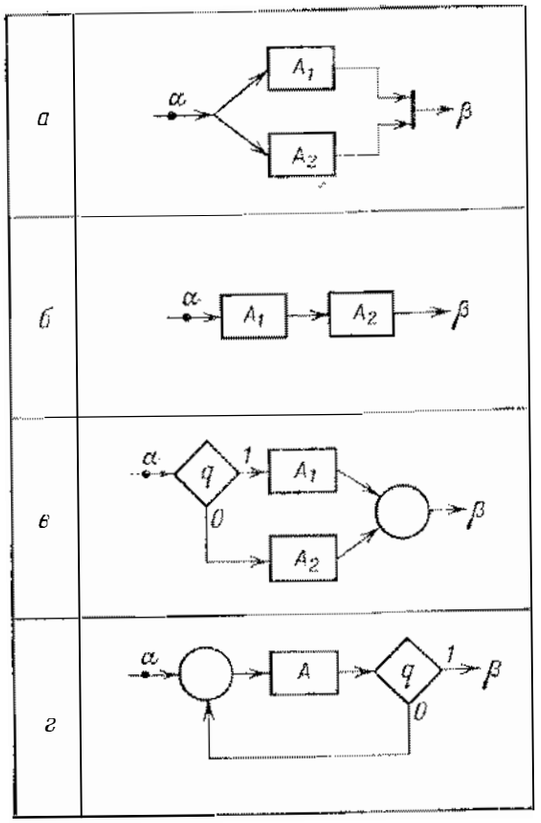


Рис. 5.6. Элементарные параллельные асинхронные блок-схемы

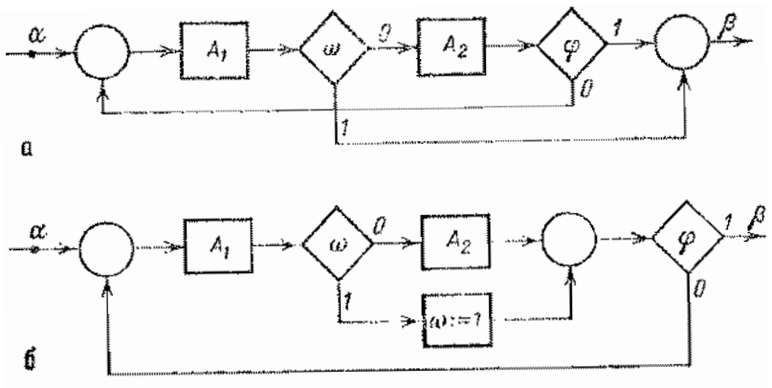


Рис. 5.7. «Неправильная» асинхронная блок-схема (*а*), приведенная к правильной (*б*)

приведенных к правильной форме. Пример такой реализации для ПАБС, показанной на рис. 5.7, *а*, будет рассмотрен ниже.

Пока везде, где не оговорена другая дисциплина, будем считать, что порядок выполнения процессов в обеих фазах совпадает, т. е. после завершения первой фазы

Таблица 5.1

Фрагмент ПАБС	Фрагмент сети Петри
Немаркированная дуга	Рис. 5.1, *а*
Маркированная дуга	Рис. 5.1, *б*
Бесповторный оператор — рис. 2.8, *а*	Рис. 5.1, *г*
Условный переход — рис. 2.8, *б*	Рис. 5.1, *ж*
Сборка — рис. 2.8, *в*	Рис. 5.1, *в*
Бифуркатор — рис. 2.8, *г*	Рис. 5.1, *е*
Синхронизатор — рис. 2.8, *д*	Рис. 5.1, *д*

всего процесса, заданного ПАБС, его вторая фаза выполняется по той же ПАБС. При этом существуют прямые аналогии между ПАБС и сетями Петри на рис. 2.13. В этом случае реализация ПАБС с помощью моделирующих схем, соответствующих их фрагментам, является наиболее простой. Это соответствие устанавливается табл. 5.1.

Сопоставление табл. 5.1 и рис. 5.1 дает *реализацию фрагментов ПАБС соответствующими модулями моделирующей схемы.* Немаркированной (маркированной) дуге ПАБС соответствует провод (инвертор), бифуркатору — разветвление провода, сборке — элемент ИЛИ (в случае, когда невозможно одновременное появление маркировок на обоих входах сборки), синхронизатору — Г-триггер, оператору — элемент задержки, включаемый между входной и выходной дугой операторной вершины. Условной вершине соответствует переключатель, схема которого приведена на рис. 5.8, *а*, а его условное обозначение — на рис. 5.8, *б*. Его входами являются управляющий сигнал a и сигналы логических условий q и $\widehat{q}$. В исходном состоянии $a = 0$. При $a = 1$ поведение переключателя зависит от значения логического условия (т. е. от блока, его вырабатывающего). Возврат в исходное состояние осуществляется после перехода 1-0 сигнала a.

Рассмотренный метод реализации ПАБС пригоден, если ПАБС *бесповторна*, т. е. никакой оператор не встречается в ней более одного раза. В противном случае (по причинам, о которых шла речь в начале этого раздела) для распространения принципа реализации бесповторной ПАБС на произвольный случай можно использовать специальную моделирующую схему, называемую далее моделирующей *схемой повторного вхождения*, на каждое вхождение повторного оператора. Такая схема должна имитировать наличие отдельных экземпляров блоков. На самом деле имеется всего один блок, соответствующий оператору, а управление этим блоком осуществляется совокупностью моделирующих схем повторного вхождения. Таким образом, синтез моделирующей схемы для произвольной ПАБС по фрагментам осуществляется также в соответствии с табл. 5.1 с той лишь разницей, что каждому вхождению повторного оператора соответствует включение в общую моделирующую схему экземпляра схемы повторного вхождения. Синтезу последней посвящен следующий пункт.

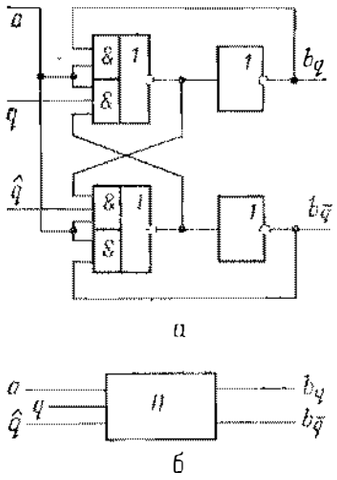
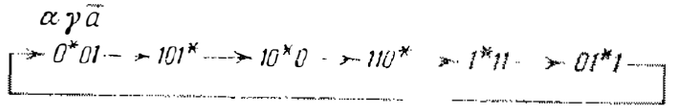


Рис. 5.8. Схема переключателя, реализующего условный переход (*а*), и его обозначение (*б*)

5.2.2. Схема повторного вхождения. *Моделирующая схема j-го вхождения оператора* A_i имеет входами выход α_{ij} общей моделирующей схемы и выход b_i сигнала окончания переходных процессов блока B_i (реализующего оператор A_i), а выходами — вход β_{ij} общей моделирующей схемы и вход $\bar{a}_{ij}$ элемента И-НЕ, выход a_i которого является фазовым сигналом блока B_i. В соответствии со сказанным в предыдущем разделе в рабочей фазе (которая инициируется по $\alpha_{ij}=1$) моделирующая схема j-го вхождения i-го оператора должна последовательно инициировать обе фазы работы блока ($\bar{a}_{ij}=0$ и $\bar{a}_{ij}=1$). Для того чтобы различить состояния схемы, в которых $\alpha_{ij}=1$ и $\bar{a}_{ij}=1$, сначала до выполнения рабочей фазы блока, а потом после выполнения его фазы гашения необходимо ввести внутреннюю переменную γ_{ij}.

В фазе гашения (при $\alpha_{ij}=0$) схема j-го вхождения i-го оператора должна возвратиться в исходное состояние, не меняя состояния блока. Таким образом, схеме соответствует диаграмма переходов вида



Нетрудно убедиться в том, что функции α_{ij}, γ_{ij} и $\bar{a}_{ij}$, полученные по этой диаграмме, не монотонны, и для их реализации требуются три триггера па элементах И-ИЛИ-НЕ.

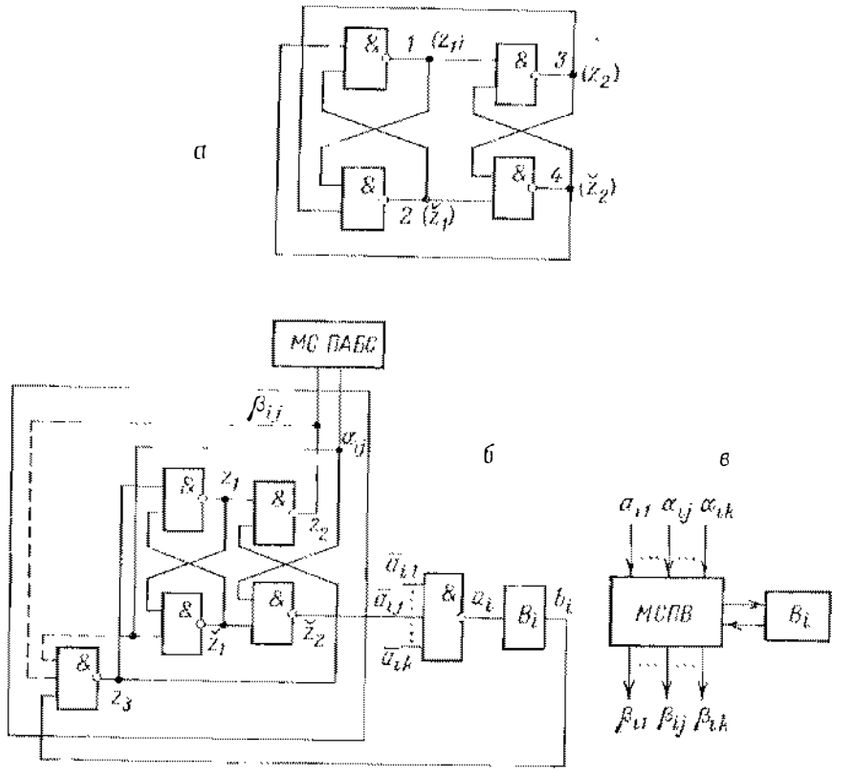


Рис. 5.9. Схема (автомат) повторного вхождения: базовая конструкция (*а*), реализация и способ включения (*б*) (МС — моделирующая схема), условное обозначение (*в*)

Лучший результат получится, если построить схему повторного вхождения на базе схемы, представляющей собой два RS-триггера, соединенные в кольцо (рис. 5.9, *а*).

В ней пара сигналов z_2 и $\check{z}_2$ изменяет значения так же, как и сигналы α_{ij} и $\bar{a}_{ij}$ в приведенной выше диаграмме. Следовательно, реализацию рис. 5.9, *а* можно использовать как основу для построения моделирующей схемы повторного вхождения. Она отвечает следующему требованию: при j-м запуске i-го блока должна срабатывать только j-я схема повторного вхождения, а остальные моделирующие схемы не должны менять своего состояния. Схема рис. 5.9, *б* (как, впрочем, и схема, непосредственно построенная по диаграмме) не удовлетворяет этому требованию. Поэтому введена блокировка сигнала b_i при α_{ij} $z_2 = 0$, что осуществляется введением дополнительных связей, показанных на рис. 5.9, *б* штриховыми линиями.

Заметим, что моделирующая схема повторного вхождения (ее условное обозначение приведено на рис. 5.9, *в*) может быть использована и для управления повторяющимися фрагментами ПАБС.

5.2.3. Схема управления циклом. Будем рассматривать реализацию, которая подобно моделирующей схеме повторного вхождения в рабочей фазе инициирует обе фазы управляемого блока, а в нерабочей фазе (иногда говорят — фазе гашения) переходит в исходное состояние, не изменяя состояния блока. Следовательно, для построения моделирующей схемы управления циклом, как и в предыдущем случае, можно использовать схему рис. 5.9, *а*.

Схема управления циклом должна иметь два режима: условие выполнено и условие не выполнено. В первом режиме схема должна вести себя так же, как схема повторного вхождения (рис. 5.9, *б*). В противном случае схема управления циклом осуществляла бы повторные запуски блока B_i (инициировала обе фазы его работы) до тех пор, пока не будет выполнено условие φ.

Корректно реализованная схема управления циклом показана на рис. 5.10, *а*. Она отличается от схемы рис. 5.9, *б* наличием дополнительного вентиля z_4 и дополнительного входа φ вентиля z_3. Для анализа работы этой схемы рассмотрим три случая.

1. $\varphi = 1$, $\hat{\varphi} = 0$. Очевидно, что $z_4 = 1$, и работа схем рис. 5.10, *а* и 5.9, *б* аналогична.

2. $\varphi = 0$, $\hat{\varphi} = 1$. Поскольку $z_3 = 1$, значения на выходах элементов z_1, z_2 и $\check{z}_1$ остаются фиксированными при

любых значениях α_{ij}, а элементы $\check{z}_2$, a_i и z_4 образуют кольцо. Блок, включенный в это кольцо, циклически меняет фазы работы до тех пор, пока сохраняется условие $\hat{\varphi} = 1$.

3. $\varphi = \hat{\varphi} = 0$. При этом $z_3 = z_4 = 1$, и вся схема находится в устойчивом состоянии до тех пор, пока один из сигналов φ или $\hat{\varphi}$ не примет значения 1.

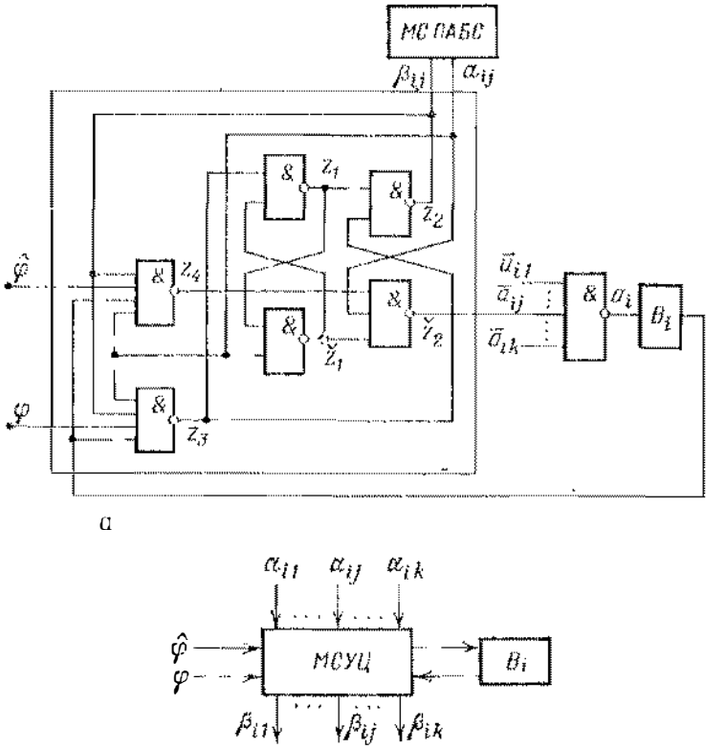


Рис. 5.10. Схема (автомат) управления циклом: реализация и способ включения (*а*), условное обозначение (*б*)

Диаграмма переходов этой схемы изображена на рис. 5.11 и состоит из двух циклов — «длинного», который осуществляется при $\varphi = 1$ ($\hat{\varphi} = 0$), и короткого — при $\varphi = 0$ ($\hat{\varphi} = 1$).

Композиция схемы рис. 5.10, *а* по j вместе с элементом a_i и блоком B_i изображается так, как показано на рис. 5.10, *б*.

Несколько усложнив последнюю моделирующую схему, можно реализовать ПАБС рис. 5.7, *а*, задающую циклическую работу двух блоков B_1 и B_2. Как видно из этой ПАБС, выход из цикла возможен как после последовательного гашения обоих блоков, так и после гаше-

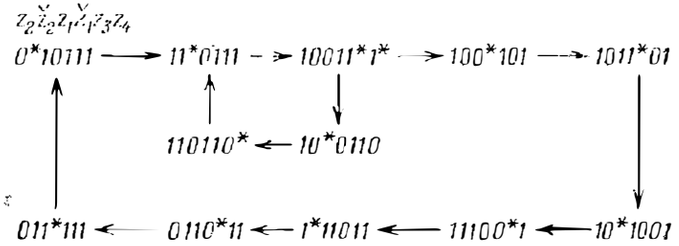


Рис. 5.11. Диаграмма переходов схемы управления циклом

ния только одного из них — B_1. По сравнению со схемой рис. 5.10, *а* схема рис. 5.12 содержит дополнительный вентиль z_5, вырабатывающий инициирующий сигнал для блока B_2. Другое отличие этой схемы состоит в

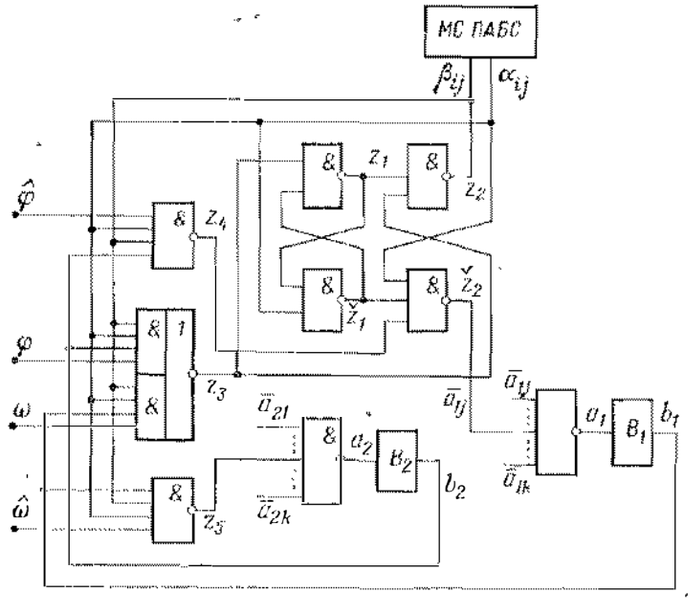


Рис. 5.12. Реализация «неправильного» цикла рис. 5.7, *а*

том, что для обеспечения возможности выхода из цикла не только после срабатывания блока B_1, но и после срабатывания блока B_2 к элементу z_3 добавлен расширитель.

Рассмотренные схемы можно считать более или менее удачными примерами построения некоторых моделирующих схем на основе модификации схемы повторного вхождения. Этот метод, хотя и не является регулярным, обладает тем достоинством, что полученные на его основе схемы экономичны. Следует, однако, заключить, что вопросы выбора исходных моделирующих схем в общем случае остаются открытыми.

5.2.4. Использование арбитра. *Арбитражной* называется ситуация, возникающая при одновременном обращении двух ветвей ПАБС к одному и тому же ресурсу (блоку). В этом случае ресурс должен быть вначале предоставлен одной (все равно какой) ветви, а после завершения использования ресурса в этой ветви — второй ветви.

Пример 5.1. На рис. 5.13, *а* арбитражная ситуация возникает, когда одновременно запрашивается оператор A_3 в левой и правой ветвях ПАБС.

Программные средства разрешения арбитражных ситуаций (*синхропримитивы* типа *семафоров Дейкстры*) вряд ли могут быть рекомендованы для преодоления трудностей, возникающих при реализации этих операций моделирующими схемами. На схемотехническом уровне требование неделимости семафоров (аналогично требованию строгой одновременности переноса маркеров из входных позиций перехода в его выходные позиции в сетях Петри) приводит к аномальному поведению схем — этот вопрос детально рассматривается в гл. 9. Здесь отметим лишь тот факт, что переходный процесс в арбитрах содержит *аномальные режимы*, и длительности переходных процессов существенно (до двух порядков) превосходят длительности в нормальных режимах. Одним из способов борьбы с возникновением аномальных режимов является использование достаточно больших встроенных задержек. Здесь используется, в отличие от гл. 9, этот метод.

На рис. 5.13, *б* показаны моделирующая схема, реализующая ПАБС рис. 5.13, *а* и содержащая арбитр (А), две моделирующие схемы управления циклом и моделирующая схема повторного вхождения. Пояснений в основном здесь требует работа арбитра. При запросе на B_3 в рабочей фазе (реализация оператора A_3) по одной из ветвей (например, сигналом b_1) арбитр выставляет сигнал на выходе α_{31}, не реагируя на сигнал b_2 до полной отработки B_3 по запросу b_1, т. е. пока не установятся сигналы $\alpha_{31} = \beta_{31} = \beta_{11} = 1$. Лишь после этого арбитр готов к запросу по входу b_2.

Если на арбитр одновременно поступают два запроса, то он некоторым случайным образом (для блокировки выходов на время переходных процессов и нужны задержки) выбирает один из входов и далее функционирует так, как было описано выше. После

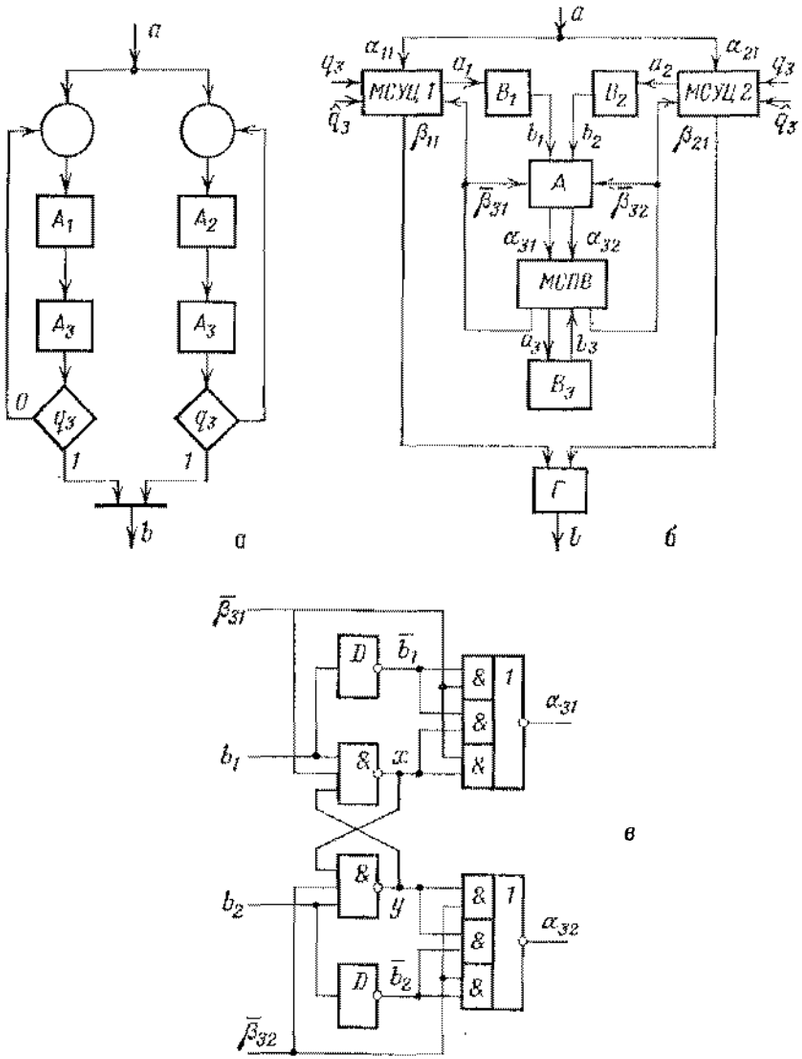


Рис. 5.13. Ситуация, ведущая к возникновению арбитража (*а*), соответствующая блок-схема, в состав которой входит арбитр (*б*), реализация арбитра со встроенными задержками (*в*)

срабатывания B_3 по запросу b_2 при $b = 1$ завершается рабочая фаза, в которой блоки B_1, B_2 и B_3 отработали обе фазы. В нерабочей фазе (после репозиции АП рис. 5.13, *а*) происходит обнуление всех сигналов. Реализация арбитра со встроенными задержками показана на рис. 5.13, *в*. По $b_1 = 1$ и (или) $b_2 = 1$ происходит переключение триггера (x, y), и через время, определяемое задержками D и элемента α_{31} или α_{32}, изменяется значение одного (но не двух!) сигналов α_{31} или α_{32}. После переключения триггера арбитр не реагирует на изменение другого входа (по $x = 0$ — на b_2, по $y = 0$ — на b_1). После срабатывания через моделирующую схему повторного вхождения блока, к которому подключен арбитр, изменяется значение одного из сигналов $\bar{\beta}_{31}$ или $\bar{\beta}_{32}$, что изменяет состояние триггера и разрешает запрос по второму каналу.

Описанные выше схемы достаточны для построения моделирующих схем, представляющих некоторый класс ПАБС. Однако прямой переход от ПАБС к схеме не всегда возможен. Это связано, как уже говорилось, с тем, что ПАБС не учитывает двухфазной дисциплины работы апериодических блоков, представляющих операторы ПАБС. Возникающие здесь трудности и способы их преодоления рассмотрим на примере.

Пример 5.2. Пусть задана ПАБС рис. 5.14, *а*. Рассмотрение ее показывает, что до завершения работы цикла $x_2 — (A_2A_3)$ неизвестно, результат работы какой из остальных ветвей алгоритма понадобится дальше. Если окажется, что $q_3 = 0$, то используется результат оператора A_1, если $q_3 = 1$ — результат оператора A_2. Другая ветвь при этом работает «впрок», и окончание ее работы не индицируется, что может привести к неопределенности в дальнейшей работе моделирующей схемы. ПАБС рис. 5.14, *а* содержит вложенные циклы, что само по себе не является принципиальным затруднением при переходе к реализации, но трудность его связана с тем, что цикл $x_4 — (A_1A_2) — x_5 — (A_3A_4)$ имеет два входа из разных мест по y_1 и y_2 и два выхода из цикла, во-первых, из разных мест и, во-вторых, по различным переменным. Последнее приводит к тому, что во внешнем цикле выход из него происходит из середины тела описанного выше цикла, оформление же цикла по точке выхода приводит к зацеплению циклов, что еще больше усложняет ситуацию, препятствуя использованию стандартных схемных модулей. Однако идея применения последних в моделирующих схемах настолько привлекательна, что следует рассмотреть вопрос о ее использовании для более широкого класса ПАБС. Эту задачу можно решить путем преобразования исходной ПАБС к виду, пригодному для стандартной реализации. Приведение ПАБС рис. 5.14, *а* к алгоритмически эквивалентной ПАБС рис. 5.14, *б*, как нетрудно убедиться, устраняет указанные выше недостатки. Достигнуто это следующим путем.

На рис. 5.15, *а* показан способ преобразования ПАБС цикла при наличии вхождения в тело цикла (вне его заголовка). Это преобразование требует введения дополнительного канала схемы повторного вхождения независимо от того, является размножаемая часть тела цикла одним оператором или сложной ПАБС. На рис. 5.15, *б* показан способ преобразования цикла при наличии выхода из цикла в середине его тела и оформления условия выхода из цикла для случая, когда логические переменные могут быть непосредственно поданы в моделирующую схему управления циклом. В случае, если это затруднительно (когда схема реализована

в виде стандартного модуля), условие выхода из цикла оформляется логической схемой и выходным переключателем.

Вопрос об индикации окончания работы неиспользованных ветвей решается так, как показано на рис. 5.14, *б*. Альтернативное

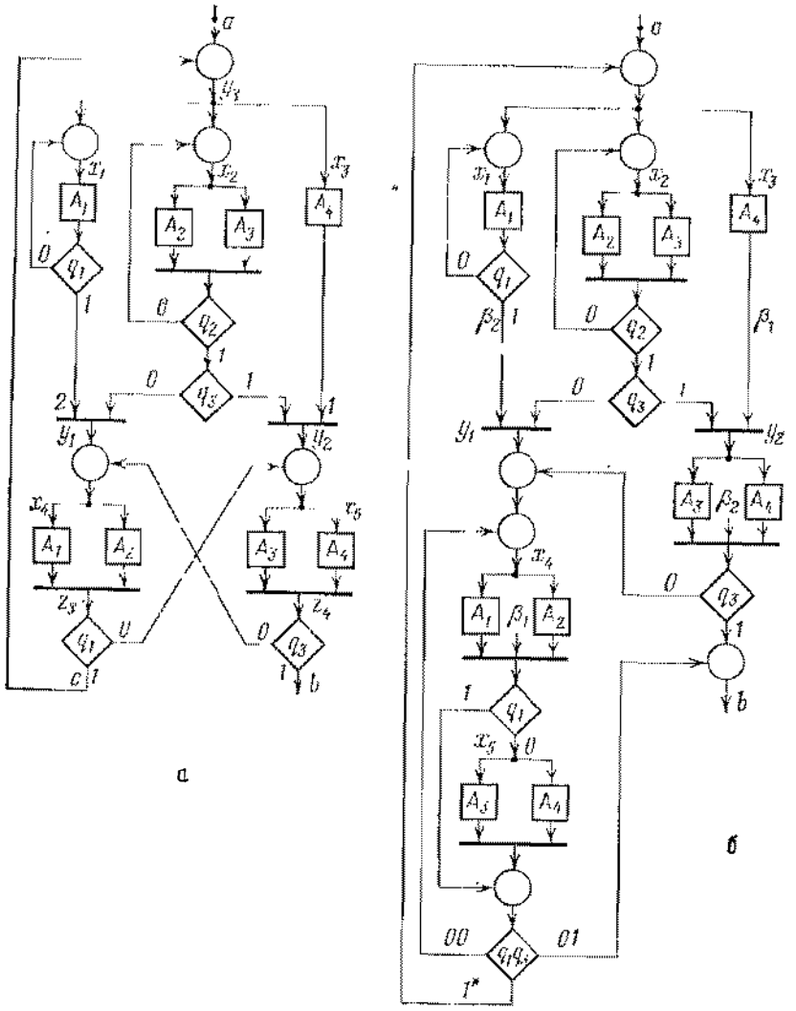


Рис. 5.14. Пример параллельной асинхронной блок-схемы (*а*) и результат ее приведения к виду, удобному для стандартной реализации (*б*)

решение достигается использованием арбитров, но в этом случае неиспользованные ветви должны быть синхронизированы с выходным сигналом моделирующей схемы.

В результате преобразования ПАБС рис. 5.14, *б* допускает непосредственный переход к моделирующей схеме, содержащей 4

схемы управления циклом, 2 двухвходовые (для A_1 и A_2) и 2 трехвходовые (для A_3 и A_4) схемы повторного вхождения, 6 синхронизаторов (Г-триггеров), 4 условных перехода (переключателя) и 3 сборки.

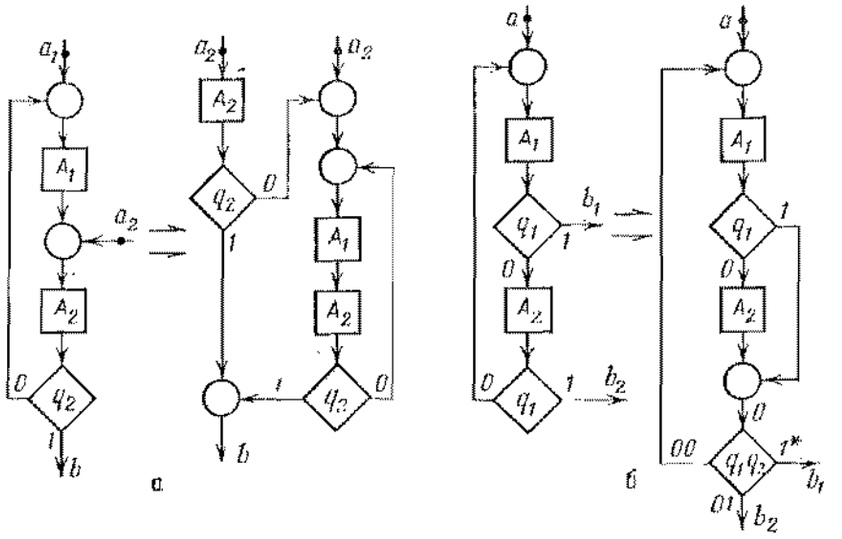


Рис. 5.15. Способы преобразования блок-схемы к виду, удобному для стандартной реализации

5.2.5. Реализация по спусковым функциям. Универсальность метода построения моделирующих схем по стандартным фрагментам ПАБС (с учетом необходимости преобразования ПАБС) позволяет предположить, что этот подход должен давать в результате достаточно сложные реализации. С этой точки зрения более эффективным является подход, использующий спусковые функции.

Спусковой функцией называется предикат, зависящий от состояния процесса и разрешающий активизацию оператора. При реализации спусковой функции в составе моделирующей схемы необходимо учесть двухфазную дисциплину реализующих операторы блоков и обеспечить независимость поведения моделирующих схем как от длительности протекания процессов в этих блоках, так и от величин задержек элементов собственно моделирующих схем.

Составление спусковых функций по ПАБС содержит неформальные моменты, и поэтому экономичность получаемых решений зависит от искусства проектировщика.

Дело в том, что ПАБС по существу определяет лишь условия инициации блока, а условия гашения могут быть определены достаточно произвольно. Следует лишь учитывать, что до момента следующей инициации блок обязательно должен быть погашен. Необходимо иметь в виду, что анализ моделирующих схем на принадлежность классу полумодулярных требует, вообще говоря, применения методов, излагаемых в гл. 8.

Построение моделирующих схем по спусковым функциям рассмотрим на примере.

Пример 5.3. Возвратимся к ПАБС рис. 5.14, *а*, в которой входящие дуги операторов обозначены через x_i (при бифуркации эти обозначения сохраняются). В данном случае $1 \leqslant i \leqslant 5$. Условием инициации оператора является дизъюнкция переменных, обозначающих входящие дуги оператора. Если использовать элементы И-НЕ, то удобнее считать, что инициация оператора происходит по $x_i = 0$. Тогда для рассматриваемой ПАБС имеем

$$a_1 = \overline{x_1 x_4}, \quad a_2 = \overline{x_2 x_4}, \quad a_3 = \overline{x_2 x_5}, \quad a_4 = \overline{x_3 x_5},$$

где через a_i обозначен фазовый сигнал блока B_i, соответствующего оператору A_i. Заметим, что переход 1-0 переменной x_i имеет место тогда и только тогда, когда значения всех других переменных, инициирующих оператор, равны 1.

Далее выделим входные дуги циклов ПАБС. Таких дуг в данном случае три. Этим дугам приписаны переменные y_i (эти обозначения сохраняются при бифуркации). Нижний цикл имеет два выхода, один из них является выходом ПАБС и обозначен b, второй — c. Сигнал ответа блока B_i (оператора A_i) обозначен b_i.

Каждой спусковой функции соответствуют два условия — перехода 1-0 и перехода 0-1. Переход 1-0 x_1 осуществляется в том случае, если инициирована переменная y_3 ($y_3 = 0$) и B_1 находится в нерабочем состоянии. Условие этого перехода можно записать в виде $y_3 \vee b_1 = 0$. Условие $y_3 = 1$ должно сохраняться независимо от переходов других переменных схемы. Если же в цикле, активизированном по $y_3 = 0$, блок B_1 сработал и $q_1 = 0$, то должен повториться цикл работы блока, т. е. имеет место условие $y_3 \vee b_1\bar{q}_1$. Итак $x_1 = y_3 \vee x_1 b_1 \vee b_1\bar{q}_1$; реализация показана на рис. 5.16, *а*. Аналогично строятся и реализуются x_2 и x_3: $x_2 = y_3 \vee x_2 b_2 \vee x_2 b_3 \vee b_2 b_3 \bar{q}$, $x_3 = y_3 \vee x_3 b_4$. Спусковые функции x_4 и x_5 несколько отличаются от рассмотренных ранее в силу того, что они могут управляться как по входу цикла, так и по внутренним связям внутри тела цикла. Пусть x_4 и x_5 реализуются в виде $x_4 = y_1 \vee x_4 b_1 \vee x_4 b_2 \vee b_3 b_4 \bar{q}_3$, $x_5 = y_2 \vee x_5 b_3 \vee x_5 b_4 \vee b_1 b_2 \bar{q}_1$ (по аналогии с $x_1 - x_3$), а указанные особенности учтем при построении y_1 и y_2. Взаимодействие y_1, y_2 и y_3 поясним с помощью рис. 5.16, *б*. В начальном состоянии они имеют значение 111. Инициация ПАБС сигналом $a = 1$ переводит их в состояние 110, что активизирует параллельные ветви верхней части ПАБС. После выполнения условия перехода к нижнему циклу должна активизироваться либо переменная y_1, либо y_2, но этого недостаточно для движения по нижнему циклу — используемые в нем блоки должны

быть переведены в нерабочие состояния. Последнее можно выполнить только после завершения всех процессов, инициированных спусковой функцией y_3. Следовательно, условием активизации y_1 и y_2 должно быть полное завершение переходных процессов в блоках верхней части ПАБС. После перехода в состояния 010 или 100 изменяется значение y_3, осуществляя переход в одно из состояний — 011 или 101, что приводит к гашению всех блоков и разрешает активизацию нижнего цикла по одному из входов. Если

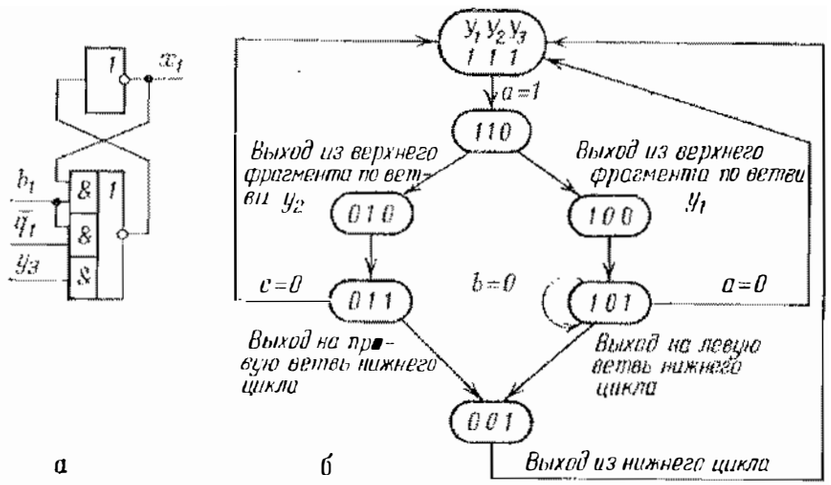


Рис. 5.16. Типичная реализация спусковой функции (*а*) и возможный подход к построению схемы, моделирующей ПАБС (*б*)

вход в нижний цикл был осуществлен по y_1, то после выполнения A_1 и A_2 управление либо передается y_2, либо ПАБС выходит из цикла, формируя функцию c, которая вызывает переход 0-1 y_1. Если управление передается y_2, то либо после выполнения операторов A_3 и A_4 моделирующая схема прекращает работу, формируя сигнал b окончания, либо продолжает функционировать в цикле. Из сказанного следует:

$$y_3 = \overline{a y_1 y_2}, \quad \bar{y}_1 = c\bar{y}_3 b_1 b_2 b_3 b_4 q_1 q_2 \bar{q}_3 \vee \bar{y}_2 b_3 b_4 \bar{q}_3 \vee \bar{y}_1 c \vee \bar{y}_1 a \vee \bar{y}_1 \bar{y}_3,$$
$$\bar{y}_2 = c\bar{y}_3 b_1 b_2 b_3 b_4 q_1 q_2 q_3 \vee \bar{y}_1 b_1 b_2 \bar{q}_1 \vee \bar{y}_2 c \vee \bar{y}_2 a \vee \bar{y}_2 \bar{y}_3.$$

Завершают построение системы функции $c = \overline{y_1 b_1 b_2 q_1}$, $\bar{b} = y_2 b_3 b_4 q_3 \vee \overline{by_1} \vee \overline{by_2} \vee \overline{by_3} \vee \bar{b}a$.

Функции y_3 и c, как и $a_1 - a_4$, реализуются на элементах И-НЕ, остальные функции — на элементах типа Г-триггеров по типу рис. 5.16, *а* (инверсия осуществляется на инверторах). Нетрудно видеть, что в рассмотренном примере реализация ПАБС по спусковым функциям экономичнее реализации по стандартным фрагментам.

§ 5.3. Функциональная полнота и синтез полумодулярных схем

В § 2.4 было показано, что модель Маллера позволяет адекватно задавать поведение асинхронных схем. Последние можно классифицировать в соответствии с видом диаграмм переходов, описывающих происходящие в схемах АП. Выделяются следующие *классы асинхронных схем*, называемых ниже для краткости просто схемами:

— *схемы, не зависящие от скорости*, реализующие управляемые диаграммы,

— *полумодулярные схемы*, реализующие полумодулярные диаграммы,

— *дистрибутивные схемы*, реализующие дистрибутивные диаграммы,

— *последовательные схемы*, реализующие последовательные диаграммы.

Сохраним для этих классов обозначения, использованные для диаграмм переходов (W, U, D, K), а для автономных диаграмм — (W^*, U^*, D^*, K^*).

Здесь будет проведено исследование возможностей реализации классов (моделирующих) схем в базисах И-ИЛИ-НЕ и И-НЕ. Приведенные доказательства являются конструктивными и, по существу, дают методы стандартной реализации рассматриваемых классов схем.

5.3.1. Постановка задачи. Пусть схема, реализующая управляемую диаграмму, задана *системой уравнений* вида

$$z_i = \varphi_i(z_1, \ldots, z_i, \ldots, z_n), \qquad 1 \leqslant i \leqslant n, \qquad (5.1)$$

где φ_i — собственная функция i-го элемента z_i схемы.

Если бы имелись элементы z_i, $i = 1, 2, \ldots, n$, с указанными собственными функциями, то работа схемы, полученной соединением этих элементов в соответствии с ССФ (5.1), не зависела бы от величин задержек элементов и их соотношения.

Пусть теперь имеются только элементы, образующие ССФ $L = \{f_1, f_2, \ldots, f_m\}$. Если хотя бы одна из функций φ_i в (5.1) не принадлежит L, то возникает задача декомпозиции φ_i на составляющие f_j такие, что $f_j \in L$. Не всякая декомпозиция такого вида приводит к схеме, принадлежащей классу схем, не зависящих от скорости.

Определение 5.1. *Декомпозицией* системы уравнений (5.1) на составляющие из L будем называть систему уравнений вида

$$\begin{gathered} z_i = f_{i0}(z_1, \ldots, z_i, \ldots, z_n, z_{n+1}, \ldots, z_{n+k}), \\ z_{n+t} = f_{it}(z_1, \ldots, z_i, \ldots, z_n, z_{n+1}, \ldots, z_{n+k}), \\ 1 \leqslant i \leqslant n, \quad 1 \leqslant t \leqslant k, \end{gathered} \qquad (5.2)$$

где f_{i0}, $f_{it} \in L$, такую, что

$$f_{i0}(z_1, \ldots, z_i, \ldots, z_n, f_{i1}, \ldots, f_{ik}) = \varphi_i(z_1, \ldots, z_i, \ldots, z_n).$$

Определение 5.2. Пусть имеются элементы с собственными функциями $f_{it} \in L$ и схема, полученная их соединением в соответствии с (5.2). Будем называть такую схему *корректной реализацией*, если она принадлежит тому же классу, что исходная схема, описываемая системой (5.1).

Определение 5.3. Систему функций $L = \{f_j\}$, $1 \leqslant j \leqslant m$, будем называть *функционально полной в классе* U^* (D^*, K^*), если для любой схемы из этого класса можно получить ее корректную реализацию на элементах с собственными функциями $f_j \in L$.

Предметом дальнейшего рассмотрения будет нахождение простейших систем функций, полных в классах U^*, D^*, K^*.

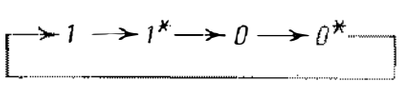
5.3.2. Некоторые свойства полумодулярных схем. Для решения поставленной задачи потребуется изучение локальных свойств элементов схемы класса U^*.

Напомним определение 2.25: считается, что элемент z_i находится в *возбужденном состоянии*, если $z_i \neq \varphi_i(z_1, \ldots, z_n)$, и в *устойчивом состоянии* в противном случае. Иначе говоря, элемент считается возбужденным, если после подачи нового входного набора он еще не сработал, т. е. значение его выхода не соответствует этому набору. Переход из возбужденного состояния в устойчивое, вообще говоря, может осуществляться двумя путями: 1) изменением значения выхода элемента по истечении времени, равного задержке его срабатывания, 2) изменением значения входов возбужденного элемента, в результате чего их набор снова окажется соответствующим значению входов. *Для полумодулярных схем допустим только первый способ снятия возбуждения элемента*, так что имеет место следующее определение полу-

модулярной схемы, эквивалентное теоретико-структурному (доказательство эквивалентности ввиду громоздкости опущено).

Определение 5.4. Схема является *полумодулярной*, если в ней каждый элемент, будучи возбужден, переходит в устойчивое состояние только путем изменения значения своего выхода.

Иначе говоря, в полумодулярной схеме допустимы лишь следующие переходы каждого элемента:



и невозможны переходы типов 1*-1 и 0*-0. При этом неявно предполагается, что реализация элемента свободна от состязаний.

Обозначим через U_n^* класс всех полумодулярных схем, содержащих в точности n элементов. Так, классу U_1^* принадлежит схема, изображенная на рис. 5.17, собственная функция ее единственного элемента имеет вид $z = \bar{z}$.

Теорема 5.1. *В схеме из U_n^*, $n > 1$, собственная функция любого элемента z_i либо изотонна* (см. определение

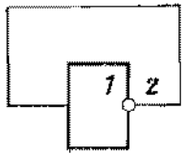


Рис. 5.17. Единственная одноэлементная полумодулярная схема

3.1) *относительно переменной z_i, либо от нее не зависит.*

Доказательство. Предположим противное: собственная функция некоторого элемента z_i схемы из U_n^*, $n > 1$, не изотонна по z_i. Тогда функция z_i может быть представлена в виде

$$z_i = z_i x_i \vee \bar{z}_i y_i,$$

где x_i и y_i — некоторые логические функции, не зависящие от z_i. Возможны четыре комбинации значений x_i и y_i. При $x_i = 1$, $y_i = 1$ z_i принимает значение 1; при $x_i = 1$, $y_i = 0$ — значение z_i; при $x_i = 0$, $y_i = 0$ — значение 0, т. е. на этих наборах значений x_i и y_i собственная функция z_i изотонна по z_i. Таким образом, если в схеме не встречается комбинация значений $x_i = 0$, $y_i = 1$, принятое предположение ведет к противоречию. С другой стороны, при этой последней комбинации элемент z_i возбужден независимо от значения z_i. При этом любое изменение значений x_i и y_i может привести к тому, что z_i окажется в устойчивом состоянии, не изменив значения своего выхода, что противоречит определению 5.4. Следовательно, при $x_i = 0$, $y_i = 1$ обе функции x_i и y_i должны быть неизменны. Последнее возможно либо, если схема, попав в такие состояния, больше из них не выйдет, либо для всех состояний $x_i = 0$ и $y_i = 1$. Но тогда либо состояние схемы не образует один класс эквивалентности, что противоречит условию теоремы, либо собственная функция z_i представима в виде $z_i = \bar{z}_i$, т. е. схема принадлежит U_1^*, что также противоречит условию теоремы ($n > 1$). Принятое предположение приводит к противоречию для любого элемента схемы, так как определение 5.4 относится ко всем ее элементам.

Из теоремы вытекает

Следствие 5.1. *В схеме U_n^*, $n > 1$, собственная функция любого элемента представима в виде*

$$z_i = z_i \bar{w}_i \vee v_i, \qquad w_i v_i = 0, \tag{5.3}$$

где w_i и v_i — логические функции, не зависящие от z_i.

Для *доказательства* (5.3) достаточно положить $v_i = y_i$, $w_i = \bar{x}_i \bar{y}_i$.

5.3.3. Совершенная реализация. Заметим, что в уравнении (5.3) на функции w_i и v_i не накладывалось никаких ограничений, кроме того, что они не зависят от z_i. Естественно, они могут быть и немонотонными, т. е. содержать некоторые переменные z_j как в прямом, так и в инверсном виде. Между тем собственные функции элементов И-ИЛИ-НЕ *антитонны по входным переменным*, в силу чего на одном элементе не может быть реализована ни изотонная, ни немонотонная функции, как это требуется для реализации уравнения (5.3). Отсюда ясно, что в реализации на элементах И-ИЛИ-НЕ должны иметь место элементы с выходами z_i и $\hat{z}_i$ (или $\check{z}_i$), т. е. реализации должны быть *двухканальными*, а в устойчивом состоянии должно выполняться $\hat{z}_i \neq z_i$ или $\check{z}_i \neq z_i$.

Если для получения $\hat{z}_i$ (или $\check{z}_i$) использовать инвертор, то полумодулярность реализации не может быть гарантирована. Действительно, в некоторый момент может иметь место $z_j = \hat{z}_j$ (случай $z_j = \check{z}_j$ далее пока не будем рассматривать в силу двойственности). В этих состояниях истинное значение переменной z_j не определено, и возможно срабатывание элемента, реализующего z_i, после чего может измениться значение z_j, а следовательно, снимутся условия возбуждения инвертора, реализующего $\hat{z}_j$. Из сказанного следует, что если в схеме используются одновременно z_i или $\hat{z}_i$, то реализация в виде $\hat{z}_i = \overline{z_i x_i \vee y_i}$, $z_i = \hat{z}_i$ (на элементе И-ИЛИ-НЕ и инверторе), не всегда корректна.

С точки зрения функционирования элемента z_i все допустимые состояния Z схемы из U^*, т. е. состояния,

в которых схема оказывается при своем функционировании, можно условно разделить на *три класса*. К первому отнесем состояния $R_i \subset Z$, в которых элемент z_i возбужден и $z_i = 1$. Ко второму классу отнесем состояния $S_i \subset Z$, в которых элемент z_i возбужден и $z_i = 0$, к третьему — состояния $T_i \subset Z$, в которых элемент z_i находится в устойчивом состоянии (значение z_i не изменяется).

Теорема 5.2. *Если в схеме из* U_n^*, $n > 1$, *каждая переменная реализована в виде* (5.3), *т. е. в виде*

$$z_j = z_j\bar{w}_j \vee v_j, \qquad w_jv_j = 0,$$

то состояния, в которых $z_i = \hat{z}_i = 0$, *относятся к классу* T_j, $j \neq i$.

Доказательство. Пусть функции w_j и v_j не изотонны и не антитонны по переменной z_j. Тогда, если их представить в виде

$$w_j = w_j(z_1, \ldots, z_i, \hat{z}_i, \ldots, z_n),$$
$$v_j = v_j(z_1, \ldots, z_i, \hat{z}_i, \ldots, z_n),$$

где переменные z_i и $\hat{z}_i$ рассматриваются как разные, то можно считать, что функции w_j и v_j изотонны как по z_i, так и по $\hat{z}_i$. При этом любой набор, на котором $z_i = \hat{z}_i = 0$, меньше, чем пара соответствующих наборов, на которых $z_i = \hat{z}_i$, а значит (по определению изотонной функции), $w_j^0 = w_j(z_1, \ldots, z_i = 0, \hat{z}_i = 0, \ldots, z_n) \leqslant w_j$, $v_j^0 = v_j(z_1, \ldots, z_i = 0, \hat{z}_i = 0, \ldots, z_n) \leqslant v_j$. Иными словами, при переходе от наборов, на которых $z_i \neq \hat{z}_i$, к наборам, на которых $z_i = \hat{z}_i = 0$, значения w_j и v_j могут только уменьшиться. В силу того, что $w_jv_j = 0$, возможны только следующие комбинации значений w_j и v_j: 10, 01 и 00, которым соответствуют устойчивые значения z_j: 0, 1 и z_j. Следовательно, уменьшение значения функций w_j и v_j (переход 10-00 или 01-00) не вызывает изменения z_j, что и требовалось доказать.

Следующая теорема дает способ декомпозиции выражений типа (5.3), обеспечивающий выполнение условия теоремы 5.2.

Теорема 5.3. *Собственная функция любого элемента* z_i *схемы из* U_n^*, $n > 1$, *может быть представлена в виде*

$$z_i = \overline{w_i \vee \overline{z_i \vee v_i}}, \quad w_iv_i = 0, \qquad (5.4)$$

где w_i *и* v_i — *функции, не зависящие от* z_i.

Доказательство. С учетом того, что $w_iv_i = 0$, (5.3) можно преобразовать следующим образом:

$$z_i = z_i\bar{w}_i \vee v_i = z_i\bar{w}_i \vee v_i\bar{w}_i \vee v_iw_i =$$
$$= z_i\bar{w}_i \vee v_i\bar{w}_i = \bar{w}_i(z_i \vee v_i) = \overline{w_i \vee \overline{z_i \vee v_i}}.$$

Очевидно, что (5.4) можно записать в виде

$$z_i = \overline{w_i \vee \hat{z}_i}, \quad \hat{z}_i = \overline{v_i \vee z_i}, \quad w_iv_i = 0. \qquad (5.5)$$

Обратим внимание на то, что (5.5) суть уравнения симметричного RS-триггера (рис. 5.18) с функциями возбуждения $S_i = v_i$ и $R_i = w_i$, $S_iR_i = 0$. Заметим, что при переключении триггера рис. 5.18 возникает состояние, когда $z_i = \hat{z}_i = 0$, а состояния $z_i = \check{z}_i = 1$ не возникает. Тогда, если каждой переменной полумодулярной схемы поставить в соответствие триггер типа рис. 5.18, то такая реализация в силу теоремы 5.3 является корректной. Использование элементов типа И-ИЛИ-НЕ для построения триггеров позволяет реализовать сокращенные д. н. ф. функций возбуждения триггера непосредственно на элементах его плеч.

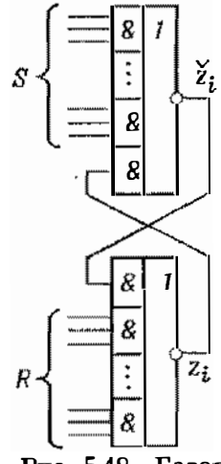


Рис. 5.18. Базовая конструкция для совершенной реализации. Функции возбуждения триггера представляются сокращенными (не минимальными!) д.н.ф.

Теоремы 5.2 и 5.3 показывают возможность корректной реализации собственных функций z_i на элементах типа И-ИЛИ-НЕ, причем состояние, в котором $z_i = \check{z}_i = 0$, возникающее при изменении значения z_i, не может вызвать возбуждение z_j, $j \neq i$. Однако в схемах из U_n^*, $n > 1$, возможно одновременное возбуждение нескольких элементов. Пусть ими будут, например, z_i и z_j. Тогда, очевидно, в корректной реализации переключение одного из элементов не должно снимать возбуждение другого — в нашем случае при $z_i = \check{z}_i$ z_j должен оставаться возбужденным и наоборот. Следующие две леммы показывают, что в корректной реализации схемы из U_n^*, $n > 1$, это условие легко может быть выполнено.

Лемма 5.1. *Если в схеме из* , $n > 1$, *возбужден некоторый элемент* z_i, *то не могут быть одновременно возбуждены все элементы, от значений которых существенно зависит собственная функция* z_i.

Доказательство. Предположим противное: все элементы, от значений которых существенно зависит собственная функция z_i, одновременно возбуждены. Но тогда на входе z_i возможно появление любого набора значений, в частности и такого, который соответ-

ствует значению выхода z_i. Но это значит, что z_i может перейти из возбужденного состояния в устойчивое, что противоречит определению 5.4.

Лемма 5.2. *Пусть в схеме из* U_n^*, $n>1$, *в состоянии* α *возбужден элемент* z_i *и его значение равно* 1 (0) *и пусть при этом элементы* z_j, z_{j+1}, ..., z_{j+m} *находятся в устойчивом состоянии, а остальные элементы возбуждены. Тогда, если представить собственную функцию* z_i *в виде* (5.3), *то сокращенная форма* $w_i(v_i)$ *содержит терм, включающий переменные* z_j, z_{j+1}, ..., z_{j+m}, *и только эти переменные.*

Доказательство. Достаточно рассмотреть построение сокращенной д. н. ф. функции $w_i(v_i)$. Очевидно, что в таблице истинности значение этой функции, соответствующей состоянию α, равно 1. Но это же значение в силу полумодулярности схемы функция имеет и на тех наборах, где возбужденные в состоянии α элементы изменили свои значения. Проведя склеивание всех этих наборов, получим терм, не содержащий ни одной из переменных, соответствующих переменным, возбужденным в α. В результате склеивания и поглощения ранг этого терма может быть и еще уменьшен, однако он будет отличен от 0, так как в противном случае собственная функция z_i будет равна константе, что противоречит условию.

Из леммы 5.1 и теоремы 5.2 следует, что реализация должна быть основана на представлении z_i в виде выражения (5.5), а из лемм 5.1 и 5.2 — что она должна соответствовать сокращенной д. н. ф. функций w_i и v_i, входящих в это выражение. Если все эти условия выполнены, полученную реализацию будем называть совершенной. *Совершенная реализация* является корректной. Из сказанного очевидна справедливость следующей теоремы.

Теорема 5.4. *Система собственных функций элементов И-ИЛИ-НЕ функциональна полна в классе* U_n^*, *т. е. на элементах И-ИЛИ-НЕ можно корректно реализовать любую схему из* U_n^*. (*Предполагается, что названный базис является неограниченным, т. е. число входов элементов, как и их нагрузочная способность, сверху не ограничены*).

Пример 5.4. Пусть схема из четырех элементов задана диаграммой рис. 5.19, *а*. По ней получим (см. § 2.4) систему уравнений

$$z_1 = \bar{z}_4, \quad z_2 = \bar{z}_4, \quad z_3 = z_3(z_1z_2 \vee \bar{z}_4) \vee \bar{z}_4(z_1 \vee z_2),$$
$$z_4 = z_4(z_1 \vee z_2 \vee z_3) \vee z_1z_2z_3.$$

Приведем их к виду (5.3):

$$z_1 = z_1z_4 \vee \bar{z}_4, \quad z_2 = z_2z_4 \vee \bar{z}_4,$$
$$z_3 = z_3\overline{z_4z_1} \vee z_4\bar{z}_2 \vee \bar{z}_4z_1 \vee \bar{z}_4z_2, \quad z_4 = z_4\overline{z_1z_2z_3} \vee z_1z_2z_3.$$

Совершенная реализация последней системы на триггерах типа рис. 5.18 приведена на рис. 5.19, *б*.

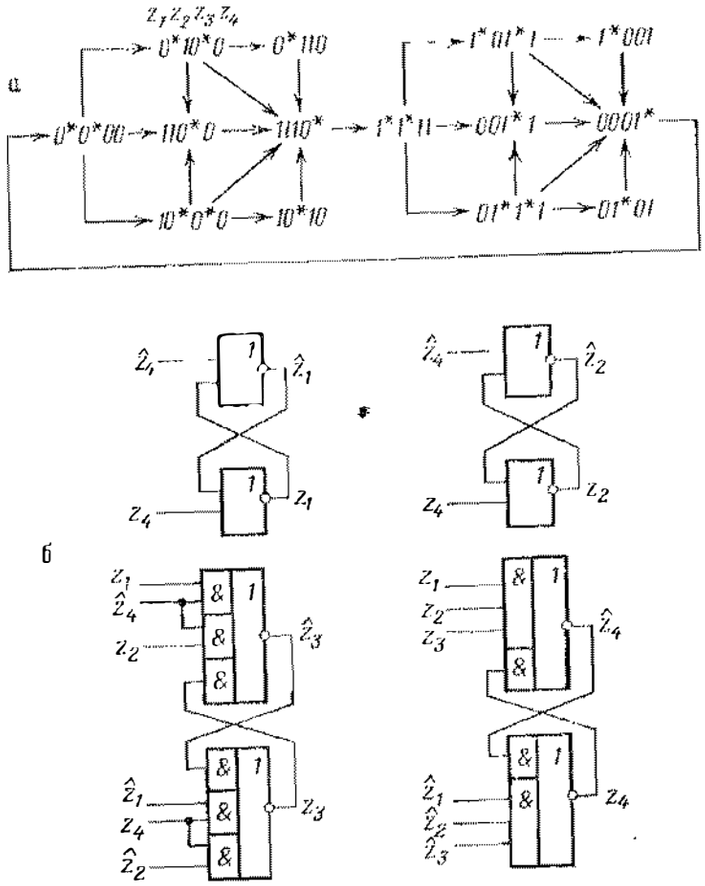


Рис. 5.19. Диаграмма переходов (*а*) и ее совершенная реализация (*б*)

5.3.4. Простые схемы. В этом пункте будет подробно изучен процесс переключения элементов в полумодулярных схемах и выделен весьма важный класс простых схем.

Процесс функционирования схемы заключается в переходах значений переменных. Эти переходы можно ну-

меровать. Каждая переменная схема в процессе циклической работы может переключиться некоторое число раз прежде, чем схема вернется в начальное состояние. Любому i-му переходу переменной z_j, который будет обозначаться парой натуральных чисел (j, i), помечаемой иногда для наглядности знаком + или — для указания направления перехода z_j: 0-1 и 1-0 соответственно, предшествует ее $(i-1)$-е возбуждение. Это возбуждение может впервые возникать в одном или нескольких неупорядоченных состояниях. Уточним сказанное.

Определение 5.5. Связное множество A состояний (т. е. такое множество, в котором любые два состояния соединены траекторией, целиком лежащей в этом множестве), являющееся подмножеством множества допустимых (рабочих) состояний схемы, назовем *зоной возбуждения переменной* z_j, если z_j имеет во всех состояниях из A одно и то же возбужденное значение (0* или 1*) и множество A максимально, т. е. для любых α, β таких, что $\alpha \notin A$, $\beta \in A$ и $\alpha \to \beta$ или $\beta \to \alpha$, имеет место $\alpha_j \neq \beta_j$. Зону возбуждения будем обозначать $A(j, i)$, если это возбуждение вызывает i-й переход переменной z_j. Состояние $\alpha \in A$ назовем *минимальным состоянием зоны возбуждения*, если для любого состояния $\beta \in A$ $(\beta \neq \alpha)$ не существует траектории состояний $\beta \Rightarrow \alpha$, целиком лежащей в A. Будем говорить, что зона возбуждения имеет *нижнюю грань*, если она содержит только одно минимальное состояние.

Легко видеть, что каждая зона возбуждения содержит хотя бы одно минимальное состояние. С другой стороны, в гл. 8 будет приведено локальное характеристическое свойство дистрибутивных схем, которое может быть сформулировано следующим образом.

Схема, полумодулярная относительно начального состояния α, дистрибутивна относительно α тогда и только тогда, когда каждая ее зона возбуждения на множестве рабочих состояний (состояний, достижимых из начального) имеет нижнюю грань.

В недистрибутивных схемах обязательно имеются зоны возбуждения без нижних граней. Такие зоны возбуждения и соответствующие им переходы значений переменных получили название *детонантных*, а число минимальных состояний в детонантной зоне — *ранга детонантности зоны*.

Собственную функцию каждого элемента схемы можно представить в виде разложения по Шеннону:

$$f_j(z_1, \ldots, z_n) = z_j f_j(z_j = 1) \vee \bar{z}_j f_j(z_j = 0).$$

Функции $f_j(z_j = 0)$ и $\bar{f}_j(z_j = 1)$ назовем *функциями возбуждения переменной* z_j и будем обозначать их через S_j и R_j.

Если зона возбуждения имеет нижнюю грань, то возбуждение может впервые возникнуть лишь в одном состоянии, а именно в том, которое является нижней гранью зоны. Данное возбуждение переменной z_j впервые вызывается одним термом T сокращенной д. н. ф. ее функции возбуждения S_j или R_j. В процессе дальнейшего функционирования схемы переменные, входящие в терм T, могут изменить свое значение прежде, чем z_j успеет переключиться. При этом в силу полумодулярности схемы возбуждение переменной z_j должно поддерживаться другими термами функции возбуждения. Это явление будем называть *перехватом термов функции возбуждения*. Естественно, перехват термов может иметь место и в детонантной зоне возбуждения.

Пример 5.5. В схеме рис. 5.20, заданной системой уравнений $a = 1$, $b = 1$, $c = a \vee b$, зона первого возбуждения переменной c состоит из трех состояний 100, 010, 110, и является детонантной,

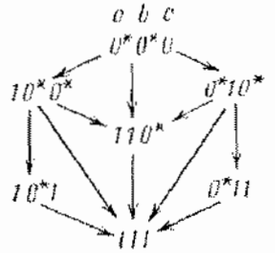


Рис. 5.20. Схема с детонантной зоной возбуждения

так как первые два из этих состояний — 100 и 010 минимальны для этой зоны и неупорядочены между собой. Функция возбуждения переменной c $S_c = a \vee b$. Возбуждение переменной c в состоянии 100 вызывается термом a, а в состоянии 010 — термом b. В состоянии 110 возбуждение поддерживается обоими термами a, b. Ранг детонантности зоны равен двум.

Дадим строгое определение перехвата термов.

Определение 5.6. Будем говорить, что в зоне возбуждения $A(j, i)$ переменной z_j имеет место перехват термов функции возбуждения S_j или R_j, если найдется траектория $\alpha^1 \to \alpha^2 \to \ldots \to \alpha^h$, где $\alpha^1, \alpha^2, \ldots, \alpha^h \in A(j, i)$ такая,

что все термы сокращенной д. н. ф. S_j или R_j, имеющие значение 1 в состоянии α^1, в состоянии α^h имеют значение 0.

Наличие перехвата термов существенно затрудняет решение задачи декомпозиции собственных функций элементов, так как между элементами, реализующими отдельные термы, участвующие в перехвате, возможны *логические состязания.*

Определение 5.7. Асинхронную схему будем называть простой (относительно начального состояния α), если все зоны возбуждения на множестве рабочих состояний свободны от перехвата термов.

Отсутствие перехвата термов в простой схеме означает, что если возбуждение переменной z_j возникло впервые в одном или нескольких неупорядоченных состояниях под воздействием одного или нескольких (детонантное изменение) термов S_j или R_j, то ни одна переменная, входящая в эти термы, не может изменить своего значения до тех пор, пока не переключится переменная z_j.

Процесс функционирования схем удобно отражать на диаграмме изменений. Формально *диаграмма изменений* представляет собой тройку $\langle V, \vdash, \rightarrow \rangle$, где V — множество пар натуральных чисел вида (j, i), которые интерпретируются как переход переменных: (j, i) — i-й переход переменной z_j; $\vdash$ и $\rightarrow$ — отношения порядка на множестве V, имеющие разную семантическую интерпретацию. Если $a, b \in V$ и $a \vdash b$, то это означает, что переход a обязательно предшествует переходу b. Если $a \rightarrow b$, то это означает, что после перехода a и, возможно, других переходов произойдет переход b, причем переход b может произойти и без перехода a под воздействием некоторого другого перехода c, если $c \rightarrow b$.

Процесс функционирования дистрибутивных схем однозначно задается лишь одним отношением $\vdash$.

Исследование свойств диаграмм изменений, задающих схемы и являющихся одним из способов их формального описания, не является самоцелью, а будет использоваться лишь как удобное иллюстративное средство. Диаграмма переходов будет задаваться трехдольным графом. Вершинам первого типа ставятся в соответствие переходы из множества V, которые для наглядности могут помечаться знаками + или — в зависимости от направления переключения переменных. Эти вершины непосредственно связаны дугами, соответствующими отношению $\vdash$ и опосредованно (через вершины других типов) в случае отношения $\rightarrow$.

Кроме того, на графе не изображаются дуги, соответствующие транзитивному замыканию дуг, подобно тому, как это делается на *диаграмме Хассе.*

Пример 5.6. Диаграмма рис. 5.21, *а* означает, что $(+1,1)\vdash(-3,2)$ и $(+2,1)\vdash(-3,2)$; рис. 5.21, *б* — $(+1.1)\vdash(-3,2)$ и $(+2,1)\vdash(-3,2)$; рис. 5.21, *в* — $(+2,1)\vdash(-3,2)$ и $\{(+1,1),(+4,1)\}\vdash(-3,2)$. Последняя запись говорит о том, что для выполнения (—3,2) необходимо

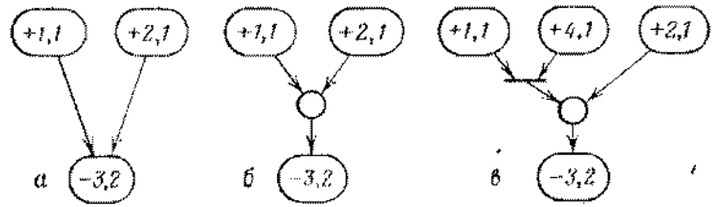


Рис. 5.21. Три диаграммы изменений

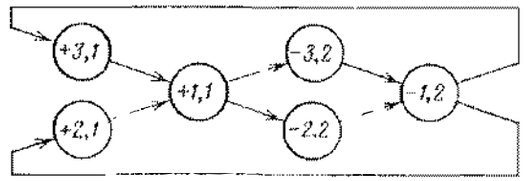


Рис. 5.22. Диаграмма изменений с совместными переходами, соответствующая диаграмме переходов на рис. 2.11, *а*

либо выполнение (+2,1), либо совместное выполнение переходов (+1,1) и (+4,1). На рис. 5.22 приведена диаграмма изменений для схемы, диаграмма переходов которой изображена на рис. 2.11, *а*.

Легко убедиться, что графическое изображение диаграммы изменений для дистрибутивных схем является сигнальным графом, у которого удалены все маркеры.

На множестве переходов V введем еще одно бинарное отношение — отношение совместности. *Будем говорить, что переходы* (s, p) и (t, q) *совместны* и писать $(s, p)\|(t, q)$, если неверно, что $(s, p)\vdash(t, q)$ или $(t, q)\vdash(s, p)$, т. е. совершение любого из переходов не обусловлено совершением другого. Очевидно, что два различных перехода одной и той же переменной в диаграмме изменений, соответствующей полумодулярной схеме, не могут быть совместны. На диаграмме перехо-

дов отношение совместности проявляется в том, что $A(s, p) \cap A(t, q) \neq \varnothing$, т. е. в некотором состоянии возбуждена как переменная z_s, так и z_t, и эти возбуждения повлекут переходы указанных переменных с номерами p и q соответственно.

Пример 5.6 (продолжение). На рис. 5.22 $(+3,1) \| (+2,1)$ и $(-3,2) \| (-2,2)$.

Как уже говорилось, для каждой переменной z_j достаточно иметь конечное число переходов $(j, 0), \ldots, (j, m_j - 1)$, где m_j — число переходов переменной z_j до того момента, как работа схемы зациклится.

Пусть α — произвольное состояние зоны возбуждения $A(j, i)$, а Q_j^i — часть д. н. ф. функции возбуждения R_j или S_j, соответствующая переходу (j, i). Переход (k, l) назовем *входным* для перехода (j, i) и зоны $A(j, i)$, если на множестве рабочих состояний найдется состояние β такое, что $\beta \xrightarrow{1}_{z_k} \alpha$, причем этот переход осуществляется в результате l-го перехода переменной z_k. Это означает, что l-й переход переменной z_k является непосредственной причиной i-го возбуждения переменной z_j. Очевидно, если в зоне возбуждения $A(j, i)$ изменяет свое значение хотя бы одна входная для этой зоны переменная, то в ней наверняка имеется перехват. Такой перехват назовем *сложным*, а если в зоне не изменяется значение ни одной входной для этой зоны переменной, то перехват назовем *простым*. Наличие любого перехвата в зоне $A(j, i)$ является следствием совместности перехода (j, i) и очередного перехода той переменной, которая входила в терм функции возбуждения Q_j^i, соответствующий выбранному состоянию α. Так, в разобранном случае сложного перехвата $(j, i) \| (k, l+1)$.

Важность класса простых схем демонстрирует следующая

Теорема 5.5. *По любой полумодулярной схеме можно построить эквивалентную ей полумодулярную простую схему, причем если исходная схема дистрибутивна, то и простая схема будет дистрибутивна.* (Теорема сформулирована для инициальных схем: имеется в виду полумодулярность и дистрибутивность схем относительно начальных состояний.)

Доказательство. Отметим, что любую недистрибутивную полумодулярную схему можно эквивалентно преобразовать к такому

виду, что функции возбуждения, соответствующие детонантным переходам, будут иметь вид $z_m^\alpha \vee z_r^\beta$, т. е. возбуждающие термы элементарны, а ранг детонантности равен двум. Поэтому достаточно научиться устранять перехваты лишь в недетонантных зонах перехода.

Устранение сложных перехватов. Идея устранения сложного перехвата основана на введении дополнительной переменной x, играющей роль буфера для переходов (j, i) и $(k, l+1)$. Переход $(+x, 1)$ происходит непосредственно после (k, l) и в свою очередь предшествует (j, i) и $(k, l+1)$. Сброс переменной x — переход $(-x, 2)$ — производится непосредственно перед $(j, i+1)$. При таком преобразовании происходит задержка перехода $(k, l+1)$, являющегося причиной перехвата, до тех пор, пока не произойдет переход $(+x, 1)$. Упорядочение предыдущих переходов при этом не изменяется, так как и до введения дополнительной переменной переход (k, l) неизбежно предшествует $(k, l+1)$. Функции возбуждения преобразуются следующим образом: $Q_j^i = T_j^i \vee \varphi_j^i$ — в $\dot{Q}_j^i = x(\dot{T}_j^i \vee \varphi_j^i)$, где T_j^i — терм, соответствующий минимальному состоянию α зоны $A(j, i)$, а $\dot{T}_j^i$ получается из T_j^i вычеркиванием z_k; Q_j^{i+1} — в $\dot{Q}_j^{i+1} = \bar{x}$. Для новой переменной $S_x = z_k^{\alpha_l} \varphi_x$, где $\alpha_l = 0$, если (k, l) — это переход 1-0 и $\alpha_l = 1$ в противном случае, а функция φ_x обеспечивает блокировку изменений x под воздействием переходов $(k, l+2)$, $(k, l+4)$, ..., если это требуется; $R_x = Q_j^{i+1}$. В результате таких преобразований устранен один из сложных перехватов в $A(j, i)$. Зона $A(x, 1)$ свободна от сложного перехвата. Перехваты из $A(j, i+1)$, если они есть, переходят в зону $A(x, 2)$. Никаких других перехватов в зоне $A(x, 1)$ нет. В остальных зонах, кроме $A(k, l+1)$, никаких изменений не произошло.

Возможны два варианта. Если в исходной схеме $(k, l+1)$ и $(j, i+1)$ несовместны, то зона $A(k, l+1)$ свободна от перехвата по x.

Положим $\dot{Q}_k^{l+1} = Q_k^{l+1} x$, что устраняет один сложный перехват.

Если же $(k, l+1) \| (j, i+1)$, то в зоне $A(k, l+1)$ имеется сложный перехват по x. Для его устранения в соответствии с описанной процедурой следует ввести вторую дополнительную переменную y. Если в исходной схеме $(j, i+1) \| (k, l+2)$, то в зоне $A(x, 2)$ будет иметь место сложный перехват по y, и для его устранения тем же способом введем третью дополнительную переменную z. Этим процесс устранения одного сложного перехвата результативно завершается, так как зона $A(y, 2)$ свободна от сложного перехвата по z, потому что для выполнения $(-y, 2)$ необходимо срабатывание $(-z, 2)$, т. е. $(-y, 2)$ и $(-z, 2)$ несовместны.

Устранение простых перехватов. Метод устранения сложных перехватов здесь не всегда применим, так как задержка перехода, являющегося причиной простого перехвата, может привести к нарушению упорядочения срабатывания элементов.

Заметим, что все термы функции возбуждения Q_j^i, участвующие в простом перехвате, имеют общий подтерм P, включающий

по крайней мере все входные переменные зоны $A(j, i)$. Невозможность замены всех термов д. н. ф. для Q_j^i на их общий подтерм P объясняется тем, что в другом месте множества рабочих состояний

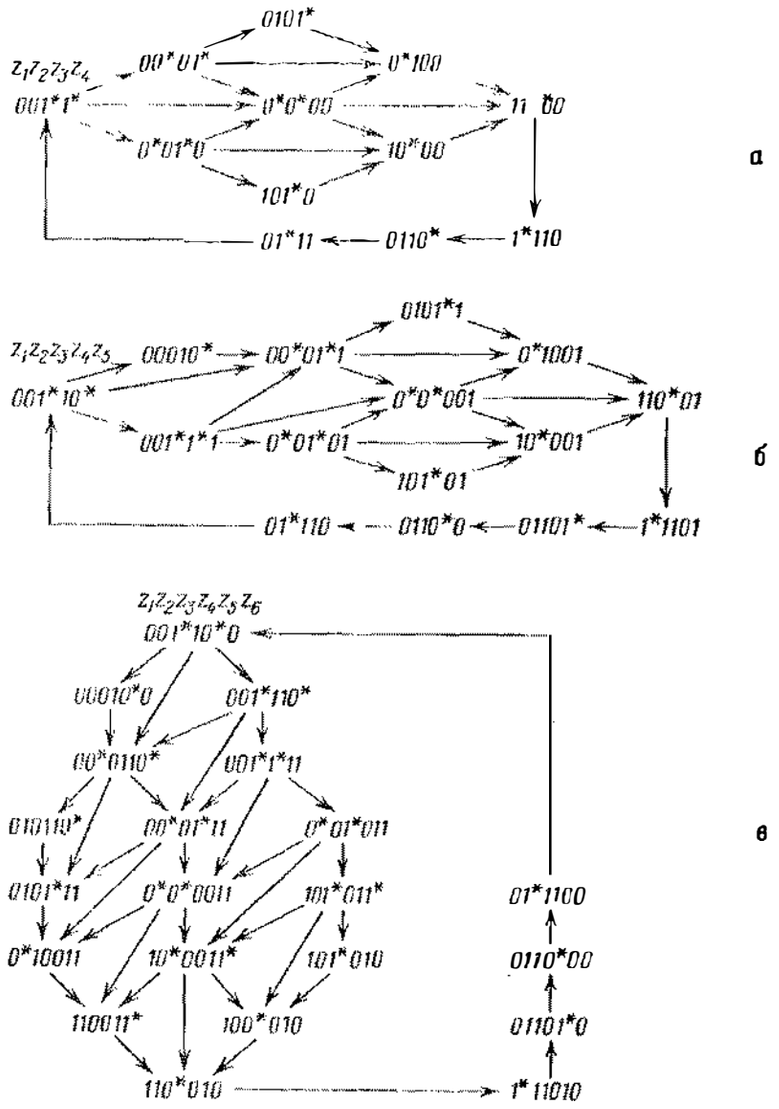


Рис. 5.23. К методу устранения перехватов: диаграммы переходов исходной схемы (*а*), схемы после устранения сложного перехвата (*б*) и простого перехвата (*в*)

схемы значения переменных, входящих в P, и z_j совпадают с их значениями в зоне $A(j, i)$, но при этом z_j устойчива. За счет введения дополнительных переменных можно так «пополнить» общий подтерм P, чтобы внутри множества рабочих состояний новой

схемы вне зоны $A(j, i)$ не встречались состояния, в которых z_j устойчива, и этот подтерм принимал бы истинное значение.

Введем переменную x, которая в процессе работы срабатывает лишь дважды. Ее переход $(+x, 1)$ непосредственно предшествует переходу (k, l), входному для $A(j, i)$, а $(-x, 2)$ следует сразу за (j, i) и предшествует тем переходам (m, t), которые в исходной схеме могли следовать сразу за (j, i). Функции возбуждения преобразуются следующим образом: $\dot{Q}_j^i = xP$; $\dot{Q}_h^l = x$; $\dot{Q}_m^t = Q_m^t \bar{x}$; $R_x = z_j^{\alpha_i}$, где $\alpha_i = 0$, если (j, i) — это переход 1-0, и $\alpha_i = 1$ в противном случае; $S_x = Q_k^{l-\alpha_l} z_j^{\alpha_l}$. Хотя $(+x, 1)$ непосредственно предшествует (k, l), функция S_x не может быть просто приравнена Q_k^l. В самом деле, $(-x, 2)$ предшествует переходу $(k, l+1)$, поэтому в состояниях, следующих за $(-x, 2)$ и предшествующих $(k, l+1)$ функция Q_k^l может все еще иметь истинное значение (см. пример 5.7) и вызвать преждевременный обратный переход 0-1 переменной x. Во избежание этого Q_k^l и умножается на $z_k^{-\alpha l}$. Сразу после $(+x, 1)$ произойдет (k, l), $z_k^{-\alpha l}$ примет значение 0 и не сможет принять истинное значение вплоть до $(k, l+1)$. Это гарантирует, что во всех упомянутых выше состояниях $S_x = 0$.

Пример 5.7 (иллюстрирующий метод устранения перехватов). Рассмотрим схему, заданную системой уравнений $z_1 = \bar{z}_4(\bar{z}_2 \vee \bar{z}_3)$, $z_2 = \bar{z}_3 \vee z_2\bar{z}_4$, $z_3 = z_2(z_1 \vee z_3)$, $z_4 = \bar{z}_1 z_2 z_3$, диаграмма переходов которой приведена на рис 5.23, *а*, а диаграмма изменений — на рис. 5.24, *а*. В схеме два перехвата термов. Первый — в зоне $A(-4, 2)$ по входной переменной z_2 — является сложным. Он осуществляется на траектории 001*1*-00*01*-0101*, в первом состоянии которого истинное значение имеет терм $\bar{z}_2$ функции возбуждения R_4^1, а в последнем — $\bar{z}_3$. Второй перехват — в зоне $A(+1, 1)$ — простой. Он осуществляется на территории 0*010*-0*0*00-0*100, где термы S_1^1 меняются с $\bar{z}_2\bar{z}_4$ (в состоянии 0*01*0) на терм $\bar{z}_3\bar{z}_4$ (в состоянии 0*100).

Устраним сначала сложный перехват. Для этого введем дополнительную переменную z_5 так, как показано на рис. 5.24, *а* и 5.23, *б*. В силу того, что $(+2, 2)$ и $(+4, 2)$ несовместны, одной дополнительной переменной оказывается достаточно. Собственные функции элементов в соответствии с изложенным методом будут иметь вид $z_1 = \bar{z}_4(\bar{z}_2 \vee \bar{z}_3)$, $z_2 = \bar{z}_3 z_5 \vee z_2\bar{z}_4$, $z_3 = z_2(z_1 \vee z_3)$, $z_4 = \bar{z}_5$, $z_5 = \bar{z}_2 \vee z_5(z_1 \vee \bar{z}_3)$.

В полученной схеме остался лишь один простой перехват в зоне $A(+1, 1)$ на траектории 0*01*01*-0*0*001-0*1001. Устраним его посредством введения еще одной дополнительной переменной z_6 — рис. 5.23, *в* и 5.24, *в*. Новые функции возбуждения имеют вид:

$$\dot{S}_1 = \bar{z}_4\bar{z}_6, \quad \dot{R}_1 = R_1 = z_4 \vee z_2z_3, \quad \dot{S}_2 = S_2 = \bar{z}_3z_5, \quad \dot{R}_2 = R_2 = z_4,$$
$$\dot{S}_3 = S_3\bar{z}_6 = z_1z_2\bar{z}_6, \quad \dot{R}_3 = R_3 = \bar{z}_2, \quad \dot{S}_4 = S_4 = \bar{z}_5,$$
$$\dot{R}_4 = z_6, \quad \dot{S}_5 = S_5 = z_2, \quad \dot{R}_5 = R_5 = \bar{z}_1z_2z_3,$$
$$\dot{S}_6 = R_4z_4 = z_4z_5, \quad \dot{R}_6 = z_1.$$

Система уравнений, задающая эквивалентную исходной схему без перехватов и соответствующая таким функциям возбуждения, имеет вид:

$$z_1 = \bar{z}_1\bar{z}_4z_6 \vee z_1\bar{z}_4(\bar{z}_2 \vee \bar{z}_3), \quad z_2 = \bar{z}_3z_5 \vee z_2\bar{z}_4,$$
$$z_3 = z_2(z_1\bar{z}_6 \vee z_3), \quad z_4 = \bar{z}_4z_5 \vee z_4\bar{z}_6,$$
$$z_5 = \bar{z}_2 \vee z_5(z_1 \vee \bar{z}_3), \quad z_6 = \bar{z}_6z_4z_5 \vee z_6\bar{z}_1.$$

Окончательно с учетом частичной минимизации за счет доопределений функций на состояниях, не принадлежащих множеству рабочих состояний, имеем: $z_1 = \bar{z}_4(z_6 \vee \bar{z}_2 \vee \bar{z}_3)$, $z_2 = \bar{z}_3z_5 \vee z_2\bar{z}_4$, $z_3 = z_2(z_1\bar{z}_6 \vee z_3)$, $z_4 = \bar{z}_4\bar{z}_5 \vee z_4\bar{z}_6$ или $z_4 = \bar{z}_6(z_4 \vee \bar{z}_5)$, $z_5 = z_1 \vee \vee \bar{z}_2 \vee \bar{z}_3$, $z_6 = z_4z_5 \vee z_6\bar{z}_1$.

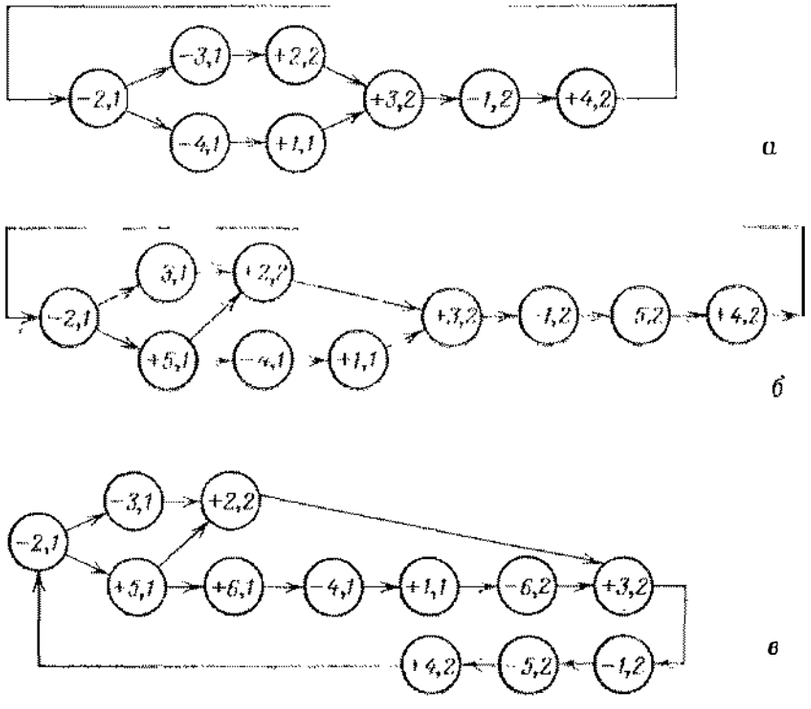


Рис. 5.24. Диаграммы изменений, соответствующие этапам устранения перехватов, показанным на рис. 5.23

5.3.5. Реализация дистрибутивных и последовательных схем. В п. 5.3.3 был описан метод совершенной реализации, позволяющий синтезировать произвольную полумодулярную схему на элементах И-ИЛИ-НЕ. Здесь будет рассмотрена возможность построения дистрибутивных и последовательных схем на минимальных системах элементов И-НЕ или (в силу двойственности) ИЛИ-НЕ.

Такая постановка имеет самостоятельный интерес. Кроме того, необходимо учитывать принятые для различных базисов гипотезы о характере задержек элементов и соединительных линий. Поясним эту мысль.

Пример 5.8. Поведение схемы рис. 5.25 описывается системой уравнений $z_1 = \bar{z}_3$, $z_2 = \bar{z}_3$, $z_3 = z_1z_2 \vee z_3(z_1 \vee z_2)$. Если подсхему, обведенную на этом рисунке штриховыми линиями, рассматривать как единое целое, то все в порядке. Однако если нарисовать полную диаграмму переходов с учетом переменных K_1, K_2 и K_3, то картина существенно изменится. На рис. 5.26 изображен фрагмент такой диаграммы, из которой, во-первых, видны возможные траектории неправильной работы (они указаны двойными стрелками на диаграмме) и, во-вторых, ясно, что если после перехода 1-0 элемента K_3 элементы K_1 и K_2 переключаются быстрее, чем любой из элементов z_2 или z_3, то нарушения правильности работы не произойдет. С другой стороны, читатель может убедиться, что схема рис. 5.27 моделирует приведенную систему уравнений независимо от задержек элементов.

Сказанное оправдывает поиск минимального схемного базиса для класса схем, не зависящих от задержек, и соответствующих методов синтеза, что имеет не только прикладной, но еще и теоретический интерес.

Как и раньше, будем исходить из представления собственной функции любого элемента z_i схемы из U_n^* в форме (5.3). По аналогии с (5.5) (см. теорему 5.2) можно получить декомпозицию (5.3), соответствующую двухъяруспой схеме на элементах $\widetilde{\text{И-НЕ}}$, где первый ярус реализует RS-триггер с выходами z_i и $\widetilde{z}_i$, а второй — термы функций w_i и v_i. Дизъюнкция этих термов осуществляется непосредственно на элементах, образующих триггеры. Но реализация, соответствующая такой декомпозиции, некорректна. Дело в том, что при наличии второго яруса по условию теоремы 5.2 необходимо инвертирование входных переменных функций w_i и v_i. Такое инвертирование подразумевает, кроме всего прочего, изменение значений выходов при $z_i = \widetilde{z}_i$. Так, если при одноярусной реализации на элементах $\widetilde{\text{И-НЕ}}$ состояния, в которых $z_i = \widetilde{z}_i = 1$, относится к классу T_j, $j \neq i$, то для двухъяруспой реализации требуется $z_i = \widehat{z}_i = 0$. Это требование удается реализовать, если использовать триггеры типа, показанного на рис. 5.28 с парафазными входами z_i и $\widehat{z}_i$.

Рассмотрение начнем с последовательных схем из класса K_n^*, т. е. схем, в которых не может быть одновременно возбуждено более одного элемента.

Теорема 5.6. *Система собственных функций элементов И-НЕ функционально полна в классе K_n^*, т. е. на элементах И-НЕ может быть корректно реализована любая последовательная схема.*

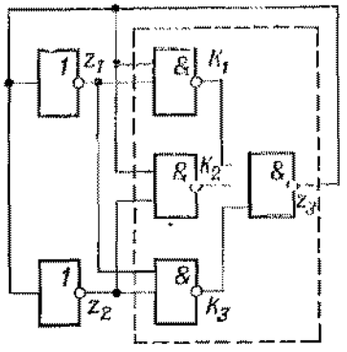


Рис. 5.25. Некорректная реализация системы уравнений двухвходового Г-триггера с инверторами

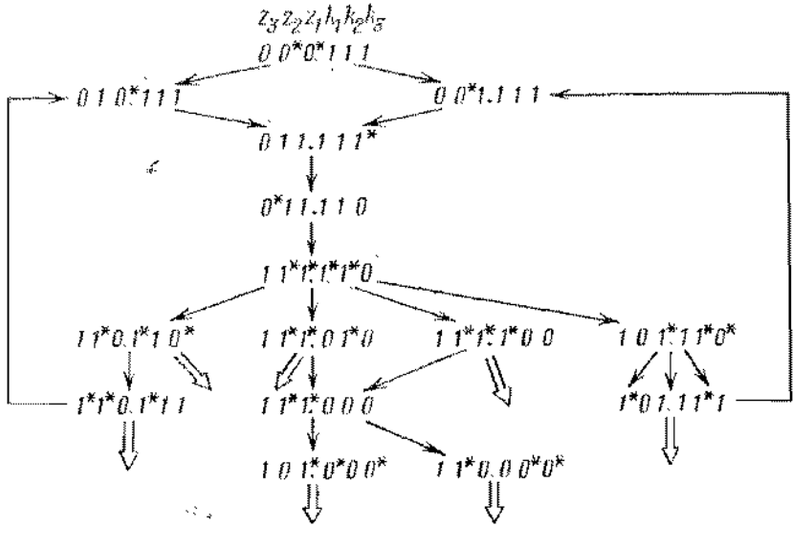


Рис. 5.26. Фрагмент диаграммы переходов схемы рис. 5.25

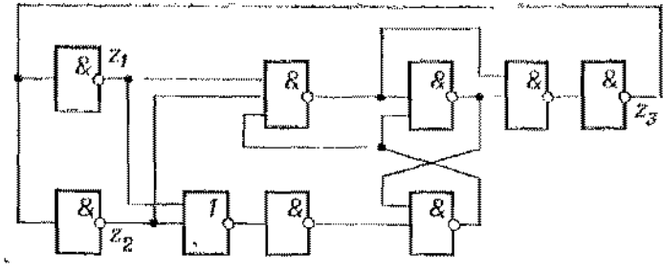


Рис. 5.27. Корректная реализация Г-триггера с инверторами

Доказательство. Каждому состоянию, принадлежащему множеству рабочих состояний последовательной схемы из K_n^*, $n>1$, $\alpha^k=\alpha_1^k\ldots\alpha_n^k$, можно поставить в соответствие элемент t_k, реализующий функцию $t_k=t_{k-1}\overline{z_1^{\alpha_1^k}\ldots z_n^{\alpha_n^k}}$, а каждому элементу z_i схемы — RS-триггер на элементах И-НЕ. Элемент t_k соединяется

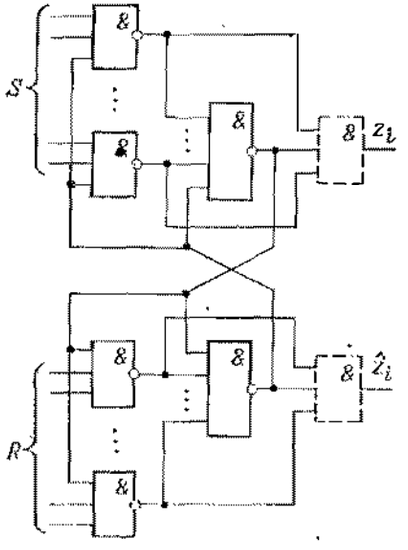


Рис. 5.28. Базовая конструкция для реализации дистрибутивных или последовательных схем

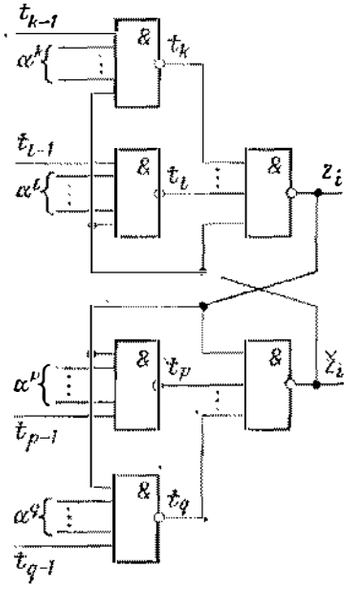


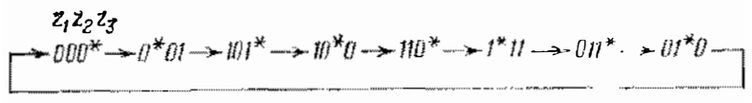
Рис. 5.29. К доказательству возможности корректной реализации последовательных схем в базисе И-НЕ

со входом R_i триггера $\check{z}_i$, если $\alpha^k\in R_i$, со входом S_i триггера $\check{z}_i$, если $\alpha^k\in S_i$, и не соединяется со входами триггера $\check{z}_i$, если $\alpha^k\in T_i$.

При нахождении схемы в состоянии α^k $t_k=0$. Пусть для определенности $\alpha^k\in S_i$ (рис. 5.29), а также $\alpha^l\in S_i$, $\alpha^p\in R_i$, $\alpha^q\in R_i$. Значение $t_k=0$ вызывает установку триггера $\check{z}_i$. После того как триггер переключится ($z_i=1$, $\check{z}_i=0$), срабатывает элемент t_k ($t_k=1$). После этого срабатывает элемент t_{k+1}, что эквивалентно переходу схемы из состояния α^k в состояние α^{k+1}, который в исходной схеме осуществляется изменением значения элемента z_i, а в реализации типа рис. 5.29 происходит путем последовательного срабатывания элементов z_i, $\check{z}_i$, t_k, t_{k+1}.

Полученная таким образом схема относится к классу последовательных K_m^*, $m\geqslant 4n$, и порядок срабатывания ее триггеров повторяет заданный порядок срабатываний элементов исходной схемы из K_n^*. Таким образом, полученная реализация корректна.

Пример 5.9. Пусть задана последовательная диаграмма переходов



в которой переходы z_3 происходят вдвое чаще, чем переходы z_1 или z_2, т. е. эта диаграмма описывает поведение счетного триггера. По диаграмме переходов получим (см. § 2.4) систему уравнений $z_1 = z_1\bar{z}_3 \vee \bar{z}_2 z_3$, $z_2 = z_1\bar{z}_3 \vee z_2 z_3$, $z_3 = z_1 z_2 \vee \bar{z}_1\bar{z}_2$. В соответствии с определением t_k в доказательстве теоремы 5.6 имеем:

$$t_0 = \overbrace{t_7 z_1 z_2 z_3}, \quad t_1 = \overbrace{t_0 z_1 z_2 z_3}, \quad t_2 = \overbrace{t_1 z_1 z_2 z_3},$$

$$t_3 = \overbrace{t_2 z_1 z_2 z_3}, \quad t_4 = \overbrace{t_3 z_1 z_2 z_3}, \quad t_5 = \overbrace{t_4 z_1 z_2 z_3},$$

$$t_6 = \overbrace{t_5 z_1 z_2 z_3}, \quad t_7 = \overbrace{t_6 z_1 z_2 z_3}.$$

Выходы элементов И-НЕ с собственными функциями $t_0 - t_7$ подключаются к RS-триггерам z_1, z_2, z_3 так, как показано на рис. 5.30.

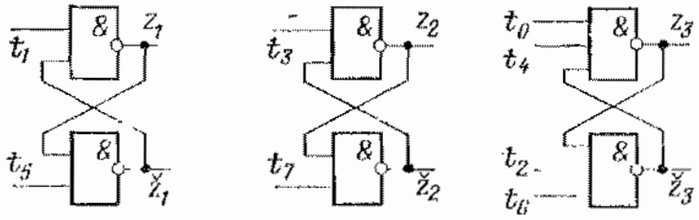


Рис. 5.30. Вариант схемы счетного триггера, построенной на основе рис. 5.29 (восемь четырехвходовых вентилей $t_0 - t_7$ не показаны)

Для реализации на триггерах типа, показанного на рис. 5.29 схем из класса $U_n^*\backslash K_n^*$ (полумодулярных, но не полностью последовательных) число элементов t_k для каждого состояния α^k должно быть увеличено по числу элементов z_i, возбужденных в этом состоянии. Проведя минимизацию собственной функции каждого элемента z_i в соответствии с теоремой 5.3, придем к тому, что элемент t_k будет соответствовать одному терму сокращенной формы функции w_i или v_i в выражениях типа (5.3). При этом вместо переменных z_i и $\bar{z}_i$ в них для элементов z_j, $j \neq i$, должны быть использованы термы $z_i t_k \ldots t_l$ и $z_i t_p \ldots t_k$ соответственно, поскольку переходы 0-1 (1-0) переменной z_i в схеме рис. 5.28 завершаются установкой

$$z_i = t_k = \ldots = t_l = 1 \quad (\bar{z}_i = t_p = \ldots = t_q = 1).$$

Тем не менее такая модификация не является достаточной для корректности реализации произвольных схем из $U_n^*\backslash K_n^*$ на элементах И-НЕ. В самом деле, поскольку функции возбуждения, вызывающие переключения элемента, могут быть не однотермными, на элементах первого яруса (триггеров рис. 5.2) возможны логические состязания, приводящие к нарушению их полумодулярности. Как уже говорилось, возможны две причины нарушения однотермности: детонантные переходы и перехват термов. В соответствии с теоремой 5.5 каждая схема может быть приведена к виду без перехвата термов, а поскольку в дистрибутивных схемах отсутствуют детонантные переходы, справедлива

Теорема 5.7. *Система собственных функций элементов И-НЕ функционально полна в классе D_n^*, т. е. на элементах И-НЕ может быть корректно реализована любая дистрибутивная схема.*

Примеры 5.7, 5.8 и 5.9 по существу иллюстрируют универсальные канонические подходы к синтезу полумодулярных, последовательных и дистрибутивных схем. Полученные в результате применения этих подходов схемы можно отнести к классу стандартных реализаций, т. е. реализаций, которые гарантируют минимальные затраты на процесс синтеза, возможно, в ущерб экономичности схем.

§ 5.4. Синтез полумодулярных схем в ограниченных базисах

До сих пор не рассматривался вопрос о том, какие ограничения накладывает на возможность реализации полумодулярных схем учет допустимого числа входов элементов и их нагрузочной способности.

Далее будет конструктивно доказан тот факт, что *любая дистрибутивная схема из D_n^* может быть корректно реализована в базисе 2И-НЕ-2*, т. е. с минимально возможными значениями коэффициентов разветвления по входу и выходу.

Каждая переменная z_j схемы в процессе работы может переключиться некоторое число раз m_j прежде, чем схема вернется в начальное состояние. Поэтому число переходов каждой переменной z_j достаточно считать, по модулю m_j. Всякому переходу (j, i), где $1 \leqslant i \leqslant m_j$, поставим в соответствие булеву переменную z_j^i,

которая будет принимать значение 1 после того, как совершится переход $(j, 1)$, но до того, как совершится переход $(j, i+1)$. (Сложение и вычитание производится по модулю m_j, т. е. $m_j + 1 = 1$, а $1 - 1 = m_j$). Точнее, $z_j^i = 1$ на состояниях, следующих в диаграмме переходов исходной схемы за состоянием из $A(j, i)$, включая состояния зоны $A(j, i+1)$ и $z_j^i = 0$ на всех остальных состояниях.

На значениях последовательности переменных $z_j^1 z_j^2 \dots z_j^{m_j}$ процесс переключений переменной z_j может изображаться в виде циклического сдвига унитарного кода $00\dots01 \to 10\dots00 \to 01\dots00 \to$ $\to 00\dots01 \to \dots$ Обозначим функцию управления i-м сдвигом, соответствующую функции возбуждения перехода $0 - 1$ переменной z_j^i, через ψ_j^i. Через Q_j^i обозначим функцию возбуждения для перехода (j, i) в исходной схеме, т. е. ту часть функции S_j или R_j, которая вызывает возбуждение z_j в зоне $A(j, i)$. Для простых дистрибутивных схем все Q_j^i содержат по одному терму (однотермны). ψ_j^i получается из Q_j^i заменой каждого вхождения переменной z_k (или $\bar{z}_k$) переменной z_k^l, где l — номер последнего перехода переменной z_k, предшествующего зоне $A(j, i)$ в исходной схеме.

Пример 5.10. Рассмотрим частный случай простой дистрибутивной схемы — простую последовательную схему, диаграмма переходов которой приведена на рис. 5.31, *а* (начальное состояние

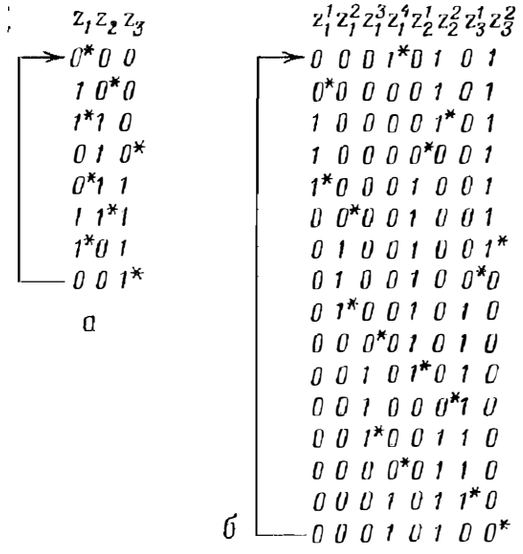


Рис. 5.31. Диаграммы переходов простой последовательной схемы (*а*) и переменных с номерами их вхождений (*б*)

0*00). Этой диаграмме соответствует система $z_1 = \bar{z}_2 z_3 \vee z_2 z_3$, $z_2 = z_1 \bar{z}_3 \vee z_2 \bar{z}_1$, $z_3 = \bar{z}_1 z_2 \vee z_3 z_1$. Переменные z_2 и z_3 переключаются дважды за цикл, в то время как z_1 — четыре раза. Соответствующие функции возбуждения имеют вид:

$$Q_1^1 = \bar{z}_2 \bar{z}_3, \quad Q_1^2 = z_2 \bar{z}_3, \quad Q_1^3 = z_2 z_3, \quad Q_1^4 = \bar{z}_2 z_3,$$
$$Q_2^1 = z_1 \bar{z}_3, \quad Q_2^2 = z_1 z_3, \quad Q_3^1 = \bar{z}_1 z_2, \quad Q_3^2 = \bar{z}_1 \bar{z}_2.$$

Диаграмма переходов для переменных z_h^l приведена на рис. 5.30 *б*. Функции управления сдвигом таковы: $\psi_1^1 = z_2^2 z_3^2$, $\psi_1^2 = z_2^1 z_3^2$, $\psi_1^3 = z_2^1 z_3^1$, $\psi_1^4 = z_2^2 z_3^1$, $\psi_2^1 = z_1^1 z_3^2$, $\psi_2^2 = z_1^3 z_3^1$, $\psi_3^1 = z_1^2 z_2^1$, $\psi_3^2 = z_1^4 z_2^2$.

Основой аппаратной реализации дистрибутивных схем может служить схема распределителя, построенного на основе базовой ячейки (так называемой *ячейки Давида*), схема которой была приведена на рис. 5.3. Поведение этой ячейки во взаимодействии с соседями описывается следующей диаграммой переходов (дуги, направленные сверху вниз, опущены):

y_{j-1}	$\check{x}_{j-1}$	x_{j-1}	y_j	$\check{x}_j$	
→ 1*	1	0	1	1	
0	1	0*	1	1	— инициация j-й ячейки со стороны $(j-1)$-й,
0	1*	1	1	1	
0*	0	1	1	1	— j-я ячейка ответила $(j-1)$-й ячейке, что приняла инициирующий сигнал,
1	0	1	1*	1	— снятие инициирующего сигнала $(j-1)$-й ячейкой,
1	0	1	0	1*	— инициация $(j+1)$-й ячейки со стороны j-й,
1	0*	1	0	0	— $(j-1)$-я ячейка ответила j-й ячейке, что приняла инициирующий сигнал,
1	1	1*	0	0	
1	1	0	0*	0	
1	1	0	1	0*	— снятие инициирующего сигнала j-й ячейкой.

На рис. 5.32, *а* приведена конструкция, позволяющая реализовать простую дистрибутивную схему на элементах И-НЕ.

Каждой переменной z_j ставится в соответствие блок переменной V_j, состоящий из схемы B_j, которая служит для моделирования поведения самой переменной z_j, и распределителя S_j. Блок V_j содержит одинаковые по своей структуре блоки переключения V_j^i, замкнутые в кольцо. Блок переключения V_j^i содержит две базовые ячейки Давида типа, оказанного на рис. 5.3, которые обозначены через $D_{j,i}^1$ и $D_{j,i}^2$, инвертор, с выхода которого на дру-

гие блоки снимается значение переменной z_j^{i-1}, и элемент $\widetilde{\psi}_j^i$, на входах которого набирается конъюнкция ψ_j^i. Запись единицы ($x_{j-1}=1$, $\widetilde{x}_{j-1}=0$) в ячейку $D_{j,i}^1$ означает, что блок V_j готов к моделированию перехода (j, i), а на блоки других переменных

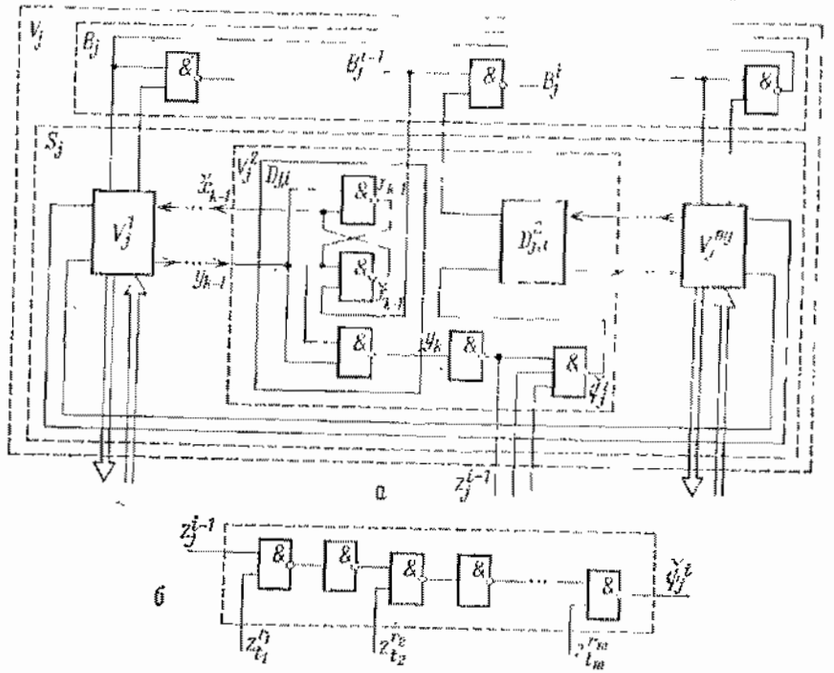


Рис. 5.32. К доказательству возможности реализации дистрибутивных схем в базисе 2И-НЕ-2: конструкция (*а*), реализация терма $\widetilde{\psi}_j^i$ (*б*)

с выхода z_j^{i-1} можно снимать информацию о завершении предыдущего перехода $(j, i-1)$. После того как конъюнкция ψ_j^i станет равной 1, в ячейку $D_{j,i}^2$ запишется единица, элемент B_j^i переключится из 0 в 1, что вызовет волну переключений всех элементов триггера B_j, которая завершается установкой $B_j^{i-1}=0$. Этот сигнал устанавливает $D_{j,i}^1$ в состояние 0, и происходит передача единицы из $D_{j,i}^2$ в $D_{j,i+1}^1$. Схема V_j готова к моделированию следующего перехода переменной z_j, и на другие блоки поступает информация о завершении перехода (j, i) (с выхода z_j^i).

Легко убедиться, что, поставив в соответствие каждой переменной z_j любую нечетную переменную схемы B_j (например, B_j^1), если в начальном состоянии исходной схемы значение z_j равнялось 0, и четную переменную B_j^2 в противном случае, получим схему, в точности моделирующую порядок срабатывания элементов исходной схемы.

Из рис. 5.32, *а* видно, что все элементы, кроме z_j^{i-1}, имеют нагрузку, равную двум. Вопрос об увеличении нагрузочной способности элемента z_j^{i-1} решается тривиально, путем включения в цепь z_j^{i-1} последовательности инверторов, число которых равно удвоенной нагрузке. Тем самым минимальная нагрузка на элемент сводится к двум.

Теперь естественно поставить *вопрос о минимальном числе входов элементов И-НЕ.*

Все элементы, кроме того, который реализует терм ψ_j^i, имеют не более двух входов. Элемент $\widetilde{\psi}_j^i$ может быть реализован на элементах 2И-НЕ-2 и инверторах так, как показано на рис. 5.32, *б*. При этом для обеспечения независимости поведения схемы от задержек элементов должен быть решен вопрос о допустимой последовательности вхождения переменных в терм. Действительно при $z_j^{i-1}=1$ переменная $z_{t_1}^{r_1}$ должна либо сохранять значение 1, либо (если она была равна 0) однократно изменить свое значение, совершив переход 0-1. Аналогичное требование предъявляется к переменной $z_{t_2}^{r_2}$ при $z_j^{i-1}z_{t_1}^{r_1}=1$ и т. д. Можно доказать, что для простых схем такое упорядочение всегда существует. Доказательство базируется на том факте, что в момент перехода 0-1 переменной z_j^{i-1} должна существовать по крайней мере одна переменная, входящая в терм ψ_j^i и не равная 1. В противном случае условие перехода уже выполнено. В силу того, что каждая из переменных $z_{t_l}^{r_l}$ изменяется в течение цикла работы дважды, переход 0-1 этой переменной будет однократным. Любая из переменных, отвечающих этому требованию, может быть выбрана первой. После того как $z_j^{i-1}z_{t_1}^{r_1}=1$, условие перехода либо выполнено, либо существует переменная, входящая в терм ψ_j^i и не равная 1, и т. д.

Таким образом, справедлива

Теорема 5.8. *Система собственных функций элементов 2И-НЕ-2 функционально полна в классе D_n^*, т. е. на элементах 2И-НЕ-2 может быть корректно реализована любая дистрибутивная схема.*

В самом деле, в соответствии с теоремой 5.7, по каждой дистрибутивной схеме C может быть построена дистрибутивная простая схема C', эквивалентная C. На

основе описанной выше конструкции по C' можно построить схему C^*, удовлетворяющую трем условиям: 1) C^* построена из элементов 2И-НЕ-2, 2) C^* полумодулярна и дистрибутивна и 3) C^* эквивалентна C по множеству Z переменных исходной схемы C.

Отметим, что в конструкции рис. 5.32, *а* между элементами, моделирующими поведение различных основных переменных (переменных исходной схемы), нет непосредственных связей. Однако выходы этих элементов можно использовать для связи с внешней средой, вставляя в разрывы проводов, соединяющих элементы V_j^i и V_j^{i+1}, согласованные апериодические реализации.

Теперь пришло время рассмотреть *вопрос о реализации схем из класса U_n^* в ограниченном базисе.* Было сказано, что достаточно исследовать реализации простых

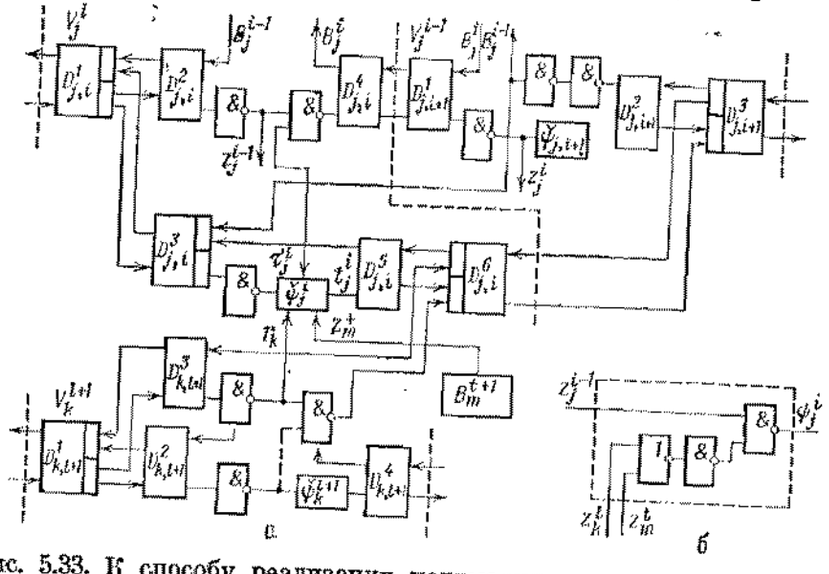


Рис. 5.33. К способу реализации полумодулярных схем в ограниченном базисе: вариант моделирования детонантного перехода (*а*), некорректная реализация функции возбуждения детонантного перехода (*б*)

полумодулярных схем, ранг детонантности которых не превосходит двух, а термы функций возбуждения детонантных переходов элементарны, т. е. имеют вид $\psi_j^i = z_k^\sigma \vee z_m^\sigma$.

Общая идея моделирования соответствует структурной схеме рис. 5.33, *а*. Отличия должны быть связаны

именно со способом реализации детонантных переходов, для которых ψ_j^i имеет вид $\psi_j^i = z_k^l \vee z_m^t$ и не может быть реализована по схеме рис. 5.33, *б*. Кажущееся очевидным решение, связанное с построением схемы, вычисляющей функцию $z_j^{i-1}(z_k^l \vee z_m^t)$, некорректно, ибо не позволяет обеспечить полумодулярность схемы. Таким образом, первая задача заключается в реализации функции ψ_j^i, соответствующей детонантному переходу (j, i). Вторая задача состоит в выборе способа гашения схемы, реализующей ψ_j^i. Для недетонантных переходов это гашение предшествовало снятию информации о срабатывании со входов блоков других переменных. Подобная организация недопустима для детонантных переходов. В самом деле, нет уверенности в том, что после совершения детонантного перехода (в отличие от недетонантного) оба входных для него перехода произошли; известно лишь, что имел место один из них. Процесс гашения, происходящий сразу после (j, i) может идти параллельно со вторым для (j, i) переходом, в результате чего может быть нарушена полумодулярность.

Идея реализации детонантного перехода (j, i) заключается в распараллеливании процессов в блоке V_j^i. Последний делается двухъярусным (рис. 5.33, *а*). Первый ярус, служащий для моделирования собственно перехода (j, i), состоит из распараллеливающей процесс ячейки — бифуркатора $D_{j,i}^1$, ячеек $D_{j,i}^2$ и $D_{j,i}^4$ и пары инверторов, с выхода первого из которых снимается значение z_j^{i-1}. Как и для дистрибутивных схем, снижение нагрузки z_j^{i-1} достигается использованием цепочек инверторов. Второй ярус, в котором «набирается» функция возбуждения, состоит из схемы $\tilde{\psi}_j^i$ и из ячеек $D_{j,i}^3$, $D_{j,i}^5$ и, возможно, $D_{j,i}^6$.

Из ячейки бифуркатора (рис. 5.34, *а*) единица переписывается в следующие за ней ячейки $D_{j,i}^2$, $D_{j,i}^3$. После того как произошла запись единицы в обе эти ячейки, происходит гашение 1 в бифуркаторе и затем распространение процессов по ярусам: на выходе z_j^{i-1} устанавливается значение 1, воспринимаемое в соответствующих блоках как сигнал о переходе $(j, i-1)$, открывая для восприятия входных переходов схему $\tilde{\psi}_j^i$. Реализация последней (рис. 5.34, *б*) содержит трехвходовый Г-триггер (рис. 5.34, *в*). Он реализует функцию $z = x_1x_2x_3 \vee z(x_1 \vee x_2 \vee x_3)$, где x_1, x_2, x_3 — входные, а z — выходная переменная триггера. После совершения хотя бы одного из переходов (k, l) или (m, t) $(z_k^l = 1$ или $z_m^t = 1)$ на одном из выходов τ_j^i схемы $\tilde{\psi}_j^i$ установится 1. В ячейку $D_{j,i}^4$ запишется 1, пройдет волна переключений элементов триг-

гера B_j, погасится 1 в $D^2_{j,i}$ и произойдет перезапись 1 из $D^4_{j,i}$ в первую ячейку следующего блока изменений $D^1_{j,i+1}$. Затем $z_j = 1$, и дальнейшие процессы идут, как было описано.

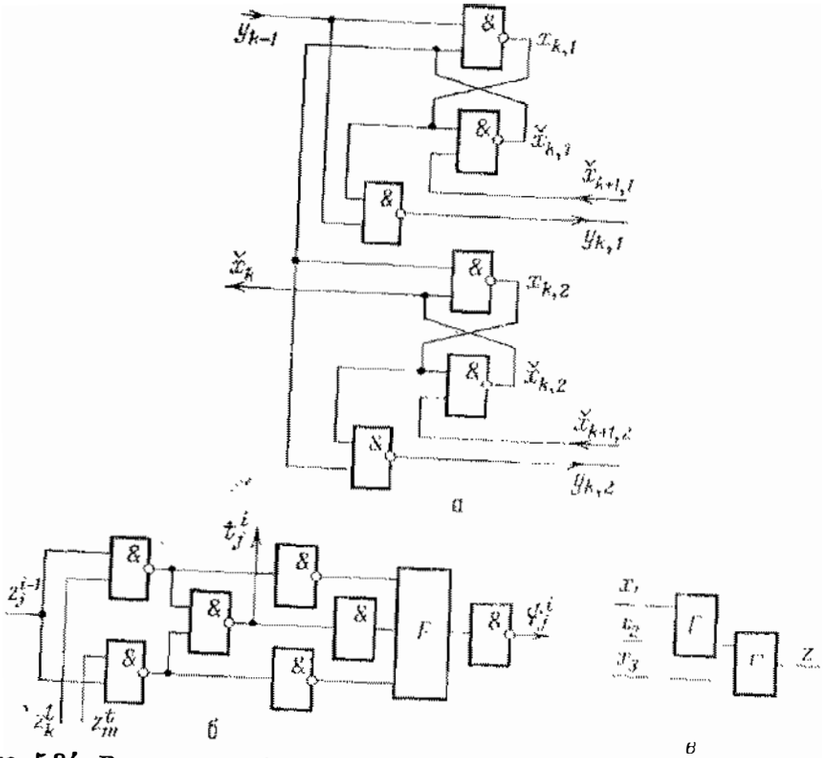


Рис. 5.34. Реализация базовых ячеек на элементах Давида: бифуркатор (*а*), схема $\check{\psi}^i_j$ с трехвходовым Г-триггером (*б*), реализация последнего на двухвходовых Г-триггерах (*в*)

Для пояснения процесса гашения схемы $\check{\psi}^i_j$ введем ряд новых понятий.

Определение 5.8. Переход (r, s) назовем *мажорантой детонантного перехода* (j, i), если: 1) (r, s) не может произойти раньше, чем произойдут оба входных для (j, i) перехода (k, l) и (m, t) и сам переход (j, i); 2) не найдется перехода (r', s'), удовлетворяющего условию 1 и предшествующего (r, s).

Вообще говоря, у детонантного перехода может быть несколько не упорядоченных между собой мажорант.

Переход $(j, i+l)$, $l>0$, будем называть *синхронизирующим* для детонантного перехода (j, i), если он — первый переход переменной z_j после (j, i), который не может произойти ранее всех мажорант перехода (j, i).

Иными словами, можно быть уверенными, что к моменту совершения синхронизирующего перехода все мажорантные переходы уже произошли, и уж подавно произошли оба входных для (j, i) перехода. Это позволяет синхронизировать процессы, распараллеленные в блоке детонантного перехода, в блоке соответствующего ему синхронизирующего перехода.

Введенные понятия иллюстрируются рис. 5.35, *а*. На рис. 5.33, *а* синхронизирующим для (j, i) является уже следующий переход

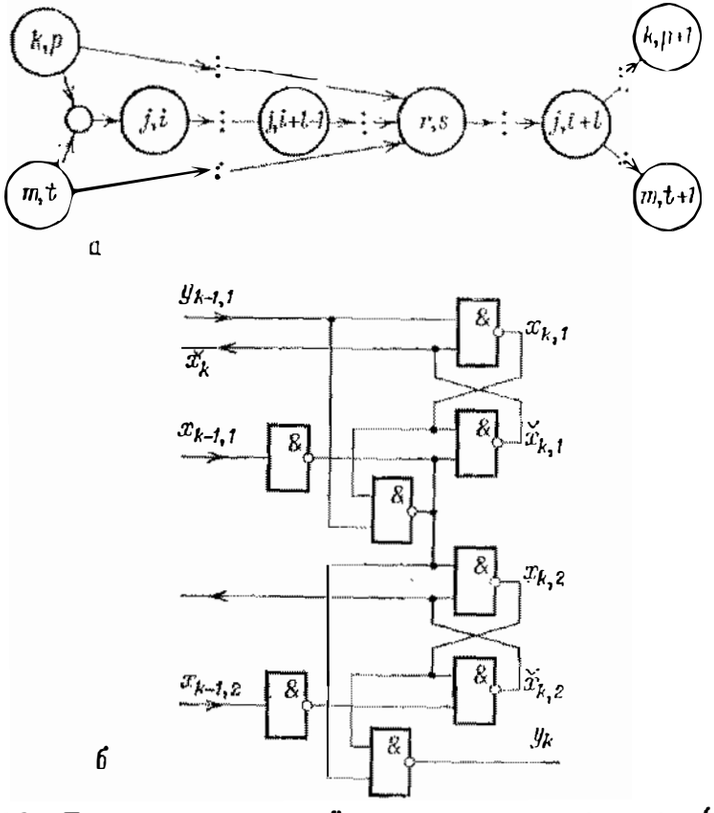


Рис. 5.35. Диаграмма изменений с детонантным переходом (*а*), реализация синхронизатора на ячейках Давида (*б*)

$(j, i+1)$. В блок V^{i+1}_j вводится дополнительная ячейка — синхронизатор $D^3_{j,i+1}$ (рис. 5.35, *б*). Запись 1 в синхронизатор происходит после того, как на обоих входах, соединенных с ячейка-

ми-предшественниками, установится 0. После записи обе названные ячейки гасятся, и 1 передается в следующую за синхронизатором ячейку блока V_j^{i+2}. Ячейка второго яруса $D_{j,i}^3$ делается по схеме бифуркатора.

Если следующие переходы $(k, l+1)$ и $(m, t+1)$ входных для перехода (j, i) переменных z_k и z_m следуют только после синхронизирующего для (j, i) перехода (как на рис. 5.35, *а*), то никаких дополнений в схему не вносится. Пусть теперь $(k, l+1)$ может произойти раньше синхроперехода. При этом установка $z_k^l = 0$ может произойти раньше, чем погасится схема $\breve{\psi}_j^i$, что делает последнюю схему не полумодулярной. Задержать переход $(k, l+1)$ до того момента, когда завершится гашение схемы $\breve{\psi}_j^i$, нельзя, так как это повлекло бы необходимость введения нарушающей исходную упорядоченность событий дополнительной синхронизации $(k, l+1)$ от (m, t). Корректное решение может быть получено за счет использования дополнительной ячейки $D_{k,l+1}^3$ в блоке V_k^{l+1}, играющей роль буфера, на котором специально для схемы $\breve{\psi}_j^i$ запоминается факт совершения перехода (k, l). Сброс этой информации не является необходимым условием для совершения следующего перехода $(k, l+1)$, он произойдет лишь после гашения $\breve{\psi}_j^i$.

Легко видеть, что в предлагаемой конструкции может потребоваться распараллелить или синхронизировать более двух процессов (в частности, в случае вложенных детонантных переходов). Соответствующие схемы могут быть построены в виде пирамидальных структур из бифуркаторов (рис. 5.34, *б*) или синхронизаторов (рис. 5.35, *б*). Отметим, что в том случае, когда номер синхронизирующего перехода $i+l$ по модулю m_j равен i, используется простой прием удвоения числа переходов переменной z_j.

Итак, предложенный метод реализации полумодулярных схем позволяет доказать следующую теорему.

Теорема 5.9. *Система собственных функций элементов 2И-НЕ-2 и 2ИЛИ-НЕ-2 функционально полна в классе U_n^*, т. е. на элементах 2И-НЕ-2 и 2ИЛИ-НЕ-2 может быть корректно реализована любая полумодулярная схема.*

Приведенный выше материал § 5.4 имел целью доказательство существования корректных реализаций полумодулярных схем с минимальными значениями коэффициентов разветвлений элементов по входам и выходам. Такая постановка с неизбежностью обусловила громоздкость предложенных схемных конструкций. Однако сам факт их существования дает разработчику уверенность в результативности процесса синтеза. Кроме того, предложенные схемы становятся существенно компактнее

с ростом допустимых значений коэффициентов разветвления элементов.

Вопрос о минимальности базиса реализации полумодулярных схем {2И-НЕ-2 + 2ИЛИ-НЕ-2}, т. е. *вопрос о том, нельзя ли реализовать любую полумодулярную схему на элементах 2И-НЕ-2 (2ИЛИ-НЕ-2), остается открытым.* Более того, открыт вопрос о возможности синтеза полумодулярных схем на элементах И-НЕ без учета ограничений по входам и выходам. Решение такой задачи оказалось бы возможным, если отказаться от требования полумодулярности реализации по всем элементам. Однако получаемые при этом схемы (полумодулярные по части переменных) могли бы корректно функционировать лишь на элементах с идеальными инерциальными собственными задержками. Очевидно, что построение таких элементов невозможно.

В заключение параграфа отметим два факта. Первый. Следствие теоремы 5.9 гласит: на элементах с собственными функциями типа $ab \vee c$ (трехвходовых элементах И-ИЛИ-НЕ) может быть корректно реализована любая полумодулярная схема из класса U_n^*. Второй факт: все результаты §§ 5.3 и 5.4, полученные для схем классов U^*, D^*, K^*, обобщаются на классы U, D, K без ограничений на вид множества рабочих состояний.

§ 5.5. Моделирование конвейерных процессов

Возвращаясь к определению 2.10 *конвейерного АП*, отметим, что в примере 2.7 i-е событие сети Петри рис. 2.8, *б* становится возможным лишь после того, как наступят $(i-1)$-е и $(i+1)$-е события. В соответствии с рис. 5.1, *а*, *б* и *д* легко получить схему, моделирующую эту сеть. Если учесть наличие двух фаз каждой задачи (рабочей фазы и фазы гашения), то упорядочение фаз тройки следующих друг за другом задач P_{i-1}, P_i и P_{i+1} (интерпретируемых в сети Петри как $(i-1)$-е, i-е и $(i+1)$-е события соответственно), задается диаграммой переходов, представленной на рис. 5.36. По таблице истинности для этой диаграммы можно получить уравнение i-го элемента моделирующей схемы

$$z_i = z_{i-1}\bar{z}_{i+1} \vee z_i(z_{i-1} \vee \bar{z}_{i+1}).$$

Непосредственная реализация этого уравнения и показана на рис. 5.37, *а*. Число инверторов может быть сокращено,

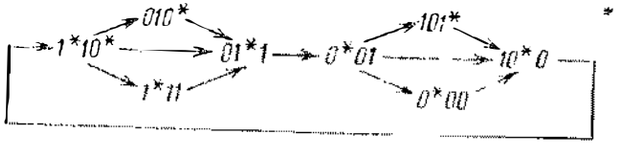


Рис. 5.36. Диаграмма переходов конвейерного процесса с переменными z_{i-1}, z_i, z_{i+1}

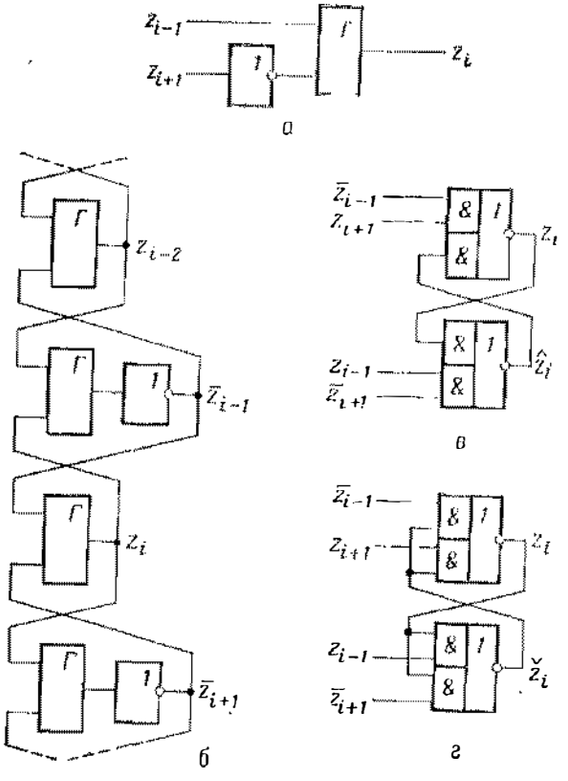


Рис. 5.37. Реализация конвейерного процесса: конструкция ячейки с Г-триггером и инвертором (*а*), способ соединения ячеек, экономящий число инверторов (*б*), конструкции ячеек на триггерах с раздельными входами (*в* и *г*)

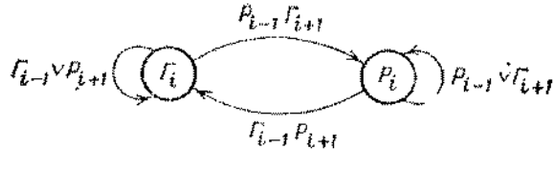


Рис. 5.38. Диаграмма переходов моделирующей схемы, соответствующей реализациям рис. 5.37

если перейти к схеме рис. 5.37, *б*. На рис. 5.37, *в* и *г* приведены конвейерные схемы на триггерах с нулевым и единичным транзитными состояниями соответственно. Время переключения этих триггеров, так же, как и Г-триггера, равно суммарной задержке двух элементов.

Для дальнейшего изложения будет удобно обозначить состояние $z_i = 1$ через P_i (рабочее состояние), а $z_i = 0$ — через Γ_i (состояние гашения) и задавать поведение моделирующей схемы в виде графа типа, показанного на рис. 5.38, соответствующего реализациям на рис. 5.37, *в* и *г*.

Рассмотрим некоторые свойства моделирующих конвейерных схем.

Событие r_2 на рис. 2.8, *б* моделирует приемник. Если он по каким-либо причинам «не работает», т. е. событие r_2 не наступает, то источник, моделируемый событием r_1, может «загрузить» в сеть не более четырех точек. В общем случае при неработающем приемнике источник может сработать n раз, т. е. n раз инициировать смену фаз конвейерного процесса. Поскольку каждая задача имеет две фазы, то всего при этом будет инициировано $n/2$ задач. Аналогично и приемник в свою очередь может «разгрузить» сеть при неработающем источнике, приняв из нее не более n точек. Отсюда следует, что *конвейерные моделирующие схемы обладают буферными свойствами.*

Важно отметить, что *темп поступления задач не зависит от числа событий в моделирующей конвейерной сети.* Для последовательного процесса (рис. 2.8, *а*) он равен nt, где n — число событий. Таким образом, при конвейерной организации процесса средняя скорость его протекания возрастает в $]n/2[$ раз по сравнению с неконвейерной организацией.

Попытаемся найти такие моделирующие схемы, на основе которых можно построить конвейерные устройства, допускающие «*плотное*» заполнение информацией, т. е. такие, что в сети из n событий может решаться n задач. В основу построения таких схем положим следующие рассуждения. Возьмем две идентичные схемы рис. 5.37, *в* или *г*. Пусть i-й элемент одной из них находится в рабочем состоянии Р, а другой — в состоянии гашения Г. Плотное заполнение этой пары достигается, если источник информации посылает точки попеременно то в одну, то в другую сеть, и эти точки продвигаются

по разным сетям. При композиции этой пары в одну необходимо учесть следующее обстоятельство. Во избежание нарушения требуемого порядка продвижения точек следует запретить комбинацию РР их состояний, которая возникает при обгоне одной точки другой. Остальные комбинации состояний этих сетей обозначим так: РГ-Р; ГР-Р*; ГГ-Г. Напомним, что в описанной модели переходы Г-Р и Г-Р* обязательно чередуются. С учетом этого обстоятельства граф переходов i-го элемента соответствующей моделирующей схемы имеет вид рис. 5.39, *а*;

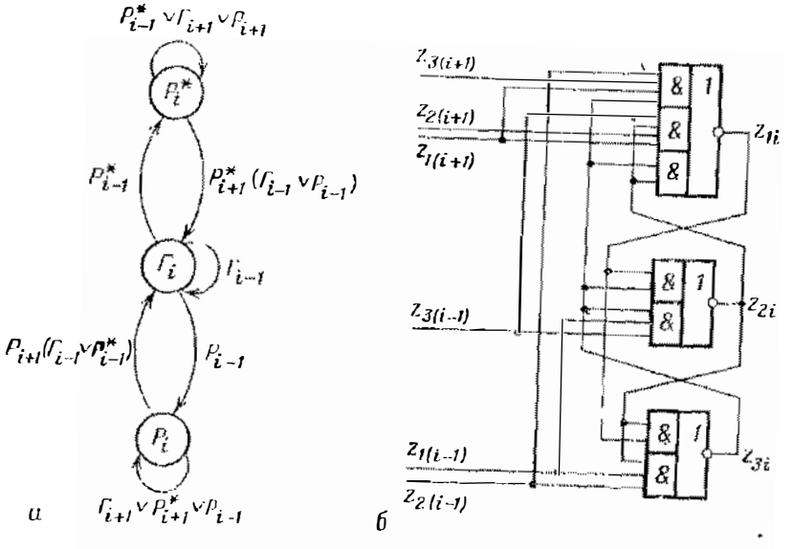


Рис. 5.39. Вариант ячейки «плотной» моделирующей схемы: граф переходов (*а*), реализация (*б*)

его полумодулярная реализация приведена на рис. 5.39, *б*. Состоянию Г в этой реализации соответствует набор 011 значений входов z_1, z_2, z_3, состоянию Р — набор 101, состоянию Р* — набор 110.

Моделирующую схему конвейерного типа с «плотным» заполнением информации можно получить и на основе другого подхода. Пусть сеть рис. 2.8, *б* содержит $2n$ событий, но только половина из них (например, события с четными номерами) активно используются (интерпретируются как решение n задач). Тогда $2i$-е событие, находящееся в фазе Р, должно перейти в фазу Г, если $(2i-1)$-е перешло в фазу Р. Изменим теперь граф переходов так, чтобы $2i$-е событие перешло в фазу Р после того, как в ту же фазу перейдет событие $2i+2$, минуя $(2i+1)$-е,— так, как показано на рис. 5.40, *а*. Граф на рис. 5.40, *б* для случая события с нечетными номерами отличается от него только в случае, когда события $2i$, $2i+1$ и $2i+2$ находятся в фазах Г, Р и Г соответственно; такая комбинация состояний в графе рис. 5.40, *а* возникнуть не может.

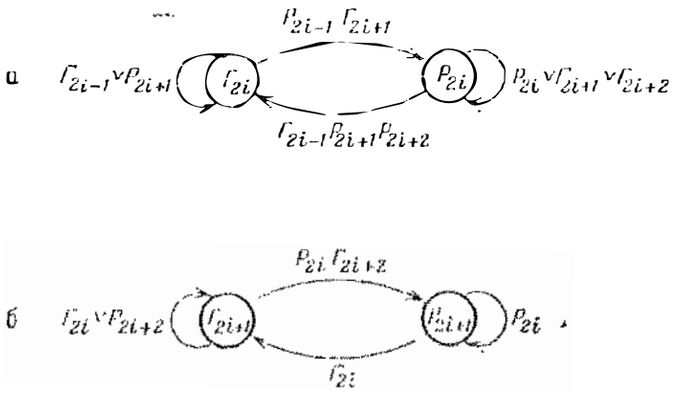


Рис. 5.40. Альтернативные варианты графов переходов ячеек с четными (*а*) и нечетными (*б*) номерами

Реализация графов рис. 5.40 может быть построена на основе триггеров, аналогичных приведенным на рис. 5.37, *в* и *г*.

В моделирующих схемах, построенных на основе графов рис. 5.38, комбинация фаз Г и Р, $2i$-го и $(2i+1)$-го элементов соответственно, является неустойчивой, т. е. переходит (независимо от состояний соседних с данной парой элементов) в комбинацию ГГ. Наличие этой неустойчивой комбинации снижает быстродействие моделирующих схем и в силу этого является нежелательным. Для ее устранения проведем композицию этой пары элементов (как и в предыдущем случае), обозначив комбинации состояний $2i$-го и $(2i+1)$-го элементов следующим образом: ГГ-Г, РГ-Р*, РР-Р. В результате получим граф переходов на рис. 5.41, *а*, полумодулярная реализация которого приведена на рис. 5.41, *б*. Состоянию Г этой реализации соответствует набор 001 значений переменных z_1, z_2, z_3, состоянию Р — набор 010, состоянию Р* — набор 100.

В сети рис. 2.8, *б*, представляющей конвейерный процесс, помимо цепочки условий, ориентированных в направлении распространения процесса по сети, содержится также цепочка условий, ориентированная в обратном направлении. Если некоторый не обязательно последовательный процесс задан сетью Петри, то, введя в эту сеть

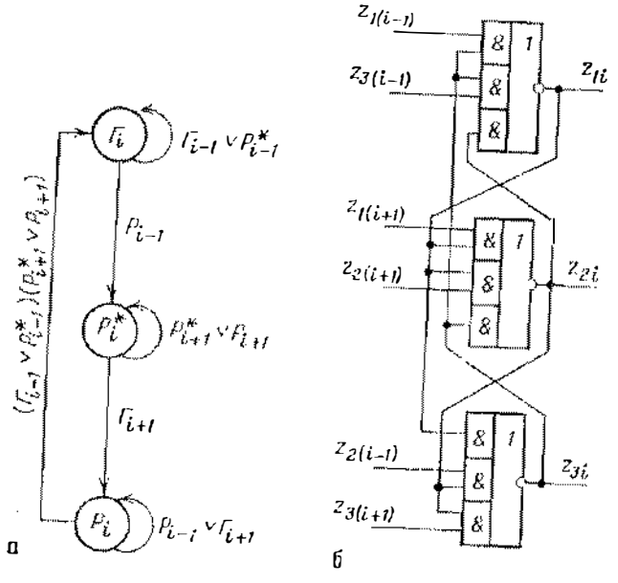


Рис. 5.41. Граф переходов, полученный в результате композиции графов рис. 5.40 (*а*) и конструкция соответствующей ячейки (*б*)

дополнительные цепочки условий, ориентированные в направлении, обратном распространению процесса, получим, как и в предыдущем случае, конвейерный АП с тем же упорядочением процессов, что и исходный. При этом увеличение скорости АП будет иметь место только в тех случаях, когда в исходной сети нет петель, содержащих менее четырех событий (условий).

Конвейерные эквиваленты фрагментов простых сетей Петри, реализация которых рассматривалась в § 5.2 (рис. 5.1), показаны на рис. 5.42. Построение моделирующих схем, соответствующих фрагментам рис. 5.42, осуществляется обычными методами, если сети Петри, представляющие эти фрагменты (так же, как и исходные), являются устойчивыми и безопасными. В общем случае

формальный переход к конвейерному процессу в соответствии с рис. 5.42 не гарантирует устойчивости полученной сети. Тогда в последнюю следует внести дополнительные условия (события), делающие ее устойчивой. Эти средства вытекают из анализа АП исходной сети.

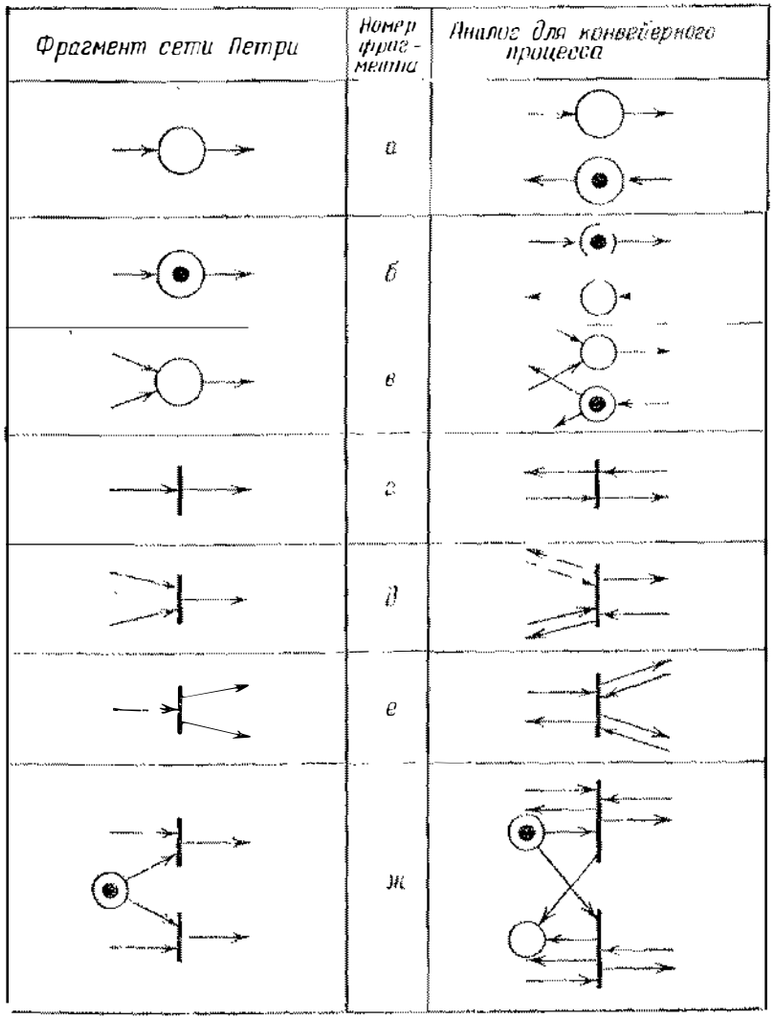


Рис. 5.42. Фрагменты сетей Петри и их конвейерные аналоги

Далее ставится задача построения по данному простому АП такого ее конвейерного аналога, который адекватен исходному процессу при гарантии максимальной эффективности решаемой задачи.

Поскольку проверка принадлежности автономного АП классу конвейерных сложна, воспользуемся интерпретацией сетей Петри. Изложение будет вестись на неформальном уровне с привлечением понятий устойчивости, безопасности и живости. Они были определены в § 2.2.

Преобразование последовательных участков было описано в примере 2.7.

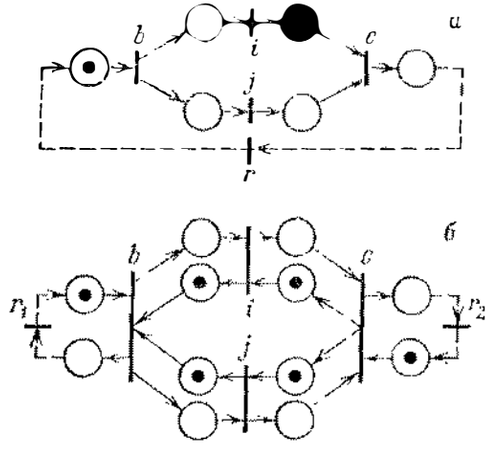


Рис. 5.43. Устойчивая и безопасная сеть Петри с реконвергентным участком (*а*) и ее конвейерный аналог (*б*)

Конвейеризация параллельных участков. Примером простейшей сети Петри может служить так называемая *реконвергентная сеть*, представленная на рис. 5.43, *а*. В ней события i и j интерпретируют *параллельно работающие* операторы. Тупиковая маркировка, содержащая только одну точку в выходном условии события c (называемого *конвергентным*), является результантом, а маркировка, содержащая точку только во входном условии события b (называемого бифуркантным),— инициатором. Тривиальная репозиция осуществляется с помощью события r. Эта устойчивая сеть безопасна. В соответствии с условием, изложенным в начале § 5.5 (событие возможно лишь после того, как наступили непосредственно предшествующие и непосредственно следующие события), по этой сети легко получить конвейерный аналог (рис. 5.43, *б*). Репозиция осуществляется посредством введения событий r_1 и r_2. Условия устойчивости и безопасности аналогичны случаю последовательного процесса. Очевидно, что конвейеризация исходного АП предполагает возможность увеличения числа одновременно решаемых задач. Обобщение на случай m ($m > 2$) параллельных ветвей тривиально.

Конвейеризация условного перехода. Пример АП, содержащего *условный переход*, приведен на рис. 5.44, *а*. Эта сеть устойчива относительно маркировок, при которых в условиях α и $\bar{\alpha}$ одновременно не содержатся точки. Сеть безопасна относительно приведенной на рис. 5.44, *а* начальной маркировки и двойственной ей (т. е. такой, при которой точка содержится в условиях a и $\bar{\alpha}$). Репозиция АП (тривиальная) осуществляется через событие r. Сеть Петри изображает в данном случае процесс решения только одной задачи; при этом используется только одна ветвь.

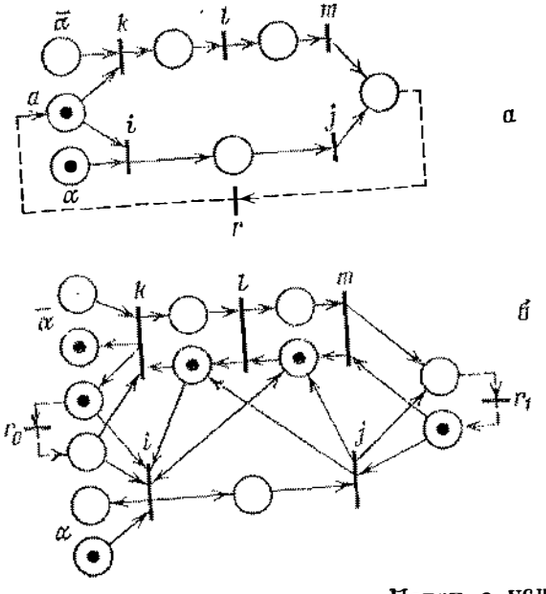


Рис. 5.44. Устойчивая и безопасная сеть Петри с условным переходом (*а*) и ее конвейерный аналог (*б*)

При нахождении конвейерного аналога в соответствии с принятым подходом возникают трудности, связанные с обеспечением устойчивости сети. Для объяснения этих трудностей воспользуемся следующей аналогией. Условную передачу управления естественно сравнить с железнодорожной стрелкой. Когда шарик подкатывается к стрелке, она должна направить его на один из двух путей. Что же может произойти дальше? Вследствие того, что шарики катятся с неодинаковой скоростью, а также из-за того, что разные ветви могут иметь разную длину, после развилки шарик может

обогнать предшествующий, катящийся по другой ветке. Таким образом, на месте конвергенции ветвей возможно как «столкновение» шариков, что соответствует нарушению устойчивости сети, так и «переформирование» состава из шариков, что соответствует нарушению исходной упорядоченности задач.

Идея, лежащая в основе построения устойчивой сети для конвейерного АП, состоит в синхронизации прохождения задач по параллельным ветвям аналогично тому, как это делается на участках железной дороги с малой пропускной способностью. Средством такой синхронизации может служить, например, цепочка дополнительных условий, вводимых, как и ранее, в исходную сеть для ее конвейеризации.

Результат преобразования исходной сети с условным переходом (рис. 5.44, *а*) приведен на рис. 5.44, *б*. Здесь пара событий, расположенных на одном ярусе в параллельных ветвях (например, j и m), имеют общее входное дополнительное условие, что обеспечивает синхронизацию прохождения задач на этих участках. При синхронизации событий с параллельной ветвью, содержащей более одного события (например, события i с участком k, l другой ветви), событие должно становиться возможным, если в этой ветви не находится на решении ни одна задача. Это требование выполняется, когда во всех дополнительных условиях участка сети содержатся точки. Синхронизируемые участки параллельных ветвей можно выбирать с учетом параметров реальной системы, оптимизируя тем самым ее производительность.

Другим способом синхронизации является введение оператора-буфера, в который записываются значения условия α при решении каждой очередной задачи. Эти значения позволяют считывать результат решения задачи в том порядке, в котором эти задачи поступили на решение. Длина буфера влияет на производительность системы: при числе разрядов буфера, не превышающем число операторов в самой короткой ветви, число задач, находящихся в системе, равно числу разрядов буфера.

Преобразование цикла. Пример сети Петри, содержащей *цикл* с фиксированным числом повторений, приведен на рис. 5.45, *а*, из которого видно, что в зависимости от значения условия α либо ($\alpha = 1$) повторяются события i, j, k, i, ..., либо ($\alpha = 0$) наступает событие l, после чего возможна репозиция процесса (через событие r). Эта сеть устойчива и безопасна для приведенной начальной маркировки или двойственной ей.

Вновь прибегнем к аналогии для пояснения возникающих при конвейеризации трудностей. Циклический участок сети представим как кольцевой желоб, в котором установлена железнодорожная

стрелка. В зависимости от положения стрелки шарик либо выкатывается из кольцевого желоба, либо продолжает путь по кольцу. Кроме того, в кольце может находиться более одного шарика, и максимальное их количество определяется длиной кольца. Если кольцо не пусто (содержит хотя бы один шарик), на выезде возможно столкновение вновь прибывающего шарика с находящимся в кольце, что, как уже говорилось, эквивалентно нарушению устойчивости сети. Идея, положенная в основу построения устойчивой сети с циклом, состоит в следующем. Въезд в кольцо должен

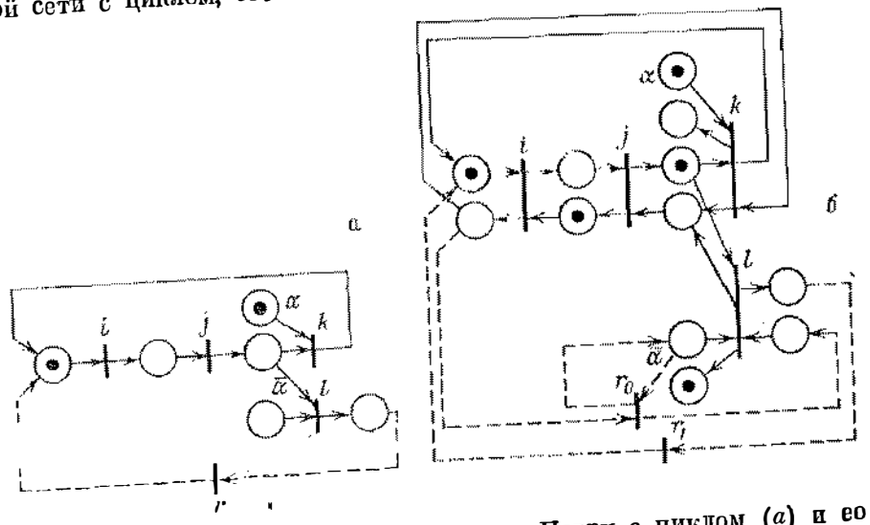


Рис. 5.45. Устойчивая и безопасная сеть Петри с циклом (*а*) и ее конвейерный аналог (*б*)

регулироваться той же стрелкой, что и выезд. Пока закрыт выезд из кольца, должен быть закрыт и въезд. Тогда, очевидно, столкновения шариков произойти не может. Въезд в кольцо (выезд из кольца) открывается, чтобы пропустить один шарик лишь после того, как стрелка разрешит одному из шариков покинуть кольцо (прибыть в кольцо). При этом число шариков в кольце остается постоянным.

Максимальная производительность конвейерной системы достигается при ее полузагрузке, т. е. когда число задач равно половине числа операторов. Поэтому при начальной установке кольцо должно быть загружено задачами (возможно, фиктивными) наполовину. Высказанная идея реализована сетью рис. 5.45, *б*.

Конвейеризация при повторных вхождениях. Реализация АП упрощается при наличии *повторных вхождений участков*. Рассмотрим простейший пример — рис. 5.46, *а*. В нем участок A сети используется после наступления как события j, так и события i. Условия b и

c служат для запоминания ветви, в которой очередной раз используется участок (адрес возврата). Заметим, что в принципе такая сеть может быть и неустойчивой, так как события i и j могут стать возможными одновременно, находясь, например, в параллельных участках сети. Для

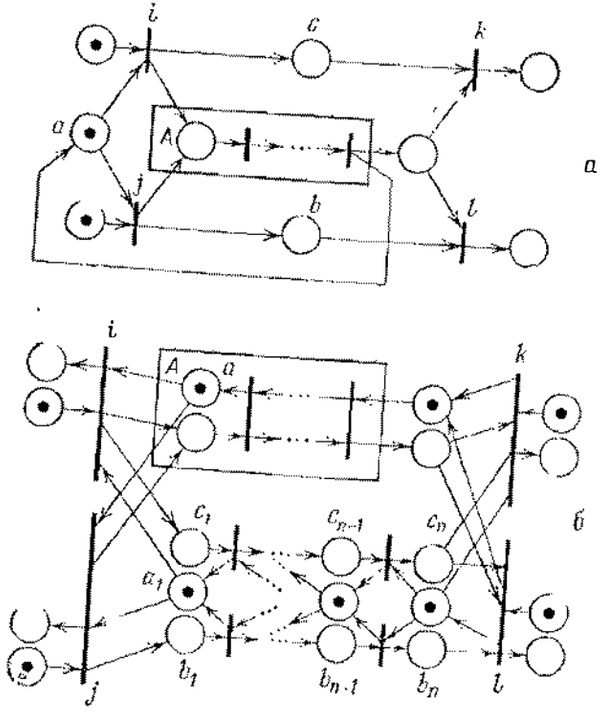


Рис. 5.46. Сеть Петри с повторным вхождением участка ($а$) и ее конвейерный аналог ($б$)

разрешения конфликтной ситуации в таких случаях используются синхропримитивы типа семафоров, реализуемых с помощью арбитров (гл. 9). В данном случае маркировка условия a не более чем одной точкой (безопасная маркировка) решает конфликт. Можно сказать, что неустойчивость данной сети Петри *локализована* в условии a. При конвейеризации АП потребуем, чтобы нарушение устойчивости сети было по-прежнему локализовано в одном месте. При построении соответствующей сети (рис. 5.46, $б$) предполагается, что участок A также конвейеризован. Для запоминания адреса возврата вместо пары условий b и c исходной сети используется последовательность условий b_1c_1, b_2c_2, ..., b_nc_n, которая совместно с соответствующими событиями моделирует работу двоичного буферного регистра. Число разрядов этого регистра должно быть равно числу участков ярусов участка A. Можно считать, что неустойчивость этой сети локализована в условиях a_1 и a.

В заключение этой главы следует отметить, что материал в основной его части тяготеет к принадлежащей М. А. Гаврилову идее блочного синтеза и его концепции стандартной реализации, состоящей в том, что по фрагментам языка блочного описания непосредственно из некоторого набора базисных элементов (обладающего функциональной полнотой) строится схема. Этот подход характеризуется малой трудоемкостью процедур синтеза, возможно, в ущерб экономичности аппаратуры.

С другой стороны, главным методологическим приемом, положенным В. М. Глушковым в основу проектирования ЭВМ, является разделение ее на операционную и управляющую части, хотя такое разделение носит относительный характер: необходимо учитывать характер взаимодействия и особенно вопросы согласования времени. Свойства апериодических схем позволяют довести эту идею до логического завершения. В данном случае роль управляющего устройства выполняет моделирующая схема, а блоки операционной части вставляются в разрывы соединяющих элементы проводов моделирующей схемы. При этом подходе синтез управляющего устройства ведется автономно от разработки операционных блоков, что упрощает композицию устройства.

§ 5.6. Замечания по библиографии

Идея реализации фрагментов сетей Петри стандартными модулями (по событиям) высказана в [225, 269]. Группой французских ученых, работающих в Центре аэрокосмических исследований в Тулузе, был предложен набор (схемно реализованных) модулей и методика перехода от сетей Петри к схемам из этих модулей [197, 211, 301]. Следует отметить, однако, что авторы не ставили перед собой задачу обеспечить принадлежность получаемых реализаций классу полумодулярных схем, так что непосредственное использование их результатов не представляется возможным. Тем не менее конструкция одного из модулей, так называемой ячейки Давида [197], была модифицирована, в результате чего удалось предложить распределитель типа, показанного на рис. 5.3 [27, 40]. Вопросы универсальности другого (гипотетического) набора модулей (физическая реализация которых, по-видимому, может быть выполнена, если привлечь, ту же интерпретацию, что использова-

лась при реализации сетей Петри по условиям), рассматривалась также в [206].

Подходы к реализации параллельных асинхронных блок-схем (на основе идеи блочного синтеза М. А. Гаврилова [69]) и конвейерных процессов были предложены в [6] и развиты затем в [53, 62, 307].

Конструкция блока повторного вхождения, представляющая собой усовершенствованный вариант приведенной в [6], описана в [56].

Использованное в § 5.4 понятие функциональной полноты отличается от общеизвестного [74], хотя и не противоречит ему. Результаты этого параграфа опубликованы в [6, 64, 178, 180] и базируются на классификации схем, не зависящих от скорости, описанной в [120]. Доказательство теоремы 5.9 содержится в [64].

В [249] был предложен набор не зависящих от скорости универсальных модулей (с памятью) и поставлен вопрос о том, можно ли сократить число модулей и уменьшить общее число их входов и выходов. Теоремы 5.8 и 5.9 дают, по-видимому, окончательный ответ на этот вопрос (если отказаться от требования учета чувствительности схемы к задержкам в проводах). Другой (достаточно громоздкий) способ реализации предлагается в [274], но он не гарантирует полумодулярности синтезируемой схемы по переменным, не встречавшимся в исходной диаграмме переходов.

Не вошедший в книгу материал, касающийся синтеза и анализа антитонных линейных (последовательных) схем, описан в серии статей [161] и для более широкого класса схем в книге [163]. Этот подход основан на языке циклограмм (и его развитии — языке таксограмм), использующем идею представления информации в изменениях сигналов, предложенную М. А. Гавриловым (таблицы включения) [71]. Линейная запись циклограмм родственна предложенной А. А. Ляпуновым [110]. Указанный подход использует также материал книги [143].

Каждый на свою стенку лезет,
а истина лежит, между тем,
внизу, у всех под ногами.

Ф. Кривин

Глава 6. КОМПОЗИЦИЯ АСИНХРОННЫХ ПРОЦЕССОВ И СХЕМ

Современные ЭВМ и дискретные системы строятся на основе модульного (агрегатного) принципа: аппаратные средства, равно как и программное обеспечение, выполняются в виде отдельных функциональных и структурных единиц (модулей). Последние объединяются (компонуются, комплексируются, агрегатируются), образуя сложные системы.

Поскольку в книге рассматривается класс обладающих определенной спецификой самосинхронизирующихся систем, необходим адекватный аппарат, который, прежде всего, позволяет строить описание функционирования системы по описаниям ее фрагментов. Такой аппарат основан на введении определенных операций над композируемыми асинхронными процессами и отличается от обычно используемых в автоматной проблематике методов композиции учетом динамических особенностей процессов.

В эту главу выделен материал, касающийся как методов композиции асинхронных процессов и апериодических схем, так и методов преобразования асинхронных схем в рамках модели Маллера.

§ 6.1. Композиция асинхронных процессов

6.1.1. Приведенный процесс. Возвращаясь к определениям 2.2 АП и 2.8 его репозиции, введем следующее определение.

Определение 6.1. *Приведенным асинхронным процессом* назовем процесс $P^{\pi} = \langle S^{\pi}, F^{\pi}, I^{\pi}, R^{\pi} \rangle$ такой, что

$S^{\text{п}} = S \cup S^{\text{д}}$, $F^{\text{п}} = F \cup (F' \setminus (F' \cap (S' \times I)))$, $I^{\text{п}} \subseteq I$, $R^{\text{п}} = R \cup S^{\text{д}}$.

Иными словами, приведенный процесс является объединением АП и его репозиции, в котором из отношения F выброшены пары, задающие переходы к инициаторам процесса (образующие пары $F' \cap (S' \times I)$).

В зависимости от того, какую репозицию имеет процесс P, соответствующий приведенный процесс $P^{\text{п}}$ будем называть *полностью* или *частично приведенным*. Если для АП репозиции не существует, то он называется *неприводимым*. Для приведенного процесса со структурированными путем выделения входной и выходной компонент ситуациями понятия результанта и инициатора можно конкретизировать, например, следующим образом. Отнесем ситуацию к результантам, если в ситуациях, принадлежащих всем траекториям, проходящим через данную ситуацию, после нее сохраняется то же значение выходной компоненты, что и в самой этой ситуации. К инициаторам отнесем те ситуации, которые образованы из результантов такой заменой значений входной компоненты, что полученная ситуация не относится к результантам.

Если некоторый процесс можно рассматривать как модель совместного функционирования нескольких подпроцессов, то структурными единицами (компонентами) ситуаций процесса могут являться ситуации подпроцессов.

Пример 6.1 (продолжение примера 2.6). Возможная репозиция процесса P, заданного сетью Петри рис. 2.7:

P: $S' = \{C, \phi, A, B, AB\}$, $F' = \{(C, \phi), (C, A), (C, B), (C, AB), (\phi, A), (\phi, B), (\phi, AB), (A, AB), (B, AB)\}$, $I' = \{C\}$, $S^{\text{д}} = \{\phi, A, B\}$, $R' = \{AB\}$.

Приведенный процесс $P^{\text{п}}$: $S^{\text{п}} = \{A, B, C, \varnothing, AB\}$, $F^{\text{п}} = \{(AB, C), (C, \phi), (C, A), (C, B), (\phi, A), (\phi, B)\}$, $I^{\text{п}} = \{AB\}$, $R^{\text{п}} = \{C, \phi, A, B\}$. Строго говоря, этот приведенный процесс описывается не одной, а четырьмя сетями Петри на рис. 2.7 с начальными маркировками AB, $\varnothing$, A, B.

6.1.2. Редукция процесса. В этом пункте будет описана операция редукции, суть которой состоит в сведении данного АП к более простому. Очевидно, что такая операция необходима тогда, когда из полного описания процесса хочется выделить некоторую его часть, рассмотрение которой интересно по тем или иным причинам.

Пусть задан неприведенный АП $P = \langle S, F, I, R \rangle$, ситуации которого структурированы так, как это было сделано в последнем абзаце п. 2.1.4, т. е. множество ситуаций S представимо упорядоченной тройкой $S = (X, Y, Z)$, где X, Y и Z — соответственно множества значений (символов) входной компоненты, выходной компоненты и компоненты, не являющейся ни входной ни выходной.

Образуем p-блочное разбиение π множества S ситуаций процесса P, в ситуациях каждого блока которого входная компонента принимает фиксированное значение x_j, $1 \leqslant j \leqslant p$. Выберем $r < p$ различных значений входной компоненты (составляющих множество $X^* \subset X$). Ситуации, входящие в те блоки разбиения π, которые соответствуют выбранным значениям входной компоненты, составляют подмножество S^*, $S^* \subset S$.

Для каждого инициатора $s_i \in I$ построим множество ситуаций $S(s_i)$, встречающихся на траекториях процесса P, ведущих из указанного инициатора. Образуем множество $S(X^*)$ как объединение тех множеств $S(s_i)$, для которых справедливо $S(s_i) \leqslant S^*$, т. е.

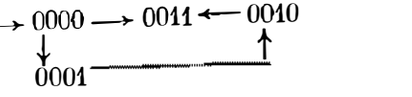
$$S(X^*) = \bigcup_{S(s_i) \subseteq S^*} S(s_i).$$

Построим также $F(X^*) = F \cap (S(X^*) \times S(X^*))$, $I(X^*) = I \cap S(X^*)$, $R(X^*) = R \cap S(X^*)$.

Определение 6.2. Назовем процесс $P(X^*) = \langle S(X^*), F(X^*), I(X^*), R(X^*) \rangle$ *редукцией* неприведенного процесса $P = \langle S, F, I, R \rangle$ по выбранному множеству X^* значений входной компоненты.

Аналогично определяется редукция $P(Y^*)$ неприведенного процесса по выбранному множеству Y^* значений выходной компоненты.

Пример 6.2. Пусть процесс P задан диаграммой переходов



в которой звездочки опущены. Здесь $S = \{0100, 0000, 0010, 0001, 0011, 1011, 1010\}$, $I = \{0100, 0010, 1011\}$, $R = \{0011, 1010\}$. Два первых элемента вектора полного состояния выберем в качестве входной компоненты. Тогда редукцией процесса P по множеству $X^* = \{00, 10\}$ будет процесс $P(X^*)$ с диаграммой полных состояний

причем $S^* = \{0000, 0010, 0001, 0011, 1011, 1010\}$, $S(X^*) = \{0010, 0011, 1011, 1010\}$, $I(X^*) = \{0010, 1011\}$, $R(X^*) = \{0011, 1010\}$.

Пример 6.3 (продолжение примера 2.7). Если для сети на рис. 2.8, *б* положить, что A является входным условием, а E — выходным, то конвейерный процесс можно описать как приведенный. Инициаторами в нем будут ситуации с точкой в кружке A, а результатами — ситуации с точкой в E или без точек в кружках A и E.

Подобным же образом сеть на рис. 2.8, *б* с учетом ранее оговоренной дисциплины может быть описана как приведенный процесс с выходным условием H и выходным D. Если же в качестве входных выбрать одновременно условия A и H, а выходных — E и D, выделить множества инициаторов и результантов не удается.

Из правил построения редукции следует, что редукция $P(X^*)$ процесса P может быть определена не на всем множестве X^*, а на некотором подмножестве X^{**}, $X^{**} \subset X$, так как при построении редукции из исходного процесса исключаются не отдельные ситуации, а пучки траекторий. В общем случае $X^{**} \subset X$. Аналогичное утверждение справедливо для $P(Y^*)$. Легко видеть, что для любого X', $X^* \supseteq X' \supseteq X^{**}$, $P(X') = P(X^{**}) = P(X^*)$.

Рассмотрим теперь репозицию редукции $P(X^*)$ процесса P. Образуем p-блочное разбиение π' ситуаций S' процесса P', каждый блок которого соответствует фиксированному значению x_j, $1 \leqslant j \leqslant p$, входной компоненты. Образуем множество S'^* как объединение тех блоков π', в ситуациях которых $x_j \in X^*$. Далее для каждого инициатора $i'_j \in R(X^*)$ процесса P' и каждого результанта $r'_m \in I(X^*)$ рассмотрим множество ситуаций $S'_k(i'_j, r'_m)$, принадлежащих k-й траектории из i'_j в r'_m.

Построим:

1) множество $S'(X^*) = \bigcup_{S'_k(i'_j, r'_m) \subset S'^*} S_k(i'_j, r'_m)$;

2) отношение $F'(X^*)$, задающее следование ситуаций на этих траекториях, $F'(X^*) = F \cap (S'(X^*) \times S'(X^*))$;

3) множества $I'(X^*) = I' \cap S'(X^*)$, $R'(X^*) = R' \cap S'(X^*)$.

Процесс $P'(X^*) = \langle S'(X^*), F'(X^*), I'(X^*), R'(X^*)\rangle$ и будет искомой *репозицией редукции* $P(X^*)$ процесса P.

Аналогично по репозиции P' процесса P строится репозиция редукции $P(Y^*)$. Редукция приведенного процесса $P^{\text{п}}$ строится по редукции неприведенного процесса P и репозиции редукции $P'(X^*)$.

Очевидны следующие утверждения (см. определения 2.3 и 2.4).

1. *Редукция неэффективного процесса может быть эффективным процессом.*

2. *Редукция эффективного процесса (если она существует) — всегда эффективный процесс.*

3. *Редукция управляемого процесса (если она существует) — всегда управляемый процесс.*

4. *Редукция частично приведенного процесса может быть полностью приведенным процессом, а редукция полностью приведенного процесса может оказаться частично приведенным или даже неприводимым процессом.*

6.1.3. Композиция процессов. Здесь предлагаются операции над АП, позволяющие композировать процессы.

Рассмотрим два АП — процесс $P_1 = \langle S_1, F_1, I_1, R_1\rangle$ (не обязательно приведенный) и приведенный процесс $P_2^{\text{п}} = \langle S_2, F_2, I_2, R_2\rangle$, ситуации которых структурированы: в ситуациях P_1 выделена выходная, а в ситуациях $P_2^{\text{п}}$ — входная компоненты. Пусть все или некоторые значения выходной компоненты y^1 ситуаций из S_1 соответствуют всем или некоторым значениям входной компоненты x^2 ситуаций из S_2. Обозначим проекции множества пар соответствующих друг другу значений компонент на множества Y_1 и X_2 через Y_1^* и X_2^* соответственно. По множествам Y_1^* и X_2^* построим редукции $P_1(Y_1^*)$ и $P_2^{\text{п}}(X_2^*)$ процессов P_1 и P_2. Пусть $P_2(X_2^*)$ — полностью приведенный процесс и $Y_1^{**} = Y_1^*$, $X_2^{**} = X_2^*$.

Построим, если это возможно, процесс $P_3 = \langle S_3, F_3, I_3, R_3\rangle$, ситуации которого представимы в виде пар $s^3 = (s^1, s^2)$, такой, что:

1) $s^1 \in S_1(Y_1^*)$, $s^2 \in S_2(X_2^*)$, т. е.

$$S_3 \subseteq S_1(Y_1^*) \times S_2(X_2^*);$$

2) выходная компонента y^1 ситуации s^1 равна входной компоненте x^2 ситуации s^2 — $y^1 = x^2$;

3) если в s^3 компонента $s^2 \in I_2(X_2^*)$, то $s^1 \in R_1(Y_1^*)$;

4) если $(s_i^1, s_j^2) F_3 (s_k^1, s_l^2)$, то либо $(s_i^1 F_1 s_k^1) \,\&\, (s_j^2 F_2 s_l^2)$, либо $(s_i^1 F_1 s_k^1) \,\&\, (s_j^2 = s_l^2)$, либо $(s_i^1 = s_k^1) \,\&\, (s_j^2 F_2 s_l^2)$.

Определение 6.3. Назовем *последовательной композицией* асинхронных процессов P_1 и $P_2^{\text{п}}$ асинхронный процесс P_3, образованный отождествлением значений входной и выходной компонент ситуаций редукционированных процессов $P_1(Y_1^*)$ и $P_2(X_2^*)$ соответственно при выполнении перечисленных выше ограничений 1)—4).

Смысл ограничения 1) состоит в том, чтобы при функционировании процесса $P_2^{\text{п}}$ в составе процесса P_3 в редукции $P_2^{\text{п}}(X_2^*)$ не встречались ситуации, не принадлежащие множеству $S_2(X_2^*)$. Ограничение 4) говорит о том, что для редукции $P_2^{\text{п}}(X_2^*)$ процесса P_2 в составе P_3 не может возникнуть новых траекторий.

Из определения 6.4 следует, что:

1) поскольку ситуации $s^3 \in S_3$ строятся из пар вида $(s^1 = (x^1, y^1, z^1), s^2 = (x^2, y^2, z^2))$, то компоненты $s^3 = (x^3, y^3, z^3)$ определяются как $x^3 = x^1$, $y^3 = y^2$, $z^3 = (z^1, y^1 = x^2, z^2)$, $S_3 \subseteq S_1(Y_1^*) \times S_2(X_2^*)$, причем проекции S_3 на $S_1(Y_1^*)$ и $S_2(X_2^*)$ равны соответственно $S_1(Y_1^*)$ и $S_2(X_2^*)$;

2) $I_3 \subseteq (I_1(Y_1^*) \times R_2(X_2^*)) \cap S_3$, причем проекции I_3 на $I_1(Y_1^*)$ и $R_2(X_2^*)$ равны соответственно $I_1(Y_1^*)$ и $R_2(X_2^*)$;

3) $R_3 \subseteq (R_1(Y_1^*) \times R_2(X_2^*)) \cap S_3$, причем проекции R_3 на $R_1(Y_1^*)$ и $R_2(X_2^*)$ равны соответственно $R_1(Y_1^*)$ и $R_2(X_2^*)$.

Рассмотрим два асинхронных (не обязательно приведенных) процесса P_1 и P_2 со структурированными ситуациями: выделенными входными компонентами x^1 и x^2. Пусть все или некоторые значения входной компоненты x^1 ситуаций из S_1 семантически эквивалентны всем или некоторым значениям входной компоненты x^2 ситуаций из S_2. Обозначим проекции эквивалентных пар значений компонент на множества X_1 и X_2 через X^*. По X^* построим редукции $P_1(X^*)$ и $P_2(X^*)$ процессов P_1 и P_2. Пусть $X_1^{**} = X_2^{**} = X^*$.

Построим, если это возможно, процесс P_3, ситуации $s_3 \in S_3$ которого представимы в виде пар $s^3 = (s^1, s^2)$ таких, что:

1) $s^1 \in S_1(X^*)$, $s^2 \in S_2(X^*)$, т. е. $S_3 \subseteq S_1(X^*) \times S_2(X^*)$;

2) входные компоненты x^1 и x^2 ситуаций s^1 и s^2 эквивалентны: $x^1 = x^2 = x$.

Определение 6.4. Назовем *параллельной композицией* асинхронных процессов P_1 и P_2 процесс P_3, образованный отождествлением значений входных компонент ситуаций редуцированных процессов $P_1(X^*)$ и $P_2(X^*)$ при выполнении ограничений 1)—2).

Рассмотрим теперь приведенный АП $P^{\text{п}} = \langle S^{\text{п}}, F^{\text{п}}, I^{\text{п}}, R^{\text{п}} \rangle$, соответствующий некоторому процессу P и его репозиции P'. Пусть ситуации процесса $P^{\text{п}}$ структурированы: в них выделены входная x и выходная y компоненты и все или некоторые значения выходной компоненты y ситуаций этого процесса соответствуют всем или некоторым значениям входной компоненты x ситуаций того же процесса. Обозначим проекцию множества эквивалентных пар значений компонент на множества X и Y через X^* и Y^* соответственно. По множествам X^* и Y^* построим редукции $P^{\text{п}}(X^*)$ и $P^{\text{п}}(Y^*)$ процесса P.

Построим автономный процесс $P^{\text{з}}(X^*, Y^*) = \langle S^{\text{з}}, F^{\text{з}} \rangle$, удовлетворяющий следующим ограничениям:

1) $S^{\text{з}} \subseteq S^{\text{п}}(X^*) \cap S^{\text{п}}(Y^*)$;

2) выходная компонента $y^{\text{з}}$ ситуации $s^{\text{з}}$ эквивалентна компоненте $x^{\text{з}} - y^{\text{з}} = x^{\text{з}}$;

3) $F^{\text{з}} = (F(X^*) \cup F'(X^*)) \cap (F'(Y^*) \cup F(Y))$, где F и F' задают причинно-следственные связи для неприведенного процесса и его репозиции, соответствующих процессу $P^{\text{п}}$.

Определение 6.5. Автономный процесс $P^{\text{з}}$, удовлетворяющий ограничениям 1)—3), назовем *замыканием* процесса $P^{\text{п}}$.

Поскольку замыкание определено только для приведенных процессов, необходимым условием существования замыкания последовательной или параллельной композиции является принадлежность композируемых процессов классу приведенных.

Введенный аппарат позволяет рассматривать АП произвольного вида как суперпозицию последовательной композиции, параллельной композиции и замыкания композируемых процессов. Тем самым возможна композиция процессов, заданных на любом из языков, интерпретируемых в терминах асинхронных процессов.

Пример 6.4. На рис. 6.1, *а* задан процесс P_1, в структурированных ситуациях которого выделена входная компонента y^1 (пятая и шестая позиции ситуации), а на рис. 6.1, *б* — процесс P_2 с выделенной входной компонентой x^1 (вторая и третья позиции ситуации). Репозиция процесса P_2 показана на рис. 6.1, *в*, а автономный процесс, образованный объединением процесса P_2 и его репозиции P_2' — на рис. 6.1, *г*. Приведенный процесс, как было сказано выше, получается из автономного процесса исключением путей, ведущих непосредственно в инициатор P_2, т. е. в данном случае на рис. 6.1, *г* исключается дуга, помеченная перекрестием. Если y^1 эквивалентна x^2, то последовательная композиция P_1 и $P_2^{\text{п}}$ будет иметь вид, показанный на рис. 6.1, *д*. В ситуациях процесса,

образованного композицией, первые четыре позиции суть соответствующие позиции процесса P_1, пятая и шестая — компонента $y^1 = x^2$, седьмая и восьмая — первая и вторая позиции ситуаций из P_2, девятая и десятая — пятая и шестая позиции того же процесса.

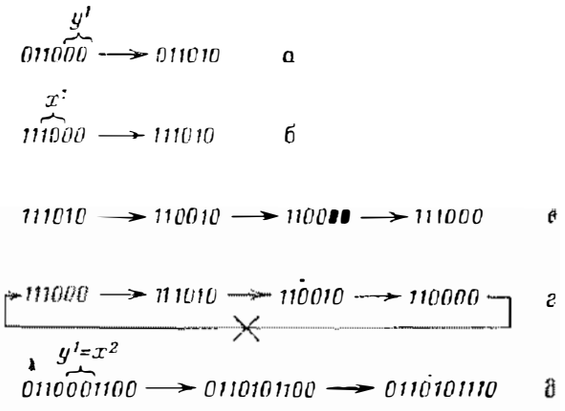


Рис. 6.1. Пример композиции двух процессов: *а* — первый процесс с выходной компонентой y^1; *б* — второй процесс с входной компонентой x^2; *в* — репозиция второго процесса; *г* — автономный процесс, образованный вторым процессом и его репозицией, и приведенный процесс (в нем переход 110 000-111 000 отсутствует); *д* — результат композиции

Отдельно следует рассмотреть вопросы композиции апериодических схем.

§ 6.2. Композиция апериодических схем

Возвратимся к определению 2.25. Смысл его состоит в следующем. Наборы входов каждого элемента z_i схемы разделяются на два класса — возбуждающие элемент при $z_i = 1$ и при $z_i = 0$ соответственно. В апериодической схеме класс входных наборов не может быть сменен, пока элемент не перейдет в устойчивое состояние. Поэтому для каждого элемента функционирование, как и всей схемы, происходит также по принципу «запрос — ответ».

6.2.1. Теорема Маллера. Специфика апериодических схем позволяет использовать методы композиции, нетипичные для синхронных и асинхронных схем. Один из таких методов дает приведенная ниже теорема, фигурирующая в литературе под названием *теоремы соединения* или *теоремы Маллера*. Прежде чем привести ее формулировку, условимся о следующих обозначениях. Запись

$x = x(x_0, X)$ означает, что элемент x реализует некоторую булеву функцию от x_0 и некоторых других аргументов из множества X (каких именно — несущественно). Обозначение $x^* = x(x_0 \equiv c, X)$ имеет следующий смысл: вход x_0 отключен от элемента x, а вместо него подключен вход c, в результате чего элемент реализует новую функцию x^*.

Теорема 6.1. *Пусть имеются две апериодические схемы A и B такие, что в схеме A можно выделить элемент y_0 и элементы $y_j = y_j(y_0, Y)$, $1 \leqslant j \leqslant m$, а в схеме B — инвертор z_0 и элементы $z_i = z_i(z_0, Z)$, $1 \leqslant i \leqslant k$. Тогда схема C, представляющая собой композицию схемы A и схемы B без инвертора z_0, такая, что $y_j^* = y_j(y_0 \equiv \overline{z}_0, Y)$, $z_i^* = z_i(z_i \equiv y_0, Z)$, $1 \leqslant j \leqslant m, 1 \leqslant i \leqslant k$, также является апериодической.*

Смысл теоремы иллюстрируется рис. 6.2, *а*. В схеме A разрывается шина, связывающая выход элемента y_0 со входами элементов $y_1, \ldots, y_m$, а в схеме B удаляется инвертор,

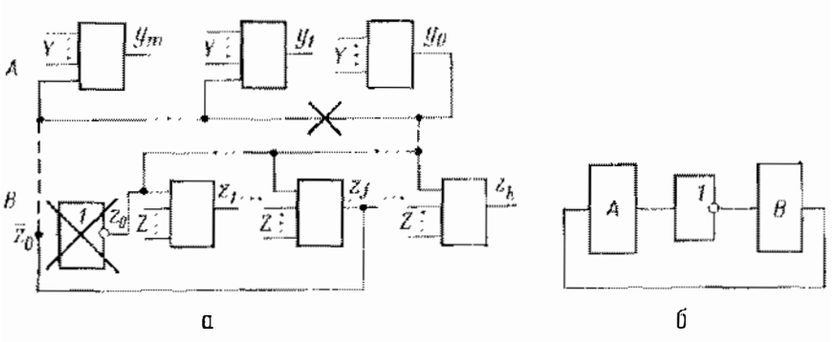


Рис. 6.2. Иллюстрация теоремы Маллера для случаев, когда в одной из композируемых схем имеется (*а*) или отсутствует (*б*) инвертор

после чего его вход $\overline{z}_0$ соединяется со входами элементов $y_1, \ldots, y_m$, а выход y_0 — со входами элементов $z_1, \ldots, z_k$. Полученная схема является апериодической.

Недостатком композиции, описываемой теоремой 6.1, является необходимость наличия инвертора в одной из схем. Однако существует искусственный прием, позволяющий обойти это ограничение. Включим в схеме B два инвертора последовательно с некоторым элементом. Очевидно, что поведение остальных элементов схемы B от этого не изменится, так как такое включение эквивалент-

но увеличению задержки соответствующего элемента и не отразится на работоспособности схемы. Теперь по теореме 6.1 один из этих инверторов можно удалить, получив в результате композицию, состоящую из A и B с дополнительным инвертором (рис. 6.2, *б*).

6.2.2. Обобщения теоремы Маллера. Будем в дальнейшем полагать, что принята гипотеза о характере задержек элементов, изложенная в п. 4.1.1.

Введем еще одно обозначение, после чего перейдем к доказательству теоремы, являющейся обобщением теоремы 6.1. Запись $x^* = x(x_0 = f, X)$ означает, что элемент x заменен элементом x^*, реализующим функцию, полученную подстановкой функции f вместо аргумента x_0 в функции x. Например, пусть $x = x_0x_1 \vee \bar{x}_1\bar{x}_2$, $f = \bar{x}_1y_1 \vee \bar{x}_1\bar{y}_2$. Тогда $x^* = (\bar{x}_1y_1 \vee \bar{x}_1\bar{y}_2)x_1 \vee \bar{x}_1\bar{x}_2 = \bar{x}_1\bar{x}_2$.

Теорема 6.2. *Пусть имеются две апериодические схемы A и B такие, что в схеме A можно выделить элемент y_0 и элементы $y_j = y_j(y_0, Y)$, $1 \leqslant j \leqslant m$, а в схеме B — элемент z_0 и элементы $z_i = z_i(z_0, Z)$, $1 \leqslant i \leqslant k$. Тогда схема C, представляющая собой композицию схемы A и схемы B без элемента z_0, такая, что*

$$y_j^* = y_j(y_0 = \bar{z}_0, Y), \quad z_i^* = z_i(z_0 \equiv y_0, Z),$$
$$1 \leqslant j \leqslant m, \ 1 \leqslant i \leqslant k,$$

также является апериодической.

Доказательство. Представим элемент z_0 в виде последовательного соединения безынерционного логического элемента, реализующего функцию $\bar{z}_0$, и инвертора, обладающего произвольной задержкой. Возможность такого представления при принятой гипотезе о характере задержек очевидна. Теперь, в соответствии с теоремой 6.1, при композиции может быть изъят инвертор, и на образовавшемся выходе схемы B будет реализована функция $\bar{z}_0$. После отсоединения выхода элемента y_0 от входов элементов $y_1, \ldots, y_m$ в схеме A эти элементы реализуют некоторые функции $y_j(y_0 \equiv c, Y)$, где через c обозначен оставшийся пока неподключенным общий вход этих элементов. Последние также могут быть представлены в виде последовательного соединения безынерционных элементов, реализующих функции $y_j(c, Y)$ или $\bar{y}_j(c, Y)$ с обладающими произвольными конечными задержками встроенными элементами задержки или инверторами соответственно. Соединение входа c с выходом безынерционного элемента $\bar{z}_0$ позволяет перейти к безынерционным элементам, реализующим функции $y_j^* = y_j(c = \bar{z}_0, Y)$ или их инверсии (если они допускают реализацию одним элементом). Возвращаясь вновь к инерционным функциональным элементам и замыкая связь y_0 с общими входами элементов $z_1, \ldots, z_k$, получим утверждение теоремы.

Итак, по теореме 6.2 в схеме A разрывается шина, связывающая выход элемента y_0 со входами элементов $y_1, \ldots, y_m$, последние заменяются элементами с собственными функциями y_j^*. В схеме B удаляется элемент z_0, образовавшиеся выходные шины соединяются с соответствующими входами элементов y_j^*, а выходная шина — с выходом y_0 схемы A.

Важно отметить, что в результате сложность композированной схемы (скажем, число элементов или общее число входов/выходов схемы) может оказаться меньше суммарной сложности композируемых схем, так как происходит удаление элемента z_0 и замена элементов y_j. Во-вторых, схема, образованная в результате такого рода композиции, может обладать большим быстродействием, чем схемы, композируемые каким-либо иным методом. Эти эффекты в совокупности, как правило, недостижимы в традиционной схемотехнике.

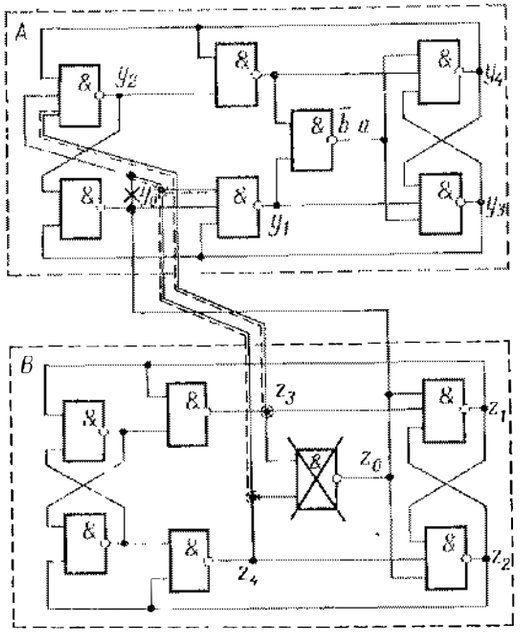


Рис. 6.3. Способ построения счетчика из двух счетных триггеров

Пример 6.5. Рассмотрим два апериодических счетных триггера A и B (рис 6.3), повторяющие схемы на рис. 4.12, *б* и отличающиеся только тем, что в первом из них (A) шины a и b запроса и ответа разорваны, а во втором (B) — замкнуты так, что образована

автономная апериодическая схема. Целью композиции A и B является синтез апериодического двухразрядного счетчика в соответствии с теоремой 6.2. Выполняется синтез следующим образом.

В схеме A разрывается связь, идущая от выхода элемента y_0 ко входам элементов $y_1 = y_0y_3$ и $y_2 = y_0y_4$ (до разветвления). В схеме B удаляется элемент $z_0 = z_3z_4$. Теперь элемент y_0 следует соединить со входами элементов z_1 и z_2, а элементы y_1 и y_2 заменить элементами y_1^* и y_2^*:

$$y_1^* = y_1(y_0 = \bar{z}_0) = \overline{z_3z_4y_0},\quad y_2^* = y_2(y_0 \equiv \bar{z}_0) = \overline{z_3z_4y_4}.$$

Таким образом, двухвходовые элементы y_1 и y_2 заменяются трехвходовыми. Новые связи в счетчике показаны штриховыми линиями.

По сравнению с обычным счетчиком (из триггеров рис. 4.12, *б* — см. ниже рис. 11.2) предложенный имеет более простой индикатор (при числе разрядов, превышающим два) и, следовательно, более экономичен. Более того, он обладает большим быстродействием, поскольку запуск второго и последующих разрядов происходит с выхода элемента y_0 и соответствующих ему элементов других разрядов, а не с элемента y_1, что имеет место в аналоге.

Особенностью многих апериодических схем является парафазное представление (входных, выходных и внутренних) сигналов.

Определение 6.6. Апериодическую схему будем называть *монотранзитной*, если при любом переходе ее входных переменных выходные и внутренние переменные имеют одно и то же транзитное состояние ($x_i = \hat{x}_i = 0$ — нулевое транзитное состояние или $x_i = \check{x}_i = 1$ — единичное транзитное состояние), и *совершенной*, если транзитные состояния входов ее элементов не вызывают их возбуждения.

Для совершенных апериодических схем, к которым, в частности, относятся рассмотренные в гл. 5 совершенные реализации, можно предложить методы композиции, вытекающие из следующих теорем.

Теорема 6.3. *Пусть имеются две апериодические схемы A и B такие, что в схеме A можно выделить совершенную подсхему с парафазным выходом $\tilde{y}_0$ и совершенные подсхемы с парафазными выходами $y_j = y_j(y_0, \tilde{y}_0, Y)$, $\tilde{y}_j = \tilde{y}_j(y_0, \tilde{y}_0, Y)$, $1 \leqslant j \leqslant m$, а в схеме B — совершенную подсхему с парафазными выходами $z_i = z_i(z_0, \tilde{z}_0, Z)$, $\tilde{z}_i = \tilde{z}_i(z_0, \tilde{z}_0, Z)$, причем все перечисленные подсхемы имеют одинаковые транзитные состояния. Пусть также в устойчивом состоянии $z_0 = \tilde{x}_0$, $\tilde{z}_0 = x_0$. Тогда схема C, представляющая собой композицию схемы A и схемы B без подсхемы $(z_0\ \tilde{z}_0)$ такая, что*

$$y_j^* = y_j(y_0 \equiv x_0,\ \tilde{y}_0 \equiv x_0,\ Y),\quad \tilde{y}_j^* = \tilde{y}_j(y_0 \equiv x_0,\ \tilde{y}_0 \equiv \tilde{x}_0,\ Y),$$
$$z_i^* = z_i(z_0 \equiv y_0,\ \tilde{z}_0 \equiv y_0,\ Z),\quad \tilde{z}_i^* = \tilde{z}_i(z_0 \equiv y_0,\ \tilde{z}_0 \equiv y_0,\ Z),$$
$$1 \leqslant j \leqslant m,\quad 1 \leqslant i \leqslant k,$$

также является апериодической.

Очевидно, что эта теорема является аналогом теоремы 6.1 для совершенных реализаций, причем вместо инвертора здесь выступает совершенная подсхема с выходами z_0 и $\tilde{z}_0$. Поэтому доказательство теоремы 6.3 следует из теоремы 6.1 с учетом того факта, что транзитное состояние входов не вызывает возбуждения ни одного из элементов композиции. В качестве совершенной подсхемы, в частности, может фигурировать RS-триггер на элементах ИЛИ-НЕ (И-НЕ), имеющий нулевое (единичное) транзитное состояние.

Предложим еще один метод композиции.

Теорема 6.4. *Пусть имеются две апериодические схемы A и B такие, что в схеме A можно выделить совершенную подсхему с парафазным выходным сигналом $\tilde{y}_0$, y_0 и триггеры $\tilde{y}_j$, $1 \leqslant j \leqslant m$, с функциями возбуждения S_j, R_j, $S_j = S_j(y_0, \tilde{y}_0, Y)$, $R_j = R_j(y_0, \tilde{y}_0, Y)$, а в схеме B — триггер $\tilde{z}_0$ с функциями возбуждения S_0 и R_0 и совершенные подсхемы $(z_i, \tilde{z}_i)$, $1 \leqslant i \leqslant k$, $z_i = z_i(z_0, \tilde{z}_0, Z)$, $\tilde{z}_i = \tilde{z}_i(z_0, \tilde{z}_0, Z)$, причем все перечисленные подсхемы и триггеры имеют одинаковые транзитные состояния. Тогда схема C, представляющая собой композицию схемы A и схемы B без триггера $(z_0, \tilde{z}_0)$, такая, что $S_j^* = S_j(y_0 = R_0, \tilde{y}_0 = S_0, Y)$, $R_j^* = R_j(y_0 = R_0, \tilde{y}_0 = S_0, Y)$, $z_i = z_i(z_0 \equiv y_0, \tilde{z}_0 \equiv y_0, Z)$, $\tilde{z}_i = \tilde{z}_i(z_0 \equiv y_0, \tilde{z}_0 \equiv \tilde{y}_0, Z)$, $1 \leqslant j \leqslant m$, $1 \leqslant i \leqslant k$, также является апериодической.*

Теорема 6.4 является аналогом теоремы 6.2 для случая совершенной реализации, и доказательство ее следует из теорем 6.2 и 6.3.

Пример 6.6. На рис. 6.4 изображены схемы двух апериодических счетных триггеров A и B, каждый из которых содержит по три RS-триггера. Функции возбуждения триггеров $\hat{y}_1$ и $\hat{y}_3$ схемы A и $\hat{y}_6$ схемы B имеют вид $S_1 = \overline{y_2y_3}$, $R_1 = \overline{y_2y_3}$, $S_3 = y_1 \oplus y_2$, $R_3 = y_1 \oplus \bar{y}_2$, $S_6 = y_4 \oplus y_5$, $R_6 = y_4 \oplus \bar{y}_6$. Для композиции A и B, согласно теореме 6.4, из B исключим триггер $\hat{y}_6$, а в A разорвем выходные шины триггера $\hat{y}_2$. Тогда функции возбуждения триггеров $\hat{y}_1$ и $\hat{y}_3$ должны иметь вид $S_1^* = \overline{y_3(y_4 \oplus y_5)}$, $R_1^* = y_3(y_4 \oplus \bar{y}_5)$,

$S_3^* = y_1 \oplus y_4 \oplus y_5$, $R_3^* = y_1 \oplus y_4 \oplus \bar{y}_5$, а на входы триггеров $\hat{y}_4$ и $\hat{y}_5$ вместо $\hat{y}_6$ подаются выходы триггера $\hat{y}_2$. В результате получим двухразрядный счетчик, содержащий пять *RS*-триггеров вместо шести.

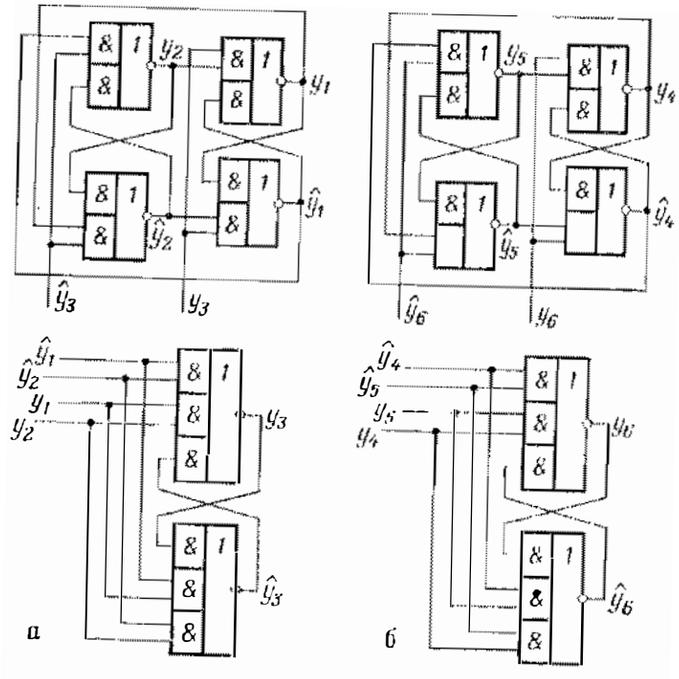


Рис. 6.4. Композиция счетных триггеров исключением совершенной подсхемы

Итак, рассмотрены методы композиции апериодических схем, не типичные для синхронных и асинхронных. Все эти методы по существу являются *обобщениями теоремы Маллера*. Заметим, однако, что приведенные способы не покрывают всех возможностей соединения апериодических схем. Например, за рамками параграфа осталась проблема соединения совершенных реализаций с разными типами транзитных состояний.

§ 6.3. Алгебра асинхронных схем

Рассмотренные в § 6.2 методы композиции апериодических схем, как правило, связаны с увеличением размерности (числа переменных) описания результирующей схемы по сравнению с композируемыми. Здесь будет

рассмотрен подход, особенность которого состоит в том, что размерности композируемых схем и результата композиции одинаковы. Иначе говоря, речь идет о преобразованиях схем, которые могут быть использованы в процессе проектирования асинхронных схем для устранения выявленных в результате анализа (гл. 8) нежелательных особенностей их поведения. Схемы рассматриваются как алгебраические объекты, на множестве которых введены алгебраические операции, с помощью которых и осуществляется преобразование.

6.3.1. Операции над схемами. Пусть асинхронные схемы (далее в § 6.3 — просто схемы) заданы на одном и том же множестве переменных $Z = \{z_1, z_2, \ldots, z_n\}$. Очевидно, мощность множества таких схем равна $2^{n \cdot 2^n}$. Пусть двум схемам S_1 и S_2 из этого множества соответствуют системы булевых уравнений

$$z_i = f_{1i}(z_1, \ldots, z_i, \ldots, z_n) \text{ и } z_i = f_{2i}(z_1, \ldots, z_i, \ldots, z_n),$$
$$i = 1, \ldots, n,$$

или для краткости $z_i = f_{1i}(Z)$ и $z_i = f_{2i}(Z)$.

Определение 6.6. *Объединением* S_{1+2} (*пересечением* $S_{1.2}$) схем S_1 и S_2 будем называть такую схему, в которой для любой пары соседних состояний (диаграммы переходов) a и b отношение $a \xrightarrow{1} b$ имеет место тогда и только тогда, когда это отношение имеет место хотя бы для одной схемы S_1 или S_2 (для обеих схем S_1 и S_2).

Наряду с обозначениями S_{1+2} и $S_{1.2}$ будем также пользоваться такими: $S_{1+2} = S_1 + S_2$, $S_{1.2} = S_1 \cdot S_2$.

Последнее определение эквивалентно следующему.

Определение 6.6*. Объединением S_{1+2} (пересечением $S_{1.2}$) схем S_1 и S_2 будем называть такую схему, в которой каждая переменная z_i возбуждена в тех и только тех состояниях, в которых она возбуждена хотя бы в одной схеме S_1 или S_2 (в обеих схемах S_1 и S_2).

Поскольку для схемы S условием возбуждения ее переменной z_i при $z_i = 1$ является $\bar{f}_i(Z) = 1$, а при $z_i = 0$ — $f_i(Z) = 1$, то имеет место

$$z_i = z_i f_i(Z) \vee \bar{z}_i f_i(Z). \qquad (6.1)$$

Из (6.1) и определения 6.6* непосредственно следует: для S_{1+2}

$$z_i = f_{(1+2)i}(Z) = z_i f_{1i}(Z) f_{2i}(Z) \vee \bar{z}_i (f_{1i}(Z) \vee f_{2i}(Z)), \qquad (6.2)$$

для $S_{1.2}$

$$z_i = f_{(1.2)i}(Z) = \bar{z}_i f_{1i}(Z) f_{2i}(Z) \vee z_i (f_{1i}(Z) \vee f_{2i}(Z)). \quad (6.3)$$

Заметим, что в объединении S_{1+2} схем S_1 и S_2 в отношение следования могут быть вовлечены такие пары несоседних состояний, которые в этом отношении не находились ни в S_1, ни в S_2. Покажем это. Пусть, например, в $S_1(S_2)$ имеет место $a \xrightarrow[z_i]{1} b \left(a \xrightarrow[z_j]{1} c\right)$, и в состоянии a переменная $z_j(z_i)$ устойчива. Пусть также в S_1 и S_2 не существует такого d, что $a \to d$. Тогда в S_{1+2} в состоянии a возбуждены обе переменные z_i и z_j и существует такое состояние d, что имеет место $a \xrightarrow[z_i, z_j]{1} d$.

Пример 6.7. Пусть схемы S_1 и S_2 заданы на множестве переменных $\{z_1, z_2, z_3\}$ системами уравнений

$$z_1 = z_2 \vee z_3, \quad z_2 = z_2\bar{z}_3, \quad z_3 = z_1 \vee \bar{z}_2 z_3 \quad (6.4)$$

и

$$z_1 = \bar{z}_3, \quad z_2 = z_1\bar{z}_3, \quad z_3 = z_1 z_2. \quad (6.5)$$

Диаграммы переходов для этих схем показаны на рис. 6.5, *а* и *б*. Пересечение $S_{1.2}$ и объединение S_{1+2} задаются в силу (6.3) и (6.4) системами

$$z_1 = z_1 \vee z_2\bar{z}_3, \quad z_2 = z_2\bar{z}_3, \quad z_3 = z_1 z_2 \vee \bar{z}_2 z_3$$

и

$$z_1 = \bar{z}_1 \vee z_2\bar{z}_3, \quad z_2 = z_1\bar{z}_2, \quad z_3 = z_1 z_2 \vee z_1\bar{z}_3$$

соответственно. Диаграмма переходов для пересечения показана на рис. 6.5, *в*; она содержит только дуги, общие для диаграмм 6.5, *а* и *б*. Диаграмма объединения должна содержать кроме дуг, общих для диаграмм составляющих схем, еще и дуги между несоседними состояниями, находящимися в отношении следования в схеме S_{1+2}. Вследствие громоздкости она не приводится.

Помимо бинарных, введем также унарные операции над схемами, целью которых является попытка заменить отношение $a \to b$ для пар состояний исходной схемы отношением $b \to a$. Сразу же оговоримся, что такая замена в принципе не всегда возможна, хотя бы в силу того, например, что число r, $0 \leqslant r \leqslant 2^n - 1$, состояний, предшествующих данному, может быть четным, в то время как число состояний, следующих за данным, равно $2^m - 1$, где m — число возбужденных в данном состоянии переменных, обязательно нечетное.

Определение 6.7. *Обращением* (*инверсией*) схемы S будем называть такую схему $\neg S(\bar{S})$, в которой для любой пары соседних состояний a и b отношение $a \xrightarrow{1} b$ имеет место тогда и только тогда, когда в схеме S состояния

a и b находятся в отношении $b \xrightarrow{1} a$ (a не предшествует непосредственно состоянию b).

Пусть в S имеет место $a \xrightarrow[z_i]{1} b$, причем для определенности $a_i = 0$. Тогда по определению обращения $b_i = 1$, т. е. $f_i(z_i = 0) = 1$. И наоборот, если $a_i = 1$, то $b_i = 0$, т. е. $f_i(z_i = 1) = 0$. В силу этого для обращения $\neg S$ схемы S имеет место

$$z_i = \neg f_i(Z) = z_i \bar{f}_i(z_i = 0) \vee \bar{z}_i \bar{f}_i(z_i = 1), \quad i = 1, \ldots, n. \quad (6.6)$$

Унарная операция обращения заменяет отношение $a \to b$ в точности на обратное только для пар соседних

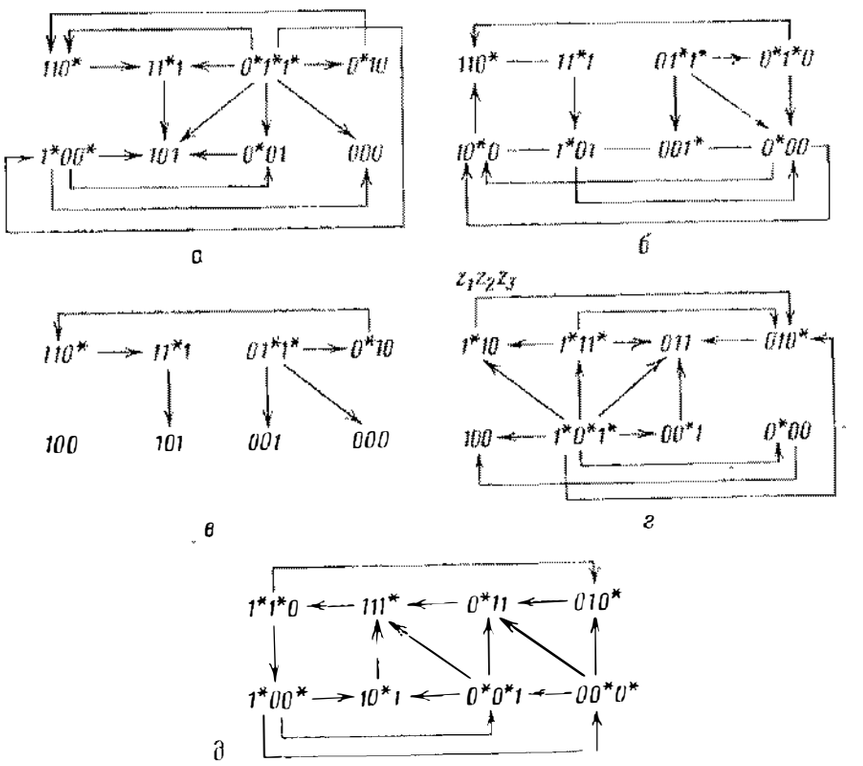


Рис. 6.5. Диаграммы переходов исходных схем (*а*, *б*), их пересечения (*в*) и обращений (*г*, *д*)

состояний. При этом в схеме $\neg S$ в отношение следования могут оказаться вовлеченными пары несоседних состояний, для которых это не имело места в схеме S, и наоборот, несоседние состояния, находящиеся в отношении

следования в S, могут оказаться не состоящими в отношении следования в схеме $\neg S$.

Из определения 6.7 очевидно также, что если в схеме S переменная z_i возбуждена (устойчива) в состоянии a, то в инверсии $\bar{S}$ в этом состоянии переменная должна быть устойчива (возбуждена). Поэтому схема $\bar{S}$ задается системой уравнений вида

$$z_i = \bar{f}_i(Z), \tag{6.7}$$

т. е. получается из системы уравнений исходной схемы инверсий их правых частей.

Пример 6.7 (продолжение). Уравнения для $\neg S_1$ и $\neg S_2$ схем S_1 и S_2 получаются из исходных (6.4) и (6.5) использованием (6.6):

$$z_1 = \bar{z}_2\bar{z}_3, \quad z_2 = z_2 \vee z_3, \quad z_3 = \bar{z}_1 z_2 \vee \bar{z}_1 z_3$$

и

$$z_1 = z_3, \quad z_2 = \bar{z}_1 \vee z_3, \quad z_3 = \bar{z}_1 \vee \bar{z}_2$$

соответственно. Диаграммы переходов этих схем показаны на рис. 6.5, *г* и *д*. Уравнения для $\bar{S}_1$ и $\bar{S}_2$ получаются из (6.4) и (6.5) применением (6.7) Для $\bar{S}_1$ получим

$$z_1 = \bar{z}_2\bar{z}_3, \quad z_2 = \bar{z}_2 \vee z_3, \quad z_3 = \bar{z}_1 z_2 \vee \bar{z}_1\bar{z}_3,$$

а инверсия $\bar{S}_2$ совпадает с ее обращением $\neg S_2$.

6.3.2. Законы и свойства. Используя выражения (6.2), (6.3), (6.6) и (6.7) для уравнений объединения, пересечения, обращения и инверсии, легко убедиться, что для введенных бинарных операций выполняются законы коммутативности, ассоциативности, идемпотентности, поглощения и дистрибутивности, а для унарных — закон инволюции. Для инверсии справедливы правила де Моргана $\overline{S_1 + S_2} = \bar{S}_1 \cdot \bar{S}_2$, $\overline{S_1 \cdot S_2} = \bar{S}_1 + \bar{S}_2$, а для обращения аналогичные правила имеют вид $\neg(S_1 + S_2) = \neg S_1 + \neg S_2$, $\neg(S_1 \cdot S_2) = \neg S_1 \cdot \neg S_2$. Кроме того, для унарных операций справедливо $\neg\bar{S} = \neg S$.

Таким образом, система $\langle\{S\}, +, \cdot\rangle$ образует *дистрибутивную структуру*. Эта структура является полной, так как содержит нулевой элемент 0 и единичный 1, которыми являются схемы, заданные системами уравнений $z_i = z_i$ и $z_i = \bar{z}_i$, $i = 1, \ldots, n$, соответственно. В схеме 0 для любого состояния имеет место $a \to \varnothing$ и $\varnothing \to a$, а в схеме 1 для любой пары состояний a и b — $a \to b$. Для любой схемы S выполняются соотношения $S \cdot 0 = 0$, $S \cdot 1 = S$, $S + 0 = S$, $S + 1 = 1$, а также $S \cdot \bar{S} = 0$ и $S + \bar{S} = 1$. Таким образом, система $\langle\{S\}, +, \cdot, -\rangle$ является булевой алгеброй. Эта алгебра содержит $2^{n \cdot 2^n}$ элементов, где n — мощность множества переменных схемы.

Определение 6.8. Схему S^0, для которой выполняется $\neg S^0 = S^0$, будем называть *самообратимой*.

Поскольку схемы 0 и 1 самообратимы (т. е. имеет место $\neg 0 = 0$ и $\neg 1 = 1$), подсистема $\langle\{S^0\}, +, \cdot\rangle$ является полной дистрибутивной структурой (подструктурой), а подсистема $\langle\{S^0\}, +, \cdot, -\rangle$ — булевой подалгеброй. Эта подалгебра содержит $2^{n \cdot 2^{n-1}}$ элементов.

Будем обозначать через $[S]$ и $]S[$ схемы, заданные выражениями $S \cdot (\neg S)$ и $S + \neg S$ соответственно. Булевы уравнения для схем $[S]$ и $]S[$, выраженные через уравнения схемы S, имеют вид

$$z_i = \bar{z}_i f(z_i = 0) \bar{f}(z_i = 1) \vee z_i (\bar{f}(z_i = 0) \vee f(z_i = 1))$$

и

$$z_i = \bar{z}_i (f(z_i = 0) \vee \bar{f}(z_i = 1)) \vee z_i \bar{f}(z_i = 0) f(z_i = 1).$$

Схемы $[S]$ и $]S[$ являются самообратимыми. Выражения, задающие эти схемы, определяют отображение булевой алгебры асинхронных схем в ее подалгебру самообратимых схем, причем $[S]$ является эпиморфизмом относительно операции пересечения, а $]S[$ — эпиморфизмом относительно операции объединения.

Теорема 6.5. *В схеме S, для которой выполняется $[S] = 0$ ($]S[= 1$), функции $f_i(Z)$ изотонны (антитонны) по переменной z_i. Если одновременно выполняются оба эти соотношения, то все переменные z_i схемы несамозависимы и имеет место $\neg S = \bar{S}$.*

Алгебра асинхронных схем (в силу того, что она является булевой алгеброй) порождается системой, содержащей, как минимум, $n +]\log_2 n[$ образующих, где $]\log_2 n[$ — целое число такое, что $\log_2 n \leqslant]\log_2 n[\leqslant \log_2 2n$.

Рассмотрим совокупность множеств схем двух типов $\langle\{S^{z_j}\}, \{S^m\}\rangle$, где схемы первого типа S^{z_j}, $j = 1, \ldots, n$, задаются системами булевых уравнений вида $z_i = \bar{z}_i z_j \vee z_i \bar{z}_j$, $i = 1, \ldots, n$, а схемы второго типа S^m, $m = 1, \ldots,]\log_2 n[$ — системами булевых уравнений вида $z_i = \bar{z}_i \sigma_m(i) \vee z_i \bar{\sigma}_m(i)$, в которых коэффициенты $\sigma_m(i)$ определяются из двоичного разложения номера переменной:

$$i = \sum_{m=1}^{]\log_2 n[} 2^{m-1} \sigma_m(i).$$

Имеют место следующие две леммы.

Лемма 6.1. *В схеме, полученной из схем первого типа применением операций объединения, пересечения и инверсии, в каждом состоянии все переменные либо возбуждены, либо устойчивы.*

Лемма 6.2. *В схеме, полученной из схем второго типа применением операций объединения, пересечения и инверсии, каждая переменная либо возбуждена, либо устойчива во всех состояниях схемы.*

Из этих лемм становится очевидной возможность задания произвольной схемы S с помощью совокупности схем $\left\langle \left\{S^{z_j}\right\}, \{S^m\} \right\rangle$ применением к последним операций объединения, пересечения и инверсии. Для этого достаточно, например, выделить в данной схеме S набор состояний и набор переменных, возбуждаемых в этих состояниях, получить, согласно леммам 6.1 и 6.2, схемы, соответствующие этим наборам состояний и переменных, а затем их пересечения, и далее объединить полученные таким образом пересечения по всем возможным наборам.

Таким образом, справедлива

Теорема 6.6. *Совокупность множеств схем* $\left\langle \left\{S^{z_j}\right\}, \{S^m\} \right\rangle$ *является минимальной системой образующих алгебры асинхронных логических схем при использовании операций объединения, пересечения и инверсии.*

6.3.3. Преобразования схем. Пусть $\varphi(Z)$ — некоторая логическая функция. Обозначим через Z^φ множество состояний схемы S такое, что некоторое состояние a принадлежит Z^φ тогда и только тогда, когда $\varphi(a)=1$.

Определение 6.9. Схемы S_1 и S_2, заданные булевыми уравнениями $z_i = f_{1i}(Z)$ и $z_i = f_{2i}(Z)$ соответственно, будем называть *эквивалентными* относительно множества состояний Z^φ, если из $a \in Z^\varphi$ следует $f_{1i}(a) = f_{2i}(a)$, $i = 1, \ldots, n$.

Первое из рассматриваемых преобразований заключается в получении по данной схеме S схемы $S^{[\varphi]}$, эквивалентной данной относительно множества состояний Z^φ и такое, что в $S^{[\varphi]}$ любое состояние a, $a \in \bar{Z^\varphi}$, является тупиковым, т. е. $a \to \varnothing$. Для получения этого преобразования используем схему S^φ, заданную системой $z_i = \bar{z}_i\varphi(Z) \vee z_i\bar{\varphi}(Z)$, $i = 1, \ldots, n$. В этой схеме для любого состояния a, $a \in \bar{Z^\varphi}$, имеет место $a \to \varnothing$, а для любой пары состояний a и b, $b \in Z^\varphi$, имеет место $b \to a$. (Схема S^φ может быть получена из образующих первого типа в соответствии с леммой 6.1.) Требуемое преобразование получается пересечением данной схемы S и схемы S^φ, т. е. $S^{[\varphi]} = S \cdot S^\varphi$.

Второе преобразование в некотором смысле обратно первому. Оно состоит в получении по данной схеме S схемы $S^{|\varphi|}$, в которой невозможен переход в состояние a из множества $Z^{\bar{\varphi}}$ из любого состояния, соседнего с a, т. е. для любой пары соседних состояний a и b, $a \in Z^{\bar{\varphi}}$, не имеет места $b \xrightarrow{1} a$. Очевидно, $S^{|\varphi|} = S \cdot (\neg S^\varphi)$.

Третье преобразование позволяет по данной схеме S получить схему $S^{(\varphi)}$, в которой состояния множества $Z^{\bar{\varphi}}$ «изолированы», т. е. для любой пары соседних состояний a и b, $a \in Z^{\bar{\varphi}}$, имеет место $a \to \varnothing$ и не имеет места $b \to a$. Схема $S^{(\varphi)}$ может быть получена как пересечение схем $S^{[\varphi]}$ и $S^{|\varphi|}$, т. е. $S^{(\varphi)} = [S^\varphi]$.

Последнее из рассматриваемых здесь преобразований позволяет по данной схеме S получить схему $S^{|\varphi|}$, в которой невозможен переход в состояния из множества $Z^{\bar{\varphi}}$ из соседних с ними состояний из множества Z^φ и наоборот: для любой пары соседних состояний a и b, $a \in Z^{\bar{\varphi}}$, $b \in Z^\varphi$, не должны иметь место $a \to b$ и $b \to a$. Схема $S^{|\varphi|}$ может быть получена как объединение схем $S^{(\varphi)}$ и $S^{(\bar{\varphi})}$: $S^{|\varphi|} = S\left(\left[S^\varphi\right] + \left[S^{\bar{\varphi}}\right]\right)$.

Пример 6.8. Пусть $\varphi(Z) = z_j$, $j = 1, \ldots, n$. Схема S^{z_j} для этого случая (образующая первого вида), как было показано, задается системой $z_i = \bar{z}_i z_j \vee z_i\bar{z}_j$, $i = 1, \ldots, n$, а схема $\neg S^{z_j}$ — системой $z_i = \bar{z}_i z_j \vee z_i\bar{z}_j$, $z_j = 1$, $i \neq j$. Система уравнений схемы $\left[S^{z_j}\right]$ имеет вид $z_i = \bar{z}_i z_j \vee \bar{z}_j z_i$, $z_j = z_j$, $i \neq j$. Для последнего из рассмотренных преобразований используется схема $\left[S^{z_j}\right] + \left[S^{\bar{z}_j}\right]$, задаваемая системой $z_i = \bar{z}_i$, $z_j = z_j$, $i \neq j$.

Из последнего примера следует, что, используя операцию обращения вместе с операциями объединения, пересечения и инверсии, можно получать образующие второго типа из образующих первого типа. Таким образом, становится очевидной справедливость следующей теоремы.

Теорема 6.7. *Множество схем* S^{z_j}, $j = 1, \ldots, n$, *является системой образующих алгебры асинхронных логи-*

ческих схем при использовании операций объединения, пересечения, инверсии и обращения.

6.3.4. Гомологические алгебры схем. Будем называть *гомологическими* различные алгебраические структуры, операции в которых определены на одном и том же множестве. В случае асинхронных логических схем для получения гомологических структур, очевидно, необходимо расширить список алгебраических операций над схемами. Интерес при этом представляют такие наборы операций, которые, с одной стороны, в совокупности со множеством схем образуют известные алгебраические структуры и, с другой стороны, имеют достаточно простую интерпретацию.

Определение 6.10. *Дизъюнкцией* (*конъюнкцией*) схем S_1 и S_2 будем называть схему $S_{1\vee 2}$ ($S_{1\wedge 2}$), заданную системой булевых уравнений вида $z_i = f_{1i}(Z) \vee f_{2i}(Z)$, $i = 1, \ldots, n$ ($z_i = f_{1i}(Z) f_{2i}(Z)$).

Непосредственной проверкой легко убедиться, что для вновь введенных бинарных операций выполняются законы коммутативности, ассоциативности, идемпотентности, поглощения и дистрибутивности. С учетом ранее определенной унарной операции инверсии справедливы правила Де Моргана $\overline{S_1 \vee S_2} = \overline{S}_1 \wedge \overline{S}_2$, $\overline{S_1 \wedge S_2} = \overline{S}_1 \vee \overline{S}_2$, а для операции обращения — аналогичные правила $\neg(S_1 \vee S_2) = \neg S_1 \wedge \neg S_2$, $\neg(S_1 \wedge S_2) = \neg S_1 \vee \neg S_2$. Таким образом, система $\langle\{S\}, \vee, \wedge\rangle$ образует *дистрибутивную структуру*. Эта структура является полной, так как содержит нулевой и единичный элементы, которыми являются соответственно схемы, заданные системами $z_i = 0$ и $z_i = 1$, $i = 1, \ldots, n$. Будем обозначать нулевой элемент этой дистрибутивной структуры через $0'$, а единичный — через $1'$.

Для любой схемы S выполняются соотношения $S \wedge 0' = 0'$, $S \wedge 1' = S$, $S \vee 0' = S$, $S \vee 1' = 1'$, а также $S \wedge \overline{S} = 0'$, $S \vee \overline{S} = 1'$. Таким образом, система $\langle\{S\}, \vee, \wedge, \neg\rangle$ является *булевой алгеброй*. Поскольку эта алгебра и рассмотренная ранее булева алгебра $\langle\{S\}, +, \cdot, \neg\rangle$ являются гомологическими и, следовательно, содержат одинаковое количество элементов — $2^{n \cdot 2^n}$, где n — мощность множества переменных схемы, справедлива

Теорема 6.8. *Гомологические алгебры схем* $\langle\{S\}, +, \cdot, \neg\rangle$ *и* $\langle\{S\}, \vee, \wedge, \neg\rangle$ *изоморфны, т. е. существует взаимооднозначное гомоморфное отображение* ψ *одной алгебры на другую, например*:

$$\psi(S_1 + S_2) = \psi(S_1 \vee S_2), \quad \psi(S_1 \cdot S_2) = \psi(S_1) \wedge \psi(S_2).$$

На рис. 6.6, *а* и *б* приведены диаграммы Хассе этих гомологических алгебр для случая, когда схемы заданы на множестве $\{z\}$, состоящем из одной переменной.

Определим для булевой алгебры схем $\langle\{S\}, +, \cdot, \neg\rangle$ ($\langle\{S\}, \vee, \wedge, \neg\rangle$) бинарную операцию равнозначности (эквивалентности) следующим образом: $S_{1\equiv 2} = S_1 \equiv S_2 = S_1 S_2 + \overline{S}_1 \overline{S}_2$ ($S_{1\infty 2} = S_1 \infty S_2 = (S_1 \wedge S_2) \vee (\overline{S}_1 \wedge \overline{S}_2)$). Булевы уравнения для равнозначности $S_{1\equiv 2}$ и эквивалентности $S_1 \infty S_2$ схем S_1 и S_2, выраженные через уравнения схем S_1 и S_2, имеют вид $z_i = z_i(f_{1i}(Z) f_{2i}(Z) \vee \overline{f}_{1i}(Z) \overline{f}_{2i}(Z)) \vee z_i(f_{1i}(Z) f_{2i}(Z) \vee \overline{f}_{1i}(Z) \overline{f}_{2i}(Z))$ и $z_i = f_{1i}(Z) f_{2i}(Z) \vee \overline{f}_{1i}(Z) \overline{f}_{2i}(Z)$ соответственно. В $S_{1\equiv 2}$ каждая переменная z_i возбуждена в состоянии a тогда и только тогда, когда $f_{1i}(a) = f_{2i}(a)$, в $S_{1\infty 2}$ в этих и только этих состояниях имеет место $f_{1\infty 2i}(a) = 1$. Отсюда для любой схемы S следует выполнение соотношений $S \equiv S = 1$ и $S \infty S = 1'$, а также $S \equiv \overline{S} = 0$ и $S \infty \overline{S} = 0'$. Таким образом, $S_{1\equiv 2}$ и $S_{1\infty 2}$ можно рассматривать как своеобразную меру

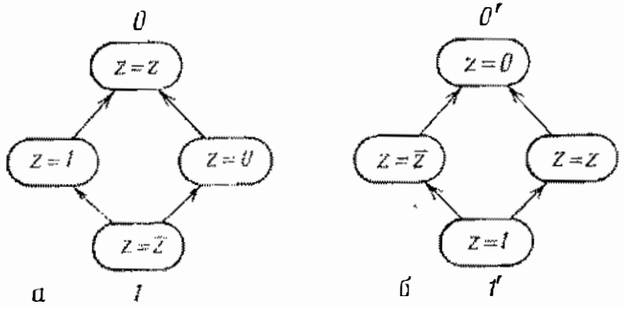


Рис. 6.6. Диаграммы Хассе двух алгебр

близости схем S_1 и S_2 в соответствующих гомологических алгебрах.

Для любой схемы S из определений равнозначности и эквивалентности очевидно выполнение соотношений $S \equiv 1 = S \infty 1' = S$ и $S \equiv 0 = S \infty 0' = \overline{S}$. Интерес представляет тот факт, что $S \equiv 1' = S \infty 1$ и $S \equiv 0' = \infty 0$ для любой схемы S, т. е. что одинаковые элементы в рассматриваемых гомологических алгебрах равноблизки не только единичным (нулевым) элементам своих алгебр, но и единичным (нулевым) элементам алгебр друг друга.

Будем обозначать через S' схему, заданную выражением $S \equiv 1'$ или $S \infty 1$. Булевы уравнения для схемы S', выраженные через уравнения схемы S, имеют вид $z_i = z_i \overline{f}_i(z) \vee \overline{z}_i f_i(Z)$. Последние можно рассматривать как определение некоторой унарной операции алгебры схем. Для этой операции выполняется закон инволюции.

В силу справедливости соотношений $(S_1 + S_2)' = S_1' \vee S_2'$, $(S_1 S_2)' = S_1' \wedge S_2'$ и $(S_1 \vee S_2)' = S_1' + S_2'$, $(S_1 \wedge S_2)' = S_1' \cdot S_2'$, а также $(\overline{S})' = \overline{S}'$, выражения, задающие схему S' (или соответствующую унарную операцию), являются изоморфным отображением булевой алгебры схем $\langle\{S\}, +, \cdot, \neg\rangle$ на булеву алгебру схем $\langle\{S\}, \vee, \wedge, \neg\rangle$ и наоборот. Введенные ранее обозначения $0'$ и $1'$ теперь получили смысл как отображение 0 и 1 соответственно. Естественно, в силу инволютивности этого отображения имеют место $(0')' = 0$ и $(1')' = 1$. Для операции обращения выполняется соотношение $\neg S' = (\neg S)'$.

Полученное изоморфное отображение для рассматриваемых гомологических алгебр позволяет выразить бинарные операции одной алгебры через бинарные операции другой $S_1 + S_2 = (S_1 \vee S_2) \wedge \wedge 1 \vee (S_1 \wedge S_2)$, $S_1 \cdot S_2 = (S_1 \vee S_2) \wedge 0 \vee (S_1 \wedge S_2)$ и $S_1 \vee S_2 = = (S_1 + S_2) \cdot 1' + S_1 \cdot S_2$, $S_1 \wedge S_2 = (S_1 + S_2) \cdot 0' + S_1 \cdot S_2$.

Система образующих алгебры схем $\langle \{S\}, \vee, \wedge, - \rangle$ может быть получена с помощью описанного отображения образующих алгебры схем $\langle \{S\}, +, \cdot, - \rangle$; отображение $\left(S^{z_j}\right)'$ схем первого типа S^{z_j} задается системой вида $z_i = z_j$, $i = 1, \ldots, n$, а отображение схемы второго типа задается системой вида $z_i = \sigma_m(i)$, $i = 1, \ldots, n$.

Очевидно, аналогичным образом могут быть порождены различные булевы алгебры схем. Поскольку при этом в качестве единичного (нулевого) элемента может быть выбрана произвольная схема S (ее инверсия $\bar{S}$), то существует $2^{n \cdot 2^n}$ различных гомологических алгебр схем, заданных на множестве переменных мощности n. В частности, положив $1' = 0$, $0' = 1$, получим алгебру, двойственную алгебре $\langle \{S\}, +, \cdot, - \rangle$, изоморфным отображением на которую является, как известно, операция инвертирования.

Из двух рассмотренных гомологических алгебр схем первая образована операциями, имеющими достаточно ясную поведенческую трактовку (объединение, пересечение, инверсия и обращение схем), а вторая — операциями, имеющими традиционный декомпозиционный смысл (конъюнкция, дизъюнкция). Тот факт, что обе эти алгебры являются булевыми и изоморфны одна другой, позволяют выразить операции одной из алгебр через операции другой. Преобразования с использованием предложенных операций могут быть использованы в процессе проектирования асинхронных схем для устранения выявленных в результате анализа нежелательных особенностей. Формальные алгоритмы анализа будут рассмотрены в гл. 8.

§ 6.4. Замечания по библиографии

Материал по композиции асинхронных процессов напечатан в [9]. Доказательство теоремы Маллера [272, 273] содержится в [120]. Обобщениям этой теоремы была посвящена работа [63]. Счетчик, изображенный на рис. 6.4, описан в [145] и защищен авторским свидетельством [23]. О существовании еще не выявленных обобщений теоремы Маллера свидетельствует возможность еще более глубокого «вложения» разрядов счетчика, показанная в [26]. Другие модели композиции, также являющиеся в некотором смысле обобщениями теоремы Маллера, рассматривались в [247].

Алгебра асинхронных схем изложена в [176, 177]. Этот материал в идейном отношении близок к работам по эквивалентным преобразованиям асинхронных схем [126, 182, 185].

Подавая сигналы в рог,
Будь всегда справедлив, но строг.

К. Прутков

Глава 7. СОГЛАСОВАНИЕ АСИНХРОННЫХ ПРОЦЕССОВ И ОРГАНИЗАЦИЯ ИНТЕРФЕЙСОВ

Современные дискретные системы и ЭВМ строятся на основе модульного (агрегатного) принципа: устройства машин, сами машины, равно как и программное обеспечение, выполняются в виде отдельных функциональных и структурных единиц (модулей). Последние могут объединяться (комплексироваться), образуя сложные системы. Организация модулей в систему с необходимостью предполагает взаимодействие между модулями, что требует значительных временных, аппаратурных и программных ресурсов. Фундаментальным понятием, отражающим аспект взаимодействия модулей в системе, является понятие интерфейса.

В широком смысле под интерфейсом (или сопряжением) понимается совокупность правил, обеспечивающих совместное функционирование аппаратных и программных средств. Иногда интерфейсом называют и сами технические средства, созданные в соответствии с этими правилами.

Сопряжение (в отличие от устройств сопряжения) не является частью аппаратных средств, а представляет собой сечение (границу) между модулями системы, на котором определены сигналы и процедуры обмена. Сопряжение точно определяет, какие сигналы или сообщения могут пересекать это сечение и каков их смысл. Оно должно обеспечивать единый метод передачи данных независимо от внутренних особенностей взаимодействующих модулей и относиться исключительно к обмену сигналами между устройствами. Правильно определенное содержание сопряжения является необходимым условием разработки интерфейса.

Совокупность правил и средств, составляющих структуру сопряжения — интерфейс — принято делить на три уровня: механический, электрический и логический.

Механический (конструктивный) уровень определяет совокупность требований, которым должны удовлетворять узлы интерфейсной аппаратуры — кабель, линии связи, разъемные соединения, интерфейсные карты (адаптеры) и т. п.

Электрический уровень сопряжения оговаривает допуски на параметры входных и выходных электрических сигналов, токовые нагрузки на источник сигнала. Они зависят от принятой элементной базы.

Логический уровень сопряжения обеспечивает координацию протекания информационных потоков между устройствами. Он определяет способ представления сигналов, символов, сообщений, обозначения и классификацию сигналов и сообщений. Логический уровень — наиболее сложная часть правил сопряжения. Логическое согласование объединяемых в систему устройств предусматривает функционирование сопрягаемых модулей по установленным алгоритмам. Основным понятием, отражающим координированное функционирование объектов логического уровня интерфейса, является понятие протокола информационного обмена.

Под протоколом информационного обмена (далее для краткости — просто протоколом) обычно понимают свод правил, которых нужно придерживаться, чтобы обеспечить упорядоченный информационный обмен между двумя или более объектами.

Термин «протокол» нашел широкое распространение в литературе по вычислительной технике и системам управления; его появление навеяно, по-видимому, некоторым сходством с понятием дипломатического протокола (совокупность правил, регулирующих порядок различных дипломатических актов) и протокола заседания (облеченное в письменную форму изложение хода заседания). Возникла даже *теория протоколов* (информационного обмена), предметом которой являются способы задания, анализа и синтеза интерфейсного взаимодействия. Несмотря на некоторые успехи этой теории, на практике для описания интерфейсных протоколов чаще всего применяются неформальные или полуформальные средства, такие, как естественный язык, временные диаграммы, циклограммы или блок-схемы алгоритмов. Вследствие этого они чрезвычайно громоздки и, что хуже, допускают неоднозначное толкование. Вообще говоря, при составлении протоколов необходимо удовлетворить несколько требований: компактное представление правил обмена, их однозначное толкование, возможность проверки задания на полноту и непротиворечивость, возможность перехода от протоколов непосредственно к синтезу интерфейсного устройства. Путь к этому открывает только формализация языка описания, которая может быть осуществлена в терминах асинхронных процессов (§§ 2.1 и 6.1).

В этой главе предметом исследования являются в основном системы, в которых протоколы обмена между объектами реализуются аппаратно. Тем не менее предложенный подход может быть также использован, скажем, и для сетей ЭВМ, где протоколы обычно реализуются программно.

Из многочисленных задач, возникающих при организации интерфейсов, здесь предметом рассмотрения будут лишь некоторые. Прежде всего, будет дано формальное толкование самого понятия «протокол», центрального в этой проблематике. После этого будет дан пример описания одного интерфейса и его протокола и предложен способ его аппаратной реализации. Далее будет показано, что и для решения задач, характерных для интерфейсов нижнего уровня, могут быть использованы «логические» средства: в частности, известное явление перекоса задержек соединительных линий, оказывается, парируется средствами апериодической схемотехники в сочетании с кодированием самосинхронизирующимися кодами.

§ 7.1. Согласованные асинхронные процессы

Пусть заданы два приведенных процесса (см. определение 6.1)

$$P_1^{\text{п}} = \langle S_1, F_1, I_1, R_1 \rangle \text{ и } P_2^{\text{п}} = \langle S_2, F_2, I_2, R_2 \rangle$$

с ситуациями, в которых выделены входная и выходная компоненты, элементы подмножества Y_1^* значений выходной компоненты ситуаций первого процесса $(Y_1^* \subseteq Y_1)$ соответствуют (эквивалентны) элементам подмножества X_2^* значений входной компоненты ситуаций второго процесса $(X_2^* \subseteq X_2)$, а элементы подмножества Y_2^* значений выходной компоненты ситуаций второго процесса

$(Y_2^* \subseteq Y_2)$ — элементам подмножества X_1^* значений входной компоненты ситуаций первого процесса $(X_1^* \subseteq X_1)$.

Осуществим, если это возможно, последовательную композицию процессов $P_1^{\text{п}}$ и $P_2^{\text{п}}$, считая, что $Y_1^* = X_2^*$ (либо $Y_2^* = X_1^*$). Результатом композиции является асинхронный процесс $P_{1,2}^{\text{п}}(P_{2,1}^{\text{п}})$. В ситуациях АП $P_{1,2}^{\text{п}}(P_{2,1}^{\text{п}})$ выделим входную компоненту, совпадающую со входной компонентой ситуаций $P_1^{\text{п}}(P_2^{\text{п}})$. Построим, если это возможно, замыкание $P_{1,2}^{\text{з}}(P_{2,1}^{\text{з}})$ АП $P_{1,2}^{\text{п}}(P_{2,1}^{\text{п}})$, считая, что поэлементно $Y_2^* = X_1^*$ $(Y_1^* = X_2^*)$. На основании введенных ранее понятий редукции и замыкания АП нетрудно доказать, что $P_{1,2}^{\text{з}} = P_{2,1}^{\text{з}}$.

Ситуации построенного таким образом асинхронного процесса имеют следующую структуру:

$$s = (u, v, z),$$

где $u = y^1 = x^2$, $v = y^2 = x^1$, $z = (z^1, z^2)$, причем компоненты u и v принимают значения из множеств $U \subseteq Y_1^* = X_2^*$ и $V \subseteq Y_2^* = X_1^*$ соответственно.

Определение 7.1. Два асинхронных процесса будем называть *согласованными* относительно множеств U и V, если результатом применения к ним последовательной композиции и замыкания является автономный процесс, заданный на непустом множестве ситуаций, компоненты u и v которых являются отождествленными входными и выходными компонентами ситуаций этих процессов.

§ 7.2. Протокол

Пусть задан асинхронный процесс $P = \langle S, F, I, R\rangle$, в ситуациях которого выделены входная x и выходная y компоненты, определенные на множествах X и Y соответственно. Построим проекции:

1) $S(X \times Y)$ множества S на $X \times Y$,
2) $F(X \times Y)$ множества F на $X \times Y \times X \times Y$,
3) $I(X \times Y)$ множества I на $X \times Y$,
4) $R(X \times Y)$ множества R на $X \times Y$.

Определение 7.2. Асинхронный процесс $P(X \times Y) = \langle S(X \times Y), F(X \times Y), I(X \times Y), R(X \times Y)\rangle$ назовем *проекцией* процесса P на $X \times Y$.

Определение 7.3. Проекцию автономного процесса $P_{1,2}^{\text{з}}$, полученного в результате согласования процессов $P_1^{\text{п}}$ и $P_2^{\text{п}}$, на $U \times V$ будем называть *протоколом* $\Pi(U, V)$ *согласования* $P_1^{\text{п}}$ и $P_2^{\text{п}}$ относительно множеств U и V.

Протокол $\Pi(U, V)$ является формальным заданием протокола обмена, он реализуется процессом $P_{1,2}^{\text{з}}$.

Построим проекции согласованных процессов $P_1^{\text{п}}$ и $P_2^{\text{п}}$ на $U \times V$. Очевидно, что эти проекции обладают свойствами

$$I_1(U \times V) \subseteq R_2(U \times V),$$
$$I_2(U \times V) \subseteq R_1(U \times V),$$
$$F_1(U \times V) \cup F_2(U \times V) = F_{1,2}^{\text{з}}(U \times V).$$

Практически задача координации работы двух устройств ставится следующим образом. Имеются два устройства и протокол обмена, которому должно удовлетворять совместное поведение этих устройств. Протокол обмена задает множество отождествляемых значений сигналов (и тем самым входы и выходы, которые нужно соединить друг с другом), а также порядок изменения этих сигналов.

Для решения задачи необходимо соединяемые устройства описать как АП с выделенными входными и выходными компонентами; определить в соответствии с протоколом обмена множества эквивалентных компонент; согласовать, если это возможно, процессы относительно множеств эквивалентных компонент; построить проекцию АП, описывающего совместное поведение согласованных АП, и, наконец, удостовериться, что полученный протокол согласования описывает заданный протокол обмена (для этого обычно требуется семантическая интерпретация протокола).

Имеется по крайней мере три случая, когда эта задача не решается.

1) Иногда значения выходной компоненты одного АП не эквивалентны значениям входной компоненты другого, например, из-за различия физических параметров сигналов, представляющих эти значения (несогласуемые непосредственно уровни сигналов, их различная физическая природа, различная кодировка сигналов, имеющих один и тот же смысл, и т. д.).

2) Может оказаться, что множества U и V, относительно которых АП удалось согласовать, включают не все значения, требуемые протоколом обмена, и из-за этого протокол обмена не реализуется или реализуется не в полном объеме.

3) В автономном процессе, описывающем совместное поведение согласованных процессов, хотя и встречаются все элементы эквивалентных множеств, но не реализуются некоторые последовательности изменений значений компонент, предусмотренные протоколом обмена.

В этих случаях устройства соединяются не непосредственно, а через дополнительный блок, называемый *адаптером*.

§ 7.3. Согласующий асинхронный процесс

Формально соединение двух устройств через адаптер описывается с помощью реализуемых ими и адаптером трех приведенных АП $P_1^{\text{п}}$, $P_2^{\text{п}}$ и $P_c^{\text{п}}$, ситуации которых имеют следующую структуру:

$$s^1=(x^1, y^1, z^1),\quad s^2=(x^2, y^2, z^2),\quad s^c=(y^1, y^2, x^1, x^2, z^c)$$

(для ситуаций процесса $P_c^{\text{п}}$, соответствующего адаптеру, компоненты y^1 и y^2 являются входными, а x^1 и x^2 — выходными).

Пара АП $P_1^{\text{п}}$, $P_c^{\text{п}}$ согласована относительно множеств U_1, V_1 допустимых значений компонент y^1, x^1. Процессы $P_2^{\text{п}}$ и $P_c^{\text{п}}$ согласованы относительно множеств U_2, V_2 допустимых значений компонент y^2, x^2. Такая композиция АП $P_1^{\text{п}}$, $P_2^{\text{п}}$ и $P_c^{\text{п}}$ может быть представлена в виде автономного АП P_{1c2}, заданного на ситуациях вида

$$s^{1c2}=(s^1, s^c, s^2).$$

Определение 7.4. Назовем $P_c^{\text{п}}$ асинхронным процессом, *согласующим АП* $P_1^{\text{п}}$ и $P_2^{\text{п}}$, если в процессе P_{1c2}:

1) существует такая ситуация s_i^{1c2} с компонентой $s_i'\in R_1$, что любая траектория процесса P_{1c2} из s_i^{1c2} неизбежно ведет к ситуации s_j^{1c2} с компонентой $s_j^2\in I_2$ и

2) существует такая ситуация s_k^{1c2} с компонентой $s_k^2\in R_2$, что любая траектория процесса P_{1c2} из s_k^{1c2} неизбежно ведет к ситуации s_l^{1c2} с компонентой $s_l'\in I_1$.

Иными словами, процесс $P_c^{\text{п}}$ будет согласующим лишь в том случае, если среди его ситуаций найдутся такие, что появление некоторого значения входной компоненты $y^1(y^2)$ по достижении результанта процессом $P_1(P_2)$ с неизбежностью вызывает такое изменение выходной компоненты $x^2(x^1)$, которое приводит в процессе $P_2(P_1)$ к переходу от результанта к инициатору.

Пример 7.1. Рассмотрим систему, состоящую из канального процессора КП, адаптера Ад и двух абонентов — Аб1 и Аб2 (рис. 7.1, *а*). КП, Аб1 и Аб2 связаны общей магистралью данных. Каждое из этих устройств имеет возможность передавать информацию в магистраль и принимать ее из магистрали. Система работает по принципу «ведущий — ведомый» и осуществляет обмен информацией между КП и абонентами. Роль ведущего играет КП, а ведомых — абоненты. КП устанавливает адрес A абонента и затем сигнал запроса $a=1$. По адресу A и этому сигналу адаптер вырабатывает либо $a_1=1$, либо $a_2=1$, которые являются сигналами, инициирующими прием или передачу информации. После окончания работы с информацией соответствующий абонент устанавливает сигнал $b_i=1$. Приняв от абонента $b_i=1$, адаптер отвечает сигналом $b=1$. После этого начинается процесс отключения абонента: КП устанавливает $a=0$ (при этом он может менять адрес), по которому адаптер сбрасывает в 0 тот сигнал a_i, который был равен 1. В ответ на $a_i=0$ i-й абонент устанавливает $b_i=0$, после чего адаптер выставляет $b=0$, и система оказывается в исходном состоянии.

На рис. 7.1, *б* изображена сеть Петри, задающая взаимодействие указанных устройств на уровне установления и разрыва связи. Эта сеть Петри описывает согласованный АП и состоит из четырех фрагментов, относящихся к перечисленным выше устройствам. В силу того, что КП может обратиться только к одному из абонентов, точка может появиться либо в кружке *5*, либо в кружке *6*, а значит, либо в *7*, либо в *8*, но не в обоих одновременно.

Для выделенных фрагментов перечислим ситуации, являющиеся инициаторами и результантами.

Результанты фрагмента КП:

1) точки в кружках *4* и *7* — установка адреса Аб1 и сигнала $a=1$;

2) точки в кружках *4* и *8* — установка адреса Аб2 и сигнала $a=1$;

3) точки в кружках *2* и *4* — установка $a=0$.

Инициатором фрагмента КП является результант 1) фрагмента Ад.

Результанты фрагмента Ад:

1) точка в кружке *9* — установка $b=1$ или $b=0$;

2) точка в кружке *11* — установка $a_1=1$ или $a_1=0$;

3) точка в кружке *13* — установка $a_2=1$ или $a_2=0$.

Инициаторы фрагмента Ад — результанты фрагментов КП, Аб1 и Аб2.

Фрагмент Аб1: результант — точка в кружке *14* (установка $b_1=1$ или $b_1=0$), инициатор — результант 2) фрагмента Ад.

Фрагмент Аб2: результант — точка в кружке *15* (установка $b_2 = 1$ или $b_2 = 0$), инициатор — результант 3) фрагмента Ад.

Сделаем следующие замечания. В приведенном примере уровень протокола, заданного на рис. 7.1, *б* таков, что он непосредственно может быть использован для реализации адаптера. Это связано с тем, что сеть на рис. 7.1, *б* относится к классу устойчивых

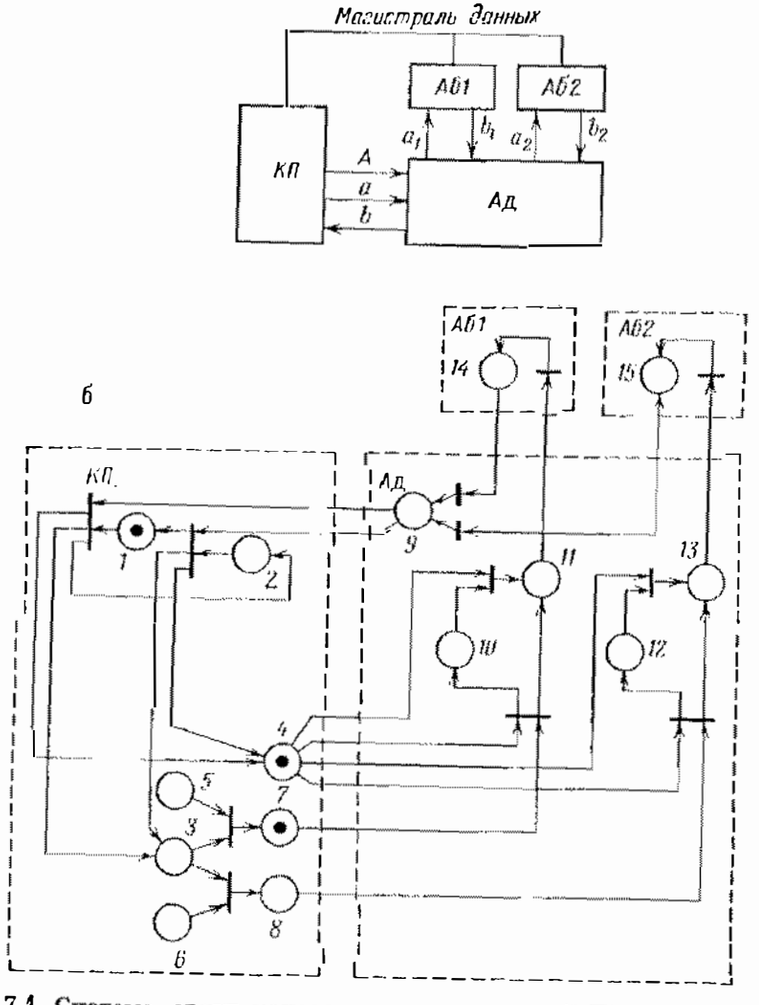


Рис. 7.1. Система связи канального процессора (КП) с двумя абонентами (Аб1, Аб2) через адаптер (Ад): структура (*а*), протокол установления и разрыва связей на языке сетей Петри (*б*)

и безопасных сетей Петри, а для этого класса существуют методы синтеза, описанные в § 5.1. Результатом синтеза является управляющий автомат адаптера, композиция которого с операционной частью (также выполняемая по методам, изложенным в гл. 6) и является реализацией адаптера.

Ниже будет дан еще один пример описания протокола обмена и показан способ перехода от языка его задания к реализации.

§ 7.4. Интерфейс Т2

В этом разделе рассматривается конкретный интерфейс, используемый в некоторой реальной системе передачи информации. Этот интерфейс, носящий название Т2, является, с одной стороны, специфическим (о чем свидетельствует, в частности, своеобразная терминология участников информационного обмена и сигналов), но, с другой стороны, сохраняет определенную степень общности с имеющимися стандартизованными интерфейсами. Включая в книгу материал этого раздела, авторы преследовали две цели. Первая — ознакомить читателя с типичным описанием интерфейса и протокола обмена на уровне управляющих сигналов, заданного теоретико-графовой моделью (пп. 7.4.1 и 7.4.2). Вторая цель — показать способ перехода от имеющегося описания протокола к его реализации (пп. 7.4.2 и 7.4.3).

7.4.1. Общие сведения. Этот интерфейс используется для связи блока управления (БУ) с абонентами, которые в данном случае называются групповыми устройствами (ГУ). Каждое ГУ содержит интерфейсную карту (ИК)[1]), реализующую согласующий АП, и ряд внешних устройств (ВУ). В качестве ВУ могут выступать не только обычные терминальные устройства (телетайпы, дисплеи, устройства печати), но и, например, аппаратура передачи данных, специализированные процессоры и т. д.

ИК и БУ имеют входные и выходные шины, которые делятся на информационные и управляющие. По информационным шинам из БУ в ИК передаются команды или данные. По информационным шинам из ИК в БУ передаются либо данные, либо информация о состоянии самой ИК и связанных с нею периферийных устройств.

Сигналы на управляющих шинах играют двоякую роль. С одной стороны, эти сигналы служат для идентификации типа информации на выходных шинах, с дру-

---

[1]) *Интерфейсной картой* или *адаптером* обычно называют устройство для соединения модулей с разным способом представления данных либо модулей, использующих различные виды унифицированных сопряжений.

гой, они являются квитанцией, подтверждающей факт приема информации со входных шин. При этом обязательно выполняются следующие правила: установка сигналов на информационных шинах должна предшествовать установке сигналов на управляющих шинах.

Временная последовательность работы БУ и ИК изображена на рис. 7.2.

В каждом цикле как в БУ, так и в ИК изменяется значение только одного управляющего выхода. Таким образом, рассмотренная в § 4.6 модель адекватно описывает поведение БУ и ИК.

БУ имеет 9 информационных выходных шин ШИБ1 — ШИБ9 и 2 управляющие выходные шины ШУБ1—ШУБ2. Каждое из ГУ имеет 9 выходных информационных шин ШИГ1 — ШИГ9 и — управляющие выходные шины

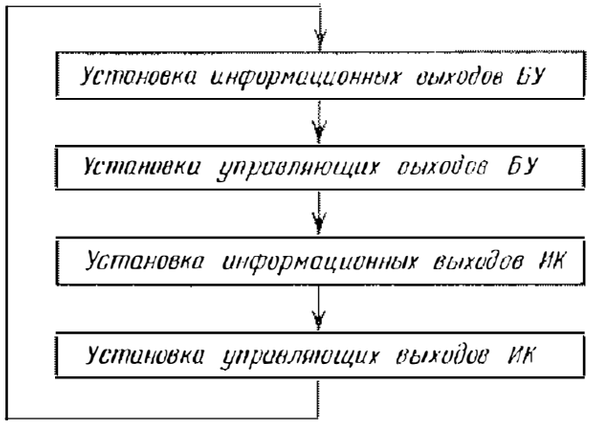


Рис. 7.2. Последовательность установки сигналов в интерфейсе Т2

ШУГ1 — ШУГ2. Сигналы, выдаваемые всеми ГУ, собираются на входах БУ (проводное ИЛИ).

В табл. 7.1 приведены: содержательная интерпретация наборов значений сигналов на ШУБ, их условные обозначения и описания типа информации, которую они сопровождают (идентифицируют).

В табл. 7.2 приведены аналогичные сведения для сигналов ГУ.

Ряд моментов в табл. 7.1 и 7.2 требует комментариев.

Байт команды имеет следующую структуру. ШИБ1 и ШИБ2 содержат код операции (00 — операция управления, 10 — ввод, 01 — вывод, 11 — обмен). ШИБ1 — 3

Таблица 7.1

ШУБ1	ШУБ2	Интерпретация	Обозначение	Вид информации на ШИБ
0	0	Команда (из БУ в ГУ); квитанция в приеме кода состояния ГУ	К	Байт команды и ее признак нечетности
1	0	Сигнал записи из БУ в ГУ при выполнении вывода и обмена; квитанция в приеме информации в БУ при выполнении ввода	З	Байт данных с признаком нечетности при выполнении вывода и обмена или несущественный байт
0	1	Адрес ГУ	А	Унитарный код адреса вызываемого ГУ
1	1	Сигнал разрешения подачи вызовов из ГУ в БУ	Р	Унитарный код адреса ГУ или несущественный байт

Таблица 7.2

ШУГ1	ШУГ2	Интерпретация	Обозначение	Вид информации на ШИГ
0	0	Состояние ГУ; квитанция в приеме адреса	С	Байт состояния ГУ с признаком нечетности, если оно логически подключено к БУ, и несущественный байт в противном случае
1	0	Чтение (ввод) данных из ГУ в БУ при выполнении ввода или обмена; квитанция в приеме команды вывода	Ч	Байт данных с признаком нечетности при выполнении ввода или обмена; несущественный байт при выполнении вывода
0	1	Вызов БУ	В	Унитарный код адреса ГУ
1	1	Модифицированное состояние ГУ, квитанция в приеме байта данных в командах вывода и обмена, квитанция в передаче последнего байта в команде ввода	М	Байт состояния ГУ с признаком нечетности

выдается при обработке последнего байта данных. ШИБ4 — ШИБ6 содержат код адреса ВУ, которому предназначена команда. Значение сигналов на ШИБ7, ШИБ8 зависит от применения интерфейса; ШИБ9 — признак нечетности кода команды.

Операция управление не связана с передачей данных и вызывает лишь изменение состояния ИК или одного из ВУ.

Операция ввод инициирует передачу байта данных из выбранного ВУ и БУ.

Операция вывод инициирует передачу байта данных из БУ в выбранное ВУ.

Операция обмен совмещает выполнение ввода и вывода.

Для задания адреса ГУ используется унитарный код, j-я позиция которого равна 1, если происходит обращение к j-му ГУ, 8 остальных позиций кода равны 0. Таким образом, интерфейс Т2 допускает подключение не более 9 ГУ.

Код, содержащий нули во всех 9 разрядах, называется несущественным байтом.

Если по ШИБ(ШИГ) передается не адрес ГУ и не несущественный байт, на ШИБ9(ШИГ9) выставляется признак нечетности $\left(\text{ШИБ9} = \overset{8}{\underset{i=1}{\oplus}} \text{ШИБ}i\right)$.

Байт состояния имеет следующую структуру. ШИГ1 и ШИГ2 содержат информацию о возможном режиме работы (00 — не готов, 10 — готов к выводу, 01 — готов к вводу, 11 — готов к обмену). ШИГ3 — 1, если обработан последний байт. Значения сигналов на ШИГ4 — ШИГ6 представляют собой код адреса ВУ, которое в данный момент подключено к ИК. Значения сигналов на ШИГ7 и ШИГ8 зависят от применения интерфейса. ШИГ9 — признак нечетности кода состояния.

7.4.2. Протокол обмена. Порядок обмена на уровне управляющих сигналов в интерфейсе Т2 может быть задан орграфом рис. 7.3.

Некоторые дуги в этом графе помечены условиями, при выполнении которых осуществляется помеченный переход. Вершины, соответствующие состояниям управляющих выходов ГУ (C_0, C_1, В, Ч, М) соединяются дугами только с вершинами, соответствующими состояниям управляющих выходов БУ (Р, А, К, З) и наоборот.

Состоянию С, указанному в табл. 7.2, соответствуют две вершины C_1 и C_0, где C_0 — исходное состояние ГУ, в котором ГУ логически не подключено к БУ. В этом состоянии на информационных выходах — несущественный байт. Состояние C_1 соответствует логическому подключению ГУ к БУ. В этом состоянии на информационных выходах ГУ — байт состояния.

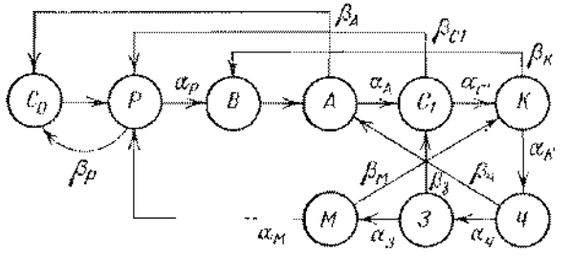


Рис. 7.3. Задание интерфейса Т2 на уровне управляющих сигналов

Необходимость расщепления состояния С на два диктуется особенностями работы с магистралью. Неработающие в данный момент ГУ выставляют на всех своих выходах нули, и на «фоне» нулей БУ может идентифицировать значения выходов ГУ, которое в данный момент работает с БУ.

Переходы ГУ, помимо состояний управляющих выходов БУ, определяются условиями α_P, β_P, α_A, β_A, α_K, β_K, $\alpha_З$, $\beta_З$. Условия α и β с одинаковым индексом ортогональны. Эти условия зависят от внутреннего состояния ГУ и его информационных входов.

В соответствии с описанием интерфейса Т2 перечисленные условия можно записать следующим образом:

$$\alpha_P = \text{ШИБ}_i \vee z_1 \vee z_2;\ \alpha_A = \text{ШИБ}_i \cdot \left(\underset{j\neq i}{\&} \text{ШИБ}_j\right);$$

$$\alpha_K = \left(\overset{9}{\underset{1}{\oplus}} \text{ШИБ}_i\right) \cdot \left(\underset{i=1,2,4,5,6}{\&} (\text{ШИБ}_i \oplus \bar{z}_i)\right) \cdot (z_1 \vee z_2);$$

$$\alpha_З = z_2 \left(\left(\overset{9}{\underset{1}{\oplus}} \text{ШИБ}_i\right) \vee (\text{Сч ГЗ} = 3)\right) \vee z_1 \bar{z}_2 z_3,$$

где z_i — i-й разряд регистра состояния ГУ, (СчГЗ = 3) — условие того, что байт информации из БУ трижды не прошел контроль на нечетность (в этом случае ГУ принимает этот байт несмотря на наличие ошибок).

Граф рис. 7.3 и выше приведенные условия однозначно задают функции возбуждения триггеров устройства реализации функций переходов (см. блок-схему рис. 4.18).

Далее на примере интерфейса Т2 мы продемонстрируем этапы синтеза этого устройства. Однако прежде опишем его поведение в удобном для синтеза виде. Примем здесь: набор сигналов на его входах и выходах образует полное состояние автомата. Пусть, например, автомат находится в состоянии C_0 и на его выходы подан сигнал Р. Его полное состояние в этом случае записывается как $РС_0$.

На рис. 7.4 приведена диаграмма переходов, в которой полное состояние кодируется пятиразрядным двоичным кодом, причем два первых разряда — код входного

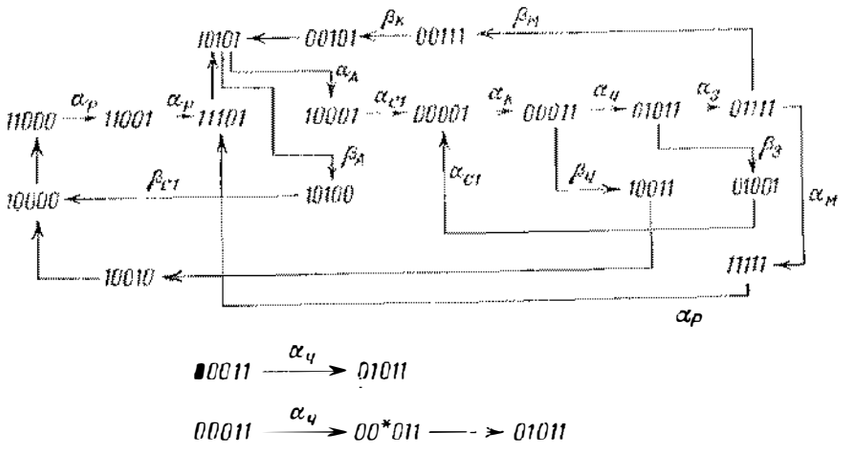


Рис. 7.4. Диаграмма переходов для интерфейса Т2

сигнала (в соответствии с табл. 7.1), третий и четвертый разряды — код выходного сигнала (в соответствии с табл. 7.2), а пятый разряд — состояние триггера Т1 логического подключения (равен 0 в состоянии C_0 и 1 во всех остальных состояниях).

От обычных диаграмм приведенная выше отличается тем, что переходы по дугам, помеченным логическими условиями, осуществляются лишь в том случае, когда эти условия принимают значение 1. Эта диаграмма соответствует рис. 7.3, за исключением двух моментов: а) согласно описанию интерфейса переход из полного состояния РМ в состояние РС невозможен, поэтому он на

диаграмме отсутствует; б) при переходах РС — РВ, АЧ — АС и АВ — АС изменяется состояние триггера Т1, поэтому для сохранения соседней дисциплины переходов указанные переходы осуществляются через транзитные состояния РС, АВТ1, АЧТ1 соответственно.

На диаграмме показаны только устойчивые полные состояния. Рассмотрим подробнее один из переходов, например:

$$00011 \xrightarrow{\alpha_ч} 01011.$$

Этот переход осуществляется следующим образом. Если выполняется условие $\alpha_ч = 1$, то возбуждается переменная, соответствующая второй позиции кода состояния, и затем изменяется значение этой переменной. Таким образом, указанный переход более подробно можно изобразить в виде

$$00011 \xrightarrow{\alpha_ч} 00{*}011 — 01011.$$

Если учесть, что из условий $\alpha_i = 1$ и $\beta_i = 1$ в силу их ортогональности выполняется только одно, то можно сделать вывод, что в каждый момент времени в системе может быть возбуждена только одна переменная, инициирующая данный переход, а снятие возбуждения происходит только путем изменения значения этой переменной. Сказанное означает, что приведенная диаграмма может быть реализована апериодической схемой при условии корректной реализации условий α_i и β_i.

7.4.3. Реализация. Далее будет последовательно изложена процедура синтеза устройства реализации функций переходов. Первым этапом этой процедуры является составление таблицы истинности для функций x_1, x_2, y_1, y_2, Т1 (первые два столбца табл. 7.3). Методика ее составления заключается в следующем.

1. В столбце 1 таблицы в лексикографическом порядке перечислены все 32 набора значений переменных.

2. Против тех наборов, которые встречаются в диаграмме переходов, во второй столбец таблицы записываются полные состояния, в которых оказывается система после перехода. При этом, если некоторая переменная изменяет свое значение независимо от условий, то в соответствующую позицию второго столбца записывается ее значение после перехода. Если переход переменной из 0 в 1 зависит от условия $\alpha_i(\beta_i)$, то в соответствующую

позицию второго столбца записывается символ $\alpha_i(\beta_i)$. Наконец, если от условия $\alpha_i(\beta_i)$ зависит переход переменной из 1 в 0, то в соответствующую позицию записывается $\overline{\alpha}_i(\overline{\beta}_i)$.

Например, состояние 10001 переходит в 00001 при условии $\alpha_{C1} = 1$, и в 11001 при условии $\beta_{C1} = 1$. Соответственно во втором столбце таблицы против набора 10001 записывается $\alpha_{C1}\beta_{C1}001$.

Таблица 7.3

x_1 x_2 y_1 y_2 T1	x_1	x_2	y_1	y_2	T_1	S_1	R_1	S_2	R_2	S_3	R_3
0 0 0 0 0	0	0	0	0	0	0	*	0	*	0	*
1 0 0 0 0	1	1	0	0	0	0	*	0	*	0	*
0 1 0 0 0	0	1	0	0	0	0	*	0	*	0	*
1 1 0 0 0	1	1	0	0	α_P	0	*	0	*	α_P	0
0 0 1 0 0	*	*	*	*	*	*	*	*	*	*	*
1 0 1 0 0	1	0	0	0	0	0	*	0	*	0	*
0 1 1 0 0	*	*	*	*	*	*	*	*	*	*	*
1 1 1 0 0	*	*	*	*	*	*	*	*	*	*	*
0 0 0 1 0	*	*	*	*	*	*	*	*	*	*	*
1 0 0 1 0	1	0	0	0	0	0	*	0	1	0	*
0 1 0 1 0	*	*	*	*	*	*	*	*	*	*	*
1 1 0 1 0	*	*	*	*	*	*	*	*	*	*	*
0 0 1 1 0	*	*	*	*	*	*	*	*	*	*	*
1 0 1 1 0	*	*	*	*	*	*	*	*	*	*	*
0 1 1 1 0	*	*	*	*	*	*	*	*	*	*	*
1 1 1 1 0	*	*	*	*	*	*	*	*	*	*	*
0 0 0 0 1	0	0	β_K	α_K	1	β_K	0	α_K	0	*	0
1 0 0 0 1	$\overline{\alpha}_{C1}$	β_{C1}	0	0	1	0	*	0	*	*	0
0 1 0 0 1	β_{C1}	$\overline{\alpha}_{C1}$	0	0	1	0	*	0	*	*	0
1 1 0 0 1	1	1	α_P	0	1	α_P	0	0	*	*	0
0 0 1 0 1	1	0	1	0	1	*	0	*	0	*	0
1 0 1 0 1	1	0	$\overline{\alpha}_A$	0	$\overline{\beta}_A$	0	α_A	0	*	0	β_A
0 1 1 0 1	*	*	*	*	*	*	*	*	*	*	*
1 1 1 0 1	1	0	1	0	1	*	0	0	*	*	0
0 0 0 1 1	$\beta_ч$	$\alpha_ч$	0	1	1	0	*	*	0	*	0
1 0 0 1 1	1	0	0	$\overline{\alpha}_A$	$\overline{\beta}_A$	0	*	0	α_A	0	β_A
0 1 0 1 1	0	1	α_3	$\overline{\beta}_3$	1	α_3	0	0	β_3	*	0
1 1 0 1 1	*	*	*	*	*	*	*	*	*	*	*
0 0 1 1 1	0	0	$\overline{\alpha}_K$	$\overline{\beta}_K$	1	0	α_K	0	β_K	*	0
1 0 1 1 1	*	*	*	*	*	*	*	*	*	*	*
0 1 1 1 1	α_M	$\overline{\beta}_M$	1	1	1	*	0	*	0	*	0
1 1 1 1 1	1	1	1	$\overline{\alpha}_P$	1	*	0	0	α_P	*	0

3. Функции на встречающихся в диаграмме переходов наборах могут быть заданы произвольно, что обозначается звездочкой в соответствующих местах таблицы.

Вообще говоря, наборы 00000 и 01000 в диаграмме не встречаются. Однако диаграмма отражает работу БУ и подключенного к нему ГУ. Когда ГУ отключено от БУ, возможно возникновение состояний, соответствующих указанным наборам. В этих случаях ГУ сохраняет свое состояние, что отражают строки 1 и 3 табл. 7.3.

Устройство реализации функций переходов строится на триггерах y_1, y_2, Т1. Функции возбуждения этих триггеров можно найти, используя таблицу истинности. Как известно уравнение RS-триггера имеет вид $y' = S \vee \overline{R}y$, $SR = 0$.

Значения S и R как функции y и y' (старое и новое состояния триггера) могут быть определены по табл. 7.4.

Столбцы 3, 4 и 5 табл. 7.3, составленные с помощью табл. 7.4, представляют функции возбуждения триггеров y_1, y_2 и Т1 соответственно. Доопределяя функции возбуждения, получим следующие выражения:

$$S_1 = \overline{x}_1\overline{x}_2\overline{y}_2 T1\beta_K \vee \vee x_1x_2T1\alpha_P \vee x_2y_2\alpha_3,$$

$$R_1 = x_1\overline{x}_2\alpha_A \vee \overline{x}_2y_2\alpha_K;$$

$$S_2 = \overline{x}_1\overline{x}_2\overline{y}_1 T1\alpha_K,$$

$$R_2 = \overline{T}1 \vee x_1\alpha_P \vee x_1\alpha_A \vee \vee x_2\overline{y}_1\beta_3 \vee \overline{x}_2y_1\beta_K;$$

$$S_{T1} = x_1x_2\alpha_P,$$

$$R_{T1} = x_1\overline{x}_2y_1\beta_A \vee x_1\overline{x}_2y_2\beta_A.$$

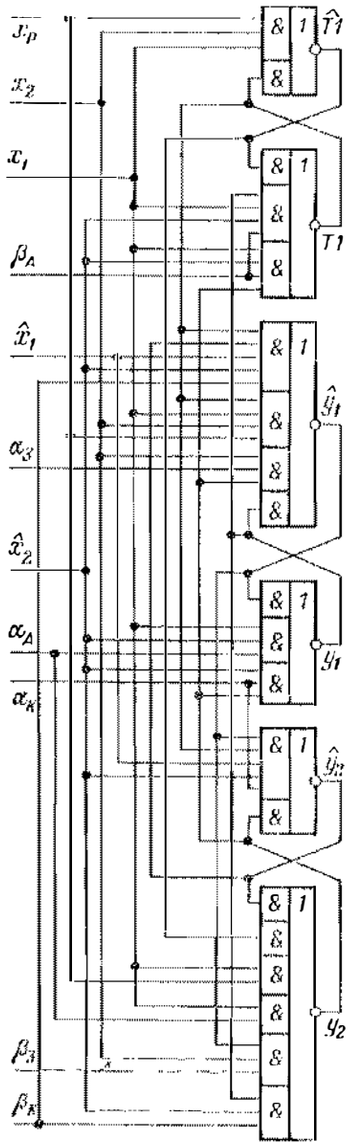


Рис. 7.5. Реализация управляющего автомата интерфейса Т2

Напомним, что все триггеры устройства реализации функций переходов (см. рис. 4.18) должны иметь нулевое транзитное состояние. Таким образом, УА состоит

Таблица 7.4

y	y'	S	R	y	y'	S	R
0	0	0	*	0	*	*	*
1	0	0	1	1	*	*	*
0	1	1	0	0	α	α	0
1	1	*	0	1	$\overline{\alpha}$	0	α

из семи триггеров T_1, T_2, T'_1, T'_2, y_1, y_2 и T1. Схема реализации приведена на рис. 7.5. Условия α_i и β_i вырабатываются в ОА.

§ 7.5. Организация асинхронного интерфейса

Здесь рассматривается «нижний» уровень асинхронного интерфейса и, в частности, одна задача о передаче данных по интерфейсным шинам. Тем не менее ее решение существенно с точки зрения выбора эффективного способа передачи данных в системах телеобработки.

Известные асинхронные интерфейсы обычно предполагают наличие по крайней мере двух групп шин для передачи сообщений по любому из направлений. Одну группу образуют *информационные шины*, по которым передаются данные (число или команда), а другую — *управляющие*, иногда называемые *шинами идентификации*. Для работы интерфейса такого типа обязательна следующая дисциплина: сначала устанавливаются сигналы на информационных шинах, а затем на управляющих шинах соответствующий передаваемой информации идентификатор. При этом задержка в установке идентификатора должна быть такой, чтобы на приемном конце сохранился тот же порядок изменения сигналов, что и на передающем. В противном случае приемник может считать с информационных шин не те значения сигналов, которые были установлены источником. Иными словами, при реализации традиционных интерфейсов необходимо учитывать разброс величин задержек шин (явление

перекоса), который, в частности, зависит от расстояния между источником и приемником. Задержку идентификатора выбирают обычно исходя из наихудших условий приема, что приводит к значительному уменьшению скорости передачи информации.

Здесь предпринята попытка показать возможность организации *интерфейса, инвариантного к явлению перекоса*.

Для исключения влияния перекоса на функционирование интерфейсных устройств необходимо применять как самосинхронизирующиеся коды для представления информации, так и апериодическую схемотехнику при аппаратной реализации.

При реализации асинхронного интерфейса приняты следующие допущения:

1) задержки в интерфейсных шинах и разность их значений (перекос) могут быть любыми, но конечными;

2) задержки элементов и соединительных проводов внутри интерфейсного автомата удовлетворяют гипотезе о задержках, принятой для апериодических автоматов (см. гл. 4).

Ясно, что указанные допущения не позволяют использовать представление входных переменных однофазными сигналами в сопровождении управляющего (фазового) сигнала, поскольку эта форма представления критична к перекосу задержек в шинах. Поэтому при передаче информации в данном случае используются *самосинхронизирующиеся коды в изменениях* (см. § 3.7).

Каждый из двух взаимодействующих процессов может функционировать в одном из двух режимов — источника и приемника. Взаимодействие процессов осуществляется по принципу «ведущий — ведомый» с *квитированием*, т. е. процесс, играющий роль ведущего, после передачи очередного кодового набора должен получить от ведомого квитанцию о приеме и использовании этого набора.

Будем считать, что каждый из взаимодействующих процессов структурно реализуется некоторым интерфейсным автоматом, состоящим из *передатчика*, *приемника* и *блока управления*, координирующего попеременную работу передатчика и приемника. В состав передатчика входит кодирующее устройство, переводящее передаваемую информацию в наборы самосинхронизирующегося кода, и выходной преобразователь наборов самосинхронизирующего кода — в код в изменениях. Приемник содержит

преобразователь наборов входных сигналов в наборы самосинхронизирующегося кода, тестор входных наборов и, возможно, блок декодирования самосинхронизирующегося кода в код, удобный для дальнейшего использования.

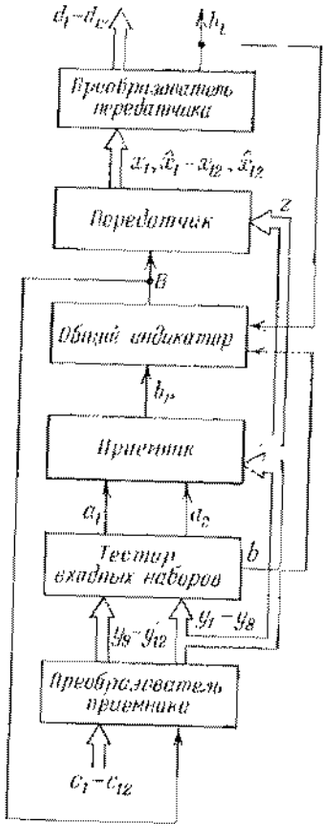


Рис. 7.6. Структурная схема интерфейсного автомата для побайтной передачи информации в кодах с идентификатором

Перечисленные блоки интерфейсного автомата, за исключением преобразователя приемника и тестора, могут быть построены с помощью вышеизложенных подходов к построению апериодических схем.

Ниже описываются способы организации интерфейсного автомата для побайтной передачи данных.

7.5.1. Передача при использовании кода с идентификатором. Один из вариантов структурной схемы интерфейсного автомата для побайтной передачи информации с использованием кода с идентификатором (§ 3.3) изображен на рис. 7.6.

Для сокращения описания функционирования схемы примем следующие обозначения:

$Z = \{z_1, \ldots, z_8\}$ — множество наборов двоичного позиционного кода, передаваемых из приемной части в передающую;

$X = \{x_1, \hat{x}_1, \ldots, x_{12}, \hat{x}_{12}\}$ — множество наборов, образующих парафазное представление передаваемого набора кода с идентификатором;

i_X — пулевой спейсер для наборов из X;

$D = \{d_1, \ldots, d_{12}\}$ — множество наборов кода в изменениях на выходе передатчика;

$Y = \{y_1, \ldots, y_{12}\}$ — множество наборов кода с идентификатором на выходе преобразователя приемника;

e_Y — единичный спейсер для наборов из Y;

$C = \{c_1, \ldots, c_{12}\}$ — множество наборов кода в изменениях на входе приемника;

$Y^* = \{y_1, \ldots, y_{12}\}$ — множество наборов значений обязательных разрядов кода с идентификатором на выходе преобразователя приемника.

Набор из некоторого множества U будем обозначать соответствующей строчной буквой с верхним индексом, например u^i.

Передающая часть схемы состоит из передатчика и преобразователя передатчика. Передатчик, используя z^i, вырабатывает x^i. Управление работой передатчика осуществляется фазовым сигналом B (выход общего индикатора). При $B = 1$ на выходе передатчика устанавливается i_X. При $B = 0$ работа передатчика не зависит от z^i, на его выходе устанавливается набор x^i.

Преобразователь передатчика реализует отображение X в D. В состав преобразователя входит индикатор b_t моментов окончания переходных процессов, который является индикатором всего передающего устройства. При i_X на выходах преобразователя сохраняется предыдущее значение d^p и $b_t = 0$, при x^q устанавливается новый набор d^q (имеет место переход $d^p - d^q$) и $b_t = 1$.

Приемная часть содержит преобразователь приемника, тестор для y^i и сам приемник.

Преобразователь приемника реализует отображение C в Y. Управление преобразователем осуществляется фазовым сигналом B (выходом общего индикатора). При $B = 0$ преобразователь хранит ранее записанный в него набор c^i. Если входной набор c^i совпадает с записанным в преобразователь, последний выдает спейсер (использование единичного спейсера вместо нулевого продиктовано желанием упростить схемную реализацию). После установки на входах нового набора c^j на выходах появляется соответствующий ему набор y^j. При $B = 1$ происходит запись c^j в регистр преобразователя, после окончания которой на выходе преобразователя устанавливается набор e_Y.

Тестор входных наборов вырабатывает $a_1 = 1$, $a_2 = 0$ ($a_1 = 0$, $a_2 = 1$), если набор y^j является обязательным (вспомогательным), а затем $b = 1$. Если же на выходах преобразователя установлен набор e_Y, то на выходах индикатора $a_1 = a_2 = 0$, а затем $b = 0$.

Приемник использует набор y^{j*}, предварительно преобразуя, если необходимо, информацию в позиционный двоичный код. Сигналы a_1 и a_2 играют роль фазовых сигналов приемника. При $a_1 = 1$, $a_2 = 0$ или $a_1 = 0$, $a_2 = 1$ в

приемник поступает набор y^{j*} и образуется код z^l, используемый передатчиком для выработки новой порции передаваемой информации. После окончания переходных процессов в приемнике $b_r = 1$. При $a_1 = a_2 = 0$ состояние приемника не зависит от y^{j*}, на его выходах сохраняется z^l и устанавливается $b_r = 0$.

Помимо указанных блоков, структурная схема рис. 7.6 содержит общий индикатор с выходом B, который управляет порядком работы приемного и передающего устройств.

Совместное функционирование приемного и передающего устройств в составе интерфейсного автомата отражено сигнальным графом рис. 7.7. Этот граф можно трактовать, если нужно, как протокол взаимодействия между

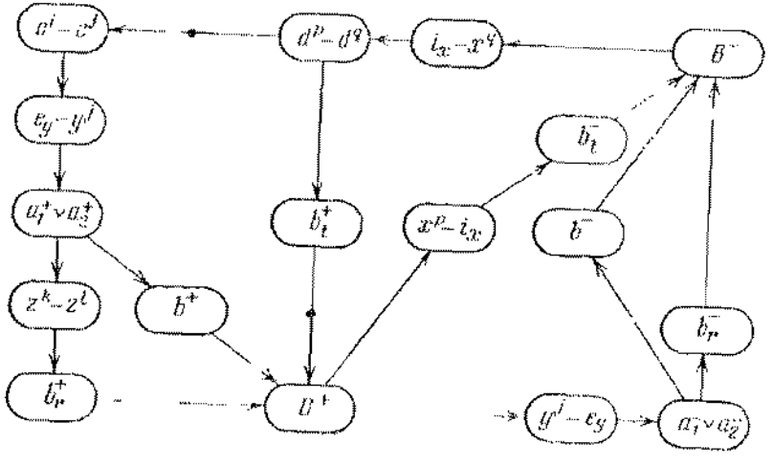


Рис. 7.7. Сигнальный граф для автомата рис. 7.4

приемным и передающим устройствами внутри одного интерфейсного автомата, однако такой протокол не служит целям описания обмена между двумя интерфейсными автоматами.

В исходном состоянии, соответствующем начальной маркировке, приемное устройство подготовлено к приему очередной порции информации, а передающее устройство находится в фазе передачи.

Ниже дается комментарий, поясняющий динамику функционирования сигнального графа рис. 7.7.

Сначала на входах преобразователя приемника устанавливается новый кодовый набор (переход $c^i - c^j$), затем происходит перекодирование его в соответствующий набор y^j (переход $e_Y - y^j$),

после чего $a_1 \vee a_2 = 1$ и $b = 1$. После $a_1 = 1$ или $a_2 = 1$ ($a_1 a_2 = 0$) приемник осуществляет прием и использование кода y^{j*}, вырабатывает код z^l, соответствующий состоянию приемника (переход $z^k - z^l$) и затем устанавливает $b_r = 1$. На выходе общего индикатора $B = 1$ устанавливается только после выполнения условия $b_t b_r b = 1$ (синхронизация условий $b_t = 1$, $b_r = 1$, $b = 1$). Появление $B = 1$ свидетельствует об окончании фазы приема и использования приемным устройством поступившей информации. Сигнал $B = 1$ переводит приемное устройство в фазу подготовки к восприятию следующей порции информации, а передающее устройство — из фазы передачи информации в фазу подготовки к следующей передаче. Этот же сигнал инициирует прием информации в преобразователь. После окончания приема на выходах преобразователя устанавливается спейсер e_Y (переход $y^j - e_Y$). Этот факт отмечается тестором входного набора ($a_1 = a_2 = b = 0$). После $a_1 = a_2 = 0$ начинается вторая фаза работы приемника (подготовка к восприятию новой порции информации), после окончания которой $b_r = 0$. По условию $b_r = b = 0$ заканчивается вторая фаза работы приемного устройства (хранение принятого набора).

В передающем устройстве после $B = 1$ и окончания в нем переходного процесса на выходе передатчика устанавливается i_X (переход $x^p - i_X$), затем $b_t = 0$. При этом выходной преобразователь сохраняет на выходах прежний набор d^p. Сигнал $b_t = 0$ свидетельствует об окончании соответствующей фазы передатчика. После выполнения условия $b_t = b_r = 0$ (синхронизация событий $b_t = 0$ и $b_r = 0$) выход общего индикатора $B = 0$. Вслед за $B = 0$ (передающее устройство переводится в фазу передачи следующего кодового набора) начинается установка на выходах передатчика очередного набора кода с идентификатором (переход $i_X - x^q$). Этот набор преобразуется в выходной набор передающего устройства (переход $d^p - d^q$) и после завершения преобразования $b_t = 1$. Схема вернулась в исходное состояние.

Говоря о реализации интерфейсного автомата, отметим следующее.

1. Передатчик может быть построен с использованием стандартных методов, изложенных в гл. 4. При этом необходимо учесть, что выход передатчика был представлен парафазным кодом.

2. Преобразователь передатчика соответствует описанному в примере 4.3 и 4.5.3.

3. Преобразователь приемника строится как регистр на D-триггерах типа рис. 4.11, *в* (§ 4.4).

4. Тестор входных наборов приемника описан в § 4.2.

5. Приемник строится как обычный согласованный автомат с фазовым сигналом $a_1 (a_2)$.

6. Общий индикатор интерфейсного автомата реализуется на базе Г-триггера в соответствии с уравнением

$$B = b_t b_r b \vee B(b_t \vee b_r \vee b).$$

7.5.2. Передача при использовании оптимального равновесного кода. Ниже на двух примерах показана возможность реализации интерфейсных устройств, использующих оптимальный равновесный код.

Полубайтный обмен. Схема, использующая $c_6^3 = 20$ кодовых наборов, построена так, что 16 из них (обязательные наборы) перекодируются в 4-разрядный позиционный двоичный код. Структура и порядок работы схемы — те же, что и схемы на рис. 7.4. Если не считать изменения числа информационных связей, модифицируются лишь блоки преобразователя приемника и тестора входных наборов. Поэтому ниже описываются схемы именно этих блоков.

Преобразователь приемника со входами $c_1 - c_6$ и выходами $y_1 - y_4$, $\hat{y}_3$, $\hat{y}_4$ наряду с перекодированием наборов кода в изменениях в наборы оптимального равновесного кода осуществляет преобразование последних в 4-разрядный позиционный двоичный код. Схема преобразователя реализуется по табл. 7.5; с учетом доопределения на запрещенных наборах реализации функций $y_1 - y_4$, $\hat{y}_3$, $\hat{y}_4$ имеют вид

$$y_1 = p_1, \; y_2 = p_2,$$

$$\hat{y}_3 = \overline{p_1p_6 \vee p_2p_4 \vee p_4p_6}, \quad \hat{y}_4 = \overline{p_1p_5 \vee p_2p_6 \vee p_5p_6}$$

$$y_3 = \overline{p_1p_3 \vee p_2p_5 \vee p_3p_5}, \quad y_4 = \overline{p_1p_4 \vee p_2p_3 \vee p_3p_4}.$$

Тестор входных наборов со входами y_3, $\hat{y}_3$, y_4, $\hat{y}_4$ вырабатывает три сигнала — a_1, a_2 и b. Комбинация $a_1 = 1$, $a_2 = 0$ означает, что на выходах $y_1 - y_4$ схемы преобразователя приемника установлен один из 16 позиционных кодов и при этом $\hat{y}_3 \neq y_3$, $\hat{y}_4 \neq y_4$, т. е. 3-й и 4-й разряды представлены парафазным кодом. При $a_1 = 0$, $a_2 = 1$ на выходах $y_1 - y_4$, $\hat{y}_3$, $\hat{y}_4$ установлен один из вспомогательных наборов, указанных в четырех последних строках табл. 7.5. При $a_1 = 0$, $a_2 = 0$ на выходах $y_1 - y_4$, $\hat{y}_3$, $\hat{y}_4$ — набор гашения i_Y. Порядок изменения сигналов a_1, a_2 и b — тот же, что и для схемы рис. 7.7.

Сигналы a_1, a_2 и b реализованы в виде

$$\overline{a_1} = \overline{y_3y_4 \vee y_3\hat{y}_4 \vee \hat{y}_3y_4 \vee \hat{y}_3\hat{y}_4}, \quad \overline{a_2} = \overline{y_3\hat{y}_3 \vee y_4\hat{y}_4},$$

$$\overline{b} = \overline{\overline{a_1}\overline{a_2}\overline{a_3} \vee \overline{b}(\overline{a_1}\overline{a_2} \vee \overline{a_3})},$$

где $\overline{a_3} = y_3 \vee \hat{y}_3 \vee y_4 \vee \hat{y}_4$.

Принадлежность схемы тестора классу апериодических проверяется посредством установления ее индицируемости в соответствии с п. 4.3.1 в силу утверждения 4.4.

Побайтный обмен. Он осуществляется на основе предложенного выше интерфейсного автомата для обмена полубайтами. В этом случае для передачи байта требуется

Таблица 7.5

$p_1\,p_2\,p_3\,p_4\,p_5\,p_6$	$y_1\,y_2\,y_3\,\hat{y}_3\,y_4\,\hat{y}_4$
0 0 0 0 0 0	0 0 0 0 0 0
0 0 0 1 1 1	0 0 0 1 0 1
0 0 1 0 1 1	0 0 1 0 0 1
0 0 1 1 0 1	0 0 0 1 1 0
0 0 1 1 1 0	0 0 1 0 1 0
1 0 0 0 1 1	1 0 0 1 0 1
1 0 1 0 1 0	1 0 1 0 0 1
1 0 0 1 0 1	1 0 0 1 1 0
1 0 1 1 0 0	1 0 1 0 1 0
0 1 0 1 0 1	0 1 0 1 0 1
0 1 0 0 1 1	0 1 1 0 0 1
0 1 1 1 0 0	0 1 0 1 1 0
0 1 1 0 1 0	0 1 1 0 1 0
1 1 0 0 0 1	1 1 0 1 0 1
1 1 0 0 1 0	1 1 1 0 0 1
1 1 0 1 0 0	1 1 0 1 1 0
1 1 1 0 0 0	1 1 1 0 1 0
1 0 0 1 1 0	1 0 0 0 1 1
1 0 1 0 0 1	1 0 1 1 0 0
0 1 0 1 1 0	0 1 1 1 0 0
0 1 1 0 0 1	0 1 0 0 1 1

12 пин, т. е. преобразователь приемного устройства имеет 12 входов $c_1 - c_{12}$. Его выходами являются $y_1 - y_4$, $\hat{y}_3$, $\hat{y}_4$, y_5, y_6, y_7, $\hat{y}_7$, y_8, $\hat{y}_8$. Сигналы y_3, $\hat{y}_3$, y_4, $\hat{y}_4$ и y_7, $\hat{y}_7$, y_8, $\hat{y}_8$ поступают соответственно на входы двух индикаторов тестора входных наборов. Обозначим выходы этих двух индикаторов через $\overline{a_1^1}$, $\overline{a_2^1}$, $\overline{a_3^1}$ и $\overline{a_1^2}$, $\overline{a_2^2}$, $\overline{a_3^2}$ (схемы сигналов $\overline{b^1}$ и $\overline{b^2}$ не используются и могут быть исключены),

тогда a_1, a_2 и b могут быть реализованы в виде

$$a_1 = \overline{a_1^1 \vee a_1^2},\quad a_2 = \overline{a_1^1 a_2^1 \vee a_1^1 a_2^2 \vee a_1^2 a_2^2 \vee a_2^1 a_2^2},$$
$$b = a_1 a_3 a_4 \vee a_2 a_3 a_4 \vee b(a_1 \vee a_2 \vee a_3 \vee a_4),$$

где $a_3 = \overline{a_3^1 a_3^2}$, $a_4 = \overline{a_3^1 \vee a_3^2}$.

При этом используется для обмена $20 \times 20 = 400$ кодовых наборов, из которых 256 предназначены для передачи 8-разрядных позиционных двоичных кодов (обязательных наборов).

Использование неравновесного кодирования. Здесь приводится пример реализации интерфейса, использующего для побайтного обмена минимальное число разрядов и неравновесное кодирование.

Минимальное число разрядов оптимального равновесного кода, требуемое для кодирования $N^0 = 2^8 = 256$ наборов 8-разрядного позиционного двоичного кода равно 11 ($c_{10}^5 = 252 < 256 < c_{11}^5 = 462$). За счет того, что требуется кодировать только 256 наборов (из 462), можно, отказавшись от равновесного кодирования, упростить функцию преобразования и использовать при этом минимальное число разрядов.

Будем использовать код, далее называемый модифицированным кодом с идентификатором. Шестью его младшими разрядами будем без перекодирования передавать шесть младших разрядов двоичного позиционного кода. Остальные пять разрядов модифицированного кода с идентификатором отведем под идентификатор, который должен выполнять следующие функции:

— определение веса w набора в основных (младших) шести разрядах,

— кодирование оставшихся двух старших разрядов позиционного кода.

Поскольку каждому набору в основных разрядах необходимо поставить в соответствие четыре комбинации значений двух старших разрядов, одному w-классу должно соответствовать не менее четырех идентификаторов.

Предлагаемый ниже вариант кодирования, в котором каждому w-классу соответствует 4 идентификатора, удовлетворяет трем условиям для кода с идентификатором из § 3.3. Соответствие идентификаторов классам наборов в основных разрядах и правило перекодирования идентификаторов в двух старших разрядах позиционного двоичного кода представлены в табл. 7.6.

Структурная схема описываемого устройства аналогична схеме рис. 7.6. По другому построены блоки преобразователя приемника и тестора входных наборов.

Таблица 7.6

w	$p_1\ p_2\ p_3\ p_4\ p_5$	$y_7\ \hat{y}_7\ y_8\ \hat{y}_8$
0	0 1 1 1 1	0 1 0 1
0	1 0 1 1 1	1 0 0 1
0	1 1 0 1 1	0 1 1 0
0	1 1 1 0 1	1 0 1 0
1	0 1 1 1 0	0 1 0 1
1	1 0 1 1 0	1 0 0 1
1	1 1 0 1 0	0 1 1 0
1	1 1 1 0 0	1 0 1 0
2	0 0 1 1 1	0 1 0 1
2	1 0 1 0 1	1 0 0 1
2	0 1 0 1 1	0 1 1 0
2	1 1 0 0 1	1 0 1 0
3	0 0 1 1 0	0 1 0 1
3	1 0 0 1 1	1 0 0 1
3	0 1 1 0 1	0 1 1 0
3	1 1 0 0 0	1 0 1 0
4	0 1 0 1 0	0 1 0 1
4	1 0 0 1 0	1 0 0 1
4	0 1 1 0 0	0 1 1 0
4	1 0 1 0 0	1 0 1 0
5	0 1 0 0 1	0 1 0 1
5	0 0 0 1 1	1 0 0 1
5	0 0 1 0 1	0 1 1 0
5	1 0 0 0 1	1 0 1 0
6	0 1 0 0 0	0 1 0 1
6	0 0 0 1 0	1 0 0 1
6	0 0 1 0 0	0 1 1 0
6	1 0 0 0 0	1 0 1 0

Преобразователь приемника:

— реализует преобразование кода в изменениях в модифицированный код с идентификатором,

— перекодирует идентификаторы модифицированного кода в наборы двухразрядного двоичного позиционного кода,

— вырабатывает промежуточные сигналы g_1 и g_2, необходимые для индикации окончания переходных процессов в схеме преобразователя.

Преобразование кода в изменениях в модифицированный код с идентификатором осуществляется 11-разрядной схемой, один разряд которой был уже описан.

С учетом доопределения на запрещенных наборах реализация функций y_7, $\hat{y}_7$, y_8, $\hat{y}_8$ имеет вид

$$y_7 = \overline{p_7p_9 \vee u_1u_2p_7p_8p_{11} \vee u_1u_2u_3p_7 \vee u_5p_{10}p_{11} \vee u_5u_6p_{10}},$$

$$\hat{y}_7 = \overline{p_8p_{10} \vee u_1u_2p_9p_{10}p_{11} \vee u_1u_2u_3p_9p_{10} \vee u_1u_2u_3p_8p_9 \vee u_5p_9p_{11} \vee u_5u_6p_9 \vee u_1u_2u_3u_4p_8},$$

$$y_8 = \overline{p_7p_8p_{11} \vee u_1p_7p_8 \vee u_1u_2p_8p_{10}p_{11} \vee u_3p_8p_9p_{11} \vee u_1u_2p_9u_3u_4 \vee u_4u_5p_7p_{11} \vee u_4u_5u_6p_7},$$

$$\hat{y}_8 = \overline{p_9p_{10}p_{11} \vee u_1p_9p_{10} \vee u_1u_2p_7p_9p_{11} \vee u_3p_7p_{10}p_{11} \vee u_5p_8p_{11} \vee u_5u_6p_8 \vee u_1u_2u_3u_4p_{10}},$$

где

$$u_1 = \overline{eh}_1, \quad u_2 = \overline{e_1e_2h_2 \vee e_2h_1h_2},$$

$$u_3 = \overline{e_1e_3h_3 \vee e_2e_3h_2h_3 \vee e_3h_1h_3},$$

$$u_4 = \overline{e_1 \vee h_1 \vee e_2h_2 \vee e_2h_3 \vee e_3h_2}, \quad u_5 = \overline{e_2 \vee h_2 \vee e_3h_3},$$

$$u_6 = \overline{e_3 \vee h_3}, \quad e_1 = \overline{p_1 \vee p_2 \vee p_3}, \quad e_2 = \overline{p_1p_2 \vee p_1p_3 \vee p_2p_3},$$

$$e_3 = \overline{p_1 \vee p_2 \vee p_3}, \quad h_1 = \overline{p_4 \vee p_5 \vee p_6},$$

$$h_2 = \overline{p_4p_5 \vee p_4p_6 \vee p_5p_6}, \quad h_3 = \overline{p_4p_5p_6}.$$

Схема тестора входных наборов реализована в соответствии с уравнениями

$$g_1 = \overline{\overline{\left(\bigvee_{i=1}^{6} u_i\right) \vee p_{11} \cdot \bigvee_{i=7}^{10} p_i}},$$

$$g_2 = \overline{\overline{\bigvee_{i=1}^{6} u_i \vee p_{11} \vee \bigvee_{i=7}^{10} p_i}},$$

$$b = (y_7 \vee \hat{y}_7)(y_8 \vee \hat{y}_8)g_1g_2 \vee b(y_7 \vee \hat{y}_7 \vee y_8 \vee y_8 \vee g_1 \vee g_2).$$

§ 7.6. Замечания по библиографии

С протоколами информационного обмена, вопросами их представления, верификации, классификации, синтеза и т. д. связана обширная библиография, например [223]. По существу, можно говорить о теории протоколов как о самостоятельном направлении; косвенным подтверждением этого является появление специального выпуска журнала [244], содержащего ряд обзоров и статей по различным аспектам этой теории [201, 202, 220, 303]. Выпущен ряд книг [51, 93, 154, 174, 190], в которых вопросам протоколов уделяется значительное внимание. Тем не менее работы по формальной теории протоколов в имеющемся потоке литературы относительно мало представлены. Среди них следует отметить [203, 204, 233, 241, 265, 267, 303, 309, 313, 314] и обзоры [266, 298]. Основные положения статьи [148] представляют собой заимствование результатов из [9]. Предложенная в гл. 7 формализованная версия понятия протокола опубликована в [9, 61, 147]. Из сделанных построений следует вывод о том, что протокол может быть описан на языке моделей, являющихся интерпретацией асинхронного процесса. Описание его в виде пар инициатор — результант позволяет отказаться от обычного для большинства подходов избыточного описания протоколов.

Пример 7.1 составлен по мотивам заметки [315].

Организация асинхронного интерфейса при использовании оптимальных равновесных кодов и кодов с идентификаторами описана в [6, 129, 130]. О явлении перекоса задержек сообщалось в [114].

Подходы к построению моделей управляемого протокола рассмотрены в [146, 255].

Один физик-теоретик заработал себе прочный авторитет тем, что, исходя из весьма общих математических соображений, вывел формулу для радиуса Вселенной. Это была очень впечатляющая формула, щедро начиненная константами e, c, $\hbar$, несколько раз в ней встречалось число π и много квадратных корней. Поскольку он был убежденным теоретиком, его не беспокоили численные значения. Прошло несколько лет, пока нашелся человек, которому захотелось узнать, чему равен радиус Вселенной.

Оказалось, десяти сантиметрам.

Ян Стюарт

Глава 8. АНАЛИЗ АСИНХРОННЫХ СХЕМ И ПРОЦЕССОВ

Сложность управления асинхронными процессами обусловливает необходимость разработки специальных методов их *анализа*. Наиболее распространенными задачами анализа при асинхронной реализации вычислений является выявление отсутствия тупиковых ситуаций или, в более общем случае, *дедлоков* (калька с английского слова deadlock — тупик) в системе и независимости результата, вырабатываемого в ходе функционирования системы, от длительности работы отдельных ее частей (фрагментов или модулей). Подобные задачи анализа возникают на всех этапах проектирования систем управления асинхронными процессами: при разработке параллельных языков и протоколов межмодульного взаимодействия, при синтезе микропрограммного управления параллельными процессами, при реализации аппаратного управления асинхронными процессами, в частности средствами апериодической схемотехники.

Здесь в центре внимания будут именно вопросы *анализа функционирования* аппаратных средств управления асинхронными процессами, т. е. *асинхронных схем*, хотя материал данной главы легко может быть использован для анализа параллельных процессов на других уровнях вычислительных систем.

Функционирование схемы в динамике представляет собой некоторым образом упорядоченную во времени последовательность срабатываний ее элементов. Длительности переключения элементов могут существенно отличаться друг от друга, так как они определяются физическими свойствами элементов и меняются с течением времени. В общем случае некоторое число элементов может срабатывать одновременно. Работа схемы может быть описана асинхронным процессом переключения ее элементов.

Все известные достоинства апериодических схем являются следствием того, что эти схемы правильно функционируют независимо от величин задержек составляющих их элементов. Проверка свойств схем, обеспечивающих такую работу, является предметом анализа апериодических схем. Задачи анализа в этом случае отличаются от традиционных для синхронных и асинхронных схем.

При исследовании синхронных схем обычно вычисляют функции, реализуемые схемой, находят реакцию схемы на заданное входное воздействие, определяют входной сигнал, вызвавший заданную реакцию, и т. п.

Считается, что асинхронная схема, свободная от критических состязаний (риска), работоспособна, поэтому традиционной задачей анализа асинхронных схем является проверка на отсутствие состязаний. Обычно в моделях асинхронных логических схем предполагается завершение переходных процессов в течение некоторого фиксированного интервала времени, который определяется с учетом того, что задержка любого элемента схемы даже в худшем случае не превосходит заданной величины. Если же отказаться от этого ограничения и считать, что схема должна быть работоспособна при любых конечных задержках элементов, то *отсутствие критических состязаний становится недостаточным условием обеспечения работоспособности схем.*

Схемы, не зависящие от скорости (в том числе полумодулярные, дистрибутивные и последовательные), работоспособны при произвольных конечных задержках элементов. Естественно, что во всех указанных классах схем отсутствуют недопустимые в асинхронных схемах всех видов всплески и колебания сигналов на выходах элементов, т. е. критические состязания.

Описание поведения схемы как асинхронного процесса удобно использовать для анализа апериодических схем на ранних стадиях их проектирования. Асинхронный процесс позволяет представить работу схемы с минимальной конкретизацией, без учета ограничений, вносимых определенной технической реализацией. В начале

разработки устройств, когда определены только «узловые» точки в работе схемы, число ситуаций в соответствующем процессе может быть невелико. Конкретизируя описание системы (например, кодируя ситуации асинхронного процесса или используя такие его интерпретации, как модель Маллера или сеть Петри) можно выявить локальные свойства, наличие которых гарантирует корректное поведение системы (например, управляемость соответствующего ей асинхронного процесса).

§ 8.1. Анализ достижимости

При исследовании функционирования дискретных систем (в том числе и схем) часто возникает необходимость определения отношения следования между ситуациями или множествами ситуаций (в терминах, используемых в асинхронных процессах), т. е. решения так называемой *задачи достижимости*.

Возможны две постановки этой задачи:

1) построение множества ситуаций, достижимых из заданной ситуации или множества ситуаций,— *прямая задача* достижимости;

2) построение множества ситуаций, из которых достижима заданная ситуация или множество ситуаций,— *обратная задача* достижимости.

Возвращаясь к материалу § 2.4 и излагая результаты, связанные с решением задачи достижимости, на языке модели Маллера, будем объект исследования здесь называть *схемой*, а переменную z_i схемы — *элементом*.

Отношение достижимости на множестве Z состояний схемы определим как транзитивное замыкание введенного в § 2.4 отношения непосредственного следования. Иными словами, состояние β достижимо из состояния α, если в диаграмме переходов найдется траектория (путь), ведущий из вершины α в вершину β. *Условимся факт достижимости* β *из* α *обозначать через* $\alpha \Rightarrow \beta$. Достижимость $\alpha \Rightarrow \beta$, таким образом, имеет место, если в диаграмме переходов существуют такие состояния $\delta, \gamma, \ldots, \psi$, что выполняется $\alpha \to \delta \to \gamma \to \ldots \to \psi \to \beta$, где $\to$ — отношение непосредственного следования, введенное в конце § 2.4.

Введем также понятие *достижимости по соседям* $\alpha \overset{1}{\Rightarrow} \beta$, являющееся сужением понятия достижимости для случая, когда любой непосредственный переход в последовательности $\alpha \to \delta \to \gamma \ldots \to \psi \to \beta$ является соседним (связан с изменением значения какой-либо одной переменной), т. е. имеет место (в соответствии с обозначениями § 2.4) $\alpha \xrightarrow{1} \delta \xrightarrow{1} \gamma \xrightarrow{1} \ldots \xrightarrow{1} \psi \xrightarrow{1} \beta$.

Обобщение понятия достижимости для множеств состояний схемы становится очевидным после того, как определены следующие два понятия.

Множество B состояний схемы является *непосредственным предшественником* множества A состояний схемы, если для каждого $\beta \in B$ найдется $\alpha \in A$ такое, что $\beta \to \alpha$ (см. § 2.4), и если для любого $\alpha \in A$ найдется $\beta \in B$ такое, что $\beta \to \alpha$. По аналогии определяется *непосредственный последователь* множества A.

Таким образом, прямая задача достижимости состоит в том, чтобы для заданного множества A состояний схемы определить все такие состояния β, что для некоторого $\alpha \in A$ будет иметь место $\alpha \Rightarrow \beta$. Обратная задача достижимости предполагает поиск всех таких β, что для $\alpha \in A$ будет выполняться $\beta \Rightarrow \alpha$.

Прямая и обратная задачи достижимости по соседям предполагают соответственно поиск всех таких состояний β, при которых будет иметь место $\alpha \overset{1}{\Rightarrow} \beta$ и $\beta \overset{1}{\Rightarrow} \alpha$.

Разработке алгоритмов решения задач достижимости и посвящен этот параграф.

Для решения задач достижимости может быть предложен итеративный алгоритм, на каждом шаге которого определяется непосредственный последователь (прямая достижимость) или непосредственный предшественник (обратная достижимость) множества состояний, полученного на предыдущем шаге, а на первом шаге — для исходного множества. Процесс анализа достижимости завершается, когда на очередном шаге не получено ни одного нового состояния. Таким образом задача сводится к нахождению непосредственного предшественника или последователя для заданного множества состояний схемы.

В принципе задачи достижимости могут решаться непосредственно по диаграмме переходов, однако для реальных схем такой способ описания работы схемы оказывается слишком громоздким, что потребовало разработки методов анализа, оперирующих с более компактным представлением схемы — системой уравнений.

Состоянию α схемы поставим в соответствие конституенту $\omega(\alpha)$ так, как это делалось в § 3.1. Тогда множе-

ство состояний схемы можно задать *характеристической булевой функцией* $\varphi_A(Z)$, где

$$\varphi_A(Z) = \bigvee_{\alpha \in A} \omega(A).$$

Введем в рассмотрение *два оператора* P и Q, которые для множества A состояний схемы определяют *непосредственного предшественника* B и *непосредственного последователя* D: $P(\varphi_A(Z)) = \varphi_B(Z)$ и $Q(\varphi_A(Z)) = \varphi_D(Z)$. Для частного случая — достижимости по соседям — введенные операторы будем обозначать через P^1 и Q^1 соответственно.

Утверждение 8.1. *$P(\varphi_A(Z))$ получается из $\varphi_A(Z)$ заданной в нормальной форме подстановкой $z_i \vee f_i(Z)$ вместо каждого вхождения z_i и $\bar{z}_i \vee \bar{f}_i(Z)$ вместо каждого вхождения $\bar{z}_i$, $1 \leqslant i \leqslant n$.*

Ограничимся здесь содержательным обоснованием утверждения. Пусть на множестве состояний $A' \subseteq A$ значение переменной $z_i = 1$. Каким условиям должна удовлетворять переменная z_i в состояниях, принадлежащих непосредственному предшественнику A'? Очевидно, должно выполняться одно из двух условий: либо $z_i = 1$, т. е. при переходе в состояния из A' значение переменной z_i не изменяется, либо $f_i(Z) = 1$, т. е. если $z_i = 0$, то переменная z_i возбуждена и при переходе в состояния из A' происходит изменение ее значения. Аналогично для случая $z_i = 0$.

Следствия утверждения 8.1:

$$P^1(\varphi_A(Z)) = \varphi_A(Z) \vee \bigvee_{i=1}^{n} \varphi_A(z_i = f_i(Z)), \qquad (8.1)$$

$$P^1(\varphi_A(Z)) = \varphi_A(Z) \vee \bigvee_{i=1}^{n} \varphi_A(z_i = \bar{z}_i)(z_i \oplus f_i(Z)). \qquad (8.2)$$

Здесь обозначение $\varphi_A(z_i = c)$ имеет тот же смысл, что в определении 3.1, т. е. $\varphi_A(z_i = c) = \varphi_A(z_1, z_2, \ldots, z_{i-1}, c, z_{i+1}, \ldots, z_n)$, а $f_i(Z)$ — собственная функция i-го элемента схемы. Легко убедиться, что формулы (8.1), (8.2) справедливы для любой формы представления $\varphi_A(Z)$.

В формуле (8.2) функция $\varphi_A(z_i = \bar{z}_i)$ задает множество B состояний, отличающихся от состояний из множества A разве лишь значением переменной z_i, функция $z_i \oplus f_i(Z)$ — множество состояний, в которых возбуждена переменная z_i, а их конъюнкция — те состояния из B, в которых z_i возбуждена. Таким образом, из этих состояний непосредственно достижимы по соседям состояния из множества A.

Аналитическое представление оператора прямой достижимости оказывается слишком громоздким, и, поскольку для целей анализа схем он не потребуется, ограничимся получением формул для оператора прямой достижимости по соседям — Q^1. Заметим, что выражение $\varphi_A(Z) \cdot (z \oplus f_i(Z))$ задает множество G состояний из A, в которых возбуждена переменная z_i, а выражение — $\varphi_A(z_i = \bar{z}_i)(\bar{z}_i \oplus f_i(z_i = \bar{z}_i))$ — множество состояний, отличающихся от состояний из G значением переменной z_i. Легко видеть, что каждое состояние α из D непосредственно достижимо по соседям из состояний $\alpha' \in G \subseteq A$, которое отличается от α только значением переменной z_i. Поэтому

$$Q^1(\varphi_A(Z)) = \varphi_A(Z) \vee \bigvee_{i=1}^{n} \varphi_A(z_i = \bar{z}_i)(\bar{z}_i \oplus f_i(z_i = \bar{z}_i)). \qquad (8.3)$$

Можно показать также, что

$$Q^1(\varphi_A(Z)) = \varphi_A(Z) \vee \bigvee_{i=1}^{n} \varphi_A(z_i = \bar{f}_i(z_i = \bar{z})). \qquad (8.4)$$

Пример 8.1. Возвратимся к системе уравнений примера 2.8, которой соответствует схема рис. 4.2, *г* (на этот раз будем пользоваться обозначениями элементов, заключенными в скобках,— z_1, z_2, z_3). Множества состояний, в которых возбуждены переменные схемы, задаются характеристическими функциями:

$$z_1 \oplus f_1 = \bar{z}_1 z_2 z_3 \vee z_1 \bar{z}_2 \bar{z}_3,$$
$$z_2 \oplus f_2 = \bar{z}_1 \bar{z}_2 \vee z_1 z_2,$$
$$z_3 \oplus f_3 = \bar{z}_1 \bar{z}_3 \vee z_1 z_3.$$

Выберем в качестве исходного множества состояний $A = \{101, 110\}$, тогда $\varphi_A = z_1 \bar{z}_2 z_3 \vee z_1 z_2 \bar{z}_3$. Вычислим значения операторов прямой и обратной достижимости по соседям:

$$P^1(\varphi_A(Z)) = z_1 \bar{z}_2 z_3 \vee z_1 z_2 \bar{z}_3 \vee (\bar{z}_1 z_2 z_3 \vee z_1 \bar{z}_2 \bar{z}_3)(\bar{z}_1 \bar{z}_2 z_3 \vee \bar{z}_1 z_2 z_3) \vee$$
$$\vee (\bar{z}_1 \bar{z}_2 \vee z_1 z_2)(z_1 z_2 z_3 \vee z_1 \bar{z}_2 \bar{z}_3) \vee (\bar{z}_1 \bar{z}_3 \vee z_1 z_3)(z_1 \bar{z}_2 \bar{z}_3 \vee z_1 z_2 z_3) =$$
$$= z_1 \bar{z}_2 z_3 \vee z_1 z_2 \bar{z}_3 \vee z_1 z_2 z_3 = z_1 z_2 \vee z_1 \bar{z}_3.$$

$$Q^1(\varphi_A(Z)) = z_1 \bar{z}_2 z_3 \vee z_1 z_2 \bar{z}_3 \vee (z_1 z_2 z_3 \vee \bar{z}_1 z_2 \bar{z}_3) \times$$
$$\times (\bar{z}_1 \bar{z}_2 z_3 \vee \bar{z}_1 z_2 \bar{z}_3) \vee (\bar{z}_1 z_2 \vee z_1 \bar{z}_2)(z_1 z_2 z_3 \vee z_1 \bar{z}_2 \bar{z}_3) \vee$$
$$\vee (\bar{z}_1 z_3 \vee z_1 \bar{z}_3)(z_1 \bar{z}_2 \bar{z}_3 \vee z_1 z_2 z_3) = z_1 \bar{z}_2 \vee z_1 \bar{z}_3.$$

Результаты вычислений соответствуют диаграмме переходов рис. 2.12, *а*.

Общую задачу достижимости в силу простоты методов вычисления операторов P^1 и Q^1 целесообразно свести к ее частному случаю — достижимости по соседям. Вопросы оценки вычислительной сложности алгоритмов и задач достижимости в общем и частном случаях выходят за рамки книги. Отметим, однако, что для большин-

ства реальных схем алгоритмы решения задач достижимости по соседям оказываются эффективнее алгоритмов решения общей задачи достижимости.

В первую очередь рассмотрим условия, когда решения общей и частной задач совпадают.

Переформулируем определение 2.21.

Определение 8.1. Состояние α *конфликтно* по переменной z_i, *если найдется состояние* β, непосредственно достижимое из α ($\alpha \to \beta$), такое, что $\alpha_i = \beta_i = f_i(\beta) \neq f_i(\alpha)$ (т. е. переменная z_i возбуждена в состоянии α и устойчива в состоянии β). Если β таково, что $\alpha \xrightarrow{1} \beta$, то α — 1-конфликтно.

Теорема 8.1. *Если все последователи состояния* α *не конфликтны, то множество состояний, достижимых из* α, *совпадает со множеством состояний, достижимых из* α *по соседям.*

Доказательство. Пусть для некоторого состояния β, $\alpha \Rightarrow \beta$, но неверно, что β достижимо из α по соседям $\left(\alpha \overset{1}{\Rightarrow} \beta\right)$. Найдется кратчайшая последовательность $\alpha^0\alpha^1 \ldots \alpha^h\alpha^{h+1}$, где $\alpha^0 = \alpha$, $\alpha^{h+1} = \beta$. Заменим всюду, где это возможно, пары $\alpha^i \to \alpha^{i+1}$ на последовательности $\alpha^i \xrightarrow{1} \beta^i \xrightarrow{1} \ldots \xrightarrow{1} \beta^r \xrightarrow{1} \alpha^{i+1}$, в результате получим максимально соседнюю последовательность $\beta^0\beta^1 \ldots \beta^m\beta^{m+1}$, где $\beta^0 = \alpha$, $\beta^{m+1} = \beta$. Найдется β^i, для которого неверно, что $\beta^i \overset{1}{\Rightarrow} \beta^{i+1}$. Пусть $\gamma \in (\beta^i, \beta^{i+1})$ и является соседним с β^{i+1}. Тогда $\beta^i \to \gamma$, но неверно, что $\gamma \to \beta^{i+1}$, иначе в силу $\gamma \xrightarrow{1} \beta^{i+1}$ приведенная последовательность не является максимально соседней. Следовательно, β^i конфликтно, что противоречит условию теоремы.

Из этой теоремы следует, что если все предшественники состояния α не конфликтны, то множество предшественников α совпадает со множеством предшественников α по соседям.

Пример 8.2. В диаграмме переходов рис. 8.1 состояние 000 является конфликтным, поэтому множество состояний, достижимых из состояния 111, не совпадает со множеством состояний, достижимых из 111 по соседям (в последнее множество не входит состояние 110).

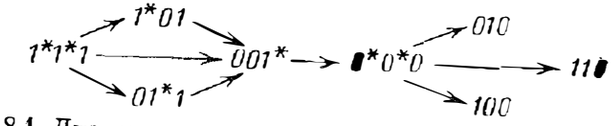


Рис. 8.1. Диаграмма переходов с конфликтным состоянием

Задача прямой достижимости может быть сведена к задаче прямой достижимости по соседям с помощью следующего преобразования схем.

Определение 8.2. *Сдвигом схемы* C на множестве состояний A будем называть схему C^A, заданную системой

$$z_i = f_i^A(Z) = z_i\left(f_i(z_i = 0) \vee Q^1(\varphi_A(Z) f_i(z_i = 0))\right) \vee z_i\left(\bar{f}_i(z_i = 1) \vee Q^1(\varphi_A(Z))\bar{f}_i(z_i = 1)\right).$$

Смысл такого преобразования заключается в том, что все 1-конфликтные в исходной схеме состояния из множества A в новой схеме становятся не 1-конфликтными состояниями. Это происходит за счет того, что некоторые возбуждения переменных сдвигаются по диаграмме переходов и при этом «открывается доступ» по соседям к состояниям, не достижимым по соседям в исходной схеме.

Основные свойства схемы C^A, полученной в результате сдвига, формулируются следующим образом:

1) в диаграмме переходов схемы C^A сохраняются возбуждения переменных, имевшие место в диаграмме исходной схемы;

2) все состояния из множества A, по которому осуществлялся сдвиг схемы C, в схеме C^A не являются 1-конфликтными;

3) в диаграмме схемы C^A могут появиться новые возбужденные переменные по сравнению с диаграммой схемы C, причем изменяются собственные функции только тех переменных, по которым в исходной схеме нарушалась полумодулярность.

Точнее, имеет место

Теорема 8.2. *Для сдвига* C^A *схемы* C *выполняется:*

1) *если* $\alpha_i \neq f_i(\alpha)$, *то* $\alpha_i \neq f_i^A(\alpha)$;

2) *все состояния из множества* A *в схеме* C^A *не являются 1-конфликтными;*

3) *если* $f_i(\beta) \neq f_i^A(\beta)$, *то найдется состояние* $\alpha \in A$ *такое, что в исходной схеме* C *имеет место* $\alpha \xrightarrow{1} \beta$ *и* α 1-*конфликтно по переменной* z_i.

Доказательство. Утверждение 1 проверяется непосредственной подстановкой.

Пусть $\alpha \in A$ 1-конфликтно по переменной z_i, т. е. найдется такое, что $\alpha \xrightarrow{1} \beta$, $\alpha_i = \beta_i = f_i(\beta) \neq f_i(\alpha)$. Предположим, что $\alpha_i = 0$. Тогда состояние α принадлежит множеству, задаваемому булевой функцией $\varphi_A(Z) f_i(z_i = 0)$, а β — задаваемому

$Q^1(\varphi_A(Z) f_i(z_i = 0))$. В силу этого $\beta_i \neq f_i^A(\beta) = 1$. Аналогично для случая $\alpha_i = 1$, что доказывает справедливость утверждения 2.

Утверждение 3 справедливо, так как функции $f_i(Z)$ и $f_i^A(Z)$ не эквивалентны только на таких состояниях α, для которых найдется 1-конфликтное состояние β такое, что $\beta \xrightarrow{1} \alpha$.

Из теоремы 8.2 непосредственно следует, что если в схеме C B — множество всех 1-конфликтных состояний из A ($B \subseteq A$), то $C^A = C^B$.

Возможность сведения задачи прямой достижимости к задаче прямой достижимости по соседям открывается благодаря следующей теореме.

Теорема 8.3. *Пусть B — множество состояний схемы C, достижимых из состояний множества A. Тогда множество состояний, достижимых из состояний множества A по соседям в сдвиге C^B схемы C, совпадает со множеством B.*

Доказательство. Пусть $\alpha^1 \in A$, $\alpha^N \in B$. Покажем, что в схеме C^B $\alpha^1 \overset{1}{\Rightarrow} \alpha^N$. Если в схеме C $\alpha^1\alpha^2 \ldots \alpha^N$ — максимально соседняя последовательность и для некоторого α^j неверно, что в схеме C $\alpha^j \overset{1}{\Rightarrow} \alpha^{j+1}$, то в силу теоремы 8.1 и теоремы 8.4, приведенной в § 8.2, состояние α^j 1-конфликтно. Согласно теореме 8.2 в схеме C^B состояние α^j не является 1-конфликтным, и значит, в схеме C^B $\alpha^j \overset{1}{\Rightarrow} \alpha^{j+1}$.

Пусть $\alpha^1 \in A$ и в схеме C^B $\alpha^1 \overset{1}{\Rightarrow} \alpha^N$. Покажем, что $\alpha^N \in B$. Если $\alpha^1\alpha^2 \ldots \alpha^N$ — последовательность состояний такая, что в схеме C^B $\alpha^k \xrightarrow{1} \alpha^{k+1}$, $1 \leqslant k \leqslant N-1$ и состояние α^j, $1 \leqslant j \leqslant N$, последнее из таких, что в схеме C не имеет места $\alpha^j \xrightarrow{1} \alpha^{j+1}$, то в силу теоремы 8.2 в схеме C найдется 1-конфликтное по переменной z_e состояние γ, $\gamma \in B$, для которого в схеме C $\gamma \xrightarrow[z_m]{1} \alpha^j$. Поскольку состояние α^{j+1} отличается от состояний γ значениями переменных z_e и z_m, причем обе эти переменные в схеме C в состоянии γ возбуждены, то $\gamma \to \alpha^{j+1}$. Следовательно, $\alpha^{j+1} \in B$, и так как в схеме C $\alpha^{j+1} \overset{1}{\Rightarrow} \alpha^N$, то и $\alpha^N \in B$.

Непосредственное использование этой теоремы, однако, затруднено: множество состояний, на котором необходимо произвести сдвиг исходной схемы, априори неизвестно. Поэтому решение задачи достижимости будем осуществлять поэтапно по следующей схеме.

На первом этапе определяется множество состояний D, достижимых из состояний заданного множества A по соседям в исходной схеме C. Далее строится сдвиг C^D схемы C и определяется множество состояний E, достижимых из состояний заданного множества A по соседям в схеме C^D. Если $E = D$ (в D не содержатся конфликтные состояния), то D является искомым множеством состояний, в противном случае процедура повторяется, причем в качестве исходной схемы берется схема C^D. В силу теоремы 8.2 процедура должна сходиться.

Пример 8.3. Рассмотрим схему 8.2, *а*, заданную на множестве переменных $\{z_1, z_2, z_3\}$ системой уравнений

$$\begin{cases} z_1 = (z_2 \vee z_3)\, z_1 \vee z_2 z_3, \\ z_2 = (\bar{z}_1 \vee \bar{z}_3)\, z_2 \vee \bar{z}_1 \bar{z}_3, \\ z_3 = (\bar{z}_1 \vee \bar{z}_2)\, z_3 \vee \bar{z}_1 z_2. \end{cases}$$

Пусть начальное множество состояний задано функцией $\bar{z}_1\bar{z}_2\bar{z}_3$. В процессе итеративного решения прямой задачи достижимости

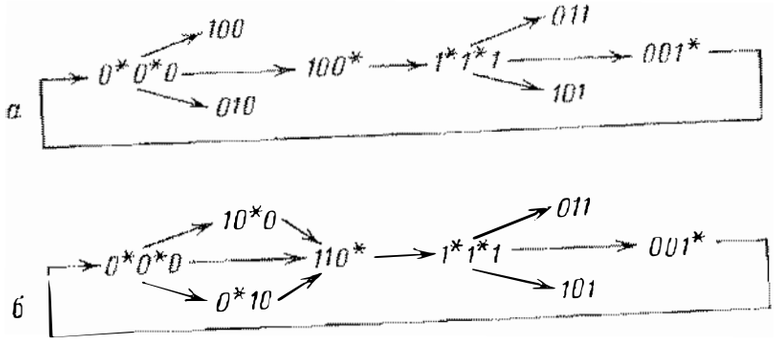


Рис. 8.2. Исходная диаграмма переходов (*а*) и ее сдвиг (*б*)

можно получить: $Q(\bar{z}_1\bar{z}_2\bar{z}_3) = \bar{z}_1$, $Q(\bar{z}_1) = \bar{z}_1 \vee z_2 z_3$, $Q(\bar{z}_1 \vee z_2 z_3) = 1$. Процесс решения задачи прямой достижимости по соседям обрывается на втором шаге: $Q^1(\bar{z}_1\bar{z}_2\bar{z}_3) = \bar{z}_1\bar{z}_3 \vee \bar{z}_1\bar{z}_2$, $Q^1(\bar{z}_1\bar{z}_3 \vee \vee \bar{z}_1\bar{z}_2) = \bar{z}_1\bar{z}_3 \vee \bar{z}_2\bar{z}_3$. Несовпадение последнего результата с результатом решения общей задачи прямой достижимости объясняется наличием в определяемом множестве состояний 1-конфликтного состояния 000. Сдвиг исходной схемы на множестве состояний, заданном булевой функцией $\bar{z}_1\bar{z}_3 \vee \bar{z}_1\bar{z}_2$ (рис. 8.2, *б*), описывается системой

$$\begin{cases} z_1 = (z_2 \vee z_3)\, z_1 \vee z_2 z_3, \\ z_2 = \bar{z}_1 \vee z_2 \bar{z}_3, \\ z_3 = \bar{z}_1 \vee \bar{z}_2 z_3. \end{cases}$$

Продолжая процесс решения задачи прямой достижимости по соседям для этой схемы получим: $Q^1(\bar{z}_1\bar{z}_3 \vee \bar{z}_1\bar{z}_2) = \bar{z}_1$, $Q^1(\bar{z}_1) = \bar{z}_1 \vee z_2z_3$, $Q^1(\bar{z}_1 \vee z_2z_3) = \bar{z}_1 \vee z_2 \vee z_3$, $Q^1(\bar{z}_1 \vee z_2 \vee z_3) = \bar{z}_1 \vee z_2 \vee z_3$. Последний результат также не совпадает с решением общей задачи прямой достижимости для исходной схемы, так как в полученном множестве содержится 1-конфликтное состояние 111. Сдвиг схемы на множестве состояний, задаваемом функцией $\bar{z}_1 \vee z_2 \vee z_3$ (рис. 2.11, *а*) описывается системой уравнений из примера 2.8. Для этой схемы $Q^1(\bar{z}_1 \vee z_2 \vee z_3) = 1$, что совпадает с решением общей задачи прямой достижимости для исходной схемы.

§ 8.2. Классификационный анализ

Важной проблематикой анализа схем является определение их принадлежности подклассам схем, не зависящих от скорости,— полумодулярным, дистрибутивным, параллельно-последовательным и последовательным схем [1]. Разработке методов такого *(классификационного) анализа* и посвящен материал этого параграфа.

Вначале уточним ряд понятий, введенных в предыдущих параграфах.

Определение 8.3. Множество всех таких состояний схем, что для любой пары состояний α, β, принадлежащих этому множеству, имеет место $\alpha \Rightarrow \beta$ и $\beta \Rightarrow \alpha$, будем называть *классом эквивалентности.*

Класс эквивалентности назовем:

— *фиктивным*, если во всех принадлежащих ему состояниях схемы некоторая переменная z_i имеет одно и то же значение и возбуждена;

— *замкнутым*, если непосредственный последователь класса совпадает с самим классом (т. е. если состояние α принадлежит классу и $\alpha \Rightarrow \beta$, то и состояние β тоже принадлежит этому классу).

Тривиальным примером замкнутого класса является тупиковое состояние схемы, т. е. такое ее состояние, в котором все переменные устойчивы.

Определение 8.4. *Схема не зависит от скорости* относительно состояния α тогда и только тогда, когда множество состояний, достижимых из α, содержит один не фиктивный класс эквивалентности.

Из этого определения следует, что множество рабочих состояний схемы, не зависящей от скорости (т. е. состояний, достижимых из некоторого инициального состояния), содержит ровно один замкнутый класс, в частности это множество может содержать не более одного тупикового состояния.

Определение 8.5. *Схема полумодулярна* относительно состояния α, если каждое состояние β, достижимое из состояния α в этой схеме не конфликтно по всем переменным.

Анализ схемы на полумодулярность подразумевает поиск конфликтных состояний. Следующая теорема позволяет упростить этот поиск.

Теорема 8.4. *Для всякого конфликтного по переменной z_i состояния α найдется 1-конфликтное по переменной z_i состояние β такое, что $\alpha \to \beta$ и $\alpha \overset{1}{\Rightarrow} \beta$.*

Доказательство. Пусть α конфликтно по переменной z_i и $\{\alpha^1, \ldots, \alpha^e\}$ — множество всех состояний таких, что $\alpha \to \alpha^j$ и $\alpha_i = \alpha_i^j = f_i(\alpha^j) \neq f_i(\alpha)$. Пусть при этом α^j отличается от α значениями k_j переменных; $k = \min\limits_{1 \leqslant j \leqslant e} k_j$ — ранг конфликтности состояния α по переменной z_i. Воспользуемся математической индукцией по рангу конфликтности. При $k = 1$ справедливость теоремы очевидна. Пусть теорема справедлива при $k = m$. Рассмотрим случай $k = m + 1$, т. е. для некоторого j состояние α отличается от α^j значениями $m + 1$ переменной. Пусть состояние γ — соседнее с α^j и $\gamma \in (\alpha, \alpha^j)$. Тогда $\alpha \to \gamma$ и $\gamma \overset{1}{\to} \alpha^j$, иначе ранг конфликтности состояния α был бы меньше $m + 1$. Следовательно, состояние γ 1-конфликтно по переменной z_i и удовлетворяет первому условию теоремы. Справедливо и второе условие: $\alpha \overset{1}{\Rightarrow} \gamma$, ибо в противном случае найдется такое состояние $\sigma \in (\alpha, \gamma)$, что $\alpha \to \sigma$. $\alpha_i = \sigma_i = f_i(\sigma) \neq f_i(\alpha)$, а тогда ранг конфликтности состояния α по переменной z_i меньше $m + 1$.

Таким образом, схема полумодулярна относительно состояния α тогда и только тогда, когда в ней из состояния α недостижимо ни одно 1-конфликтное состояние.

Кроме того, из теорем 8.1, 8.4 следует, что если схема полумодулярна относительно состояния α, то множество состояний, достижимых в этой схеме из α, совпадает со множеством состояний, достижимых в ней из α по соседям.

Поясним, почему поведение полумодулярных схем инвариантно к соотношению задержек элементов.

Вообще говоря, некоторая возбужденная переменная может стать устойчивой двумя способами: 1) изменив свое значение, 2) в результате изменения значений других переменных, от которых она зависит. В полумодулярной схеме возможен только первый способ снятия возбуждения переменной. Отсюда следует, что в полу-

модулярной схеме переключение одной или нескольких возбужденных переменных не может «помешать» переключению других возбужденных переменных.

Для отыскания множества состояний, относительно которых схема не полумодулярна, сначала определяется *множество конфликтных* (1-конфликтных) *состояний* по всем переменным, а затем решается задача обратной достижимости для этого множества. В силу теоремы 8.1 достаточно при этом решать задачу обратной достижимости по соседям.

Утверждение 8.2. *Множества конфликтных и 1-конфликтных состояний по переменной z_i задаются соответственно булевыми функциями*

$$z_i\bar{f}_i(Z)P(z_if_i(Z))\vee\bar{z}_if_i(Z)P(\bar{z}_i\bar{f}_i(Z)),$$
$$z_i\bar{f}_i(Z)P^1(z_if_i(Z))\vee\bar{z}_if_i(Z)P^1(\bar{z}_i\bar{f}_i(Z)).$$

Справедливость этих формул вытекает из того, что они задают пересечения множеств состояний, в которых переменная z_i возбуждена — $z_i\bar{f}_i(Z)$ и $\bar{z}_if_i(Z)$ — со множествами состояний, из которых без изменения значения z_i непосредственно достижимы (по соседям) состояния, где переменная z_i устойчива — $P(z_if_i(Z))$ и $P(\bar{z}_i\bar{f}_i(Z))$ ($P^1(z_if_i(Z))$ и $P^1(\bar{z}_i\bar{f}_i(Z))$).

Свойство независимости от скорости является «глобальным» в том смысле, что нельзя указать состояния, в которых это свойство нарушается (по-видимому, этим и объясняются трудности в разработке регулярных методов синтеза схем, обладающих в точности таким свойством, а не его сужениями — подклассами). Свойство полумодулярности, напротив, можно назвать «локальным». Именно локализация нарушения полумодулярности в конфликтных состояниях делает эффективным анализ схем на полумодулярность. Аналогично, «локальными» свойствами обладают дистрибутивные и последовательные схемы, а также ряд других подклассов схем, не зависящих от скорости.

Для анализа схем на принадлежность подклассам полумодулярных введем ряд понятий.

Определение 8.6. Состояние α назовем:

— *детонантным* по переменной z_i, если найдется пара состояний β и γ, непосредственно достижимых из $\alpha\left(\alpha\xrightarrow{1}\beta,\alpha\xrightarrow{1}\gamma\right)$ таких, что переменная z_i устойчива в состоянии α и возбуждена в состояниях β и γ ($\alpha_i=\beta_i=\gamma_i=f_i(\alpha)$ и $f_i(\beta)=f_i(\gamma)\neq\alpha_i$);

— *бифуркантным*, если в нем возбуждено более одной переменной;

— *гамачным*, если в каждом его непосредственном последователе β, за исключением разве лишь такого, который отличается от α значениями всех возбужденных в α переменных, возбуждены только те переменные, которые возбуждены в α и не изменили своего значения ($\alpha_i=\beta_i\neq f_i(\alpha)$).

Пример 8.4. На диаграмме переходов рис. 8.3, *а* состояние 000 детонантно, не гамачно и не конфликтно, а все остальные состояния не детонантны и гамачны. Бифуркантными являются также

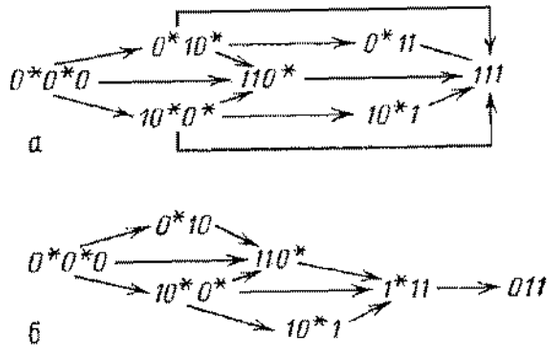


Рис. 8.3. К определению понятия бифуркантного, детонантного и гамачного состояний: примеры диаграмм переходов

состояния 100 и 010. Не бифуркантны — 101, 110, 011, 111. рис. 8.3, *б* состояние 000 не гамачно и не детонантно.

Утверждение 8.3. *Множества детонантных, бифуркантных и не гамачных состояний задаются соответственно формулами*

$$(\bar{z}_i\oplus f_i)\bigvee_{j=1}^{n}P^1((z_i\oplus f_i)z_j)P^1((z_i\oplus f_i)\bar{z}_j),\qquad(8.5)$$

$$\bigvee_{1\leqslant i,\,j\leqslant n,\,i\neq j}(z_i\oplus f_i)(z_j\oplus f_j),\qquad(8.6)$$

$$\bigvee_{1\leqslant i,j\leqslant n}(\bar{z}_i\oplus f_i)(z_j\oplus f_j)P((z_i\oplus f_i)(z_j\oplus f_j)).\qquad(8.7)$$

Справедливость формул (8.5) — (8.7) вытекает из следующего: формула (8.5) задает пересечение множеств состояний, в которых переменная z_i устойчива $(\bar{z}_i\oplus f_i)$, со

множествами состояний, из которых непосредственно достижимы по соседям два различных состояния, где переменная z_i возбуждена $P^1((z_i \oplus f_i)z_j)$ и $P^1((z_i \oplus f_i)\bar{z}_j)$, формула (8.6) — те состояния, где возбуждены хотя бы две переменные, формула (8.7) — пересечение множества состояний, в которых z_i устойчива, а z_j возбуждена, со множеством состояний, непосредственно предшествующих состояниям, в которых z_j еще не переключилась, а z_i возбудилась.

Теперь можно определить интересующие нас подклассы полумодулярных схем.

Определение 8.7. 1) *Схема дистрибутивна* относительно состояния α, если каждое состояние β, достижимое из состояния α в этой схеме, не конфликтно и не детонантно по всем переменным [1]).

2) *Схема параллельно-последовательна* относительно состояния α, если каждое состояние β этой схемы, достижимое из состояния α, не конфликтно по всем переменным и гамачно.

3) *Схема последовательна* относительно состояния α, если каждое состояние β, достижимое из α, не бифуркантно.

Для дистрибутивных схем характерна однозначность локальной предыстории в том смысле, что каждое возбуждение любой переменной может быть вызвано единственным способом — изменением значений всех переменных только из одного минимального набора переменных.

В параллельно-последовательных схемах каждая переменная возбуждается только после того, как в результате изменения своих значений станут устойчивыми все ранее возбужденные переменные. Класс параллельно-последовательных схем является подклассом дистрибутивных и образует расширение класса последовательных схем.

Отыскание состояний, относительно которых схема не дистрибутивна (не параллельно-последовательна, не последовательна), проводится аналогично поиску состояний, относительно которых схема не полумодулярна.

[1]) Можно показать, что схема дистрибутивна относительно состояния α тогда и только тогда, когда ее кумулятивная диаграмма (построенная для состояния α) образует дистрибутивную структуру.

Пример 8.5. Схема рис. 8.4, заданная системой уравнений

$$\begin{cases} z_1 = \bar{z}_3, \\ z_2 = \bar{z}_1, \\ z_3 = \bar{z}_2, \end{cases}$$

последовательна относительно всех своих состояний, кроме 000 и 111. Относительно этих последних она не полумодулярна. Схема

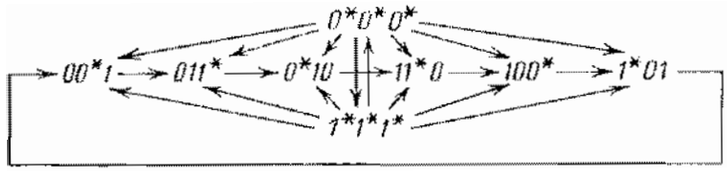


Рис. 8.4. Диаграмма переходов для трех соединенных в кольцо инверторов

рис. 2.11, *а* параллельно-последовательна относительно всех состояний, а схема рис. 8.3, *б* дистрибутивна, но не параллельно-последовательна относительно состояния 000. Схема рис. 8.3, *а*, заданная системой уравнений

$$\begin{cases} z_1 = 1, \\ z_2 = 1, \\ z_3 = z_1 \vee z_2, \end{cases}$$

не дистрибутивна, но полумодулярна относительно состояния 000.

Полным анализом будем называть процедуру разбиения множества состояний схемы на две области — *запрещенную и допустимую*. В зависимости от того, какое классификационное свойство исследуется, запрещенной области могут принадлежать состояния, относительно которых схема не полумодулярна, не дистрибутивна и т. п. Таким образом, полный анализ позволяет исследовать поведение схемы на множестве всех ее состояний.

Пример 8.6. Рассмотрим процедуру полного анализа схемы, заданной системой уравнений

$$z_1 = \bar{z}_5, \quad z_2 = \bar{z}_5, \quad z_3 = z_1 z_2 \vee z_3(z_1 \vee z_2), \qquad (8.8)$$
$$z_4 = z_2, \quad z_5 = z_3 z_4 \vee z_5(z_3 \vee z_4).$$

Ее полная диаграмма переходов содержит 32 состояния и не отличается особой наглядностью, поэтому она не приводится. В результате анализа получены следующие результаты. Множество 1 конфликтных состояний схемы описывается характеристической

функцией

$$z_2\bar{z}_4z_5 \vee \bar{z}_2z_4z_5 \vee z_1\bar{z}_3z_5(z_2 \vee \bar{z}_4) \vee \bar{z}_1z_3\bar{z}_5(\bar{z}_2 \vee z_4)$$

и содержит 12 состояний.

Множество состояний, относительно которых схема не является полумодулярной (запрещенная область) описывается функцией $\bar{z}_5(\bar{z}_1z_3 \vee \bar{z}_2z_4) \vee z_5(z_2\bar{z}_4 \vee z_1\bar{z}_3)$ и содержит 14 состояний.

Относительно остальных состояний схема полумодулярна. Допустимая область описывается функцией $z_3z_5(\bar{z}_2 \vee z_4) \vee \bar{z}_3\bar{z}_5(z_2 \vee z_4) \vee z_1\bar{z}_5(\bar{z}_4 \vee z_2z_3) \vee \bar{z}_1(z_4z_5 \vee \bar{z}_2\bar{z}_3\bar{z}_4)$ и содержит 18 состояний.

Хотя результатом полного анализа является исчерпывающая информация о поведении схемы, он не всегда необходим и удобен.

§ 8.3. Множество рабочих состояний

Число состояний, в которых может оказаться схема в процессе функционирования, конечно — 2^n (где n — число элементов схемы), поэтому существуют лишь две возможности: 1) либо схема переходит из состояния в состояние, пока не оказывается в тупике, т. е. в состоянии, не имеющем последователей, 2) либо, начиная с какого-то момента, некоторая последовательность состояний или часть последовательности бесконечно повторяется.

Если в схеме выделено некоторое инициальное состояние, то все состояния, достижимые из него, будем называть *рабочими*. Зачастую мощность множества рабочих состояний существенно меньше 2^n. Учет этого обстоятельства позволяет упростить анализ. Такое упрощение базируется на выявлении характеристических свойств множества рабочих состояний.

Построение и анализ множества рабочих состояний схемы с инициальным состоянием (множеством инициальных состояний) могут быть основаны на итеративных методах решения задачи прямой достижимости. При этом целесообразно чередовать шаги решения задачи достижимости с проверкой принадлежности схемы тому или иному классу схем, не зависящих от скорости.

Теперь остановимся на вопросах организации процесса построения множества рабочих состояний схемы, ориентируясь при этом на повышение эффективности процедуры анализа. Обозначим множество рабочих состояний через W. Для нахождения этого множества можно использовать формулы (8.3)—(8.4), задающие оператор

прямой достижимости по соседям. В качестве исходного множества $A \subseteq W$ состояний выбирается инициальное состояние (или множество инициальных состояний) схемы. На первом шаге строится множество непосредственных последователей по соседям для A. Это множество покрывается вторыми членами формул (8.3)—(8.4), имеющими вид

$$R^1(\varphi_A(Z)) = \bigvee_{i=1}^{n} \varphi_A(z_i = \bar{z}_i)(\bar{z}_i \oplus f_i(z_i = \bar{z}_i)), \qquad (8.9)$$

или

$$R^1(\varphi_A(Z)) = \bigvee_{i=1}^{n} \varphi_A(z_i = \bar{f}_i(z_i = \bar{z}_i)), \qquad (8.10)$$

которые, вообще говоря, могут включать некоторое подмножество состояний из A. На втором шаге по формулам (8.9), (8.10) находятся все непосредственные последователи по соседям для состояний, найденных на первом шаге. Очевидно, что, продолжая вычисления аналогичным образом, можно построить все множество W. Для завершения этого процесса необходимо убедиться, что на очередном шаге не получены новые состояния, не фигурировавшие на предыдущих шагах. На первый взгляд для этого кажется необходимым запоминать текущее значение множества W, включающее все полученные на предыдущих шагах состояния. Ниже будет предложена альтернативная организация нахождения множества W, не требующая манипулирования с текущими его значениями. Для этой цели введем новое понятие.

Определение 8.8. Будем называть k-м *слоем* (L_k) множества W рабочих состояний схемы с инициальным состоянием α множество состояний, достижимых из α за k шагов по соседям.

Нулевой слой $L_0 = \{\alpha\}$ образуется инициальным состоянием.

Из сказанного следует, что множество слоев упорядочено. В силу конечности числа состояний схемы, начиная с некоторого шага построения множества рабочих состояний, слои начнут повторяться.

Слой из W, полностью совпадающий с одним из предыдущих, назовем *кратным*.

Ясно, что если L_m — кратный слой, т. е. $L_m = L_k$, $m > k$, то это является признаком завершения процесса

построения множества рабочих состояний:

$$W = \bigcup_{i=0}^{m-1} L_i.$$

Вообще говоря, не каждый слой, полученный в процессе анализа, является кратным. Часть рабочих состояний может встретиться при функционировании схемы только однажды. Для слоев, не содержащих повторяющихся состояний, отсутствуют покрывающие их слои.

Если бы существовали какие-либо конструктивные соображения, позволяющие предсказать, что некоторый k-й слой в дальнейшем повторится (будем называть такой слой *контрольным*), можно было бы ориентироваться только на проверку кратности этого слоя, т. е. каждый новый слой проверять на совпадение только с контрольным. В этом случае было бы достаточно хранить только слой L_k, отказавшись от хранения текущего значения W. Одним из таких соображений является учет *ширины слоя* (числа состояний в нем). В качестве контрольного может быть выбран наиболее широкий слой.

Имеет место следующий факт: *если множество рабочих состояний представляет один класс эквивалентности, то самый широкий слой является кратным.* Если же диаграмма переходов, соответствующая W, содержит начальный фрагмент (множество состояний, которые не появляются вновь при работе схемы, но из которых достижимы состояния, принадлежащие замкнутому классу), то, вообще говоря, самый широкий слой может принадлежать начальному фрагменту и, следовательно, не быть кратным. В этом случае такой критерий окончания не гарантирует сходимости процесса анализа. Однако в большинстве реальных схем начальные фрагменты диаграмм не слишком широки, поскольку обладают меньшим параллелизмом, чем циклические участки диаграмм переходов.

Пример 8.7. В схеме рис. 2.11, *а* с инициальным состоянием 000 слои множества рабочих состояний задаются функциями:

$$L_0 = \bar{z}_1\bar{z}_2\bar{z}_3, \quad L_1 = \bar{z}_1(z_2\bar{z}_3 \vee \bar{z}_2 z_3), \quad L_2 = \bar{z}_1 z_2 z_3,$$
$$L_3 = z_1 z_2 z_3, \quad L_4 = z_1(z_2\bar{z}_3 \vee \bar{z}_2 z_3), \quad L_5 = z_1\bar{z}_2\bar{z}_3,$$
$$L_6 = \bar{z}_1\bar{z}_2\bar{z}_3, \quad L_7 = \bar{z}_1(z_2\bar{z}_3 \vee \bar{z}_2 z_3).$$

В качестве контрольного может быть выбран слой L_1 шириной 2 (шире нет). Он является кратным, так как $L_1 = L_7$. Как можно убедиться, ни один из слоев не содержит конфликтных состояний, и $\varphi_W(z) = \bigvee_{i=0}^{6} L_i = 1$. Таким образом, схема полумодулярна относительно всех своих состояний. Алгоритм построения W с проверкой окончания по самому широкому слою в данном случае потребовал 7 шагов.

Минимальными по ширине являются слои, состоящие из одного состояния. Такие слои было бы заманчиво использовать в качестве контрольных. Остановимся на условиях, когда это возможно.

Определение 8.9. 1) Последовательность состояний $\alpha^1\alpha^2 \ldots \alpha^k \ldots$ такая, что $\alpha^i \to \alpha^{i+1}$ является *полной*, если она конечна и завершается тупиковым состоянием или бесконечна и не содержит бесконечного участка, во всех состояниях которого какая-либо переменная имеет одно и то же значение и возбуждена.

2) Пусть W — множество состояний, достижимых в схеме из состояний множества A ($A \subset W$). Состояние β будем называть *узловым* для множества A в этой схеме, если все полные последовательности, начинающиеся в состояниях W, содержат β.

Прокомментируем понятие «полной последовательности». Очевидно, что последовательности на диаграмме переходов схемы могут быть как конечными (если они не образуют цикл), так и бесконечными (если они образуют цикл). Из конечных последовательностей к полным относятся лишь те, которые не могут быть продлены, т. е. последовательности, заканчивающиеся тупиковым состоянием. Среди бесконечных последовательностей (содержащих *циклы*) к полным относятся те, по которым схема может «крутиться» сколь угодно долго. А в силу гипотезы о конечности задержек элементов этим свойством не обладают те циклы, в которых какая-либо переменная не меняет своего значения и возбуждена во всех состояниях цикла, ибо рано или поздно возбужденная переменная должна изменить свое значение.

Из сказанного ясно, что к полным относятся те последовательности, которые исчерпывающе характеризуют возможный порядок смен состояний схемы.

Понятие узлового состояния можно проиллюстрировать следующим образом. В ряде апериодических схем значение сигнала на выходе индикатора моментов окончания переходных процессов (если такой индикатор есть) обычно изменяется последним и выходной элемент индикатора оказывается возбужденным только после то-

го, как переключатся все остальные элементы схемы. Состояние, в котором возбужден выходной элемент индикатора, и будет узловым.

Укажем, опустив из-за громоздкости его доказательство, следующее свойство узловых состояний.

Теорема 8.5. *Пусть схема полумодулярна относительно состояний множества A. Если состояние α является узловым для множества A, то в любом состоянии β, достижимом из состояний множества A в этой схеме, таком, что β не принадлежит фиктивному классу, $\beta \neq \alpha$ и $\beta \xrightarrow{1} \alpha$, возбуждена ровно одна переменная. Обратно: если из состояний множества A достижим один не фиктивный класс эквивалентности B, и состояние $\alpha \in B$ таково, что во всех его непосредственных предшественниках по соседям $\beta \neq \alpha$, достижимых из состояний множества A, возбуждена ровно одна переменная, то α — узловое состояние для множества A.*

Установим соответствие между узловыми состояниями и слоями.

Теорема 8.6. *Пусть схема полумодулярна относительно состояния α и из α недостижим фиктивный класс. Если некоторый слой содержит узловое для множества $\{\alpha\}$ состояние, то он не содержит других состояний. Обратно: если кратный слой содержит одно состояние, то это состояние узловое для инициального множества.*

Доказательство. 1. Пусть слой L_j содержит хотя бы два состояния, одно из которых узловое. Тогда, учитывая, что каждый непосредственный предшественник узлового состояния по соседям, достижимый из инициального состояния, не бифуркантен (теорема 8.5), нетрудно показать по индукции, что каждый слой L_l, $0 \leqslant l \leqslant j-1$, задает более одного состояния. Это противоречит условию теоремы.

2. В обратном утверждении теоремы легко убедиться, если учесть, что «между» L_k и L_m содержится весь замкнутый класс состояний, достижимый из инициального состояния.

Обратим внимание на еще два интересных и достаточно очевидных свойства узлового состояния, характеризующих схемы, не зависящие от скорости.

1) Если состояние α является узловым для множества A, то α принадлежит некоторому замкнутому классу B, причем если множество A само образует нефиктивный класс, то $A = B$.

2) Если в схеме найдется состояние α, являющееся узловым для множества состояний A, то эта схема не зависит от скорости относительно всех состояний из A.

В соответствии с теоремой 8.6 процесс построения множества рабочих состояний схемы можно считать завершенным после того, как второй раз будет получено одно и то же узловое состояние.

К сожалению, не каждая схема, полумодулярная относительно инициального состояния, имеет узловое состояние (рис. 8.5). Не обладают узловыми состояниями, например, конвейерные схемы.

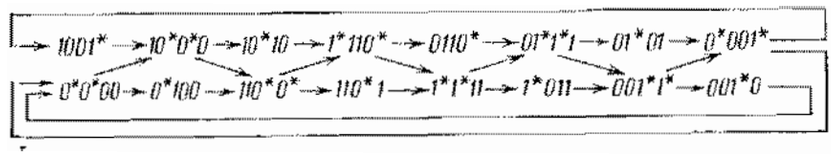


Рис. 8.5. Схема без узловых состояний

Однако некоторые подклассы полумодулярных схем все же обладают рабочими множествами с узловыми состояниями. К ним относятся, например, параллельно-последовательные схемы.

Пример 8.8 (продолжение примера 8.7). Схема рис. 2.11, *а* является параллельно-последовательной относительно всех своих состояний. Узловыми в ней являются состояния 000, 011, 111, 100, заданные функциями L_0, L_2, L_3, L_5 соответственно. Если проверять окончание построения множества рабочих состояний W по узловым состояниям, то потребуется не 7 шагов (как в примере 8.7), а 6.

Существуют схемы, которые не являются параллельно-последовательными и, тем не менее, имеют множество рабочих состояний с узловыми состояниями.

Например, схема, представленная на рис. 8.6 и соответствующая системе уравнений (8.8), не является параллельно-последовательной относительно, скажем, состояния 00000, однако если выбрать это состояние в качестве инициального, то множество W содержит четыре узловых состояния: 00000, 11110, 11111 и 00001.

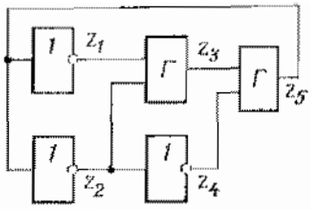


Рис. 8.6. Схема с узловыми состояниями, не являющаяся параллельно-последовательной

Пример 8.9. (Иллюстрирует «взаимодействие» анализа и синтеза при проектировании апериодических схем.) На рис. 5.2 была изображена схема регистра сдвига, представляющая собой соединение трех так называемых «ячеек Давида» (*RS*-триггера с вентилем). Эта схема, как и схема с большим числом ячеек, работоспо-

собна для начальных состояний, в которых возбужден лишь один элемент. Возникает вопрос о работоспособности таких схем для начальных состояний, в которых одновременно возбуждено два элемента, т. е. вопрос о том, можно ли использовать такие схемы в конвейерном режиме. Анализ множества рабочих состояний показал, что в этом случае регистр, содержащий четыре ячейки Давида, не работает: он быстро оказывается в тупиковом состоянии,

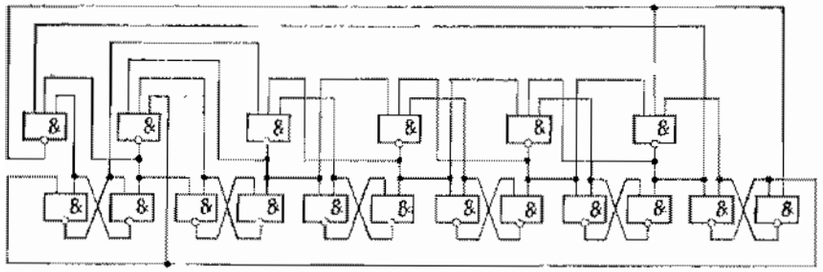


Рис. 8.7. Конвейерный аналог распределителя на элементах Давида

что для автономных схем недопустимо. Аналогичная схема с шестью ячейками Давида не попадает в тупик для инициальных состояний с парой возбужденных переменных, однако ее множество рабочих состояний содержит конфликтные состояния, а, следовательно, схема не полумодулярна. На рис. 8.7 представлен окончательный вариант конвейерного аналога регистра типа рис. 5.2, который потребовал введения дополнительных входов вентилей и соответствующих связей.

В рассмотренных методах анализа не исследуется каждая полная последовательность с начальным состоянием α в отдельности, а делаются последовательные шаги по всем путям диаграммы сразу. Такой метод соответствует «параллельному просмотру» ветвей диаграммы переходов, что ускоряет процесс анализа схемы.

§ 8.4. Влияние задержек в проводах

В предыдущих параграфах этой главы рассматривались вопросы анализа асинхронных логических схем в предположении, что задержки в соединительных проводах пренебрежимо малы по сравнению с задержками элементов. Эта гипотеза несправедлива для схем, в которых быстродействие элементов столь велико, что величины задержек элементов становятся сравнимы с величинами задержек проводов, например для схем, построенных на ЭСЛ-технологии или на логике Джозефсона. Кроме того, существует широкий класс интерфейсных устройств, в состав которых входят относительно длинные

линии связи, с задержками, значительно превышающими задержки элементов. Для подобных устройств характерно явление перекоса задержек. С повышением уровня интеграции задержки в проводах оказывают все более заметное влияние на поведение схем и в схемах сверхбольшой интеграции становятся сравнимыми с задержками логических элементов, а то и превосходят их.

Сказанное обосновывает необходимость анализа зависимости работы асинхронных логических схем от задержек в проводах.

В тех случаях, когда необходимо проанализировать схемы со связями, обладающими значительными задержками, в них могут быть введены специальные элементы — повторители вместо проводов с большой задержкой, а гипотезы, положенные в основу модели Маллера, не пересматриваются. Таким образом, можно использовать вышеизложенные методы для анализа работоспособности схем при наличии значительных задержек в некоторых проводах.

Пример 8.10. Исследуем схему, представленную на рис. 8.8. Анализ множества рабочих состояний показывает, что схема полумодулярна, скажем, относительно инициального состояния 0111100, если задержки в проводах несущественны. Учет задержек в проводах приводит к следующим результатам. Задержки, не влияющие на работоспособность схемы, помечены на рис. 8.8 звездочками.

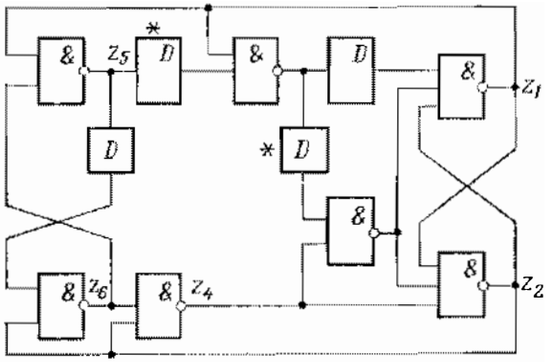


Рис. 8.8. Пример нарушения полумодулярности схемы из-за задержек в проводах

Использование связей со значительными задержками, которые на рис. 8.8 не помечены звездочками, приводит к нарушению полумодулярности схемы. Остальные связи в схеме аналогичны помеченным звездочками, поэтому анализировать схему на работоспособность при внесении задержек в эти связи не требуется.

Введение задержек во все (или многие) связи схемы приводит к резкому увеличению размерности задачи анализа и делает ее трудноразрешимой. Представляет интерес исследование закономерностей влияния задержек в проводах на поведение схемы и разработка на их основе методов анализа, не требующих введения дополнительных переменных в схему в явном виде. Такие методы, возможно, в ряде случаев позволят повысить эффективность анализа работоспособности схем с существенными задержками в проводах.

Формализуем понятие провода (связи) между элементами схемы

$$z_i = f_i(Z), \qquad 1 \leqslant i \leqslant n.$$

Определение 8.10. Пару натуральных чисел (i, j) таких, что $f_j(Z)$ существенно зависит от переменной z_i, будем называть *проводом схемы*, а z_i и z_j — входной и выходной переменными провода (i, j) соответственно.

Пусть $L = \{(i_1, j_1), \ldots, (i_m, j_m)\}$ — некоторое множество проводов схемы C. Обозначим через $I(L)$ и $O(L)$ множества входных и выходных переменных для проводов из L. Если $z_j \in O(L)$, то через $I(j, L)$ обозначим множество входных переменных для всех тех проводов из L, которые имеют z_j в качестве входной переменной.

Для изучения влияния задержек проводов из L на поведение схемы будем сравнивать поведение исходной схемы $C = \langle Z, F\rangle$ с поведением схемы C^L, полученной из исходной введением элементов задержки (повторителей) во все интересующие нас провода. Эту новую схему $C^L = \langle Z^L, F^L\rangle$ будем называть *расширением схемы C по множеству проводов L*.

Построение схемы C^L связано, во-первых, с введением множества дополнительных переменных, соответствующих проводам из L, которые будем обозначать $l_{i1}^{j1}, \ldots, l_{im}^{jm}$ и, во-вторых, с естественным преобразованием системы булевых уравнений. Для дополнительной переменной l_p^r вводится уравнение $l_p^r = z_p$, а исходные уравнения преобразуются следующим образом: если $z_r \notin O(L)$, то уравнение для z_r не изменяется, а если $z_r \in O(L)$, то в функцию $f_r(Z)$ вместо каждой переменной $z_p \in I(r, L)$ подставляется новая переменная l_p^r.

Пример 8.11. Пусть схема рис. 8.9 задана системой уравнений

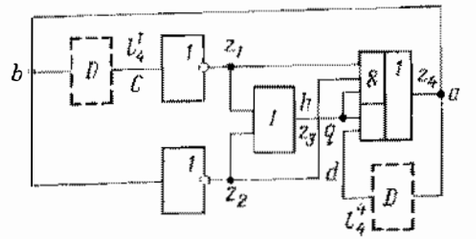
$$z_1 = \bar{z}_4, \quad z_2 = \bar{z}_4, \quad z_3 = z_1 \vee z_2, \quad z_4 = z_3(z_1 z_2 \vee z_4). \qquad (8.11)$$

Нас интересует множество проводов (4,1), (4,4). Введем две дополнительные переменные l_4^1, l_4^4 и преобразуем систему уравнений:

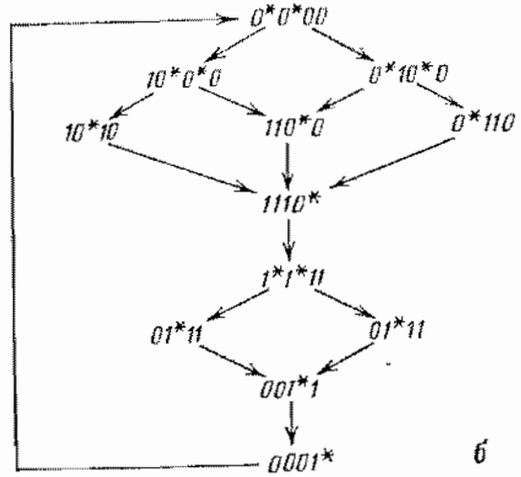
$$z_1 = \bar{l}_4^1, \quad z_2 = \bar{z}_4, \quad z_3 = z_1 \vee z_2, \quad z_4 = z_4 z_1 z_2 \vee z_3 l_4^4, \quad l_4^1 = z_4,$$

$$l_4^4 = z_4.$$

Выход элемента z_4 связан с остальными элементами схемы тремя проводами (4,1), (4,2), (4,4) и при топологии схемы, соответствующей рис. 8.9, имеется два узла разветвления a и b. Из введенной



а



б

Рис. 8.9. Метод включения элементов задержки для анализа влияния проводов

модели вытекает, что дополнительный элемент (повторитель) l_4^4 включается в разрыв участка ad провода (4,4), а элемент l_4^1 — bc провода (4,1). В разрыв участка ab, как следует из модели, задержку встраивать не нужно. Если бы нас интересовал провод (3,4), то элемент l_3^4 следовало бы вставить в разрыв участка hg, так как задержка участков провода после разветвления на разные входы одного и того же элемента (z_4) считается пренебрежимо малой.

Введем ряд вспомогательных понятий.

Определение 8.11. 1) *Проекция состояния* $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ схемы на подмножество $Z' = \{z_{i_1}, \ldots, z_{i_k}\} \subseteq Z$ ее переменных есть вектор значений всех переменных из Z' в состоянии α.

2) Пусть $\alpha^1\alpha^2 \ldots \alpha$ — последовательность состояний схемы, где $\alpha^i \to \alpha^{i+1}$, $\alpha^i \neq \alpha^{i+1}$. Проекция этой последовательности на множество переменных $Z' \subseteq Z$ получается из последовательности проекций ее состояний на это множество удалением каждого члена, совпадающего с предыдущим.

3) Две инициальные схемы $C_1 = \langle Z_1, F_1, \alpha \rangle$ и $C_2 = \langle Z_2, F_2, \beta \rangle$ будем называть *эквивалентными* по множеству переменных $Y \subseteq Z_1 \cap Z_2$, если множества различных проекций на Y последовательностей, начинающихся в инициальном состоянии для схемы C_1 и в инициальном состоянии для схемы C_2, совпадают. Обозначим эквивалентность схем C_1 и C_2 по Y через $C_1 \underset{Y}{\sim} C_2$.

Пусть C^L является расширением исходной схемы C по множеству проводов L. Произвольному состоянию α схемы C в общем случае соответствует множество состояний расширения C^L, проекция которых на переменные схемы C равна L. Среди них есть состояние $\tilde{\alpha}$, в котором все провода из α, точнее, все переменные, соответствующие этим проводам, возбуждены, и состояние $\hat{\alpha}$, в котором все они устойчивы. В дальнейшем нас будет интересовать поведение расширения схемы только относительно инициальных состояний типа $\hat{\alpha}$. Таким образом, если задана схема (инициальная) $C = \langle Z, F, \alpha \rangle$, то в качестве ее расширения по множеству проводов L берется инициальная схема $C^L = \langle Z^L, F^L, \hat{\alpha} \rangle$. Для краткости будем говорить, что инициальная схема полумодулярна, нечувствительна к задержкам и т. п., если имеется в виду выполнение указанного свойства относительно начального состояния.

Введем

Определение 8.12. Схему C будем называть *нечувствительной* ко множеству проводов L, если $C \underset{Z}{\sim} C^L$, полумодулярную (инициальную) схему C — *строго нечувствительной* ко множеству проводов L, если ее расширение C^L полумодулярно (относительно начального состояния $\hat{\alpha}$) по множеству переменных Z.

Все неполумодулярные схемы чувствительны к проводам, так как неполумодулярность означает зависимость работы схемы от задержек элементов.

Введенные понятия характеризуют различную степень влияния задержек в проводах на поведение схемы. Между ними существует следующая связь.

Если схема C строго не чувствительна ко множеству проводов L, то C не чувствительна к L. Обратное не всегда верно, так как, вообще говоря, для строго чувствительных схем, даже если они не чувствительны, расширение C^L может оказаться не полумодулярным. Однако если схема C не чувствительна ко множеству проводов L, входных для $z_j (O(L) = \{z_j\})$, и переменная z_j схемы C самонезависима, то C строго не чувствительна к L.

Очевидно, что если схема чувствительна к L и (j, i) — произвольный провод, то схема тем более чувствительна к $L \cup \{(j, i)\}$. С другой стороны, может быть показано, что для анализа чувствительности схем к задержкам в проводах нет необходимости перебирать все возможные для схемы множества проводов, а достаточно рассматривать лишь множества проводов, имеющих одну и ту же выходную переменную.

Для целей анализа воспользуемся понятием булевой производной (определение 4.6). Кроме того, пусть $Z' \subseteq Z$ — некоторое множество переменных схемы. Обозначим через $\alpha(Z')$ состояние, противоположное состоянию α по Z'. Соответственно через $f(Z' = \overline{Z}')$ обозначим функцию, полученную из $f(Z)$ инвертированием всех переменных из Z'. В частности, если $Z' = \{z_i\}$, то это обозначение вырождается в $f(z_i = \overline{z}_i)$.

Определение 8.13. 1) Будем говорить, что *переменная* z_j *активна* для z_i в состоянии α, если $\frac{\partial f_i}{\partial z_j}(\alpha) = 1$; в противном случае z_j *пассивна* для z_i в α.

2) Функцию $\frac{Sf_i(Z)}{SZ'} = f_i(Z) \oplus f_i(Z' = \overline{Z}')$ будем называть *функцией чувствительности* $f_i(Z)$ по множеству переменных Z'.

3) Состояние β схемы C назовем *критичным* для переменной z_j ко множеству $Z' \subseteq Z$, если z_j устойчива в состоянии β и $\frac{Sf_j(\beta)}{SZ'} = 1$.

4) Состояние β назовем *строго критичным* для z_j ко множеству Z', если а) оно критично для z_j ко множест-

ву Z' или б) z_j возбуждена в β, $\frac{Sf_j}{SZ'}(\beta) = 0$ и найдется переменная z_k, активная для z_j в состоянии $\beta(Z')$, причем если $z_k \notin Z'$, то z_k возбуждена и пассивна для z_j в состоянии β.

Проиллюстрируем введенные понятия примером.

Пример 8.11 (продолжение). Возвратимся к системе (8.11), которой соответствует диаграмма переходов рис. 8.9, *б*.

1) Переменная z_1 активна для z_3 в состоянии 0000:

$$\frac{\partial f_3}{\partial z_1}(Z) = f_3(Z) \oplus f_3(z_1 = \bar{z}_1) = (z_1 \vee z_2) \oplus (\bar{z}_1 \vee z_2) = \bar{z}_2,$$

$$\frac{\partial f_3}{\partial z_1}(0000) = 1.$$

Активность означает, что при переходе 0000-1000 изменяется значение функции f_3. Заметим, однако, что переменная z_4 активна для z_1 в состоянии 0010, хотя состояния 0010 и 0011 не связаны отношением непосредственного следования (связь на диаграмме переходов отсутствует).

2) Пусть $Z' = \{z_1, z_2\}$. Тогда

$$\frac{Sf_3}{SZ'} = f_3 \oplus f_3(z_1 = \bar{z}_1,\ z_2 = \bar{z}_2) = (z_1 \vee z_2) \oplus (\bar{z}_1 \vee \bar{z}_2) = z_1 z_2 \vee \bar{z}_1 \bar{z}_2.$$

3) Состояние 0000 критично для z_3 ко множеству $Z' = \{z_1, z_2\}$, поскольку z_3 устойчива в 0000 и $\frac{Sf_3}{SZ'}(0000) = 1$.

4) Состояние 0000 не является критичным для z_1 ко множеству $Z' = \{z_3\}$, но является строго критичным для z_1 к $\{z_3\}$. В самом деле: z_1 — возбуждена в 0000, $Sf_1/Sz_3 \equiv 0$ и переменная z_4 активна для z_1 в состоянии 0011 ($\partial f_1/\partial z_4 \equiv 1$).

Функция чувствительности обобщает понятие булевой производной в том смысле, что при $Z' = \{z_i\}$ $Sf_j/SZ' = \partial f_j/\partial z_i$.

Следующая теорема дает необходимые и достаточные условия чувствительности и строгой чувствительности схем к проводам.

Теорема 8.7. *Пусть L — множество проводов с выходной переменной $z_j (O(L) = \{z_j\})$. Для чувствительности (строгой чувствительности) схемы C ко множеству проводов L необходимо и достаточно, чтобы в схеме C нашлось критичное (строго критичное) для z_j ко множеству $I(L')$, $L' \subseteq L$, состояние β такое, что: 1) существует последовательность состояний, ведущая из α в β, из которой изменяются все переменные из множества $I(L')$, и 2) если на этой последовательности изменяется переменная z_j, т. е. имеет место $\alpha \overset{1}{\Rightarrow} \gamma \overset{1}{\underset{z}{\to}} \delta \overset{1}{\Rightarrow} \beta$, то $Sf_j/SZ'(\gamma) = 0$, где $Z' \subseteq I(L')$, и Z' содержит все такие (и только такие) переменные из $I(L')$, которые не изменяются на последовательности, ведущей из состояния δ в состояние β.*

Доказательство этой теоремы опущено ввиду громоздкости.

На ее основе может быть предложен метод анализа на чувствительность (строгую чувствительность) к произвольному множеству проводов L, аналогичный методам анализа схем на принадлежность подклассам, не зависящим от скорости. На первом этапе выделяются критичные (строго критичные) к L состояния, а на втором — решается задача обратной достижимости и проверяется выполнение условий теоремы 8.7.

Способ выделения критичных (строго критичных) состояний схемы дают следующие утверждения.

Утверждение 8.4. *Пусть L — множество проводов с выходной переменной z_j $(O(L) = \{z_j\})$. Множества критичных и строго критичных для z_j к $I(L)$ состояний задаются соответственно функциями:*

$$\frac{Sf_j(Z)}{SI(L)}(\bar{z}_j \oplus f_j(Z)),$$

$$\frac{Sf_j(Z)}{SI(L)}(\bar{z}_j \oplus f_j(Z)) \vee \overline{\frac{Sf_j(Z)}{SI(L)}} \left(\bigvee_{z_k \in I(L)} \frac{\partial f_j(I(L) = \overline{I(L)})}{\partial z_k} \vee \right.$$

$$\left. \vee \bigvee_{z_k \in I(z_j) \setminus I(L)} \frac{\partial f_j(I(L) = \overline{I(L)})}{\partial z_k} \cdot \overline{\frac{\partial f_j(Z)}{\partial z_k}} (z_k \oplus f_k(Z)) \right),$$

где $I(z_j)$ — множество переменных, от которых существенно зависит функция $f_j(Z)$.

Заметим, что строго критичные для z_j к $I(L)$ состояния можно выявить также, если в расширении C^L найти 1-конфликтные по z_j состояния и затем взять проекцию этих состояний на множество исходных переменных Z.

Установим зависимость между чувствительностью схемы к задержкам в проводах и свойствами булевых функций, задающих схему. Для этого введем ряд новых понятий.

Последовательность состояний $\alpha^1 \alpha^2 \ldots \alpha^k$ $(\alpha^i \to \alpha^{i+1})$ *стационарна* по переменной z_i, если $\alpha_i^1 = \alpha_i^2 = \ldots = \alpha_i^k$

и $f_i(\alpha^1) = \ldots = f_i(\alpha^k)$. Возможны четыре варианта стационарности по переменной z_i:

1) $\alpha_i^e = f_i(\alpha^e) = 0$ и

2) $\alpha_i^e = f_i(\alpha^e) = 1$ — стационарности по устойчивой переменной;

3) $\alpha_i^e = 0$, $f_i(\alpha^e) = 1$ и

4) $\alpha_i^e = 1$, $f_i(\alpha^e) = 0$ — стационарности по возбужденной переменной.

При этом в первом случае $\bar{f}_i^0(\alpha^e) = 1$, во втором — $f_i^1(\alpha^e) = 1$, в третьем — $f_i^0(\alpha^e) = 1$ и в четвертом $\bar{f}_i^1(\alpha^e) = 1$, $1 \leqslant e \leqslant k$, где $f_i^0 = f_i(z_i = 0)$ и $f_i^1 = f_i(z_i = 1)$ — члены разложения булевой функции $f_i(Z)$ по Шенону. Функции $\bar{f}_i^0$, f_i^1, f_i^0, $\bar{f}_i^1$ назовем *определяющими функциями переменной* z_i для стационарной последовательности соответствующего типа. Причем f_i^0 и $\bar{f}_i^1$ — *функции возбуждения*, а $\bar{f}_i^0$ и f_i^1 — *функции блокировки* переменной z_i.

На каждой стационарной по переменной z_i последовательности ее определяющая функция сохраняет значение 1, т. е. в каждом состоянии последовательности хотя бы один терм д. н. ф. этой функции имеет значение 1. При этом в разных состояниях стационарной последовательности значение 1 могут иметь разные термы д. н. ф. определяющей функции.

На стационарной по переменной z_i последовательности состояний $\alpha^1\alpha^2 \ldots \alpha^k$ имеет место *перехват термов*, если все термы сокращенной д. н. ф. определяющей для данной последовательности функции, имеющие значение 1 в состоянии α^1, в состоянии α^k принимает значение 0. Такой перехват назовем *перехватом для переменной* z_i по множеству переменных Z', изменяющихся на данной последовательности, если для каждой $z_j \in Z'$ среди термов сокращенной д. н. ф. определяющей функции, в которые входит переменная z_j, найдется такой терм T_1, для которого $T_1(\alpha^k) = 1$, и не найдется терм T_2, для которого $T_2(\alpha^1) = 1$.

Теорема 8.8. Если в схеме из начального состояния достижимо первое состояние последовательности, на котором происходит перехват термов для переменной z_i по множеству переменных Z', то эта схема строго чувствительна ко множеству проводов L, где $I(L) = Z'$, $O(L) = \{z_i\}$.

Доказательство. Пусть $\alpha^1\alpha^2 \ldots \alpha^m$ — стационарная по z_i последовательность, на которой имеет место указанный в теореме перехват термов по Z', причем ни на какой более короткой последовательности $\alpha^1 \ldots \alpha^l$, $l < m$, такого перехвата нет, т. е. некоторый терм T сокращенной д. н. ф. функции такой, что $T(\alpha^1) = \ldots = T(\alpha^{m-1}) = 1$ и $T(\alpha^m) = 0$. В расширении C^L, где множество L удовлетворяет условиям теоремы, построим последовательность состояний $\tilde{\alpha}^1 \ldots \tilde{\alpha}^m$, которая получается из $\alpha^1 \ldots \alpha^m$ следующим образом: если $\alpha^i \xrightarrow{1} \alpha^{i+1}$, то в строящейся последовательности дополнится $\tilde{\alpha}' \xrightarrow[z_j]{1} \tilde{\alpha}''$, и если $z_j \in Z$ и z_j изменяется на последовательности $\alpha^{i+1} \ldots \alpha^m$, то $\tilde{\alpha}'' \xrightarrow{1}_{l_j} \tilde{\alpha}'''$. Легко видеть, что проекция построенной последовательности на множество Z совпадет с $\alpha^1 \ldots \alpha^m$ и $f_i^L(\tilde{\alpha}^m) \neq f_i(\alpha^m)$, так как все термы определяющей функции, равные 1 в состоянии α^m, равны 0 в $\tilde{\alpha}^m$, а значит если переменная z_i устойчива в состоянии α^m схемы C, то она возбуждена в состоянии $\tilde{\alpha}^m$ расширения C^L и, наоборот, если z_i возбуждена в α^m, то z_i устойчива в $\tilde{\alpha}^m$. В первом случае неверно, что $C \sim C^L$, а во втором C^L не полумодулярна по z_i. Следовательно, C строго чувствительна к L.

Следствие. *Если в инициальной схеме из начального состояния достижимо первое состояние последовательности, на которой происходит перехват для z_i по z_j, причем переменная z_j входит во все термы, равные 1, в заключительной последовательности α^k, то эта схема строго чувствительна к проводу (j, i).*

Функционирование некоторых асинхронных логических схем не зависит от величин задержек проводов. К числу таких схем относится, например, тривиальный подкласс полумодулярных схем, в которых отсутствуют разветвления проводов. Более широкий класс схем, не чувствительных к задержкам в проводах, дает следующая теорема.

Теорема 8.9. *Если инициальная схема C полумодулярна и функция $f_j(Z)$ существенно зависит только от одной переменной z_i, то схема C строго не чувствительна к проводу (i, j).*

Вообще говоря, теорема 8.9 справедлива также для схем, в которых каждая функция возбуждения переменной z_j существенно зависит только от одной переменной, например $z_j = z_j z_i \vee \bar{z}_j z_k$. В этом случае схема строго не чувствительна к проводам (i, j) и (k, j). Однако и этот класс асинхронных логических схем сравнительно узок, в частности он не покрывает даже класса последовательных схем.

Пример 8.12. Схема, приведенная на рис. 5.8, *а*, состоящая из пары *RS*-триггеров, является последовательной относительно любого состояния, кроме таких, в которых оба плеча какого-либо триггера устанавливается в 0. Эта схема строго не чувствительна ко множеству проводов, соединяющих между собой триггеры. В то же время она чувствительна к проводам в обратных связях триггеров. Структура этой схемы характерна для триггерных устройств типа Master — Slave и приведенные результаты анализа справедливы для всех подобных устройств.

Параллельно-последовательная схема рис. 4.2, *г* имеет пять проводов: (z_2, z_1), (z_3, z_1), (z_1, z_2), (z_1, z_3), (z_1, z_1). Она строго не чувствительна ко всем своим проводам, кроме последнего. Структура этой схемы является типовой для параллельно-последовательного соединения полумодулярных схем, что также позволяет обобщить результаты анализа.

§ 8.5. Схемные сети Петри

В предыдущих разделах этой главы разработаны методы анализа схем, основанные на непосредственном использовании модели Маллера. Применение для описания функционирования схем такого «локального» представления позволило получить эффективные алгоритмы анализа.

Однако непосредственно в модели Маллера (если не строится диаграмма переходов) динамика работы схемы отражается в скрытой форме, косвенно, не наглядно.

Как известно, одним из наиболее распространенных формальных инструментов изучения параллельных систем являются сети Петри, рассмотренные в § 2.2. Сети Петри наглядно отражают динамику функционирования системы и, как уже было сказано, являются локальным ее описанием, т. е. в них, как и в модели Маллера, устанавливаются локальные соотношения между компонентами системы, а «глобальное» поведение системы изучается как результат локальных взаимодействий между этими компонентами.

В настоящем параграфе вводится некоторый новый *класс сетей Петри, однозначно соответствующий модели Маллера.* Этот класс специально предназначен для анализа асинхронных схем, наглядно отражает динамику их функционирования и свободен от ряда недостатков классов сетей, рассмотренных в п. 2.2.2.

В основу предлагаемого подхода положена идея представления элемента схемы некоторой подсетью сети Петри.

Пусть $P^* \subseteq P$ — подмножество условий и $T^* \subseteq T$ — подмножество событий сети N, тогда примем обозначения:

$$I(P^*) = \bigcup_{p_i \in P^*} I(p_i), \quad O(P^*) = \bigcup_{p_i \in P^*} O(p_i),$$

$$C(P^*) = I(P^*) \cap O(P^*), \quad V(P^*) = O(P^*) \setminus C(P^*),$$

$$J(P^*) = I(P^*) \setminus C(P^*), \quad I(T^*) = \bigcup_{t_i \in T^*} I(t_i),$$

$$O(T^*) = \bigcup_{t_i \in T^*} O(t_i).$$

Определение 8.15. 1) Пусть заданы сеть Петри $N = \langle P, T, M_0, H, F\rangle$ и некоторое подмножество $P^* \subseteq P$ условий этой сети. *Подсетью* N^* сети N, соответствующей P^*, будем называть такую сеть $N^* = \langle P^*, T^*, M_0^*, H^*, F^*\rangle$, что

$$T^* = I(P^*) \cup O(P^*), \quad F^*\colon P^* \times T^* \to \{0, 1\},$$

$$H^*\colon T^* \times P^* \to \{0, 1\}, \quad M_0^*\colon P^* \to \{0, 1, \ldots\}.$$

2) *Петлей* назовем подсеть $N^l = \langle P^l, T^l, M_0^l, H^l, F^l\rangle$ сети N, где $P^l = \{p_i\}$, $T^l = \{t_j\}$, $M_0^l = M_0(p_i)$, $H^l = H(p_i, t_j)$, $F^l = F(t_j, p_i)$, причем $H^l = F^l = 1$.

От обычного изображения петли на рис. 8.10, *а* перейдем к более экономичному — рис. 8.10, *б*. Очевидно, что срабатывание события t_j невозможно при $M(p_i) = 0$ и что срабатывание t_j не изменяет $M(p_i)$.

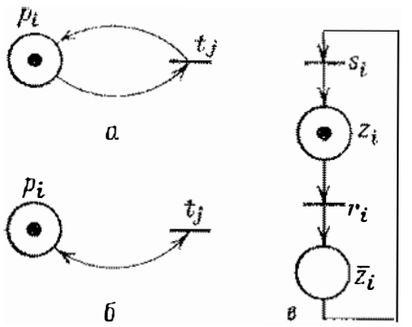


Рис. 8.10. Изображение петли (*а*, *б*) и цикла элемента (*в*) в схемной сети Петри

Поставим в соответствие переменной z_i схемы пару условий z_i и $\bar{z}_i$ сети Петри, переключениям элемента — два события s_i и r_i. Событие $s_i(r_i)$ соответствует переключению 0-1 (1-0) элемента, оно является входным для условия z_i ($\bar{z}_i$) и соответствует $z_i = 1$ ($z_i = 0$). Таким образом, поведение элемента моделируется подсетью типа рис. 8.10, *в*; назовем ее *циклом элемента.*

Собственную функцию элемента можно представить в виде: $z_i = z_i\bar{R}_i \vee S_i\bar{z}_i$. Переключению 0-1 соответствует

$S_i = 1$, а переключению 1-0 — $R_i = 1$. Поскольку переключение элемента не влияет непосредственно на его входы, дуги сети Петри, отражающие связь входов одного элемента с выходами других, должны входить в петлю.

Воспользуемся конъюнктивным разложением функций S_i и R_i для переменной z_i; тогда в моделирующей схему сети Петри необходимо иметь вспомогательные условия, соответствующие дизъюнктивным термам этих функций, такие условия будут входными для соответствующих событий типа $s_j(r_j)$ (входящих в циклы отличных от z_i элементов) и выходными для событий типа $r_j(s_j)$. Таким образом, можно построить сеть, моделирующую функционирование любой схемы.

Пример 8.13. На рис. 8.11, *а* приведена схема, а на рис. 8.11, *б* — моделирующая ее сеть Петри. Начальная разметка сети соответствует исходному состоянию 111 схемы. Функция $S_1 = \bar{z}_2 \vee \bar{z}_3$ содержит один дизъюнктивный терм. Ему поставлено в

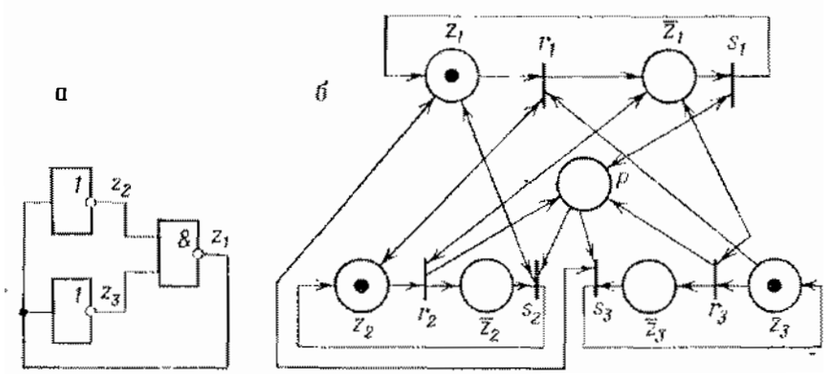


Рис. 8.11. Пример схемной сети Петри

соответствие условие, для которого $J(P) = \{r_2, r_3\}$, $V(P) = \{s_2, s_3\}$, а $C(P) = \{s_1\}$.

Описанную сеть можно отнести к специальному классу сетей Петри. Такие сети обладают структурными особенностями (каждое событие входит в какой-нибудь цикл элемента, при этом все условия, инцидентные этому событию и не входящие в цикл элемента, образуют с ним петлю). Будем называть такие сети *схемными сетями Петри*. Точнее, схемные сети Петри, соответствующие конъюнктивной форме представления собственных функций элементов, будем именовать *конъюнктивными*

схемными сетями Петри. Воспользовавшись дизъюнктивным разложением функций S_i и R_i, можно построить *дизъюнктивные схемные сети Петри*.

Для схемы из n элементов конъюнктивная сеть содержит $2n$ событий. В множестве P можно выделить два подмножества: P_1 — собственные условия, входящие в какой-то из циклов элементов, $|P_1| = 2n$, и P_2 — условия, соответствующие дизъюнктивным термам собственных функций, $|P_2| \leqslant \sum_{i=1}^{n} m_i$, где m_i — число термов к. н. ф. функций S_i и R_i элемента z_i.

Дизъюнктивная сеть содержит $2n$ условий, число переходов (событий) $|T| = \sum_{i=1}^{n} m_i$, где m_i — число термов д. н. ф. функций S_i и R_i.

Каждая разметка схемной сети Петри соответствует некоторому состоянию моделируемой схемы, поэтому множество R_N всех ее разметок, достижимых из M_0, ограничено — $|R_N| \leqslant 2^n$, где n — число переменных схемы.

Основной для нас проблемой анализа сетей Петри является определение их устойчивости. Напомним, что конфликтность разметки есть локальный признак нарушения устойчивости сети.

Разметка M конфликтна, если после реализации некоторых, возможных при M событий (T^*), при разметке M^* ($M \xrightarrow{T^*} M^*$) в каком-нибудь условии (p_i) «не хватает» точек для реализации возможных при M событий, если же точек «хватает», M не является конфликтной.

Следующая теорема дает признак конфликтности разметки в конъюнктивной сети.

Теорема 8.10. *Пусть в конъюнктивной схемной сети Петри найдется условие $p_r \in P$ и событие $t_j \in O(p_r)$, и $V'(p_r)$ — подмножество множества выходных событий условия $P(V'(p_r) \subseteq V(p_r))$, $t_j \notin V'(p_r)$ и $M(p_r)$ — некоторая разметка, существует натуральное число $l = M(p_r)$. Разметка M конфликтна тогда и только тогда, когда возможны все события и $V'(p_r)$ и событие t_j и, кроме того, $|V'(p_r)| \geqslant l$.*

Доказательство теоремы непосредственно следует из определения конфликтности разметки и правила образования множества $V(P)$.

Сформулируем теперь центральное для анализа асинхронных процессов и схем на языке сетей Петри положение.

Теорема 8.11. *Пусть разметка M конъюнктивной схемной сети Петри N соответствует состоянию U схемы C. Схема полумодулярна относительно состояния U, если любая разметка M', достижимая от M в N, не является конфликтной.*

Справедливость последнего свойства немедленно следует из сравнения определений устойчивости сети Петри и полумодулярности схемы.

Метод поиска покрытия множества всех конфликтных разметок схемных сетей Петри основан на теореме 8.10, из которой вытекает существование алгоритма поиска покрытия множества состояний схемы, относительно которых она не полумодулярна. Первая часть этого алгоритма состоит в определении конфликтных разметок сети, а вторая в отыскании разметок, от которых достижимы конфликтные разметки. Аналогичным образом можно показать, что все алгоритмы анализа, рассмотренные в предыдущих разделах этой главы, могут быть переформулированы применительно к анализу схемных сетей Петри, что, впрочем, следует и из однозначного соответствия между схемными сетями и моделью Маллера. Заметим, что анализ схем с помощью схемных сетей Петри, таким образом, редуктивно сводится к решению проблемы достижимости в сетях Петри (с учетом факта их ограниченности).

Схемные сети Петри обладают рядом не приведенных здесь интересных свойств, позволяющих, в частности, анализировать схемы на отсутствие *дедлоков* и *живость*, т. е. способность всех элементов схемы переключаться неограниченное число раз при циклической работе схемы.

Использование для анализа схем такой распространенной модели параллельных процессов, как сети Петри, не только наглядно иллюстрирует асинхронность и параллельность переключения элементов схемы, но и открывает возможность использования рассмотренных в настоящем параграфе алгоритмов для анализа корректности асинхронных параллельных программ.

Предложенные методы анализа иногда удобно совмещать с методами проверки индицируемости, которые будут изложены в гл. 9.

§ 8.6. Оценка сложности алгоритмов анализа

Будем говорить, что функция $f(n)$ *по порядку не превосходит* $g(n)$, и писать $f(n)=O(g(n))$, если $f(n)\leqslant cg(n)$ почти для всех n, где c — некоторая константа.

Анализ достижимости, живости, независимости от скорости и ее частных случаев (полумодулярности, дистрибутивности, последовательности и т. п.) можно проводить в рамках неинтерпретированного асинхронного процесса, используя в качестве исходной информации задание отношения F на множестве ситуаций, например в виде матрицы инциденций асинхронного процесса. Тогда проблема анализа достижимости сводится к построению матрицы связности, что может быть сделано, например, алгоритмом Уоршелла за $O(|S|^3)$ шагов, где $|S|$ — число ситуаций асинхронного процесса. Поиск точек нарушения полумодулярности, живости, и последовательности осуществляется на основе проверки локальных свойств графа (матрицы инциденций) процесса и требует $O(|S|^2)$ шагов. Таким образом, проверка того, является ли асинхронный процесс полумодулярным, дистрибутивным или последовательным относительно произвольной ситуации, может осуществляться за $O(|S|^3)$ шагов.

Однако этот путь анализа не всегда удобен, во-первых, в силу того, что число ситуаций асинхронного процесса $|S|$ может быть велико и исходное его задание, длина которого $O(|S|^2)$, будет громоздким, и, во-вторых, такой подход не предоставляет компактных средств анализа относительно множеств ситуаций.

Более удобными для анализа являются интерпретации асинхронного процесса, например в рамках модели Маллера (или схемных сетей Петри), в которой за счет кодирования ситуаций и описания переменных формулами длина задания процесса становится по порядку равной $c(m)\log_2(|S|^2)$, где $c(m)=O(2^m)$ — параметрическая константа, параметр m — максимальное число кодовых переменных, от которых существенно зависит данная переменная ($m<\log_2|S|$, причем на практике часто $m<<\log_2|S|$). Кроме того, в указанных интерпретациях эффективность анализа может повышаться за счет различных, в том числе эвристических, методов, а также имеются компактные средства анализа множеств ситуаций (см. предыдущие разделы настоящей главы).

В табл. 8.1 приводится краткая сводка оценок сложности для решения проблем анализа в рамках модели Маллера и схемных сетей Петри, где $c_1(m)$, $c_2(m)$, $c_3(m)$ — параметрические константы:

$$c_1(m) = O\left(2^{\min(m^2, \log_2 |S|)}\right),$$

$$c_2(m) = O\left(2^{\min(m^3, \log_2 |S|)}\right),$$

$$c_3(m) = O\left(2^{\min(m^2, \log_2 |S|)}\right).$$

Отметим в заключение, что изложенные в настоящей главе методы анализа асинхронных процессов и схем в рамках модели Маллера (на основе применения характеристических функций) и схемных сетей Петри позволяют получить достаточно эффективные и используемые

Таблица 8.1

Проблема (свойство)	Сложность по порядку
Достижимость	$\|S\|^3$
Поиск ситуаций нарушения полумодулярности	$c_1(m) \log_2 \|S\|$
Поиск ситуаций нарушения дистрибутивности	$c_2(m) \log_2^3 \|S\|$
Поиск ситуаций нарушения последовательности	$c_3(m) \log_2^2 \|S\|$

на практике методы машинного анализа асинхронных процессов ограниченной размерности, и приведенные оценки сложности следует считать пессимистическими.

§ 8.7. Замечания по библиографии

Вопросам анализа функционирования асинхронных параллельных систем на основе рассмотренных ранее моделей (с ориентацией, впрочем, на вопросы теоретического программирования) посвящено большое количество работ. Среди них можно выделить публикации по анализу на сетях Петри [119, 197, 238, 239, 245, 254, 305, 306] маркированных графах [260] и системе переходов [248] (модели, близкой к асинхронному процессу [9]). Более полную информацию можно получить по обзорам [197, 198, 284] и тематическим сборникам [275, 292].

Различные постановки задачи анализа схем, отличные от рассмотренных в данной главе, приводятся в [169, 181, 187, 199, 229].

Работы по анализу схем, не зависящих от скорости, проводились под руководством Д. Е. Маллера.

Методы решения прямой и обратной задач достижимости состояний схемы изучаются в [4, 181].

В [249] предлагается подход к построению схем, работа которых не зависит от задержек в проводах, основанных на использовании некоторого универсального набора модулей. Схемы, построенные из таких модулей, не зависят от скорости работы модулей и нечувствительны к задержкам межмодульных проводов. Недостатком этого подхода является громоздкость полученных схемных решений и, главное, зависимость работы модулей от задержек элементов, из которых они построены, и от задержек внутримодульных проводов.

Эффект повышения влияния задержек проводов с увеличением уровня интеграции схем рассмотрен в [207, 291].

Вопросы анализа переключательных схем с помощью сетей Петри специального вида рассмотрены в [99, 237].

Алгоритм Уоршелла построения матрицы связности графа предложен в [308].

Методы анализа схем на живость и отсутствие дедлоков излагаются в [99, 199, 229, 232].

Материал гл. 8 базируется на работах [4, 99]. Доказательство теорем 8.5 и 8.11 можно найти в [4].

Рассмотренные в данной главе методы анализа могут быть использованы для проверки корректности протоколов обмена данными [65, 98] и управляющих структур параллельных программ.

Я понял, что должен выбрать между ними раз и навсегда. Мои две натуры обладали общей памятью...

Р. Л. Стивенсон

Глава 9. АНОМАЛЬНОЕ ПОВЕДЕНИЕ ЛОГИЧЕСКИХ СХЕМ И ПРОБЛЕМА АРБИТРАЖА

Развитие архитектуры вычислительных и управляющих систем, увеличение их быстродействия привели к необходимости решения проблемы временного согласования обмена между блоками систем и проблемы разделения общего ресурса. В середине 60-х годов появились первые упоминания о трудностях, возникающих при решении указанных проблем. Эти трудности можно проиллюстрировать на процессах в асинхронном приемнике, на вход которого подается асинхронный сигнал (рис. 9.1). Приемник состоит из D-триггера T, прямой и инверсный выходы которого обозначены Q и $\overline{Q}$, и элемента И, выход которого соединен со входом D триггера T. На входной вентиль приемника подается внешняя асинхронная последовательность x и последовательность внутренних синхроимпульсов c. На выходе вентиля могут появиться короткие импульсы (рис. 9.1, *б*), энергии которых достаточно для инициации переходного процесса в триггере приемника, но недостаточно для того, чтобы триггер изменил свое состояние. В результате возникающего внутри триггера явления арбитража, он может оказаться в состоянии неустойчивого равновесия (*метастабильном состоянии*), при котором значения сигналов совпадают и не представляют ни логического 0, ни логической 1. Подобные устройства, согласующие асинхронно приходящие на их входы последовательности сигналов, получили название *синхронизаторов*, а указанное поведение этих устройств стали называть *аномальным*.

В дальнейшем было обнаружено, что аномальное поведение характерно и для *арбитров* — устройств, служащих

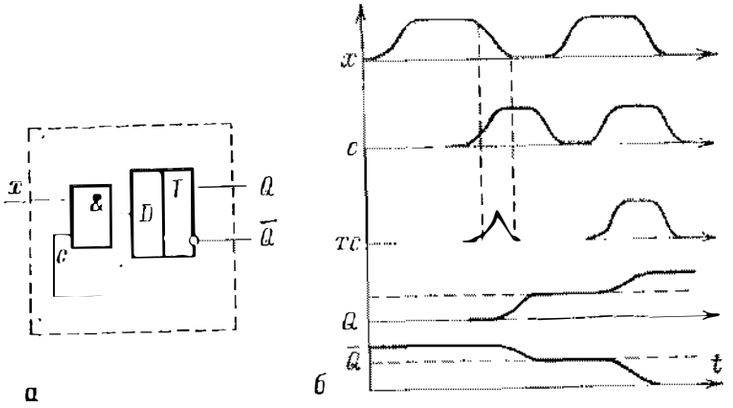


Рис. 9.1. Поведение синхронной схемы при подаче асинхронного сигнала: схема (*а*), осциллограмма сигналов (*б*)

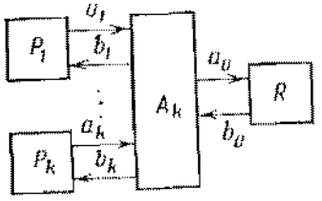


Рис. 9.2. Арбитр с k входами (A_k — схема)

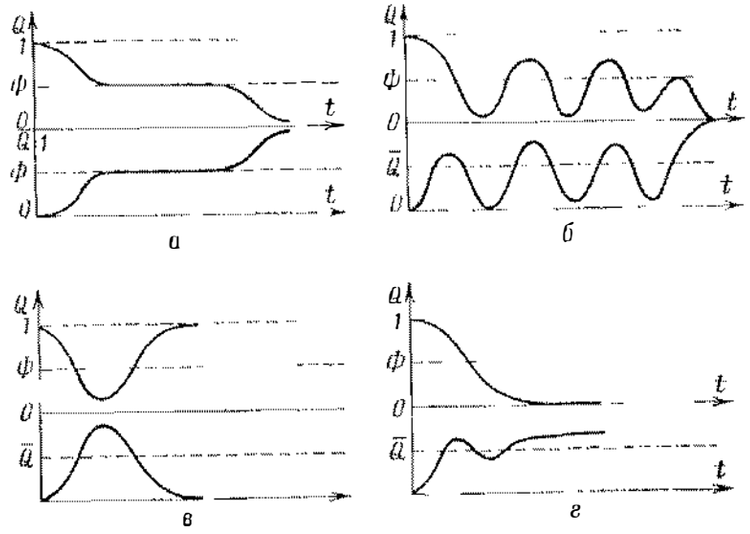


Рис. 9.3. Виды аномалий в схемах, реализующих арбитры: метастабильная аномалия (*а*), колебательная аномалия (*б*), нестабильный переходный процесс (*в*), затягивание фронта (*г*)

для предоставления общего ресурса ровно одному из процессов, запрашивающих этот ресурс. *Проблема арбитража* встает вне зависимости от природы процессов и общего ресурса. Даже если они программны, проблема в конечном счете сводится к построению работоспособного аппаратного арбитра. Общепринятая структура внешних соединений k-входового арбитра (рис. 9.2) содержит: $a_1, \ldots, a_k$ — сигналы запроса у арбитра общего ресурса R от процессов $P_1, \ldots, P_k$; $b_1, \ldots, b_k$ — сигналы ответа от арбитра на запросы $P_1, \ldots, P_k$; a_0 — сигнал запроса ресурса от арбитра; b_0 — сигнал ответа ресурса арбитру.

В поведении логических схем, реализующих арбитры и синхронизаторы, известны четыре вида аномалий: уже упомянутая *метастабильная аномалия* (рис. 9.3, *а*); *колебательная аномалия* (рис. 9.3, *б*) — синфазное колебательное изменение сигналов на обоих выходах, при котором значения сигналов в каждый момент времени совпадают; *нестабильный переходный процесс* (рис. 9.3, *в*); *затягивание одного из фронтов* в переходном процессе (рис. 9.3, *г*). Аномальное поведение может возникнуть при подаче короткого импульса, амплитуда которого незначительно выше порога срабатывания элемента, на один из входов арбитра или синхронизатора, или в случае, когда запросы на общий ресурс приходят от разных процессов с промежутком меньшим, чем время завершения переходных процессов в арбитре. Возможность возникновения аномалий двух последних типов зависит от структуры устройства, и борьба с ними несложна, в то время как аномалии первых двух типов носят фундаментальный характер, и избавление от них представляет наибольшие трудности.

Экспериментальные данные показывают, что аномальное поведение арбитров и синхронизаторов, реализованных на двоичных логических элементах, являются существенным источником сбоев вычислительных и управляющих систем. Ниже дано теоретическое обоснование аномальных явлений и указаны пути построения работоспособных арбитров. Подобное обоснование представляется актуальным, ибо по сей день в практике проектирования появляется большое число некорректно реализованных арбитров и синхронизаторов.

§ 9.1. Арбитры

Прежде чем формально определить арбитры, введем ряд вспомогательных определений.

Асинхронная логическая схема C есть тройка вида $\langle Z, Z_0, F\rangle$, где $Z = \{z_1, \ldots, z_n\}$ — множество двоичных переменных схемы, $Z_0 \subset Z$ — множество входных переменных, система булевых функций — F — имеет вид $z_i = f_i(z_1, \ldots, z_n)$, $z_i \in Z \setminus Z_0$.

Вектор γ значений переменных из $Z' \subset Z$ назовем *подсостоянием схемы*. Подсостояние размерности l задает подкуб, содержащий 2^{n-l} состояний. Подсостояние γ будем называть *стабильным* (относительно фиксации входных переменных), если во всех задаваемых им состояниях переменные $z_i \subset Z'$ ($z_i \in Z' \cap (Z \setminus Z_0)$) стабильны. Так как мы будем исследовать поведение арбитров при фиксированных входных сигналах запроса, то достаточно считать, что входные переменные стабильны во всех состояниях.

Обозначим множества всех состояний, достижимых из состояния α и непосредственно достижимых из α, через $R(\alpha)$ и $r(\alpha)$ соответственно. Напомним, что множество состояний $D = \{\alpha_1, \ldots, \alpha_n\}$ называется замкнутым классом, если для каждого $\alpha_i \in D$ имеем $R(\alpha_i) = D$. Через $E(\alpha)$ обозначим множество замкнутых классов, достижимых из α.

Проекция множества состояний $B = \{\alpha_1, \ldots, \alpha_m\}$ на Z' ($B|Z'$) определена и равна проекции состояния $\alpha_1|Z'$, если $\alpha_1|Z' = \ldots = \alpha_m|Z'$, и не определена в противном случае. Для краткости в значении проекции мы будем писать только те переменные из Z', значения которых равны 1 в проецируемом состоянии (множестве состояний), а сами состояния задавать не двоичными векторами, а соответствующими им конъюнктивными термами. Например, если $Z' = \{z_1, z_2\}$, то $\bar{z}_1 z_2 \bar{z}_3 \bar{z}_4 | Z' = z_2$, и $\bar{z}_1 \bar{z}_2 z_3 z_4 | Z' = 0$.

Схема C_1 *покрывает* схему C_2 той же размерности, если каждая дуга, содержащаяся в диаграмме переходов (ДП) C_2, содержится и в ДП C_1.

Отметим, что используемая модель асинхронной логической схемы базируется на тех же, что и в гл. 4, гипотезах относительно ее физической реализации.

Определение 9.1. Схема $A_k = \langle Z, Z_0, \Lambda, B, F\rangle$, где $\Lambda = \{a_1, \ldots, a_k\} \subset Z_0$ — множество входных переменных

запроса, $B = \{b_1, \ldots, b_k\} \subset Z$ — множество переменных ответа, называется *арбитром ранга* k (A_k-схемой), если она удовлетворяет следующим условиям:

1) *условию установки ответа* — если состояние α таково, что $\alpha|A = a_{i1} \ldots a_{il}$, $l \leqslant k$, то а) для всякого замкнутого класса $D \in E(\alpha)$ имеет место $D|B = b_{ir}$, где $1 \leqslant r \leqslant l$, и б) всякий полный путь из α ведет в замкнутый класс;

2) *условию сброса ответа* — если состояние α таково, что $\alpha|a_i = 0$, то $E(\alpha)|b_i = 0$;

3) *условию устойчивости ответа* — если состояние α таково, что $\alpha|a_i = a_i$ и $\alpha|B = b_i$, то $r(\alpha)|B = b_i$.

Отметим, что здесь от A_k-схем не требуется, чтобы они были свободны от риска (состязаний) по переменным ответа, т. е. на переходный режим для переменных ответа не накладываются ограничения, а также, без снижения общности, считается, что используется прямое кодирование сигналов. Кроме того, поскольку причиной

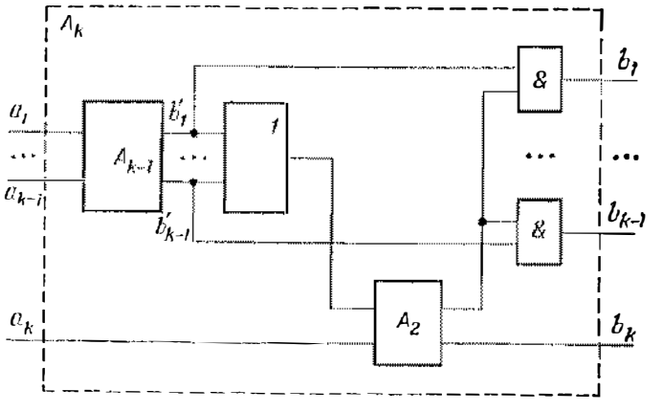


Рис. 9.4. К доказательству возможности построения произвольных арбитров из двухвходовых

аномалий в поведении арбитров является их взаимодействие с процессами, нас интересует только та часть арбитров, которая ведает этим взаимодействием.

Свойство 9.1. *Всякая* A_k*-схема* ($k > 1$) *является* A_{k-1}*-схемой.*

Свойство 9.2. *Всякая* A_k*-схема* ($k > 1$) *может быть построена из* A_{k-1}*- и* A_2*-схем.*

Доказательство для $k = 2$ тривиально. Для $k > 2$ на рис. 9.4 приведена конструкция, для которой нетрудно показать, что если A_{k-1} и A_2 удовлетворяют условиям 1)–3) определения 9.1, то им удовлетворяет и A_k. Из этих свойств ясно, что для того, чтобы строить произвольные A_k-схемы, необходимо и достаточно уметь строить A_2-схемы.

§ 9.2. Колебательная аномалия

Введем еще несколько вспомогательных понятий, которые позволяют исследовать влияние внутренних обратных связей элементов на аномальное поведение арбитрирующих схем.

Переменная z_i схемы *самозависима*, если $f_i(z_1, \ldots, z_n)$ существенно зависит от z_i, т. е. $f_i(z_i = 0) \neq f_i(z_i = 1)$. В противном случае z_i *несамозависима. Схема* C *несамозависима*, если каждая ее переменная несамозависима, иначе C *самозависима.*

Переменная z_i схемы *(не)антитонно-самозависима*, если она самозависима и f_i (не)антитонна по z_i. Схема C *антитонно-самозависима*, если все ее самозависимые переменные антитонно-самозависимы. Иначе C *неантитонно самозависима.*

Свойство 9.3. *Схема* C *несамозависима по переменной* z_i *тогда и только тогда, когда для любой пары соседних по* z_i *состояний* α *и* α' *выполняется либо* $\alpha \in r(\alpha')$, *либо* $\alpha' \in r(\alpha)$, *но не оба условия вместе.*

Свойство 9.4. *Схема антитонно-самозависима по переменной* z_i *тогда и только тогда, когда для любой пары соседних по* z_i *состояний* α *и* α' *выполняется* $\alpha \in r(\alpha')$, *либо* $\alpha' \in r(\alpha)$ *и хотя бы для одной пары оба условия вместе.*

Определение 9.2. Будем говорить, что схема C обладает *колебательной аномалией* по множеству переменных Z', если найдется цикл состояний $\alpha_1\alpha_2 \ldots \alpha_p\alpha_1\alpha_2 \ldots$ $\ldots(\alpha_i \in r(\alpha_{i-1}),\ 1 \leqslant i \leqslant p-1,\ \alpha_1 \in r(\alpha_p))$ такой, что для некоторых α_i, α_j, $1 \leqslant i, j \leqslant p$ выполняется $\alpha_i|Z' \neq \alpha_j|Z'$.

Теорема 9.1. *Никакая* A_k*-схема не принадлежит классу несамозависимых или антитонно-самозависимых схем.*

Доказательство. В силу свойств 9.1, 9.2 и того, что конструкции из свойства 9.2 сохраняет несамозависимость и антитонную самозависимость, доказательство достаточно проводить для A_2-схем. $A = \{a_1, a_2\}$, $B = \{b_1, b_2\}$.

Возможны два случая.

1. Для каждого состояния α такого, что $\alpha|A = a_1a_2$, и для произвольных D_i, $D_j \in E(\alpha)$ выполняется $D_i|B = D_j|B$. Пусть для определенности $D_i|B = b_1$. Рассмотрим два соседних по b_2 состояния: $\alpha = a_1a_2\bar{b}_1\bar{b}_2\beta$ и $\alpha' = a_1a_2\bar{b}_1b_2\beta$, где β — вектор значений остальных переменных схемы. В соответствии со свойствами 9.3, 9.4 должно выполняться $\alpha \in r(\alpha')$ или $\alpha' \in r(\alpha)$, но первое противоречит условию устойчивости ответа в A_k-схеме, а второе — исходной посылке о том, что $D_i|B = b_i$, либо условию устойчивости ответа. Таким образом, случай 1 противоречив.

2. Найдется состояние α, $\alpha|A = a_1a_2$ такое, что есть два замкнутых класса D_1, $D_2 \in E(\alpha)$, для которых имеет место $D_1|B \neq D_2|B$. Для определенности положим, что $D_1|B = b_1$, $D_2|B = \bar{b}_2$. Рассмотрим два множества состояний R_1 и R_2 таких, что $R_1|A = R_2|A = a_1a_2$, $R_1|B = b_1$, $R_2|B = b_2$. Очевидно, что $D_1 \subset R_1$, $D_2 \subset R_2$, $R_1 \cap R_2 \neq \varnothing$, и в силу условия устойчивости ответа R_1 и R_2 являются замкнутыми множествами состояний ($a_1 = a_2 = 1$). Рассмотрим четыре состояния: $\alpha_1 = a_1a_2\bar{b}_1\bar{b}_2\beta$, $\alpha_2 = a_1a_2b_1\bar{b}_2\beta$, $\alpha_3 = a_1a_2\bar{b}_1b_2\beta$, $\alpha_4 = a_1a_2b_1b_2\beta$. Очевидно, что α_1, $\alpha_4 \notin R_1$, R_2 и не принадлежат ни одному замкнутому классу из $E(\alpha_1)$. Состояние α_1 является соседним с α_2 и α_3, состояние α_4 — с α_2 и α_3, кроме того, α_1, $\alpha_4 \notin r(\alpha_2)$, $r(\alpha_3)$. Следовательно, согласно свойствам 9.3 и 9.4, α_2, $\alpha_3 \in r(\alpha_1)$, $r(\alpha_4)$. Однако тогда $\alpha_1 \in r(\alpha_4)$, $\alpha_4 \in r(\alpha_1)$, что противоречит условию установки ответа (б).

Следствие 1. *Всякая несамозависимая или антитонно-самозависимая схема, покрывающая A_k-схему, обладает колебательной аномалией по переменным ответа.*

Это следствие основывается непосредственно на доказательстве теоремы 9.1. В самом деле, $\alpha_1\alpha_4$ — колебательная аномалия по переменным ответа независимо от β.

Если под элементом понимать один полупроводниковый вентиль, то никакой элемент не может быть самозависимым. Тем не менее совокупность вентилей, размещенных в одном корпусе с одним входным инвертором-усилителем сигнала, можно рассматривать как один элемент, задержка которого приведена к выходному инвертору. Такая модель адекватно отражает поведение существующих дискретных элементов и интегральных микросхем. Замыкание выхода такого элемента на вход дает схему, которую с достаточной степенью точности можно рассматривать как антитонно-самозависимую. Для реализации неантитонно-самозависимой схемы необходимо получить положительный коэффициент усиления в контуре, что может быть достигнуто двумя путями: 1) в контуре нет инверторов или 2) в контуре четное число инвертирующих каскадов. В первом случае невозможно получить усиление сигнала в контуре, а во втором несправедливо предположение о приведении задержки к выходному инвертору; при анализе поведения такого элемента каждый инвертирующий каскад необходимо рассматривать как отдельный элемент, и, следовательно, элемент не будет самозависимым.

Таким образом, арбитр нереализуем средствами двоичной логической схемотехники, так как всякая такая реализация будет обладать колебательной аномалией по переменным ответа.

§ 9.3. Метастабильная аномалия

Под *троичными* (Т) *функциями* понимаются отображения n-й степени трехэлементного множества $\{0, \varnothing, 1\}^n$ в себя. Определим класс *булево-троичных* (БТ) функций как троичные функции, сохраняющие множество $\{0, 1\}$, т. е. БТ-функция на любом двоичном наборе из 0 и 1 принимает значения 0 или 1, но не $\varnothing$.

Рассмотрим примеры элементарных троичных функций.

1. 0, $\varnothing$, 1 — константы.
2. $x \vee y = \max(x, y)$ — обобщенная дизъюнкция (считаем, что $0 < \varnothing < 1$).
3. $x \cdot y = \min(x, y)$ — обобщенная конъюнкция.
4. $\bar{x} = 1 - x$ — обобщенное отрицание ($1 - \varnothing = \varnothing$).
5. $x^a = \begin{cases} 1, & x = a, \\ 0, & x \neq a \end{cases}$ — характеристические функции для трех значений a ($a = 0, \varnothing, 1$).

Все приведенные функции, за исключением константы $\varnothing$, являются БТ-функциями.

Определение 9.3. *Троичной схемой* (ТС) C^T будем называть тройку $\langle Z^T, Z_0^T, F^T \rangle$, где $Z^T = \{z_1, \ldots, z_n\}$ — множество троичных переменных схемы, $Z_0^T \subset Z^T$ — множество входных троичных переменных схемы, F^T — система троичных уравнений вида $z_i = F_i(z_1, \ldots, z_n)$, $z_i \in Z^T \setminus Z_0^T$, где F_i — T-функция.

Все понятия, приведенные выше для асинхронных схем, можно обобщить для ТС. *Переменная $z_i \in Z^T$ метастабильна* в состоянии α, если $\alpha|z_i = F_i(\alpha) = \varnothing$. Подсостояние γ, соответствующее значениям переменных из $Z' \subset Z$, будем называть метастабильным, если во всех задаваемых им состояниях хотя бы одна переменная $z_i \in Z'$ метастабильна, а остальные переменные из Z' не возбуждены. Очевидным частным случаем метастабиль-

ного подсостояния является метастабильное состояние (при $Z' = Z$). Для ТС можно ввести понятие троичной диаграммы переходов подобно тому, как это было сделано для двоичных схем.

Пусть в схеме поведение элемента (переменной) z_i задавалось булевым уравнением $z_i = f_i(z_1, \ldots, z_n)$, где собственная функция f_i задана в базисе И-ИЛИ-НЕ. Обобщением функции $f_i(z_1, \ldots, z_n)$, которое будем обозначать также через $f_i(z_1, \ldots, z_n)$, назовем БТ-функцию, полученную из исходной булевой функции обобщением операций конъюнкции, дизъюнкции и отрицания и заменой двоичных переменных на троичные. Для получения обобщения булевой функции, представленной в другом базисе, необходимо выразить каждую операцию из базиса формулой над базисом И-ИЛИ-НЕ, а затем обобщить операции и переменные.

В троичной схеме, описывающей поведение реальной схемы в переходном режиме, поведение элемента (переменной) z_i будем задавать троичным уравнением $z_i = F_i(z_1, \ldots, z_n)$. Значение Т-функции F_i зависят от значений обобщенной собственной функции элемента $f_i(z_1, \ldots, z_n)$, а также от соотношения уровней сигнала,

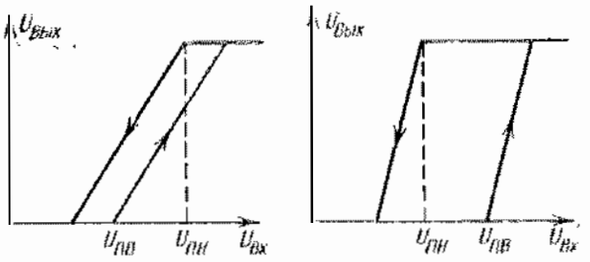


Рис. 9.5. Идеализированные вход-выходные характеристики логических элементов с разными крутизной характеристик и шириной гистерезиса

представляющего $\varnothing$, и сигналов представляющих 0 и 1. Пусть $U_{пн}$, $U_{пв}$, $U_{\varnothing}$ обозначают уровни порогов элемента при переключении в логический 0, 1 и уровень $\varnothing$ соответственно.

Анализ вход-выходных характеристик логических элементов (рис. 9.5) показывает, что в зависимости от крутизны характеристик и ширины гистерезиса возможны

лодующие варианты:

1) $U_{\varnothing} \leqslant U_{пн} \leqslant U_{пв}$, 2) $U_{пн} \leqslant U_{\varnothing} \leqslant U_{пв}$,
3) $U_{пн} \leqslant U_{пв} \leqslant U_{\varnothing}$, 4) $U_{\varnothing} \leqslant U_{пв} \leqslant U_{пн}$,
5) $U_{пв} \leqslant U_{\varnothing} \leqslant U_{пн}$, 6) $U_{пв} \leqslant U_{пн} \leqslant U_{\varnothing}$,
7) уровень $U_{\varnothing}$ не фиксирован («плывет») и в отдельные моменты может удовлетворять различным соотношениям.

Этим случаям соответствуют четыре различных типа Т-функций, которые могут быть заданы троичными формулами (9.1):

$$
\begin{aligned}
F_{1i}(Z) &= \bar{z}_i^0 f_i(Z) \vee \varnothing f_i(Z) \vee \varnothing z_i^1,\\
F_{2i}(Z) &= \bar{z}_i^0 f_i(Z) \vee \varnothing \bar{f}_i^0(Z) \vee \varnothing z_i^1,\\
F_{3i}(Z) &= \bar{z}_i^0 f_i(Z) \vee z_i^1 f_i^{\varnothing}(Z) \vee \varnothing \bar{f}_i^0(Z) \vee \varnothing z_i^1,\\
F_{4i}(Z) &= z_i^0 f_i(Z) \vee z_i^1 f_i^{\varnothing}(Z) \vee \varnothing f_i^1(Z) \vee \varnothing z_i^1.
\end{aligned}
\qquad (9.1)
$$

Заметим, что во всех вариантах, если $z_i = f_i(r) = \varnothing$, то $F_i(Z) = \varnothing$, т. е. переменная z_i в состоянии $\varnothing$ не возбуждена. Это объясняется тем, что уровню $\varnothing$ соответствует коэффициент усиления логического элемента, равный 1. Троичная схема, задаваемая уравнениями $z_i = F_{ji}(Z)$, $j \in \{1, 2, 3, 4\}$, где для разных переменных z_i тип функции (j) может, вообще говоря, различаться, обладают следующими особенностями. Если переменная z_i была устойчива в схеме в состоянии α, то она устойчива в этом состоянии и в ТС. Переменная z_i, возбужденная в состоянии α в исходной схеме, остается возбужденной в этом состоянии и в ТС, но при этом $F_i(\alpha) = \varnothing$, так как переключение элементов 0-1 и 1-0 происходит через $\varnothing$. Если переменная z_i несамозависима в исходной схеме, то после переключения в $\varnothing$ она будет снова возбуждена, продолжая переключение в 0 или в 1. Наконец, если $f_i(\alpha) = \varnothing$, то z_i в зависимости от ее значения в этом состоянии и от соотношения уровней $U_{пн}$, $U_{пв}$, $U_{\varnothing}$ может быть возбуждена или стабильна, что определяется типом функции F_i.

Задаваемые таким образом троичные функции F_{1i}, F_{2i}, F_{3i}, F_{4i} будем называть *нормальными расширениями булевой функции* $f_i(Z)$ типа 1)—4) соответственно.

Свойство 9.5. *Пусть* $f_i(Z)$ — *булева функция, а* $F_i(Z)$ — *ее нормальное расширение любого типа, тогда*:

1) *Если* $f_i(\alpha)=0$ *и* $\alpha'|z_i\neq 1$, *то* $F_i(\alpha')\leqslant\varnothing$, *где троичный вектор* α' *получен из* α *заменой некоторых двоичных значений на* $\varnothing$.

2) *Если* $f_i(\alpha)=1$ *и* $\alpha'|z_i\neq 0$, *то* $F_i(\alpha')\geqslant\varnothing$.

3) *Если* $f_i(\alpha)=0$, $f_i(\beta)=1$, $\alpha|z_i\neq\beta|z_i$, *то* $F_i(\alpha/\beta)=\varnothing$, *где* $\alpha|\beta$ *— троичный набор, i-й разряд которого* $(\alpha|\beta)_i$ *определяется следующим образом:*

$$(\alpha\,|\,\beta)_i=\begin{cases}\alpha_i, & \text{если } \alpha_i=\beta_i,\\ \varnothing, & \text{если } \alpha_i\neq\beta_i.\end{cases}$$

Определение 9.4. Троичную схему C^T назовем *троичным представлением* схемы C, если $|Z^T|=|Z|$, $|Z_0^T|=|Z_0|$, а каждая Т-функция F_i является нормальным расширением (произвольного типа) соответствующей булевой функции f_i.

Определение 9.5. Троичная схема *обладает метастабильной аномалией* по множеству переменных Z', если при некоторой двоичной фиксации входных переменных найдется метастабильное (под)состояние γ, достижимое из некоторого состояния α и в γ хотя бы одна переменная $z_i\in Z'$ метастабильна.

Теорема 9.2. *Пусть* C *— несамозависимая или антитонно-самозависимая схема, покрывающая некоторую* A_k*-схему и удовлетворяющая условию устойчивости ответа (по определению 9.1). Тогда ее троичное представление* C^T *обладает метастабильной аномалией по множеству переменных ответа.*

Доказательство. Его достаточно провести для A_2-схем.

Рассмотрим состояния $\alpha_2=a_1a_2b_1\bar b_2\beta$ и $\alpha_3=a_1a_2\bar b_1b_2\beta$ схемы C, где β — произвольный вектор значений остальных переменных схемы.

Так как выполнено условие устойчивости ответа, то $f_{b_1}(\alpha_2)=f_{b_2}(\alpha_3)=1$ и $f_{b_1}(\alpha_3)=f_{b_2}(\alpha_2)=0$ независимо от значения β. В троичном представлении схемы по свойству 9.5 (3) выполняется $F_{b_1}(\alpha_2|\alpha_3)=F_{b_1}(a_1a_2\varnothing\varnothing\beta)=F_{b_2}(\alpha_2\,|\,\alpha_3)=F_{b_2}(a_1a_2\varnothing\varnothing\beta)$ и вне зависимости от типа расширения и от значений переменных из β. Следовательно, подсостояние, задаваемое равенствами $a_1=a_2=1$, $b_1=b_2=\varnothing$, метастабильно. Покажем, что оно достижимо из двоичных состояний, используя доказательство теоремы 9.1. Рассмотрим состояния $\alpha_1=a_1a_2\bar b_1\bar b_2\beta$ и $\alpha_4=a_1a_2b_1b_2\beta$. Переменные b_1, b_2 возбуждены в α_1 и α_4. Следовательно, в троичном представлении из α_2 и α_4 достижимо подсостояние $a_1a_2b_1^{\varnothing}b_2^{\varnothing}$.

Из этой теоремы следует, что всякая реализация арбитра средствами двоичной логической схемотехники обладает метастабильной аномалией по переменным ответа.

Ниже приводится свойство, на котором основан метод поиска метастабильных (под)состояний.

Определим *функцию возбуждения переменной* z_i как

$$\varphi_i(Z)=z_i^0\overline{F_i^0(Z)}\vee z_i^{\varnothing}\overline{F_i^{\varnothing}(Z)}\vee z_i^1\overline{F_i^1(Z)},$$

где $F_i(Z)$ — собственная Т-функция переменной z_i в троичной схеме. Очевидно, что если $\varphi_i(\alpha)=0$ и $\alpha|z_i=\varnothing$, то z_i в состоянии α метастабильна, если $\varphi_i(\alpha)=0$ и $\alpha|z_i\neq\varnothing$, то z_i в α стабильна, а если $\varphi_i(\alpha)=1$, то z_i в α возбуждена.

Свойство 9.6. *Пусть* C^T *— троичная схема,* γ *— (под)состояние, соответствующее значениям переменных из* $Y'\subset Z$. *При этом* γ *метастабильно тогда и только тогда, когда* $\bigvee\limits_{z_i\in Z'}\varphi_i(\gamma)\equiv 0$, *где* $\varphi_i(\gamma)$ *— значение функции возбуждения переменной* z_i *в подсостоянии* γ.

Пример 9.1. Простейшей несамозависимой реализацией двухвходового арбитра (с инверсными входными переменными запроса) может служить следующая асинхронная логическая схема:

$$b_1=\overline{\bar a_1\vee b_2},\quad b_2=\overline{\bar a_2\vee b_1}.$$

Построив фрагмент диаграммы переходов для $a_1=a_2=1$, нетрудно убедиться в наличии колебательной аномалии $a_1a_2b_1b_2 — a_1a_2\bar b_1\bar b_2$ по переменным ответа.

Анализ двоичного представления этой схемы показывает, что состояние, задаваемое равенствами $a_1=a_2=1$, $b_1=b_2=\varnothing$, является метастабильным относительно переменных ответа и достижимо из двоичных состояний $a_1a_2b_1b_2$, $a_1a_2\bar b_1\bar b_2$ вне зависимости от типа расширения.

§ 9.4. Построение работоспособных арбитров

Для построения работоспособных арбитров можно воспользоваться элементами, поведение которых не может быть адекватно выражено средствами двоичной логики, но которые возможно на модельном уровне описывать БТ-функциями.

Возможны два типа работоспособных арбитров: *ограниченный* и *неограниченный*. Арбитры первого типа имеют ограниченное время ответа на запросы процессов вне зависимости от того, приходят ли они на входы арбитра в одиночестве или одновременно с другими запросами. Точнее, если на входы свободного в некоторый момент

времени арбитра приходит один или несколько запросов, то через конечное время, ограниченное некоторой (линейной) функцией от величин собственных задержек элементов, на выходе арбитра появится ответ на один из запросов. Возможный, казалось бы, подход к построению ограниченного арбитра основан на исключении аномалий посредством использования элементов, которые в переходных режимах ведут себя не так, как двоичные логические элементы; при этом не могут быть исключены колебательные аномалии, ибо они не зависят от характера переходного процесса. Исключение метастабильных аномалий возможно посредством применения *прямоугольного гистерезисного элемента* (ПГЭ), собственную функцию которого можно описать БТ-функцией $f(Z) = x^1 \vee x^{\varnothing} z^1$, где x — входная, а z — выходная переменная элемента.

На рис. 9.6, *а* приведена схема входного каскада арбитра с ПГЭ, по которой, используя методы предыдущего параграфа, можно убедиться в отсутствии метастабильных аномалий.

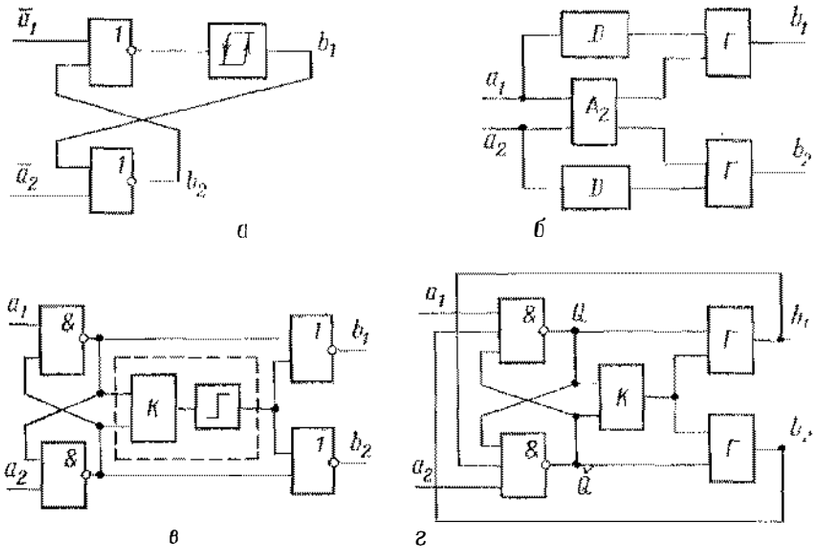


Рис. 9.6. Реализация арбитров: содержащая элемент с прямоугольной петлей гистерезиса (*а*), с Г-триггерами и задержками (*б*), с компараторами (*в*), с компараторами и Г-триггерами (*г*)

В этой схеме, однако, возможно возникновение колебательных аномалий. Кроме того, так как идеальный ПГЭ нереализуем, имеется вероятность возникновения метастабильного состояния.

Анализ переходных характеристик схемы рис. 9.6, *а* с *триггером Шмидта* вместо ПГЭ показывает наличие метастабильного состояния.

Арбитры второго типа имеют неопределенное время ответа на одновременные или близкие по времени запросы, т. е. время ответа такого арбитра не определяется величинами задержек элементов.

Суть подхода к построению «неограниченного» арбитра состоит в том, что сигналы на выходах арбитра блокируются на время пребывания схемы в аномальном состоянии и начинают изменяться после того, как устройство вышло из аномального режима. Корректность подобных арбитров объясняется тем, что в силу физико-статистических закономерностей схемы выходят из аномального состояния через, возможно, весьма длительный, но конечный и ограниченный сверху период времени.

Указанная блокировка может осуществляться встроенными задержками (рис. 9.6, *б*). Чтобы обеспечить достаточно малую вероятность того, что аномалия «проскочит» на выходы ответа b_1, b_2, длительность задержек D необходимо выбирать в десятки, а то и в сотни раз больше, чем длительность нормального переходного процесса в схеме, что в соответствующее число раз снижает быстродействие арбитра. Отметим, что для корректной работы арбитра на выходах схемы необходимо поставить Г-триггеры, а не элементы И.

Другой способ блокировки основан на применении *компаратора* с порогом, который используется как индикатор окончания аномального поведения арбитра. Такая индикация возможна в силу того, что при колебательной аномалии сигналы на выходах арбитра изменяются синфазно (см. рис. 9.3, *б*) и, как и в метастабильном состоянии, совпадают между собой в каждый момент времени с определенной точностью. Порог компаратора выбирается таким образом, чтобы при расхождении сигналов на выходе арбитра более чем на величину порога гарантировался выход из аномального состояния.

Схема входного каскада такого арбитра приведена на рис. 9.6, *в*. При таком решении, в отличие от предыдущего, переходный процесс затягивается лишь при условии аномального поведения входного триггера арбитра. На рис. 9.6, *в* компаратор и пороговый элемент для наглядности показаны отдельно, однако их можно выполнить как единый элемент. Недостатком приведенного на рис. 9.6, *в* решения является то, что переменная, моделирующая поведение компаратора, неполумодулярна, т. е. на выходе компаратора возможно появление коротких импульсов в случае

реальной (неидеальной) инерциальной задержки компаратора. Эти импульсы могут пройти на выходы b_1, b_2 схемы, и для получения корректного решения их необходимо фильтровать. Следствием указанного недостатка является также то, что схема не является самопроверяемой относительно неисправности типа «константный 0» на выходе компаратора. При такой неисправности компаратор перестает выполнять блокирующие функции, и аномальное поведение будет иметь место на выходах b_1, b_2.

Указанных недостатков можно избежать, если в схеме рис. 9.6,*в* заменить выходные вентили на Г-триггеры, с выходов которых завести перекрестные обратные связи на элементы входного триггера (рис. 9.6, *г*). Более изящное решение позволяет получить идея разнесения компаратора на две симметричные подсхемы, каждая из которых может быть выполнена на одном транзисторе.

На основании приведенных базовых арбитражных схем, которые являются апериодическими в том смысле, что их работоспособность не зависит от соотношений величин задержек элементов, могут быть построены арбитры с произвольными входными дисциплинами и многоканальные арбитры, т. е. арбитры, обслуживающие более двух процессов.

Многоканальные арбитры можно разделить на *централизованные* и *децентрализованные* (распределенные).

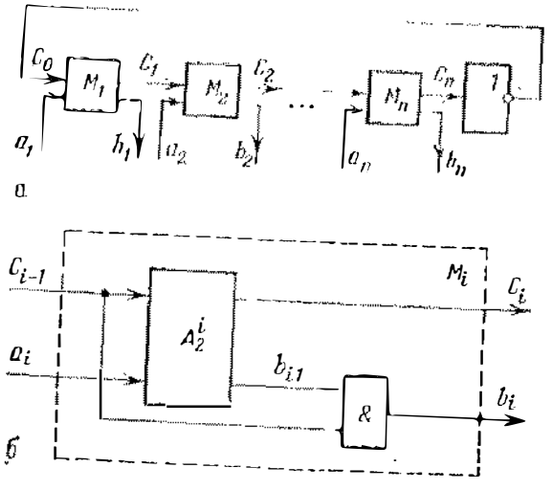


Рис. 9.7. Распределенный арбитр: циклическая структура (*а*), реализация модуля (*б*)

Первые могут выполняться в виде отдельного модуля, ибо между отдельными его частями имеется большое число соединительных проводов, которые не являются

общими для всех частей. Распределенные арбитры строятся на основе передачи информации между модулями по общим шинам, число которых относительно невелико. Централизованные арбитры могут строиться на основе древовидных или других рекурсивных структур (например, структуры рис. 9.4) или на основе многостабильных триггеров. В основе распределенных арбитров, как правило, лежат циклические структуры, простейшая из которых приведена на рис. 9.7.

Схема рис. 9.7, *а* функционирует следующим образом. В начальный момент времени $c_0 = 1$, $c_1 = \ldots = c_n = 0$. При этом арбитражные функции выполняет первый модуль M_1, который выбирает между системным запросом c_0 и внешним запросом a_1 (в случае наличия последнего). Если выбран системный запрос, то устанавливается значение $c_1 = 1$, в противном случае установится внешний ответ $b_1 = 1$, и только после сброса внешнего запроса $a_1 = 0$ установится значение $c_1 = 1$. В результате продвижения системного запроса в работу включается следующий модуль M_2. После отработки последнего модуля M_n установится значение $c_n = 1$ и затем сбросится сигнал c_0 ($c_0 = 0$). В результате по циклу пойдет волна гашений модулей, которая приведет к сбросу всех системных запросов ($c_0 = c_1 = \ldots = c_n = 0$) и к возврату в исходное состояние посредством установки первого системного запроса $c_0 = 1$. В таком арбитре процесс арбитража внешних запросов и процесс продвижения системного запроса протекают параллельно в том смысле, что при значениях $c_{i-1} = 0$ и $a_i = 1$ арбитр i-го модуля (рис. 9.7, *б*) захватывает внешний запрос (установка значения $b_{i1} = 1$) и выдает внешний ответ (установка значения $b_i = 1$) сразу вслед за появлением соответствующего системного запроса (т. е. установки значения $c_{i-1} = 1$), не тратя времени на процесс арбитража. В качестве A_2^i могут быть использованы, например, арбитры типа приведенных на рис. 9.6.

Для организации процесса самодиагностики арбитра может применяться схема с индикацией. Установка определенного значения на выходе индикатора должна блокировать процесс обработки того запроса, который в данный момент не обслуживается, и не затрагивает процесса обработки обслуживаемого запроса.

Схема самодиагностируемого арбитра приведена на рис. 9.8. На шинах A и B осуществляется сборка по ИЛИ всех внешних запросов и ответов соответственно. Общий сигнал индикации I вырабатывается инвертированием сигнала на шине B. Схема модуля M_i имеет вид рис. 9.8, *б*. Константная неисправность на выходе какого-либо элемента модуля приведет к отсутствию изменений сигнала на входе индикатора, причем в случае, если неисправность возникла в элементах того плеча модуля, которое обслуживает системный запрос, к этому приводит отсутствие изменений сигнала на системной шине.

Остановимся теперь на способах построения устройств, распределяющих общий ресурс, составной частью которых является арбитр. Общение процесса с общим ресурсом можно расчленить на две части: 1) получение доступа к ресурсу через систему арбитража и 2) собственно работа с ресурсом. Возможны две стратегии организации указанного общения. При первой на систему

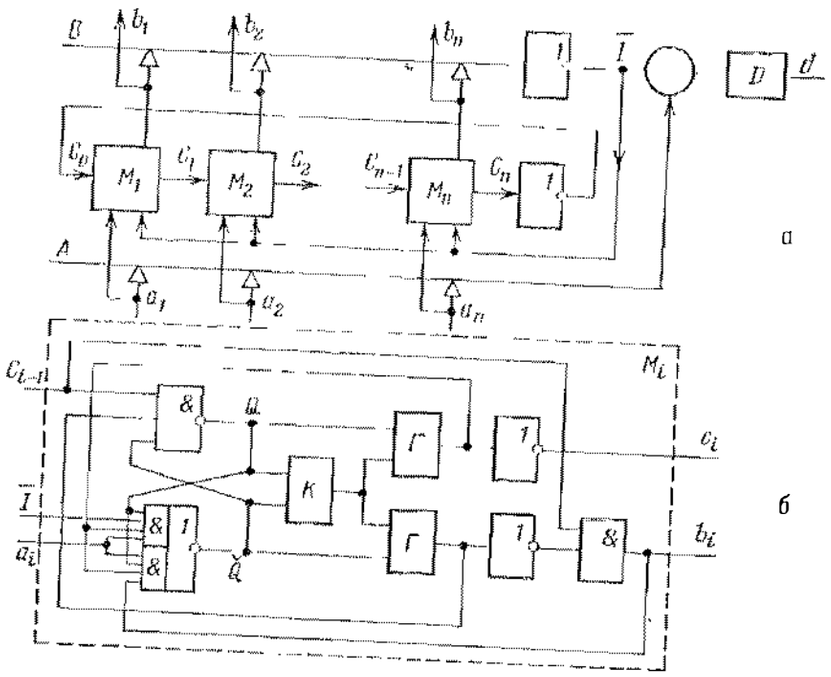


Рис. 9.8. Самодиагностирующийся арбитр: структура (*а*), реализация модуля (*б*)

арбитража возлагаются только функции выдачи разрешения процессу на работу с общим ресурсом, а обе фазы работы апериодического ресурса (рабочую и гашения) организует сам запрашивающий ресурс процесс (рис. 9.9, *а*). В этом случае автомат распределения ресурса представляет собой собственно многоканальный арбитр. Достоинством такого подхода является относительная простота автомата распределения ресурса, а недостатком — малое совмещение во времени процесса арбитража и функционирования ресурса. При второй стратегии устройство распределения ресурса выполняет

функции арбитража запросов и само организует общение процесса и ресурса (рис. 9.9, *б*). При этом каждый из процессов $P_1, \ldots, P_n$ с точки зрения управления имеет выход к ресурсу только через устройство распределения ресурса, ядром которого является арбитр.

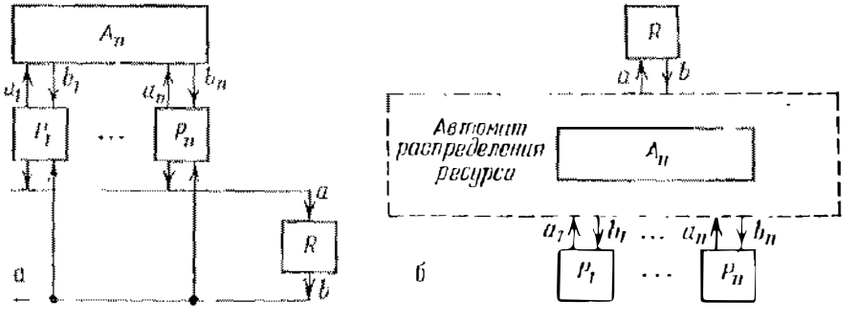


Рис. 9.9. Распределение ресурса: доступ через систему арбитража (*а*), доступ через автомат распределения ресурса с арбитром (*б*)

В гл. 5 было описано устройство распределения ресурса, названное автоматом повторного вхождения, для случая последовательных запросов, когда два запроса не могут прийти на ресурс одновременно (рис. 5.10). На рис. 9.10, *а* изображен сигнальный граф, отражающий порядок изменений внешних сигналов автомата повторного вхождения, обслуживающего i-й запрос. После прихода запроса a_i^+ схема инициирует рабочую фазу ресурса R^+, а затем фазу гашения ресурса R^-, и только после этого выдает ответ b_i^+. Сброс запроса a_i^- вызывает сброс ответа b_i^- и не затрагивает ресурса. К моменту сброса ресурса нельзя сбрасывать информацию, которую он выработал. Задача сохранения информации может быть решена схемами коммутации, которые вставляются в разрыв проводов α_i и выполняют функции управления буферными регистрами, на которых сохраняется информация. Обслуживание очередного запроса возможно при такой организации лишь после освобождения ресурса.

Схема повторного вхождения рис. 9.10, *б*, реализующая сигнальный граф рис. 9.10, *в*, позволяет распараллеливать фазу гашения ресурса и обработку ответа b_i внутри запрашивающего процесса. Это возможно за счет введения в схему обслуживания i-го запроса внутренней памяти r_i, на которой запоминается тот факт, что ресурс после обслуживания i-го запроса был сброшен. Обслуживание очередного запроса на ресурсе может начаться лишь после того, как запомнился факт сброса ресурса после обслуживания предыдущего запроса. Носителем этой информации в схеме выступает сигнал $\beta_i = 0$.

Для построения автомата распределения ресурса, способного обслуживать параллельно приходящие запросы, необходимо состыковать схему n-канального арбитра A_n с n-канальной схемой повторного вхождения S_n. Пример такой стыковки приведен на

рис. 9.11, *а*. Заметим, что конъюнктивное объединение запроса a_i и сигнала d_i может осуществляться непосредственно на элементах арбитра A_n. В случае использования в качестве S_n схемы рис. 5.10 порядок смены сигналов может быть описан сигнальным графом рис. 9.11, *б*. При этом если в качестве A_n использовать распределенный арбитр, то и вся схема устройства является распределенной, так как между отдельными ячейками схемы повторного вхождения рис. 5.10 нет связей. Если в качестве схемы многократного

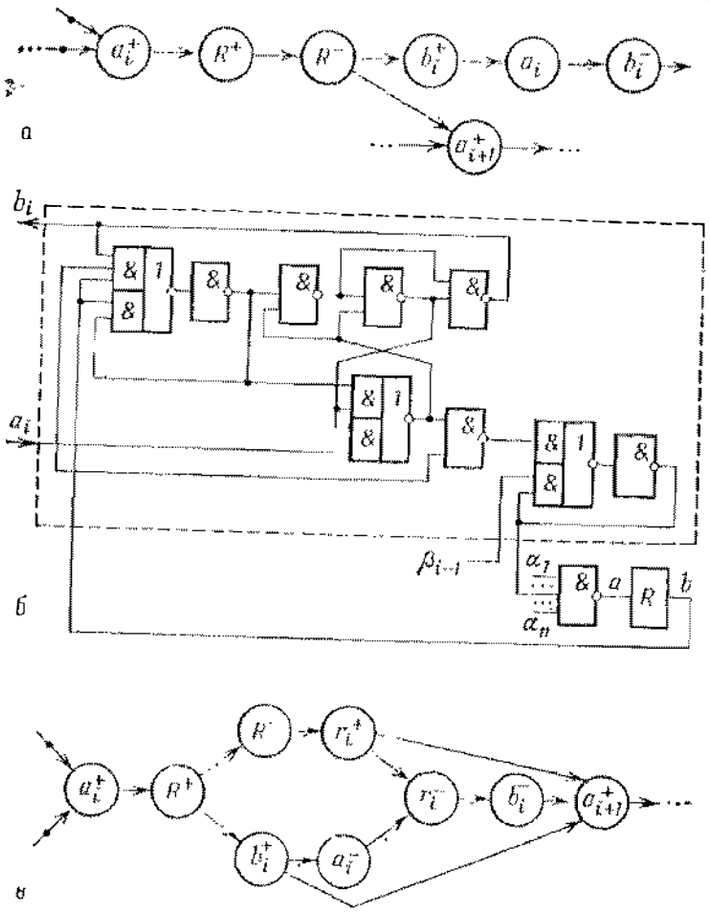


Рис. 9.10. Организация распределения ресурсов: сигнальный граф, задающий порядок срабатывания внешних сигналов при использовании автомата повторного вхождения (*а*), вариант автомата повторного вхождения (*б*), сигнальный граф последнего (*в*)

вхождения использовать схему рис. 9.10, *б*, то реализуется сигнальный граф рис. 9.11, *в*. Однако для выполнения такого устройства в распределенном виде необходимо решить вопрос связи между

ячейками схемы повторного вхождения. Один из возможных вариантов заключается в том, что по сигналам β_i на шине посред-

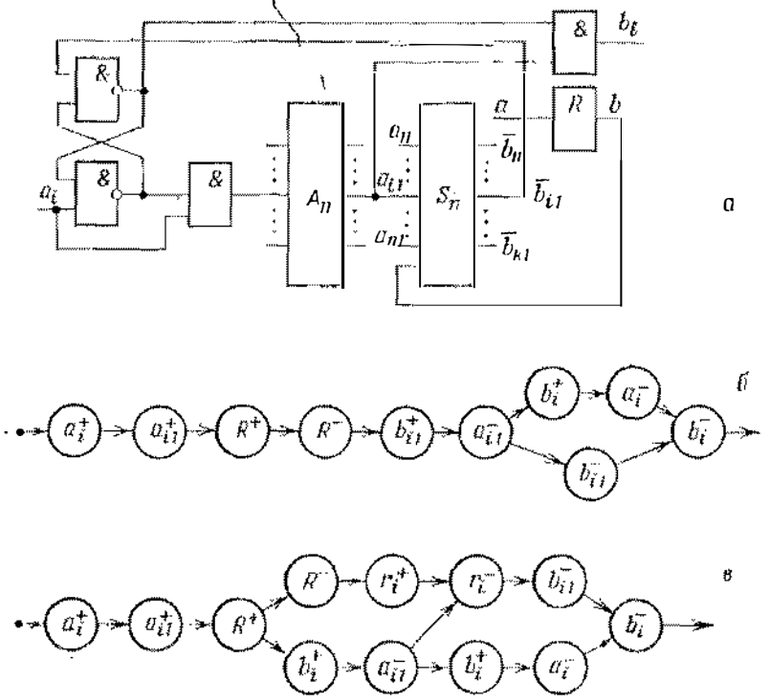


Рис. 9.11. Распределение ресурса для обслуживания параллельных запросов: схема автомата (*а*), его сигнальный граф при использовании схемы рис. 5.10, *б*, сигнальный граф при использовании схемы рис. 9.10, *в*

ством проводного ИЛИ формируется общий сигнал β, который и подается на входы β_{i-1} каждой i-й ячейки.

§ 9.5. «Ограниченные» арбитры и безопасные инерциальные задержки

Здесь мы остановимся на сложностях корректного решения проблемы построения «ограниченного» арбитра и на связи этой проблемы с проблемой построения безопасной инерциальной задержки.

Элементы задержки, сдвигающие во времени свой выходной сигнал относительно входного, принято разделять на *чистые* (совершенные) и *инерциальные* задержки. Чистая задержка осуществляет временной сдвиг произвольного входного сигнала, не изменяя его формы, т. е. преобразует входной сигнал $f(t)$ в выходной $f(t-D)$,

где D — величина задержки. Поскольку каждое физическое устройство не реагирует на входные воздействия, находящиеся за пределами его порога чувствительности, чистая задержка не может быть в точности воспроизведена физически. Приближенно ее поведение моделируется линией передач.

Поведение идеальной инерциальной задержки иллюстрируется рис. 9.12. Сигналы на входе Y, длительность которых меньше D,

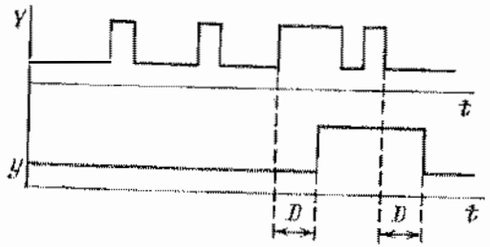


Рис. 9.12. Поведение идеальной инерциальной задержки

«не пропускаются» на выход y (фильтруются), а сигналы, длительность которых превышает D, появляются на выходе со сдвигом на время D. Приближением идеальной инерциальной задержки может служить схема, приведенная на рис. 9.13, построенная на основе RC-фильтров и мажоритарного элемента M. Для сколь угодно малого $E > 0$ идеальная инерциальная задержка должна фильтровать сигналы длительностью $D - E$ и пропускать сигналы длительностью $D + E$ вне зависимости от длительности последующего сигнала, которая может быть сколь угодно малой. Однако в

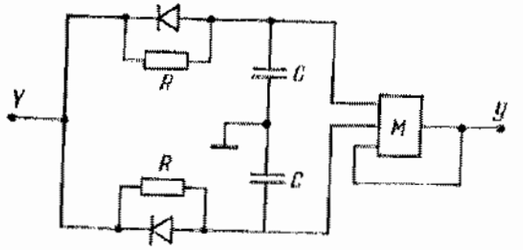


Рис. 9.13. «Реализация» идеальной инерциальной задержки

силу высказанных соображений о пороге чувствительности такая модель не поддается точной физической реализации. Иными словами, всякая инерциальная задержка обладает некоторой «зоной неопределенности» $[D - E, D + E]$, и нельзя заранее сказать, как она отреагирует на сигнал, длительность которого находится в пределах этой зоны.

На рис. 9.14 приведена известная схема Фридмена, моделирующая поведение идеальной инерциальной задержки с помощью

...той задержки D и мажоритарного элемента M с нулевой задержкой. Входные импульсы (нулевой, если $y = 1$, и единичный, если $y = 0$) длительностью меньше D фильтруются при условии, что период между соседними импульсами не меньше D. Следовательно, на поведение инерциальной задержки оказывает воздействие дисциплина изменения входного сигнала. Это объясняется

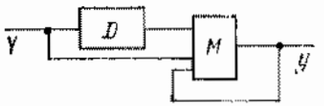


Рис. 9.14. Модель идеальной инерциальной задержки

...рционностью физических элементов, в силу которой схеме необходимо некоторое время для подготовки к фильтрации очередного короткого импульса (время релаксации).

Сказанное обусловливает правомерность введения другой модели инерциальной задержки, которая, обладая достаточной мере идеальными свойствами, будет учитывать наличие зоны неопределенности и входной дисциплины.

Введем следующие параметры инерциальных задержек.

1. *Входная дисциплина.* На вход инерциальной задержки подается последовательность чередующихся двоичных сигналов высокого и низкого логического уровня. Ее можно интерпретировать как последовательность чередующихся положительных и отрицательных импульсов (рис. 9.12) вида $\tau_0^+\tau_1^-\tau_2^+\tau_3^- \ldots$ или $\tau_0^-\tau_1^+\tau_2^-\tau_3^+ \ldots$, где τ_i^+, τ_j^- — длительность положительных и отрицательных импульсов. Входная дисциплина инерциальной задержки накладывает ограничения на допустимые соотношения длительностей следующих друг за другом импульсов. Так, для примера рис. 9.14 входная дисциплина задает следующее ограничение на входную последовательность: если $\tau_i < D$, то $\tau_{i+1} > D$.

2. *Пороги фильтрации.* Положительные вещественные числа D_m^+, D_m^- назовем порогами фильтрации, если положительные импульсы длиной $\tau^+ < D_m^+$ и отрицательные импульсы длиной $\tau^- < D_m^-$ не пропускаются на выход задержки.

Если на входе задержки после импульса $\tau_i^+(\tau_i^-)$ может появиться не только двоичный сигнал $\tau_{i+1}^-(\tau_{i+1}^+)$, но и сигнал $\tau_{i+1}^{\varnothing}$ некоторой промежуточной амплитуды,

то естественно считать, что если $\tau^{\varnothing}_{i+1} < D^-_m (\tau^{\varnothing}_{i+1} < D^+_m)$, то этот сигнал фильтруется. На обработку задержкой более длительных импульсов промежуточной амплитуды не накладываются никакие ограничения.

3. *Пороги пропускания.* Положительные вещественные числа D^+_M, D^-_M назовем порогами пропускания, если положительные импульсы длиной $\tau^+ > D^+_M$ и отрицательные импульсы длиной $\tau^- > D^-_M$ пропускаются на выход задержки без искажения или с допустимыми временными искажениями. Для оценки временного искажения может быть задано максимально возможное относительное (ε) или абсолютное (Δ) искажение пропускаемого сигнала во времени. Тогда, если входной импульс τ_i на выходе задержки превращается в импульс τ'_i, то должно выполняться условие $|\tau'_i - \tau_i| < \Delta$, или $|\tau'_i - \tau_i| < \varepsilon \cdot \tau_i$.

Интервалы $[D^+_m, D^+_M]$ и $[D^-_m, D^-_M]$ назовем зонами неопределенности для положительных и отрицательных импульсов соответственно.

4. *Величины задержек.* Это величины D^+ и D^-, на которые осуществляется сдвиг пропускаемого положительного или отрицательного сигнала. Очевидно, что $D^+ \geqslant D^+_m$ и $D^- \geqslant D^-_m$, так как задержка не может «принять решение» о том, пропускать ли импульс на выход до того, как «определит» в реальном масштабе времени, что $\tau^+ \geqslant D^+_m$ или $\tau^- \geqslant D^-_m$. Следовательно, изменение на выходе задержки может начаться не ранее чем через D^+_m (или D^-_m) после изменения на входе.

Отметим, что в общем случае параметры $D^+_m, D^-_m, D^+_M, D^-_M, D^+, D^-$ могут быть функциями времени и входных последовательностей.

Таким образом, инерциальная задержка обладает временными зонами трех типов: зонами фильтрации, зонами неопределенности и зонами пропускания. Если на ее вход подается последовательность импульсов, соответствующая входной дисциплине, то импульсы из зон фильтрации не пропускаются на выход, а импульсы из зон пропускания проходят на выход с задержкой, но сохраняя свою длительность (с требуемой точностью).

Определение 9.6. Инерциальную задержку будем называть *безопасной*, если каждый импульс из зоны неопределенности $(D^+_m \leqslant \tau^+ \leqslant D^+_M$ или $D^-_m \leqslant \tau^- \leqslant D^-_M)$,

подаваемый на ее вход в соответствии со входной дисциплиной, либо фильтруется, либо пропускается на выход с задержкой, но без искажения (с заданной точностью).

При подаче импульса из зоны неопределенности на вход небезопасной задержки на ее выходе может возникнуть сильно искаженный сигнал, в частности короткий паразитный импульс, появление которого, как уже говорилось, может привести к сбою и возникновению аномалий в поведении устройства.

Рассмотрим схему (рис. 9.15), состоящую из двух одинаковых каскадов, каждый из которых составляется задержкой и арбитром. У арбитров второй вход запроса и первый вход ответа являются инверсными, и в схеме используется только первый выход ответа.

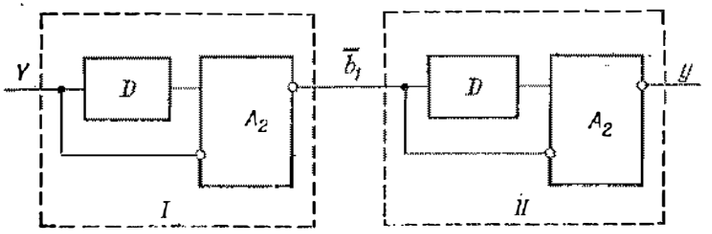


Рис. 9.15. Идеальная инерциальная задержка на основе арбитра

Для простоты будем считать, что задержки и арбитры первого и второго каскадов идентичны. Пусть d — чистая задержка, а A_2 — идеальный безынерциальный арбитр. Тогда нетрудно убедиться в том, что схема рис. 9.15 реализует идеальную инерциальную задержку, которая фильтрует короткие импульсы длительностью меньше d при условии, что период между соседними короткими импульсами больше d. Импульсы, длительность которых больше d, пропускаются без искажений с задержкой $2d$. Заметим, что положительные короткие импульсы ($\tau^+ < d$) фильтруются первым, а отрицательные ($\tau^- < d$) — вторым каскадами. Импульсы, длительность которых равна d, могут недетерминированным образом либо отфильтроваться, либо пройти на выход с задержкой $2d$.

В том случае, если в качестве задержки d используется небезопасная инерциальная задержка с порогами фильтрации $d^+_m = d^-_m = 0$, а в качестве A_2 — ограниченный арбитр, приведенная конструкция реализует безопасную инерциальную задержку, правда с другими параметрами.

Известно, что ограниченный арбитр может быть построен на основе идеальных инерциальных задержек. Для этого можно также использовать безопасные инерциальные задержки. Таким образом, проблемы синтеза

ограниченного арбитра и безопасной задержки *эквивалентны* в том смысле, что, разрешив одну из них, можно разрешить и другую [1]).

В заключение параграфа остановимся на трудностях, встающих при попытках построения ограниченного арбитра.

Идея построения арбитра с ограниченным временем ответа могла бы заключаться в следующем. Подобно схеме рис. 9.6, *в* необходимо специальными средствами индицировать аномальное поведение входного каскада, но при этом не просто блокировать выходы арбитра на все время аномалии, но еще и активно влиять на входы арбитра (блокируя один из запросов) с целью наискорейшего вывода арбитра из аномального состояния.

Рассмотрим схему рис. 9.16, *а*, которая отличается от схемы рис. 9.6, *в* лишь наличием дополнительного элемента И-ИЛИ-НЕ, блокирующего запрос a_2, двух небезопасных инерциальных задержек на выходах арбитра. Кроме того, запрос a_2 подается на арбитр в инверсной форме (хотя эта инверсия могла бы формироваться и внутри арбитра). Диаграмма переходов приведенной схемы содержит две области: область нормальной работы и область риска по переменным ответа. Последняя в свою очередь состоит из участка статического и динамического риска [2]) по переменным b_1', b_2', длительность которого ограничена длительностью переключения нескольких элементов схемы, и участка динамического риска по переменной b_2', длительность которого неограничена. Последнее объясняется тем, что длительность аномального поведения триггера (b_1^*, b_2^*) непредсказуема. Возможно, что в тот момент, когда произошла индикация аномального поведения и под воздействием обратной связи начинает сбрасываться второй запрос, триггер (b_1^*, b_2^*) выйдет из аномального состояния. Это может уловить компаратор, и второй запрос будет восстановлен. Триггер (b_1^*, b_2^*) может войти во второй цикл аномального поведения и т. д. Фрагмент диаграммы переходов, соответствующий описанному функционированию, приведен на рис. 9.16, *б* (значения переменных в состояниях пространственно расположены так же, как элементы на рис. 9.16, *а*). Из него видно, что наличие обратной

[1]) Проблема синтеза *ограниченного синхронизатора*, т. е. синхронизатора с ограниченным сверху временем переходных процессов также эквивалентна проблемам синтеза ограниченного арбитра и безопасной задержки.

[2]) Под статическим риском здесь понимается возможность появления одного, а под динамическим — нескольких подряд идущих коротких импульсов на выходах элементов b_1', b_2'. На участке нормальной работы переменные b_1', b_2' свободны от риска.

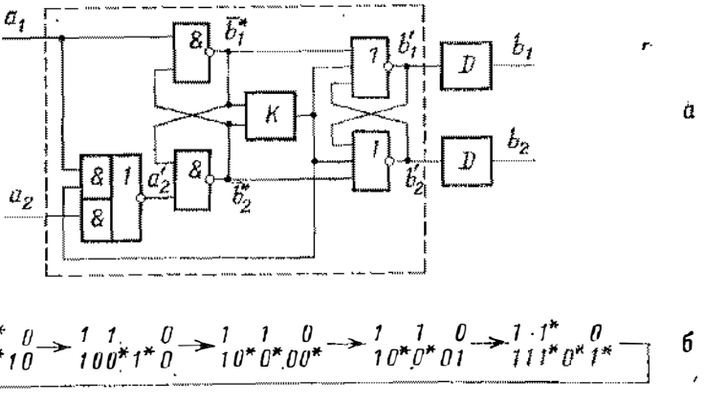


Рис. 9.16. Попытка реализации арбитра с ограниченным временем ответа: схема (*а*), диаграмма переходов (*б*)

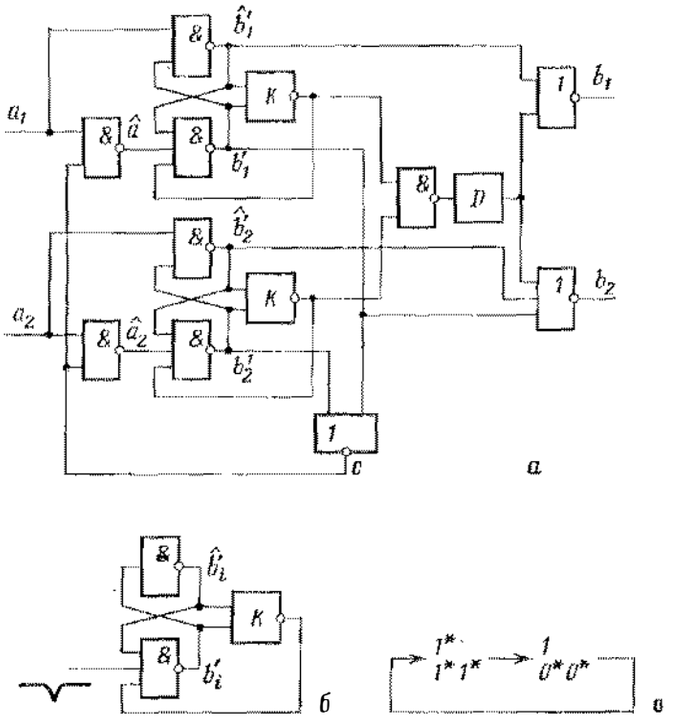


Рис. 9.17. Вторая попытка реализации арбитра с ограниченным временем ответа: схема (*а*), фрагмент схемы (*б*), диаграмма переходов, демонстрирующая возможность входа фрагмента в режим генерации (*в*)

связи с выхода индикатора аномалий делает возможной внутреннюю «подкачку» аномального поведения триггера (b_1^*, b_2^*), в результате которой схема может войти в режим генерации неопределенной длительности даже при фиксированных входах. В силу того, что арбитр работает в «запрос-ответном» режиме, риск ограниченной длительности может быть отфильтрован.

Однако наличие участка неограниченного риска по b_2' делает схему рис. 9.16, *а* некорректной.

Аналогичные трудности встают и при использовании другого метода компоновки арбитра с применением входного регистра хранения запросов и самосинхронизации (рис. 9.17, *а*). Если два запроса приходят на выходы a_1, a_2 с небольшим промежутком времени, то возможно, что второй из них (пусть a_1) придет на вход в тот момент, когда арбитр под воздействием первого (пусть a_1) отсекается от своих входов ($c = 0$). Триггер $(\hat{b}_1', b_1')$ под воздействием короткого отрицательного импульса на входе может войти в аномальное состояние. Если индикация аномального состояния компаратором произойдет в тот момент, когда триггер выйдет из аномального состояния в состояние (1, 0), то участок схемы (рис. 9.17, *б*) может войти в режим генерации (рис. 9.17, *в*) за счет обратной связи с выхода компаратора, и на выходе арбитра b_2 может возникнуть динамический риск неограниченной длительности.

Итак, приведен ряд аргументов в пользу невозможности активного влияния на выход схемы из аномального состояния. Эти практические аргументы подтверждаются теоретическими результатами. Поэтому можно считать, что изложенный в параграфе 9.4 подход к построению арбитров, работоспособность которых не зависит от задержек элементов, является единственно возможным.

§ 9.6. Замечания по библиографии

Первые упоминания о трудностях, встающих при решении проблем арбитража и синхронизации, появились, по-видимому, в 1966 г. [210, 256]. Большое количество последующих работ было посвящено сбору экспериментальных данных о поведении различных устройств (в основном триггеров) в аномальных режимах [214, 215, 216, 219, 250, 251, 282]. Статистическая обработка полученных данных, проведенная рядом авторов, показывает, что аномальное поведение арбитров и синхронизаторов является существенным источником сбоев вычислительных систем [219, 251, 262]. Так, в работе [251] оценивается вероятность сбоя синхронизатора асинхронных запросов на прерывания. При частоте 1 МГц и задержке вентиля 10 нс можно ожидать в среднем четырех сбоев системы в секунду. Очевидно, что с увеличением рабочей частоты системы и усложнением ее архитектуры (ростом числа синхронизирующихся сигналов) вероятность сбоя существенно возрастает.

Значительный экспериментальный материал позволил выработать рекомендации к архитектуре вычислительных систем, имеющие целью снижение вероятности сбоев из-за аномального поведения элементов систем [251, 297].

В работах [213, 216, 228, 251, 268, 280, 281, 290, 310] предлагаются разнообразные варианты построения схем арбитров и синхронизаторов, учитывающие возможность аномального поведения.

Схема арбитра с триггером Шмидта (рис. 9.6, *а*) предложена в [310]. В [212] показано, что такое решение не позволяет избежать аномальных режимов работы, с чем согласился автор схемы рис. 9.6, *а* [311]. В [290] сделана попытка построения ограниченного арбитра на основе схемного решения рис. 9.12, *а*. Однако, как показано в § 9.5, это решение некорректно. Описывались также методы построения многоканальных арбитров [217, 218, 242, 286], в частности с различными приоритетными дисциплинами [287].

Приведенное в данной главе доказательство неизбежности возникновения колебательных и метастабильных аномалий в двоичных логических схемах, реализующих арбитр, и рассмотрение методов построения работоспособных арбитров дано в [55] и основано на применении недвоичных логических элементов, описание которых может быть дано в рамках булево-троичной логики [6, 312].

В [127] доказывается невозможность реализации недетерминированных автоматов в классе самонезависимых схем. Однако понятие схемы, реализующей недетерминированный автомат [127], не вполне адекватно описывает арбитры и синхронизаторы.

Понятие безопасной инерциальной задержки из 9.5 близко понятию идеальной инерциальной задержки из [296]. Схемы инерциальных задержек (рис. 9.7) были предложены в [230] (см. также [5]). Анализ различных конструкций инерциальных задержек приведен в [259]. Схема рис. 9.10 предложена в [296]. В [207, 296] устанавливается связь проблемы построения идеальных арбитров и синхронизаторов с ограниченным временем ответа с проблемой построения идеальной инерциальной задержки.

В [258] показано, что для широкого класса динамических систем можно подобрать короткое входное воздействие, при подаче которого система войдет в аномальное состояние. Этот результат является весомым аргументом в пользу невозможности реализации ограниченных синхронизаторов и арбитров, а также безопасных инерциальных задержек.

Житель английского города Ньюарк Джеймс Кук вытащил из воды некую мисс Миллер, которая чуть было не утонула вместе со своей собачкой. Вскоре мистер Кук получил по почте денежный перевод. Это общество спасения утопающих наградило его «поощрительной премией» в 1 фунт стерлингов. А недавно Кук снова получил по почте чек, но уже на 10 фунтов. Это была награда от общества по охране животных за спасение собачки мисс Миллер.

По сообщениям информационных агентств

Глава 10. ДИАГНОСТИЧЕСКИЕ СВОЙСТВА АПЕРИОДИЧЕСКИХ СХЕМ И ОРГАНИЗАЦИЯ САМОРЕМОНТА

На ранних этапах разработки теории апериодических схем казалось, что их основные преимущества перед традиционными состоят главным образом в повышении быстродействия за счет организации функционирования по реальным задержкам элементов и в устойчивости к их нестабильности. Сейчас становится ясным, что *решающим достоинством апериодических схем являются их самодиагностические свойства*. Дело в том, что возможность фиксации моментов окончания переходных процессов попутно приводит к решению задачи диагностирования ряда неисправностей. Поскольку апериодическая схема правильно функционирует при любых конечных задержках элементов, то при возрастании задержки некоторого ее элемента до бесконечности схема не в состоянии сформировать сигнал индикации момента окончания переходного процесса. В свою очередь возрастание задержки элемента до бесконечности может интерпретироваться как дефект схемы, моделью которого являются *константные неисправности*, т. е. такие неисправности, при которых на выходе элемента фиксируется сигнал «постоянный 0» или «постоянная 1».

Тип неисправностей определяется рядом факторов. Во-первых, местом возникновения дефектов. Рассматриваемые в апериодической схемотехнике вопросы самодиагностики и саморемонта касаются лишь дефектов на выходах элементов и, вообще говоря, не затрагивают дефектов внутри элементов и на их входах. Во-вторых, поведением дефектного элемента. Мы рассматриваем

только константные неисправности любой направленности, для которых уравнение $z_i = f_i(Z)$, описывающее поведение элемента до возникновения дефекта, заменяется на $z_i = 0$ или $z_i = 1$. В-третьих, кратностью неисправностей. В-четвертых, моментом возникновения дефекта или, точнее, состоянием, в котором находится элемент, в тот момент, когда на его выходе возникает неисправность. По этому признаку константные неисправности делятся на консервативные и мутационные.

К *консервативным* отнесем такие неисправности вида $z_i = \sigma$, $\sigma = \{0, 1\}$, в момент возникновения которых элемент z_i имел на выходе значение сигнала, соответствующее либо его устойчивому состоянию σ, либо его возбужденному состоянию $\bar{\sigma}(z_i \neq f_i = \sigma)$. Сам момент возникновения такой неисправности остается незамеченным для схемы, ибо в момент своего возникновения такая неисправность никак не влияет на работу схемы, так как неисправное значение σ совпадает либо со значением выхода исправного элемента z_i, либо со значением собственной функции элемента f_i. Обнаружение такой неисправности в апериодической схеме произойдет позже — в тот момент, когда элемент z_i должен будет переключиться.

К *мутантным* отнесем неисправности следующего вида: на выходе элемента z_i в момент возникновения неисправности $z_i = \sigma$ установлено устойчивое значение $\bar{\sigma}(z_i = f_i \neq \neq \sigma)$. Такая неисправность вызывает непредвиденное переключение и, следовательно, оказывает воздействие на поведение схемы уже в момент своего возникновения. Мутантные неисправности могут явиться причиной появления коротких импульсов на выходе дефектного элемента, которые могут привести к аномальному поведению схемы.

Схемы, в которых в процессе функционирования обнаруживаются неисправности некоторых классов, в технической диагностике называют *полностью самопроверяемыми схемами*. Идея самопроверяемости базируется на таком разбиении входных и выходных наборов схемы на классы, при котором реакции на один и тот же класс входных наборов исправной схемы и схемы с неисправностями относятся к различным классам выходных наборов. Обычно самопроверяемые схемы снабжаются устройствами (также самопроверяемыми), способными распознать принадлежность выходных наборов тому или иному классу. Любая неисправность в схеме, снабженной такой

схемой встроенного контроля, обнаруживается на ее выходах.

Общепринятым является такой способ кодирования выходов *схемы встроенного контроля* (двухвыходной): при нормальной работе на ее выходах вырабатываются сигналы 01 или 10, при неисправности из заданного класса — 00 и 11. Недостатком его является наличие критических состязаний при переходах 01-10, из-за которых в процессе работы исправной схемы кратковременно появляются наборы 00 или 11, ложно свидетельствующие о неисправностях. При традиционном подходе этот недостаток едва ли может быть полностью преодолен.

Теория полностью самопроверяемых схем развита, по существу, лишь для класса синхронных схем, хотя применение термина «полностью самопроверяемые» к нему не совсем точно, поскольку неисправности системы синхронизации в них не диагностируются.

При синтезе асинхронных полностью самопроверяемых схем возникают существенные трудности, связанные с необходимостью обеспечения, помимо самопроверяемости, отсутствия критических состязаний в схеме. Попытки раздельного решения этих задач приводят к громоздким схемам.

Апериодические схемы, не нуждающиеся в синхронизации и свободные от критических состязаний, как будет показано ниже, также относятся к классу полностью самопроверяемых схем. Таким образом, изучение «структурных» свойств апериодических схем важно и с точки зрения решения проблем функциональной диагностики. Апериодические схемы являются полностью самопроверяемыми относительно консервативных неисправностей и именно на них ориентированы методы самодиагностики и саморемонта. Разработка общих методов самодиагностики, а тем более саморемонта, в случае любых мутантных неисправностей весьма затруднительна.

§ 10.1. Полностью самопроверяемые комбинационные схемы

Неудачные попытки построения полностью самопроверяемых асинхронных схем объясняются, видимо, тем, что их специфика требует ревизии известных определений защищенности от неисправностей, самотестируемости и полной самопроверяемости, а также более тонкой проработки вопросов, касающихся кодирования переменных и допустимых дисциплин переходов между наборами. Эти вопросы тесно связаны с явлением функциональных состязаний и с требованием отсутствия логических состязаний в асинхронных схемах.

Таким образом, между понятиями индицируемости (п. 4.3.1) и полной самопроверяемости должна существовать тесная связь. Очевидно, что полная самопроверяемость в асинхронных схемах возможна лишь при определенной дисциплине переходов входных наборов, например двухфазной дисциплине.

Сохраним для асинхронных полностью самопроверяемых схем понятие допустимого перехода (определение 3.2) и обозначения, принятые в § 4.3.

Будем считать, что неисправность p проявляется в том, что комбинационная схема реализует систему $F_p(X) \neq F(X)$.

Определение 10.1. Асинхронная комбинационная схема называется *полностью самопроверяемой* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого допустимого перехода a-b справедливо либо

$$F_p(b) = F(b), \tag{10.1}$$

либо

$$F_p(b) \notin L; \tag{10.2}$$

б) для любого допустимого перехода b-a справедливо либо

$$F_p(a) = F(a), \tag{10.3}$$

либо

$$F_p(a) \notin K \tag{10.4}$$

(условия а) и б) суть условия «защищенности от неисправностей»);

в) существует хотя бы один допустимый переход a-b или b-a, для которого выполняются условия (10.2) или (10.4) («самотестируемость»).

Как видно из определения 10.1, неисправность может выявляться не сразу «в момент возникновения», а только в следующей фазе работы схемы при двухфазной дисциплине, однако это нельзя считать нарушением «идеологии»

полной самопроверяемости, так как для таких схем «моментом» или «тактом» является полный «цикл», включающий в себя обе фазы дисциплины.

Определение 4.3 индицируемости позволяет формально подойти и к вопросам самопроверяемости комбинационных схем. Имеет место следующая

Теорема 10.1. *Индицируемые (апериодические) комбинационные схемы являются полностью самопроверяемыми для одиночных и кратных константных неисправностей элементов.*

Доказательство. Пусть в комбинационной схеме, входы X которой индицируются на выходах Y, имеется неисправность какого-либо элемента типа «постоянный 0».

а) Если при допустимом переходе a-b неисправный элемент не должен переключаться, то выполняется условие (10.1) определения 10.1. Если тот же элемент должен осуществить переход 0-1, то либо $F_p(b) = F(a)$, либо $F_p(b) = F(c)$, где $c \in [a, b]$, либо $F_p(b) = h$, $h \in [k, l]$ и по определению 4.3 $F(c) \notin L$, $F(a) \notin L$, $h \notin L$. Но тогда выполняется условие (10.2) определения 10.1.

б) Если неисправный элемент существен, найдется хотя бы один допустимый переход вида a-b или b-a, где этот элемент должен переключиться. Примем условие, что для неисправности «постоянный 0» это переход вида a-b. Тогда для любой неисправности этого типа найдется допустимый переход a-b такой, что $F_p(b) = F(a)$ или $F_p(b) = F(c)$, $F_p(b) = h$, $h \in [k, l]$, следовательно, $F_p(b) \notin L$, и выполняется п. а) определения 10.1, т. е. комбинационная схема является полностью самопроверяемой. Аналогичные рассуждения можно провести для случая неисправностей типа «постоянная 1» и кратных неисправностей.

Замечание. Если на входах схемы отсутствуют разветвления или неисправности возникают до разветвлений (как это обычно бывает при обрывах ножек корпусов микросхем), обнаруживаются и константные неисправности входов схемы.

§ 10.2. Полностью самопроверяемые автоматы

Для определения полностью самопроверяемых схем с памятью (автоматов) следует обобщить понятие индицируемости на этот класс схем. Это уже было сделано в § 4.8 (определение 4.7). Автомат Мили будем считать композицией автомата Мура и комбинационной схемы. Общим случаем задания автомата можно считать систему уравнений $F(X, Z)$.

При формулировке следует иметь в виду, что в случае неисправности $p \in P$ внутренний переход c-d (состояний автомата) не может завершиться, поэтому вместо символа d последним появится некоторый символ $\delta \in (c, d]$, а при переходе d-c — символ $\gamma \in (d, c]$. Таким образом, в случае неисправности запись $F(b, d)$ заменяется записью $F_p(b, \delta)$, а запись $F(a, c)$ — записью $F_p(a, \gamma)$.

Определение 10.2. Асинхронный автомат Мура будем называть *полностью самопроверяемым* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого перехода a-b справедливо либо

$$F_p(b, \delta) = F(b, d), \tag{10.5}$$

либо

$$F_p(b, \delta) \notin D; \tag{10.6}$$

б) для любого допустимого перехода b-a справедливо либо

$$F_p(a, \gamma) = F(a, c), \tag{10.7}$$

либо

$$F_p(a, \gamma) \notin C \tag{10.8}$$

(условия а) и б) суть условия «защищенности от неисправностей»);

в) существует хотя бы один допустимый переход a-b или b-a, для которого выполняются условия (10.6) или (10.8) — (самотестируемость).

Обратим внимание на то, что в автоматах, как и комбинационных схемах, неисправность может выявиться не сразу в момент возникновения, а только в следующей фазе.

Для апериодических автоматов по аналогии с теоремой 10.1 имеет место следующее утверждение.

Утверждение 10.1. *Апериодические автоматы являются полностью самопроверяемыми для одиночных и кратных константных неисправностей элементов.*

Доказательство непосредственно следует из сопоставления определений 4.8 апериодического автомата и 10.2 полностью самопроверяемого автомата.

Механизм нахождения неисправностей необходимо связан со способом кодирования входных и выходных переменных самосинхронизирующимися кодами (гл. 3). Исходя из расчетного быстродействия, можно установить некоторый промежуток времени, в течение которого схема должна завершить переход. Тогда отсутствие по истечении этого промежутка выходного набора, принадлежащего классу D (для перехода вида a-b) или C (для обратного

перехода вида b-a), можно интерпретировать как наличие неисправности в автомате.

Теорема 10.1 и утверждение 10.1 имеют принципиальное значение: устанавливается связь между проблематиками технической диагностики и теории апериодических автоматов. Непосредственным результатом установления этой взаимосвязи является возможность использования методов анализа схем (п. 4.3.1 и гл. 8) для проверки схем на принадлежность классу полностью самопроверяемых в смысле определений 10.1 и 10.2. Результаты, излагаемые в этой книге, позволяют разрешить ряд проблем теории полностью самопроверяемых схем, а также проблему недиагностируемости неисправностей в системе синхронизации. При традиционном подходе контролируемая схема проектируется без учета диагностических требований, а затем искусственно дополняется различными механизмами, обеспечивающими ее контролепригодность посредством разбиения входных и выходных наборов схемы на классы. Затем на схему «навешивается» тестор, который распознает принадлежность наборов тому или иному классу. В апериодической схемотехнике контролепригодность обеспечивается непосредственно при проектировании устройств, что позволяет упростить методы синтеза как самих полностью самопроверяемых схем, так и тесторов.

§ 10.3. Диагностирование автономных схем

Определенные в §§ 10.1, 10.2 понятия полностью самопроверяемых комбинационных схем и автоматов могут быть распространены на случай задания моделирующих схем с помощью модели Маллера.

Если в неавтономной схеме переходный процесс завершается некоторым тупиковым состоянием диаграммы переходов, то исправная автономная схема не попадает в тупики. Константная неисправность в схемах, заданных моделью Маллера, сводится к фиксации переменной, соответствующей неисправному элементу, в исходной системе уравнений.

Пример 10.1. Пусть схема описывается системой

$$z_1 = \bar{z}_3, \quad z_2 = \bar{z}_1, \quad z_3 = \bar{z}_2;$$

полная ее диаграмма переходов была приведена на рис. 8.4. При неисправности $z_1 = 0$ система приводится к следующей:

$$z_1 = 0, \quad z_2 = \bar{z}_1, \quad z_3 = \bar{z}_2,$$

а диаграмма — к виду, показанному на рис. 10.1. Эта неисправность никак не отразится на работе элементов z_2 и z_3, но поскольку состояние 010 является тупиковым, неисправность обнаружится после нескольких переключений элементов схемы. В данном случае переход 111-010 может потребовать 6 переключений.

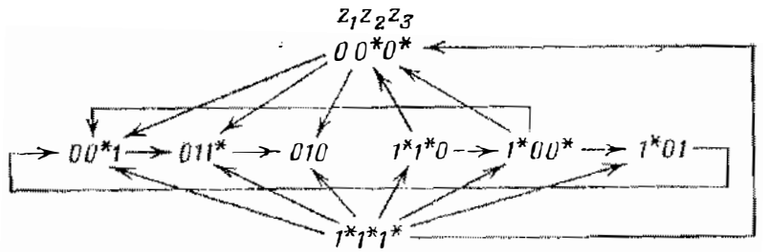


Рис. 10.1. Диаграмма переходов кольца из трех инверторов после константной неисправности одного из них

Таким образом, естественным проявлением неисправности представляется наличие тупиковых состояний в диаграмме переходов неисправной схемы.

Для определения понятия полностью самопроверяемой автономной схемы примем следующие обозначения:

$R(\alpha)$ — множество рабочих состояний, содержащее состояние α (иначе — *рабочий цикл с состоянием* α);

R — *множество бесконечных рабочих циклов схемы* (содержащее все состояния, принадлежащие хотя бы одному из входящих в него множеств рабочих состояний, каждое из которых является бесконечной последовательностью состояний);

T — *объединение множеств конечных* (содержащих тупики) *множеств рабочих состояний исправной и неисправной схем* (последнее образуется преобразованием диаграммы переходов исправной схемы по фиксированной неисправности).

Нижним индексом p по-прежнему будем помечать обозначение неисправной схемы. Для пары $R(\alpha)$, $R(\beta) \in R$ будем использовать запись $R(\alpha) \neq R(\beta)$, если эти циклы не совпадают.

Определение 10.3. Автономная схема с рабочими циклами из R называется *полностью самопроверяемой* для неисправностей класса P, если для всех $p \in P$ выполняются следующие условия:

а) для любого рабочего цикла $R(\alpha) \in R$ выполняется либо

$$R_p(\alpha) = R(\alpha), \tag{10.9}$$

либо

$$R_p(\alpha) \notin R \text{ и } R_p(\alpha) \in T \qquad (10.10)$$

(условие защищенности от неисправности);

б) существует хотя бы один рабочий цикл $R(\alpha) \in R$, для которого выполняется условие (10.10) (самотестируемость).

Таким образом, условие защищенности от неисправностей состоит в том, что либо рабочие циклы неисправной и исправной схем совпадают, либо рабочий цикл неисправной схемы не совпадает ни с одним бесконечным рабочим циклом исправной схемы и одновременно является конечным (содержит тупик).

Выясним, для каких классов неисправностей полумодулярные схемы являются полностью самопроверяемыми. Для этого сначала введем дополнительные понятия.

Определение 10.4. Константную неисправность p будем называть: 1) *выносной* для множества R, если при ней схема оказывается в состоянии ω, таком, что $\omega \notin T$, $\omega \notin R$; 2) *подменной* для множества R с рабочими циклами $R(\alpha)$ и $R(\beta)$, если найдется такое состояние $\gamma \in R_p(\alpha)$, что $\gamma \in R(\beta)$.

Выносная неисправность отличается от подменной тем, что выводит схему из той области, где она была исследована. При подменной неисправности диаграмма переходов может содержать ранее не учтенные бесконечные циклы. Таким образом, выносная неисправность потенциально может оказаться подменной.

Пример 10.2. Пусть поведение схемы описывается диаграммой

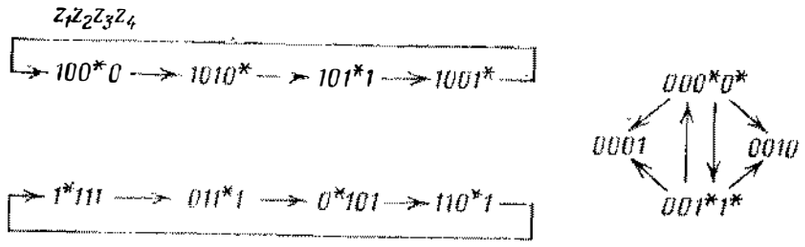


Рис. 10.2. К определению подменных и выносных неисправностей

переходов, представленной на рис. 10.2, которой соответствует система уравнений

$$z_1 = z_1\bar{z}_2 \vee z_2\bar{z}_3, \quad z_2 = z_2, \quad z_3 = z_1z_2 \vee \bar{z}_4,$$
$$z_4 = z_2 \vee z_1z_3 \vee \bar{z}_1\bar{z}_3.$$

При $z_2 = 1$ и $z_2 = 0$ эта система редуцируется до

$$z_1 = \bar{z}_3, \quad z_2 = 1, \quad z_3 = z_1 \vee \bar{z}_4, \; z_4 = 1$$

и

$$z_1 = z_1 \vee \bar{z}_3, \quad z_2 = 0, \quad z_3 = \bar{z}_4, \quad z_4 = z_1z_3 \vee \bar{z}_1\bar{z}_3$$

соответственно.

При $z_1 = z_2 = 0$ имеем $z_1 = 0$, $z_2 = 0$, $z_3 = \bar{z}_4$, $z_4 = \bar{z}_3$.

Для множества R рабочих циклов $R(1000)$ и $R(1111)$ обе константные неисправности элемента z_2 являются подменными, так как при $z_2 = 1$ схема из рабочего цикла $R(1000)$ попадает в рабочий цикл $R(1111)$, а не в тупик, а при $z_2 = 0$ имеет место обратное явление.

Неисправность $z_1 = z_2 = 0$ для этих рабочих циклов оказывается выносной, поскольку при такой неисправности схема выходит из состояний множества R, но не оказывается в тупике.

Очевидно, что для выносных неисправностей не выполняется условие $R_p(\alpha) \in T$, а для подменных — условие $R_p(\alpha) \notin R$ и $R_p(\alpha) \in T$. Таким образом, схемы с выносными и подменными неисправностями не диагностируются.

Можно показать, что в полумодулярной схеме, имеющей фиктивный класс эквивалентности (см. определение 8.3), всегда найдутся такие состояния α, β из $R(\alpha)$, $R(\beta) \in R$, достижимые из состояний φ, ψ, для которых имеет место $\varphi \Rightarrow \alpha$, $\psi \Rightarrow \beta$, причем α и β не принадлежат фиктивному классу. Фиктивный класс не является полной последовательностью. Отсюда вполне очевидно, что если часть состояний рабочего цикла $R(\alpha)$ образует фиктивный класс, в котором возбуждена и не переключается переменная z_s, то в соответствии с определением 10.3 схема с неисправностью $z_s = 1$ не будет полностью самопроверяемой. В самом деле, при этой неисправности состояния из фиктивного класса могут сменять друг друга, и схема не попадает в тупик. Ясно, что в этом случае неисправность не диагностируется. Поэтому далее в этом параграфе рассматриваются только полумодулярные схемы, не имеющие фиктивных классов.

Заметим, что присутствие фиктивных классов в диаграмме переходов обусловлено наличием в схеме элементов, входы которых изолированы от остальных элементов схемы — элементов, выходы которых замкнуты на свои входы, или являющихся генераторами 0 или 1. Рассмотрение такого рода схем нецелесообразно вследствие того, что они могут быть упрощены.

Пример 10.3. Схема $z_1 = 1$, $z_2 = \bar{z}_3$, $z_3 = z_2$ содержит генератор единиц, а ее диаграмма переходов, показанная на рис. 10.3, имеет фиктивный класс {000, 010, 011, 001}.

Имеет место следующая теорема.

Теорема 10.2. *Автономная схема с бесконечными рабочими циклами* $R(\alpha)$, $R(\beta)$, ..., $R(\lambda)$, *полумодулярная относительно* α, β, ..., λ, *является полностью самопроверяемой для невыносных и неподменных для* R *одиночных и*

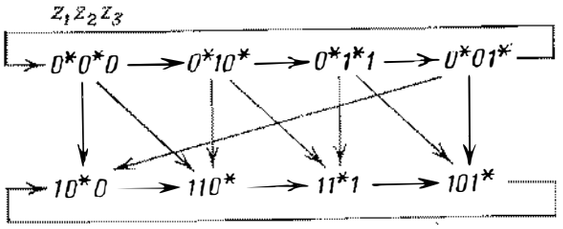


Рис. 10.3. Схема с фиктивным классом эквивалентности

кратных константных неисправностей элементов, если она не содержит фиктивных классов эквивалентности.

Доказательство. Если множество R содержит несколько рабочих циклов, в некоторых из них, например в $R(\alpha)$, могут найтись неживые элементы, т. е. элементы, которые не переключаются в этом рабочем цикле. Тогда константные неисправности таких элементов, совпадающие по значению с сигналами на выходах этих элементов, никак не проявляются в рабочем цикле $R(\alpha)$, и выполняется условие (10.8). Если же в рабочем цикле $R(\alpha)$ элемент был живым, неподменная и невыносная его неисправность

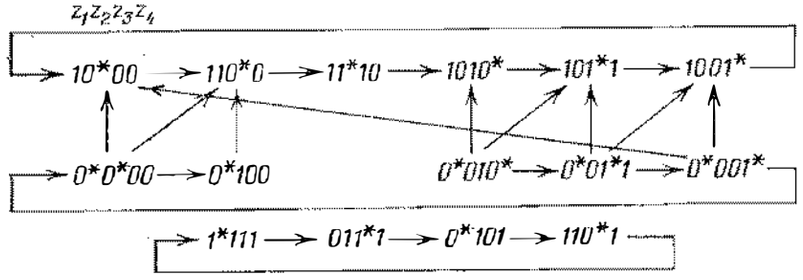


Рис. 10.4. Пример диаграммы переходов с двумя рабочими циклами

приведет к тому, что сигнал на выходе элемента станет постоянным. В силу того, что схема полумодулярна и множество ее рабочих состояний не содержит фиктивного класса, в нем найдется такое состояние ε, в котором возбужден только неисправный элемент. Но из-за неисправности этот элемент не может переключиться, и состояние ε окажется тупиковым, т. е. будет выполнено условие (10.9) — условие защищенности от неисправности.

Если в схеме нет неживого во всех рабочих циклах элемента (иначе он бесполезен и должен быть априори удален из схемы), выполняется условие б) определения 10.3 — самотестируемость. Значит, схема будет полностью самопроверяемой.

Пример 10.4. Рассмотрим схему, поведение которой описывается диаграммой переходов рис. 10.4. Ей, в частности, соответствует система уравнений

$$z_1 = \bar{z}_2 \vee \bar{z}_3 \vee \bar{z}_4,$$
$$z_2 = z_2z_4 \vee \bar{z}_3\bar{z}_4,$$
$$z_3 = z_1z_2 \vee z_3\bar{z}_4,$$
$$z_4 = z_2z_4 \vee \bar{z}_2z_3.$$

Пусть рабочими циклами являются R (1000) и R (1111). При неисправности $z_1 = 1$ состояние 1111 станет тупиковым (самотестируемость). Неисправность $z_1 = 0$ также выявляется в рабочем цикле R (1111): тупиковым оказывается состояние 0101. Неисправность $z_1 = z_2 = 0$ является для R (1000) и R (1111) выносной, хотя при задании еще одного цикла R (0000) она также диагностируется, так как схема, заданная вышеприведенной системой уравнений, при этом оказывается в тупиковом состоянии.

Необходимо отметить, что в доказательстве теоремы 10.2 существенно используется факт полумодулярности исправной схемы относительно состояний из множества R. Если же схема этим свойством не обладает, то константная невыносная и неподменная неисправность может и не выявляться.

Пример 10.5. Неисправность одного из инверторов неполумодулярной схемы с системой уравнений $z_1 = \bar{z}_3$, $z_2 = \bar{z}_3$, $z_3 = z_1 \vee$

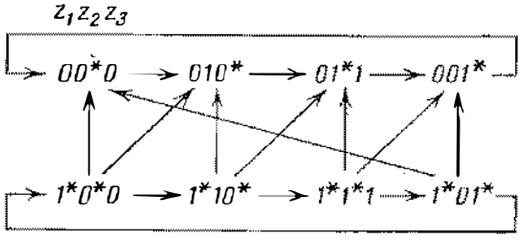


Рис. 10.5. Пример неполумодулярной схемы, в которой константная неисправность не диагностируется

$\vee z_2$, не приводит к «остановке» схемы. Действительно, неисправность $z_1 = 0$ приводит к диаграмме рис. 10.5, в которой рабочий цикл бесконечен. Система в этом случае выглядит так: $z_1 = 0$, $z_2 = \bar{z}_3$, $z_3 = z_2$.

Анализ диагностических свойств автономных схем может, вообще говоря, проводиться в двух направлениях. Первое — это проверка схемы при заданных неисправностях (из класса константных) и заданных рабочих цик-

лах. Такой анализ может осуществляться теми же методами, что и обычный анализ рабочего цикла (гл. 8), и сравнительно несложен. Целью анализа в этом случае является проверка достижимости неисправной схемы какого-либо тупикового состояния.

Второй подход к анализу диагностических свойств автономных схем в чем-то подобен полному анализу схем. Он включает определение всех бесконечных рабочих циклов, относительно состояний которых схема полумодулярна, и определение всех конечных рабочих циклов схемы. Затем на основе полученной информации можно выявить все выносные и подменные неисправности схемы. Эти типы неисправностей особенно опасны, так как для них полумодулярные схемы не являются полностью самопроверяемыми. Выявление фиксаций переменных, приводящих к таким неисправностям, позволяет установить наиболее «уязвимые» точки схемы, которые должны быть «усилены» технологическими методами. Вычислительная сложность алгоритмов при таком подходе к анализу диагностических свойств автономных схем достаточно велика. Пока для такого подхода еще не разработаны методы, существенно лучшие, чем построение полной диаграммы переходов и перебор всех возможных сочетаний константных неисправностей элементов.

Еще более сложным и недостаточно исследованным является вопрос анализа диагностируемости дефектов, приводящих к изменению топологии схемы (например, перемыкание или обрыв проводов, обрыв провода после разветвления — обрыв провода до разветвления сводится к константной неисправности, короткое замыкание входов элементов на выходы). Наиболее вероятные из подобных дефектов также могут быть исследованы методами, изложенными в гл. 8, подобно тому, как это предлагается делать для константных неисправностей. При этом для некоторых указанных дефектов автономные схемы в ряде случаев могут сохранять свойство полной самопроверяемости.

§ 10.4. Саморемонт апериодических схем

Процесс *саморемонта* некоторого устройства связан с последовательной реализацией трех функций: *диагностирования неисправности*, заключающейся в обнаружении ошибки в работе устройства; *локализации неисправности*,

заключающейся в выявлении места, в котором произошла неисправность, например с точностью до модуля регулярной структуры; *ремонта* методом замещения неисправного модуля резервным. Саморемонт, таким образом, приводит к восстановлению работоспособной структуры устройства. После завершения саморемонта для возобновления процесса, реализуемого устройством до возникновения неисправности, необходимо установить устройство в начальное

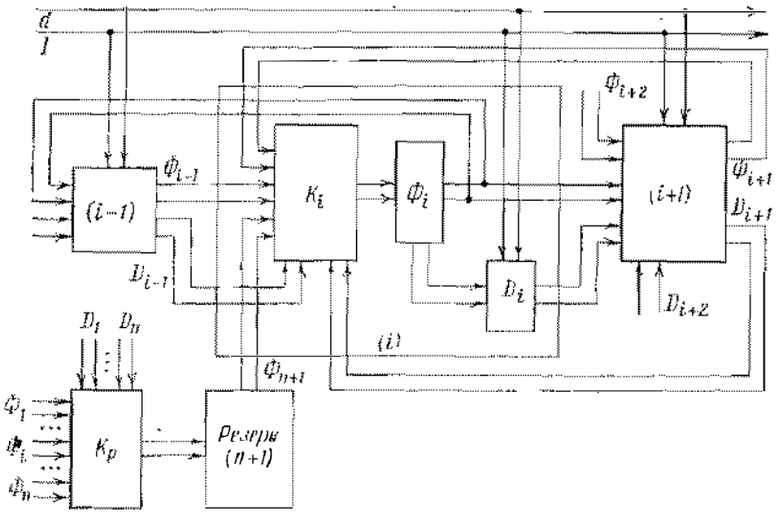


Рис. 10.6. Организация саморемонта скользящим резервированием с прямым замещением

состояние или в последнее контрольное состояние и реинициировать процесс. Будем считать, что первая из трех перечисленных функций — диагностирование неисправности — реализуется средствами, описанными выше в настоящей главе. Результатом диагностики является выработка сигнала неисправности $d = 1$.

Можно предложить два способа организации саморемонта вычислительных систем и устройств, имеющих регулярную структуру. При первом в случае возникновения дефекта неисправный модуль заменяется резервным. Один и тот же резервный модуль может заменить любой из основных модулей, в которых возникла неисправность, поэтому данный способ мы будем называть *скользящим резервированием с прямым замещением*. На рис. 10.6 приведена схема организации саморемонта однократных

неисправностей на основе общего резерва для случая однородной конвейерной или магистральной структуры, в которой каждый i-й элемент связан с предшествующим $(i-1)$-м и последующим $(i+1)$-м. Для случая произвольных связей между модулями структура схемы обобщается очевидным образом. Каждый модуль состоит из функциональной части Φ_i, детектора D_i, локализующего неисправность, и коммутатора K_i, подключающего на входы функциональной части Φ_i функциональные выходы резервного блока Φ_{n+1} вместо Φ_{i-1}, если неисправность локализовалась в $(i-1)$-м модуле, или вместо выходов Φ_{i+1}, если неисправность в $(i+1)$-м модуле. Управление коммутацией осуществляется выходами детекторов D_{i-1} и D_{i+1}. Ремонт включает в себя также коммутацию на выходы резервного модуля выходов тех модулей, которые подавались на входы неисправного модуля. Эту функцию осуществляет коммутатор K_p на входе резервного модуля на основе информации о локализации неисправности, задаваемой детекторами $D_1, \ldots, D_n$. Корректная организация саморемонта достигается за счет того, что коммутаторы приведены ко входам функциональных частей и любые неисправности в i-м модуле, как в функциональной, так и в коммутационной части удается ремонтировать «силами» других модулей.

На рис. 10.7 приведена схема основного разряда саморемонтируемого последовательного двоичного счетчика. Она содержит функциональную часть — счетный триггер с элементами u_i, $\breve{u}_i$, q_i, $\breve{q}_i$, p_i, $\breve{p}_i$, локализующий детектор, состоящий из сумматора по модулю 2 (для сигналов p_i, $\breve{p}_i$) с выходом $\bar{d}_i$, и триггера неисправности $(T_i, \breve{T}_i)$. Коммутатор выходных проводов разряда совмещен с функциональной частью и реализуется непосредственно на элементах u_i, $\breve{u}_i$. Каждый разряд двоичного счетчика содержит один входной провод $\breve{p}_{i-1}$ — перенос из предыдущего разряда, поэтому коммутатор должен реализовать одну коммутирующую функцию $a_i = \breve{T}_{i-1}p_{i-1} \vee T_{i-1}\breve{p}_g$. Следовательно,

$$u_i = \overline{p_i u_i a_i} = \overline{p_i u_i \breve{T}_{i-1}\breve{p}_{i-1} \vee p_i u_i T_{i-1}\breve{p}_g}$$

и

$$\breve{u}_i = \overline{u_i a_i \breve{p}_i} = \overline{u_i \breve{p}_i \breve{T}_{i-1}\breve{p}_{i-1} \vee u_i \breve{p}_i T_{i-1}\breve{p}_g}.$$

Схема резервного разряда состоит из счетного триггера и входного коммутатора, реализующего функцию $a_g = T_1\breve{p}_0 \vee T_2\breve{p}_1 \vee \ldots \ldots \vee T_n\breve{p}_{n-1}$, где p_0 — вход счетчика.

Если в счетчике отсутствует неисправность, то сигнал $\bar{d} = 1$ запрещает работу разрядных локализующих детекторов, и триггер $(T_i, \breve{T}_i)$ неисправности каждого разряда находится в устойчивом состоянии (0, 1). При этом $a_i = \breve{p}_{i-1}$ (как в обычном счетчике), а вход резервного разряда заблокирован $(a_g = 0)$. В случае возникновения неисправности в i-м разряде срабатывает система диагностики и устанавливается $\bar{d} = 0$, после чего срабатывает локализующий детектор $(\bar{d}_i = 0)$, и триггер неисправности $(T_i\breve{T}_i)$ устанавливается в состояние (1, 0). В результате этого функции

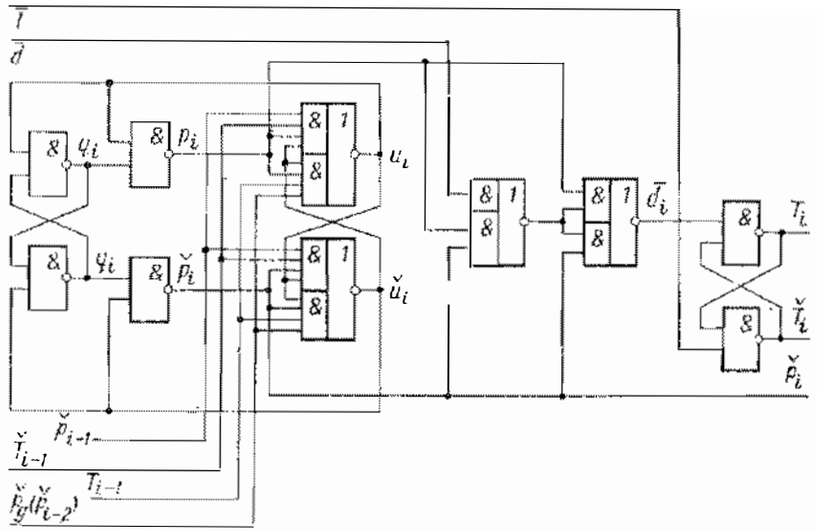


Рис. 10.7. Структура ячейки саморемонтирующегося счетчика

коммутации $(i+1)$-го и дополнительного разрядов принимают соответственно вид $a_{i+1} = p_g$, $a_g = \breve{p}_{i-1}$. Таким образом, осуществляется процесс ремонта, в результате которого вместо i-го основного разряда в схему счетчика вставлен резервный разряд с выходом $\breve{p}_g$. При необходимости после устранения неисправности триггер $(T_i, \breve{T}_i)$ может быть возвращен в исходное состояние сигналом начальной установки $\bar{I} = 0$.

Заметим, что если i-й разряд счетчика неисправен, то либо $a_i = 0$ соответствует $p_i = \breve{p}_i = 1$, либо $a_i = 1$ соответствует $p_i \neq \breve{p}_i$. Состояние дополнительного разряда до подключения — $a_g = 0$, $p_g \neq \breve{p}_g$. Если подключение дополнительного разряда происходит при $a_i = 0$, то он не переключается, и a_{i+1} сразу становится равным p_g. Если же подключение происходит при $a_i = 1$, то дополнительный разряд переключается, после чего $a_{i+1} = 1$. Следовательно, при подключении резерва фазы дополнительного разряда автоматически согласуются с фазами основных разрядов счетчика. В результате этого прерванный неисправностью переходный процесс в счетчике завершится, отработает индикатор и сбросится сигнал неисправности (вернется в состояние $\bar{d} = 1$), что свидетельствует о завершении процесса саморемонта.

С информационной точки зрения процесс может быть некорректен, так как информация, записанная в резервном разряде, может не совпадать с той, которая должна быть в i-м разряде, поэтому после завершения саморемонта необходимо повторить вычисления.

Приведенный метод позволяет саморемонтировать любую однократную консервативную неисправность. Поскольку локализующие детекторы начинают работать только после того, как в схеме (в ее функциональной или коммутирующей части) уже есть неисправность, то однократные консервативные неисправности в локализующих детекторах никак не влияют на работу схемы, а поэтому в приведенном методе никакие неисправности локализующих детекторов не саморемонтируются.

В то же время ошибочная установка $(T_i, \tilde{T}_i)$ в состояние (1, 0) при отсутствии неисправности в разряде приведет к подключению дополнительного разряда параллельно с протеканием нормальных переходных процессов в счетчике, что может повлечь сбойную, неиндицируемую ситуацию. Иными словами, однократные мутационные неисправности в локализующих детекторах могут привести к сбою в работе схемы. Этот недостаток можно устранить, если реализовать следующую функцию коммутации: $a_i = \tilde{d}' p_{i-1} \vee d' (\tilde{T}_{i-1} p_{i-1} \vee T_{i-1} p_g)$, где d', $\tilde{d}'$ — значения выходов триггера неисправности, на котором запоминается факт выработки сигнала неисправности $\bar{d} = 1$. При такой реализации любые неисправности в локализующих детекторах (одиночные и кратные, консервативные и мутационные) при условии отсутствия неисправностей в коммутирующих и функциональных частях тех же модулей никак не влияют на правильную работу функциональной части саморемонтируемого счетчика.

Другой метод саморемонта основан на применении *скользящего резервирования с замещением посредством сдвига*. В случае возникновения дефекта неисправный модуль «шунтируется» логическими средствами, т. е. просто «удаляется» из структуры подобно тому, как удаляется элемент из списка, и линейный размер структуры восстанавливается за счет подключения к ней резервного модуля. Структурная схема организации саморемонта такого типа для случая линейной однородной схемы приведена на рис. 10.8. В нормальном режиме входы каждого i-го модуля соединены только с выходами $(i-1)$-го. В случае возникновения дефекта в $(i-1)$-м модуле после

локализации неисправности коммутатор K_i подключает выходы $(i-2)$-го модуля на входы i-го модуля. Для рассматриваемого примера счетчика коммутатор K_i реализует одну функцию $a_i = \tilde{T}_{i-1} p_{i-1} \vee T_{i-1} p_{i-2}$. Схема саморемонтируемого счетчика со скользящим резервированием совпадает со схемой рис. 10.7, с той разницей, что на один из входов i-го разряда вместо сигнала p_g подается p_{i-2}.

Для обобщения метода саморемонта на основе скользящего резервирования с замещением посредством сдвига

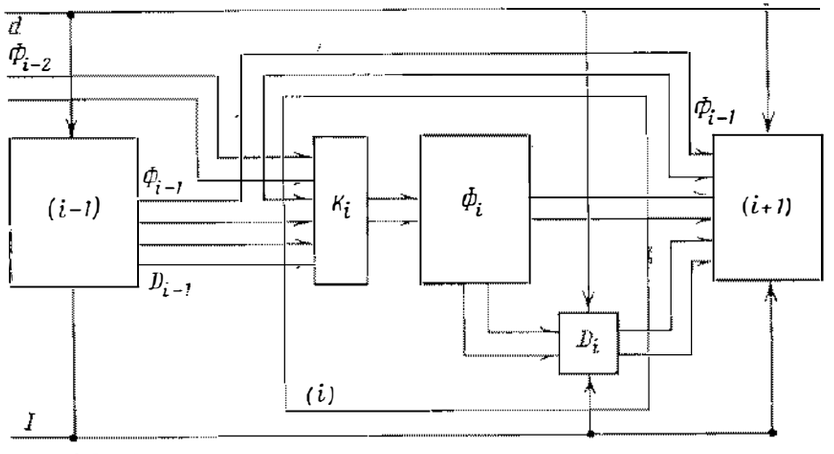


Рис. 10.8. Организация саморемонта скользящим резервированием с замещением посредством сдвига

на более широкий класс систем введем определения: 1) Под *однородной структурой* будем понимать совокупность идентичных модулей. 2) Однородную структуру будем называть *одномерной*, если можно так перенумеровать модули, что для любой пары связанных модулей $a_i \rightarrow a_{i+k}$ или $a_i \leftarrow a_{i+k}$ найдется последовательность связанных модулей $a_i \rightarrow a_{i+1} \rightarrow \ldots \rightarrow a_{i+k-1} \rightarrow a_{i+k}$ или $a_i \leftarrow a_{i+1} \leftarrow \ldots \ldots \leftarrow a_{i+k-1} \leftarrow a_{i+k}$. 3) *Цепочкой* будем называть последовательность модулей $A = \ldots a_{i_1}, a_{i_2}, \ldots, a_{i_k}, \ldots$ такую, что каждые два соседних модуля последовательности $a_{i_l}, a_{i_{l+1}}$ связаны между собой $a_{i_l} \rightarrow a_{i_{l+1}}$ и одинаково удалены друг от друга $|i_1 - i_2| = |i_2 - i_3| = \ldots = |i_{k-1} - i_k|$. 4) Цепочку будем называть *прямой*, если $i_l < i_{l+1}$, и *обратной*, если $i_l > i_{l+1}$. 5) *Областью действия модуля* $a_{i_l} \in A$ для A назовем множество $\{a_j\colon i_l < j \leqslant i_{l+1}\}$, если A — прямая цепочка, и $\{a_j\colon i_{l+1} \leqslant j < i_l\}$, если обратная.

Пример 10.6. На рис. 10.9 приведены примеры одномерных структур, первая из них (рис. 10.9, *а*) линейна. На рис. 10.9, *б* имеются три цепочки: $A_1 = \ldots a_{i-2}, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \ldots, A_2 = \ldots a_{i+3}$,

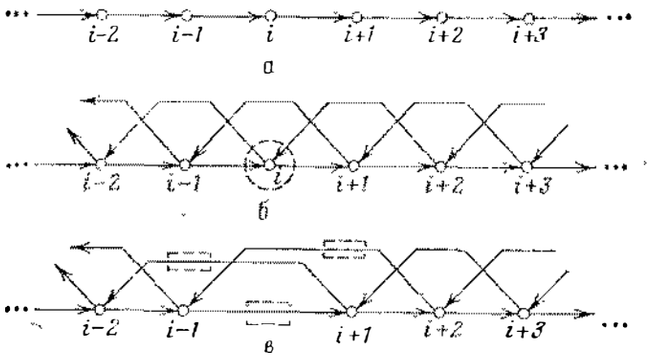


Рис. 10.9. Возможные сдвиги в одномерных структурах при удалении неисправного элемента

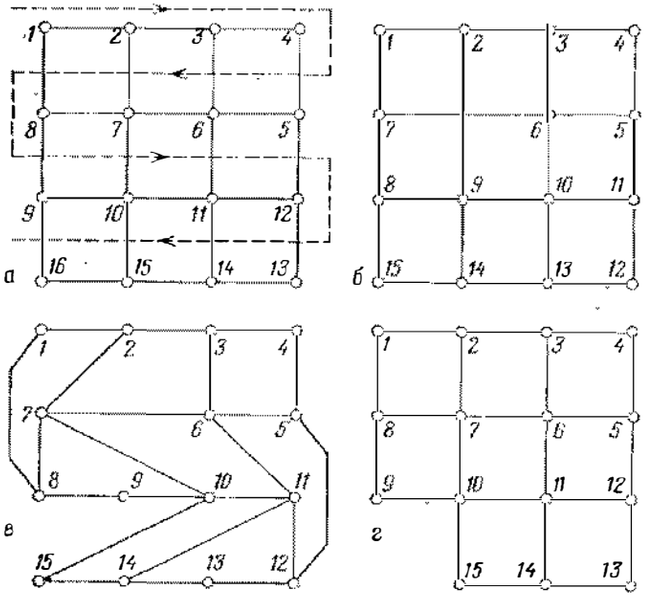


Рис. 10.10. Возможные сдвиги в двумерных структурах при удалении неисправного элемента

$a_{i+1}, a_{i-1} \ldots$ и $A_3 = \ldots a_{i+2}, a_i, a_{i-2}, \ldots A_1$ — прямая, а A_2 и A_3 — обратные. Область действия модуля a_i для цепочки A_1 — $\{a_{i+1}\}$, а для цепочки A_3 — $\{a_{i-1}, a_{i-2}\}$. Структура, получаемая в результате удаления модуля a_i, приведена на рис. 10.9, *в*, на котором помечены три новые связи.

Удаление вышедшего из строя модуля a_i в процессе саморемонта одномерной структуры связано с необходимостью коммутации входных проводов во всех модулях из множества областей действия удаляемого модуля a_i. Рассматриваемый метод саморемонта можно применять и к другим однородным структурам.

На рис. 10.10, *а* проиллюстрированы возможные преобразования двумерной однородной структуры при удалении неисправного модуля a_7. На рис. 10.10, *б* — результат простого удаления, а на рис. 10.10, *в* — удаления с сохранением топологии связей; эквивалентное изображение последней структуры дано на рис. 10.10, *г*.

Выбор конкретного метода реконфигурации зависит от семантики вычислительного устройства. Саморемонт многократных неисправностей возможен, но приводит к громоздким конструкциям и здесь не рассматривается.

§ 10.5. Замечания по библиографии

Понятие полностью самопроверяемой схемы предложено в [209]. Методам синтеза схем, обладающих этим свойством, посвящено достаточно большое число работ, среди которых обычно выделяют [160, 194, 196, 302]. В них рассмотрены основные определения и свойства самопроверяемых схем. Обзор работ по этой тематике выполнен в [157]. В работах [79, 159] для построения полностью самопроверяемых схем используется код с проверкой на четность. В [194, 261] решается задача синтеза схем встроенного контроля комбинационных схем, выходы которых закодированы кодом «m из n», а в [156, 196] — кодом с идентификатором (кодом Бергера). Аналогичные коды используются при синтезе апериодических комбинационных схем [6].

Вопросы построения самопроверяемых устройств с памятью (автоматов) гораздо менее разработаны. Основные структурные особенности таких схем впервые исследовались в [302]. Формальное определение полностью самопроверяемых автоматов на основе результатов работы [302] было дано в [227] и уточнено в [222].

В [2, 111, 112, 221, 226, 234, 252, 277, 289, 294] разработаны частные методы синтеза полностью самопроверяемых схем.

Наиболее полное изложение вопросов, связанных с самопроверяемостью (синхронных) схем, содержится в [130, 157].

Попытки синтеза асинхронных схем этого класса делались в [151, 222], однако раздельное решение задач борьбы с состязаниями и диагностируемости предопределило громоздкость предложенных решений. На диагностические свойства апериодических (полумодулярных схем указывалось в [6, 60, 120, 234, 252]. Метод обнаружения неисправностей базируется на техническом решении, защищенном авторским свидетельством [35].

Весьма перспективным представляется использование апериодической схемотехники при построении высоконадежных отказоустойчивых систем. Общие вопросы, связанные с разработкой таких систем, рассматривались в [1, 67, 85, 118, 141, 186, 188].

Машины мало меня заинтересовали, наверное, потому, что на лобовой броне у каждой сидел вдохновенный до полупрозрачности изобретатель, пространно объяснявший устройство и назначение своего детища. Изобретателей никто не слушал, да они, кажется, ни к кому в особенности не обращались.

А. Стругацкий, Б. Стругацкий

Глава 11. АПЕРИОДИЧЕСКАЯ СХЕМОТЕХНИКА

В этой главе описаны апериодические схемы базовых устройств вычислительной техники. Как правило, дается лишь краткое неформальное описание их функционирования: это сделано не столько из-за необходимости уменьшения объема книги, сколько из-за желания увеличить круг ее читателей за счет инженеров-практиков. Как и ранее, быстродействие приведенных схем будет оцениваться числом последовательно переключаемых элементов в обеих фазах работы; *среднее время переключения элемента обозначается через* T. Сложность схем можно оценивать, например, числом входов и выходов элементов, из которых они построены. Выбор такой оценки объясняется тем, что она соответствует требуемому числу транзисторов при использовании n-канальной МОП-технологии. Однако эти оценки здесь не приводятся, ибо они без труда могут быть получены читателем.

Напомним, что *триггеры, образованные парой элементов, обозначаются одной буквой — например* $\widehat{q}$ *или* $\widehat{\bar{q}}$ *— что позволяет* читателю *сразу же определить транзитное состояние триггера — единичное или нулевое соответственно.*

Конструкции наиболее употребительных типов триггеров были приведены в § 4.4. Описание базовых схем начнется с еще одного триггера, прототип которого нашел столь широкое применение, что авторы не могли обойти его стороной.

§ 11.1. JK-триггер

JK-триггер функционирует следующим образом. При $J=K=1$ он работает как T-триггер, при $J\neq K$ — как RS-триггер ($J=S$, $K=R$), при $J=K=0$ сохраняет то же самое состояние, в которое он перешел в предыдущей рабочей фазе.

На рис. 11.1 приведена схема апериодического JK-триггера. Она содержит основной триггер $\breve{q}$, вспомогательный триггер $\breve{y}$, вентили p и $\bar{p}$ и схему индикации окончания переходных процессов.

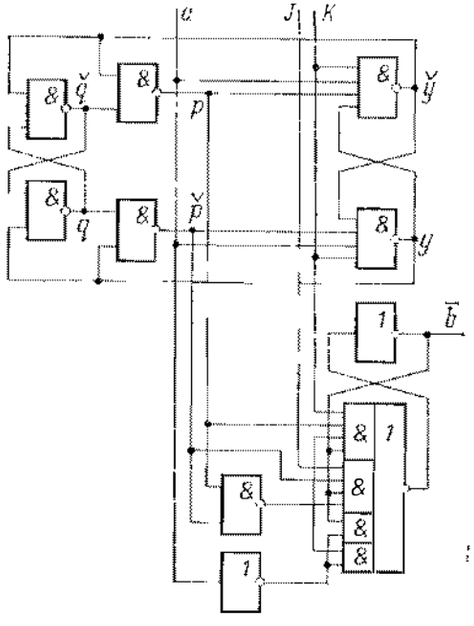


Рис. 11.1. Апериодический JK-триггер

При фазовом сигнале $a=0$ входы J и K не влияют на состояние элементов триггера и в этой фазе значение сигналов J и K может меняться произвольным образом. Выход индикатора $b=1$.

В рабочей фазе, которая инициируется переходом 0-1 сигнала a, триггер восприимчив ко входам J, K, т. е. значения этих сигналов в рабочей фазе не должны меняться. Рабочая фаза завершается переходом 1-0 сигнала b.

Цикл работы JK-триггера $T=12$ в случае, когда триггер переключается, и $T=6$, когда он сохраняет предыдущее состояние.

§ 11.2. Регистры

Регистрами называют устройства, предназначенные для приема и хранения двоичных кодов и в ряде случаев для выполнения преобразования этих кодов (например, сдвига). Вообще говоря, регистры могут быть построены с использованием любых апериодических триггеров, рассмотренных ранее. Основные различия параллельных регистров обусловливаются способом индикации.

Простейший параллельный регистр может быть построен на *RS*-триггерах типа рис. 4.9, *а* (без индикаторов $\bar{b}$), общий индикатор которого, построенный, например, с применением параллельного сжатия (см. пример 4.3), имеет входами выходы q, $\check{q}$ этих триггеров. *Общая длительность переходных процессов для такого*

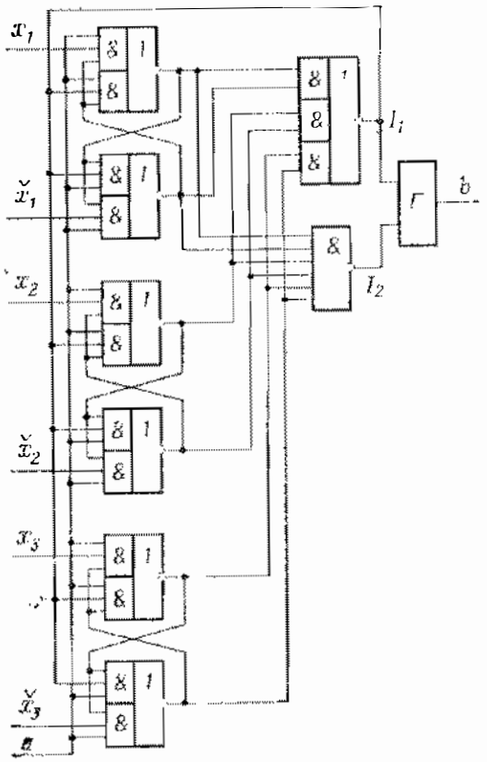


Рис. 11.2. Параллельный приемный регистр с парафазными входами

регистра равна $4T + \tau(n)$, где τ — быстродействие индикатора, n — число триггеров регистра.

Используя *RS*-триггеры типа рис. 4.10, *а*, можно применить индикатор с собственной функцией

$$b = \overline{\bar{S}_1 y_1 \vee \bar{R}_1 \check{y}_1 \vee \ldots \vee \bar{S}_n y_n \vee \bar{R}_n \check{y}_n \vee \check{a}}.$$

Альтернативный способ построения регистра на тех же триггерах дает рис. 11.2, в котором при построении индикатора используется как параллельное сжатие, так и идея коллективной ответственности (см. п. 4.3.2).

При $a = 0$ все триггеры этого регистра погашены и, следовательно, $I_1 = I_2 = 0$. После перехода 0-1 сигнала a и записи входной информации, триггеры становятся нечувствительными к изменению состояний их информационных входов («отсекаются» от входов), $I_1 = I_2 = 1$. По окончании переходного процесса $b = 1$. Переход 1-0 сигнала a вызывает гашение триггеров, в результате $I_1 = I_2 = 0$. Эта фаза завершается переходом 1-0 сигнала b.

Рассмотрим *последовательный регистр*, осуществляющий многотактную задержку: состояние его j-го запоминающего элемента совпадает со значением входного сигнала, поступившего в регистр j тактов (циклов) тому назад.

На рис. 11.3 приведен последовательный регистр, построенный на базе схемы, показанной на рис. 4.9, *а*.

При $a = 0$ все триггеры $\check{y}_j$ погашены, а триггеры $\check{q}_j$ находятся в рабочем состоянии, $\bar{b} = 1$. Переход 0-1 сигнала a инициирует последовательную перепись состояний триггеров $\check{q}_j$ в триггеры $\check{y}_j$ с последующим гашением триггеров $\check{q}_j$. В n-разрядном регистре через время, равное $2nT$, все триггеры $\check{y}_j$ перейдут в рабочее состояние, а все триггеры $\check{q}_j$ будут погашены. Гашение триггера $\check{q}_1$ вызывает переход 1-0 сигнала $\bar{b}$, чем и завершается переходный процесс в этой фазе. Аналогично переход 1-0 сигнала $\check{a}$ инициирует последовательную перепись состояний триггеров $\check{y}_{j-1}$ в триггеры $\check{q}_j$ с последующим гашением первых. Гашение триггера $\check{y}_1$ разрешает запись входного сигнала в триггер $\check{q}_1$, что вызывает переход 0-1 сигнала $\bar{b}$, чем и завершается переходный процесс.

Цикл работы этого регистра составляет $4nT + 2T$, в силу чего его можно назвать «медленным» — все элементы этой схемы переключаются последовательно.

Можно попытаться распараллелить процесс. Разделим регистр на две части, одна из которых состоит из r, а другая — из $n - r$ разрядов, и будем фазовым сигналом инициировать переходный процесс в обеих частях регистра одновременно. На рис. 11.4 показан способ сочленения обеих частей.

Промежуточный триггер $\check{u}_r$ служит для запоминания выхода одной части регистра до момента, когда будет разрешена запись информации во входной триггер его второй части. При $a = 1$ после завершения переходных процессов в этой фазе триггеры $\check{y}_r$ и $\check{y}_{r+1}$ находятся в рабочем состоянии, промежуточный триггер u_r дублирует состояние выходного триггера первой части регистра, а все триггеры $\check{q}_r$, в том числе и $\check{q}_{r+1}$, погашены. На общий индикатор подаются оба выхода триггеров q_1 и $\check{q}_{r+1}$. Переход 1-0 сигнала a инициирует гашение триггеров $\check{y}_n$ и $\check{y}_r$, чем начинаются

переходные процессы в обеих частях регистра. Так как состояние промежуточного триггера u_r в этой фазе не меняется, переходный процесс в первой части регистра завершается записью информации во входной триггер q_1, а во второй части — переходом

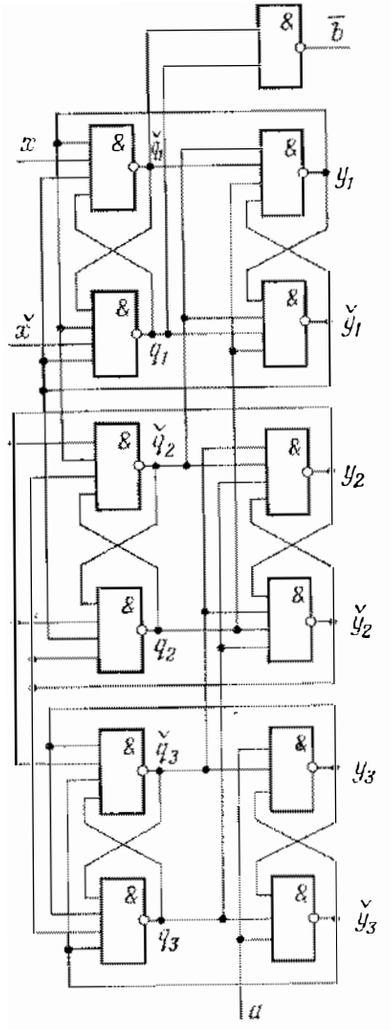


Рис. 11.3. «Медленный» последовательный регистр

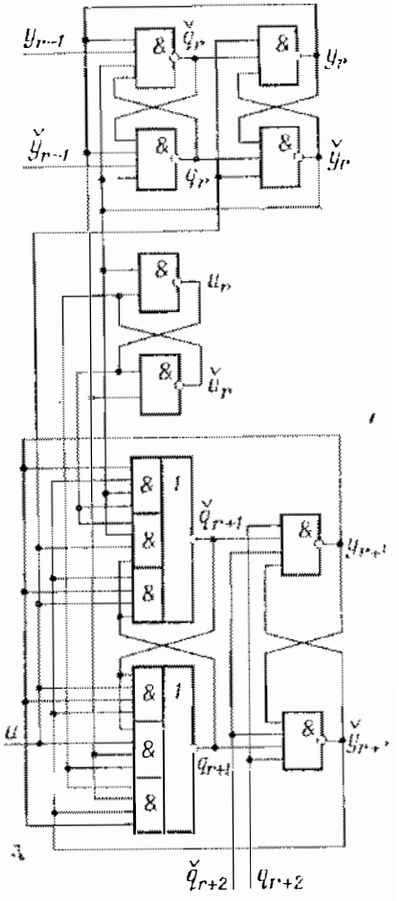


Рис. 11.4. Распараллеливание процесса записи в медленном регистре

триггера в рабочее состояние. Заметим, что передача информации из одной части регистра в другую произойдет только после гашения триггеров y_{r+1} и y_r. Переход в рабочее состояние обоих триггеров q_1 и q_{r+1} влечет переключение индикатора, чем и завершается переходный процесс. Переход 0-1 сигнала a также инициирует переходный процесс в обеих частях регистра одновременно. Эти процессы завершаются гашением триггеров q_1 и q_{r+1} соответственно, которое в свою очередь влечет переключение индикатора, завершающее переходные процессы в регистре.

Обобщение этого приема на случай разбиения n-разрядного регистра на n частей приводит к последовательному регистру; его два разряда изображены на рис. 11.5.

Следует заметить, что эта схема критична к асинхронности поступления фазового сигнала на входы различных

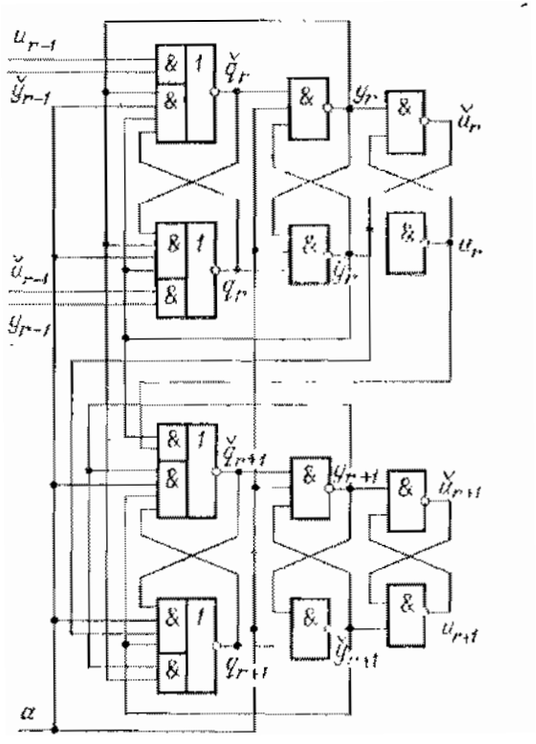


Рис. 11.5. Последовательный регистр

разрядов регистра. Таким образом, если из-за ограниченной нагрузочной способности элемента, с которого снимается фазовый сигнал для этого регистра, требуется заменить один элемент, вырабатывающий этот сигнал, несколькими, то регистр окажется неработоспособным при определенном разбросе величин задержек элементов, формирующих фазовые сигналы для его разных разрядов.

Этот недостаток можно устранить путем некоторого усложнения схемы регистра. Обозначим через a_q фазовый сигнал, подаваемый на триггеры $\check{q}$, а через a_y — фазовый сигнал, подаваемый на триггеры $\check{y}$ соответственно. Пусть a_y снимается с выхода одного элемента, т. е. поступает на входы всех триггеров одновременно, а на триггеры q поступают сигналы a_q, снимаемые с выходов разных элементов. Приведенные последовательные регистры функционируют правильно, если смена значений сигналов a_q предшествует смене значения сигнала a_y. Действительно, после завершения переходных процессов, смена значения сигнала a_q не изменяет состояния элементов схемы, и запуск переходных процессов в регистре изменением значения сигнала a_y происходит при этом в уже «подготовленной» схеме. Очевидно, одним из решений проблемы является подача сигналов a_q на индикатор, с выхода которого можно получить сигнал a_y. Однако такое решение уменьшает быстродействие регистра. Другое решение — формирование сигнала a_{qj} индикатора, которым снабжается каждый триггер q_j. Изменение фазового сигнала a_q допустимо сразу после завершения переходного процесса в триггере q. В этом случае для индикации окончания переходных процессов во всем регистре достаточно собрать на общем индикаторе все сигналы a_{qj}.

На основе приведенных последовательных регистров можно сконструировать *реверсивные регистры*. Управление направлением сдвига в апериодических реверсивных регистрах удобно осуществлять парафазным сигналом. Такой регистр может иметь два индикатора — для каждого направления сдвига свой.

§ 11.3. Конвейерные регистры

Конвейерный регистр — это последовательный регистр, который благодаря своим динамическим свойствам может выполнять роль буфера — накопителя между источником и приемником двоичной информации. Такой регистр содержит несколько последовательно соединенных ячеек,

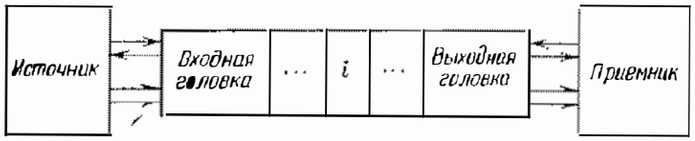


Рис. 11.6. Конвейерный регистр

первая из которых соединена с источником (входная головка), а последняя — с приемником (выходная головка) — рис. 11.6. Порция информации, выданная источником, прежде, чем поступить на вход приемника, пробегает

по всем ячейкам регистра. В зависимости от соотношения скоростей работы источника и приемника информации, конвейерный регистр может то заполняться информацией, то освобождаться от нее. Взаимодействие входной головки с источником, а выходной головки с приемником, так же как и взаимодействие каждой пары соседних ячеек регистра, осуществляется по принципу запрос-ответ. Источник и приемник могут работать независимо друг от друга до тех пор, пока регистр полностью не освободится от информации либо не будет полностью загружен ею.

Конвейерные регистры характеризуются такими параметрами, как *максимальное число порций информации* при заданном числе n разрядов регистра и *пропускная способность* (величина, обратная минимальному интервалу времени t между прохождениями через любую ячейку регистра двух последовательных порций информации).

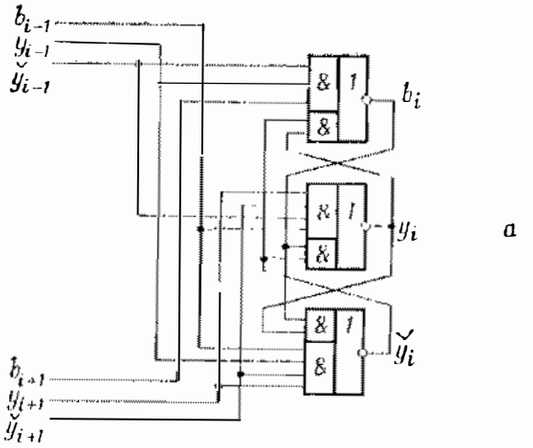
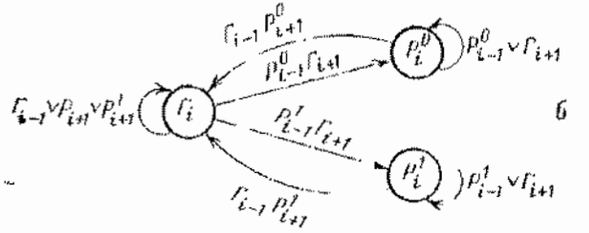


Рис. 11.7. Ячейка неплотного конвейерного регистра на основе трехстабильного триггера (*а*) и граф ее переходов (*б*)



11.3.1. Неплотные регистры. Схема ячейки для варианта регистра, представляющая собой трехстабильный триггер, приведена на рис. 11.7, *а*. Работа ячейки может быть пояснена графом переходов рис. 11.7, *б*. Одна из

вершин графа соответствует отсутствию информации в ячейке (Γ_i — состояние гашения), а две другие (P_i^0 и P_i^1 — рабочие состояния) — передаваемым битам информации (0 и 1 соответственно). Максимальное число порций информации, которое может быть помещено в такой регистр, равно $]n/2[$. Действительно, если источник информации, подключенный ко входу регистра передает информацию порция за порцией, то при неработающем приемнике (который, например, находится в состоянии Г) регистр в конце концов достигнет состояния $\Gamma_1P_2\Gamma_3P_4\ldots$ $\ldots\Gamma_{n-1}P_n$ ($P_1\Gamma_2P_3\Gamma_4\ldots\Gamma_{n-1}P_n$) при четном (нечетном) n. После этого источник не сможет поместить в регистр ни одной порции информации. В силу того, что описанный конвейерный регистр может быть заполнен информацией только наполовину (между двумя ячейками, хранящими соседние порции информации, находится хотя бы одна ячейка в состоянии гашения), он назван неплотным регистром. *Пропускная способность этого регистра составляет* $1/(4t)$, *где* $t=2T$. Действительно, минимальный интервал между двумя соседними порциями информации достигается, когда конфигурация состояний соседних ячеек регистра имеет вид $P_iP_{i+1}\Gamma_{i+2}\Gamma_{i+3}P_{i+4}P_{i+5}\Gamma_{i+6}\Gamma_{i+7}$ (пара P_iP_{i+1} соответствует одной и той же порции информации). В этом случае за время t осуществляется сдвиг указанной конфигурации на одну позицию вправо, а исходная конфигурация повторится через $4t$. При этом *в регистре содержится* $]n/4[$ или $]n/4[-1$ порции информации.

Состоянию P_i^1 соответствует устойчивое состояние триггера рис. 11.7, *а*: 110 состоянию P_i^0 — 101, а состоянию Γ_i — 011. Таким образом, признаком наличия информации в ячейке является $b_i=1$, а признаком отсутствия (гашения ячейки) является $-y_i=\check{y}_i=1$.

Практический интерес представляет вариант *неплотного конвейерного регистра* на элементах И-НЕ, ячейка которого приведена на рис. 11.8. *Для этого регистра* $t=2{,}5T$. Эта ячейка так же, как и предыдущая содержит трехстабильный триггер, состояния которого соответствуют: 011 — в ячейку записана единица, 101 — в ячейку записан нуль, 110 — информация в ячейке стерта. Запись информации в i-ю ячейку происходит, если в $(i-1)$-й ячейке записана информация, а в $(i+1)$-й — стерта, причем значения выходов дополнительных элементов этих ячеек $d_{i-1}=d_{i+1}=1$. Запись начинается переходом 1-0 дополнительного элемента d_i i-й ячейки, затем в ее трехстабильном триггере устанавливается состояние 011 или 101. После этого происходит стирание информации в $(i-1)$-й ячейке, т. е. ее триггер оказывается в состоянии 110, после чего происходит переход 0-1 дополнительного элемента d_i i-й ячейки, что разрешает перепись информации в $(i+1)$-ю ячейку.

11.3.2. Полуплотный конвейерный регистр. Две соседние порции информации в конвейерном регистре могут отличаться друг от друга своими значениями. Такие порции можно, очевидно, располагать в соседних ячейках без «зазора». Эта идея легла в основу построения схемы конвейерного регистра с полуплотным заполнением.

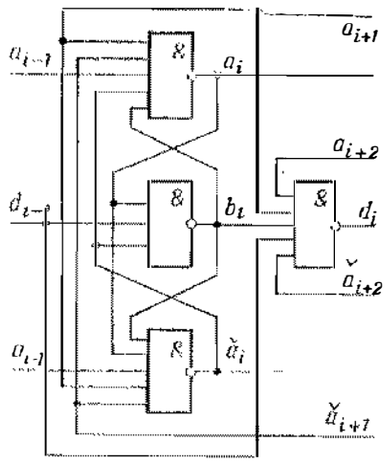


Рис. 11.8. Ячейка неплотного конвейерного регистра на элементах И-НЕ

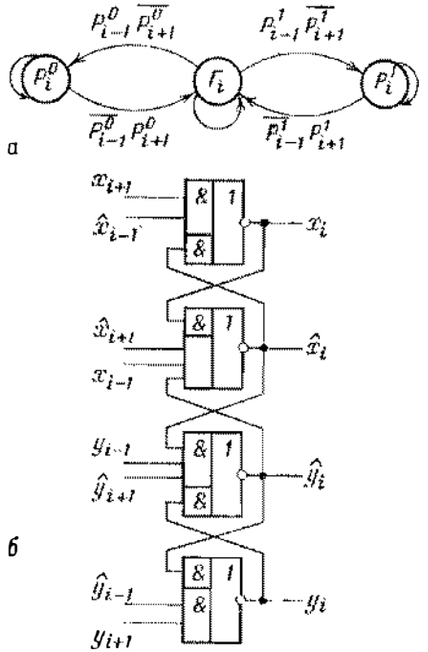


Рис. 11.9. Граф переходов ячейки полуплотного конвейерного регистра (*а*) и ее реализация (*б*)

Граф переходов ячейки *полуплотного регистра* приведен на рис. 11.9, *а*. На этом графе вершина Γ_i соответствует состоянию гашения ячейки, а P_i^0 и P_i^1 — рабочим ее состояниям (хранение нуля и единицы соответственно). Схема ячейки этого регистра приведена на рис. 11.9, *б*. Она содержит два RS-триггера $\hat{x}_i$ и $\hat{y}_i$, имеет четыре устойчивых состояния, закодированных в соответствии с табл. 11.1.

Полуплотный конвейерный регистр, построенный из ячеек этого типа, имеет следующие характеристики:

— информационная емкость лежит в пределах от $n/2$ до n, и в предположении, что входная информационная последовательность является бернуллиевской, ее среднее значение равно $3n/4$;

— пропускная способность регистра лежит в пределах от $1/(3t)$ до $1/(4t)$; верхняя оценка достигается при конфигурации состояний соседних ячеек вида

$$\ldots \Gamma_i P^{\sigma}_{i+1} P^{\sigma}_{i+2} \Gamma_{i+3} P^{\bar{\sigma}}_{i+4} P^{\bar{\sigma}}_{i+5} \Gamma_{i+6} P^{\sigma}_{i+7} P^{\sigma}_{i+8} \ldots,$$

а нижняя — при

$$\ldots \Gamma_i \Gamma_{i+1} P^{\sigma}_{i+2} P^{\sigma}_{i+3} \Gamma_{i+4} \Gamma_{i+5} P^{\sigma}_{i+6} P^{\sigma}_{i+1} \ldots,$$

среднее значение пропускной способности равно $7/(24t)$.

Наличие у рассмотренных конвейерных регистров нескольких устойчивых состояний в каждой ячейке памяти дает повод сделать следующее замечание. При включении питания ячейки регистра устанавливаются в произвольные устойчивые состояния. Поэтому необходима начальная установка, которая в случае конвейерных регистров может быть выполнена путем выкачивания случайных начальных состояний ячеек из регистра. Для этого, например, достаточно выполнить многократное считывание информации из регистра через его выходную головку. Нетрудно проверить, что в результате информация во всех ячейках будет стерта (они все установятся в состояние Γ_i).

Таблица 11.1

	x_i	$\hat{x}_i$	$\hat{y}_i$	y_i
Γ_{i_0}	0	1	1	0
P_{i_1}	1	0	1	0
P_i	0	1	0	1
—	1	0	0	1

11.3.3. Плотные конвейерные регистры. Схема конвейерного регистра с плотным заполнением информацией целесообразна в том случае, если его сложность не превышает сложности неплотного регистра (имеющего вдвое большее число ячеек). В ряде случаев такие регистры обладают и большей пропускной способностью, чем неплотные.

Граф переходов варианта ячейки *плотного буферного регистра* приведен на рис. 11.10, *а*. В соответствии с этим графом ячейка имеет четыре состояния: P_i^{00}, P_i^{10}, P_i^{01}, P_i^{11}. Первый верхний индекс соответствует значению бита информации, записанной в ячейку, а второй — метке. На

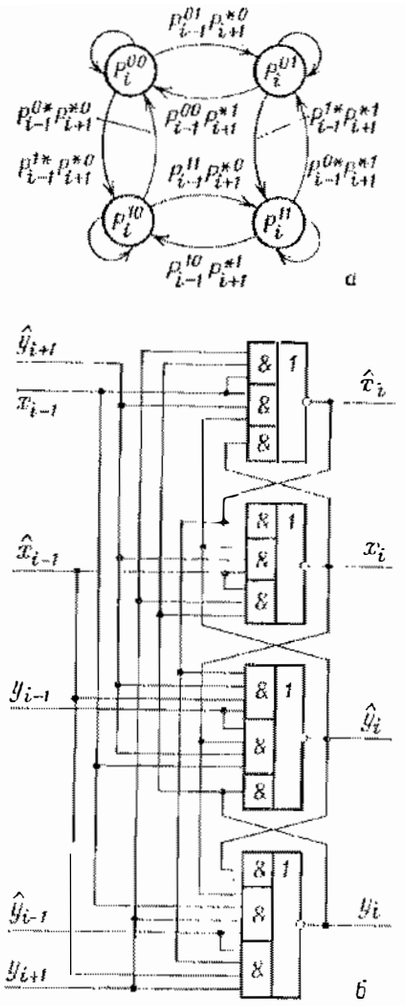


Рис. 11.10. Граф переходов ячейки плотного конвейерного регистра (*а*) и ее реализация (*б*)

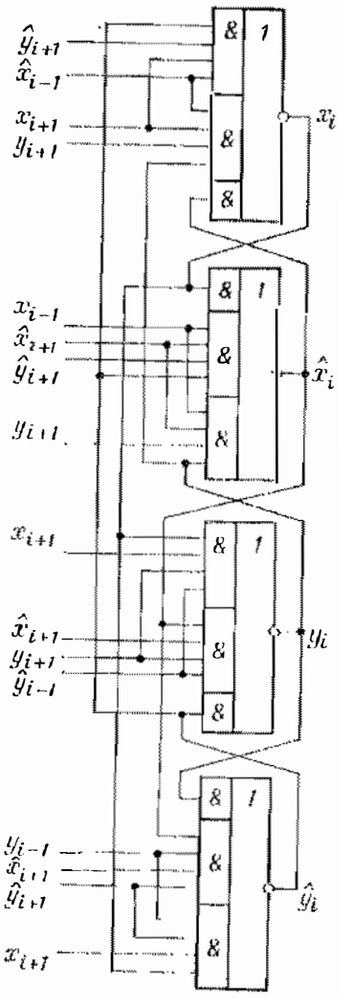


Рис. 11.11. Ячейка полуплотного конвейерного регистра с повышенной (по сравнению с ячейкой рис. 11.10) пропускной способностью

графе приняты следующие обозначения:

$$P^{\sigma *}_{i-1} = P^{\sigma_0}_{i-1} \vee P^{\sigma_1}_{i-1},\ P^{*\sigma}_{i+1} = P^{0\sigma}_{i+1} \vee P^{1\sigma}_{i+1}.$$

Ячейка не имеет состояния гашения, т. е. состояния, в котором информация в ней стерта, поэтому регистр всегда хранит информацию, хотя бы один бит. В последнем случае все ячейки регистра находятся в одном и том же состоянии. Два разных бита, хранящиеся в соседних ячейках, помечены разными метками. Таким образом, в полностью заполненном регистре состояния любой пары соседних ячеек имеют разные метки.

Особенностью приведенной ячейки является то, что все ее состояния могут быть не только устойчивыми, но и транзитными. Например, при переходе из состояния P_i^{00} в состояние P_i^{11} ячейка побывает в состоянии P_i^{10}, которое в этом случае является транзитным, а при переходе из состояния P_i^{11} в состояние P_i^{10}, последнее является устойчивым.

Схема этой ячейки приведена на рис. 11.10, *б*. Как и предыдущая она содержит два *RS*-триггера $\hat{x}_i$ и $\hat{y}_i$, первый из которых хранит информацию, записанную в ячейку, а второй — метку. Состояния ячейки закодированы в соответствии с табл. 11.2.

Таблица 11.2

	$\hat{x}_i$ x_i $\hat{y}_i$ y_i
P_i^{00}	1 0 1 0
P_i^{10}	0 1 1 0
P_i^{01}	1 0 0 1
P_i^{11}	0 1 0 1

Пропускная способность плотного регистра, построенного из таких ячеек, лежит в пределах от $1/(2t)$ до $1/(3t)$, где $t = 2T$. Верхняя граница достигается, когда значение всех битов информации, записанной в регистр, одинаково, конфигурация состояний соседних ячеек регистра при этом имеет вид

$$\ldots P_i^{\sigma_0} P_{i+1}^{\sigma_0} P_{i+2}^{\sigma_1} P_{i+3}^{\sigma_1} P_{i+4}^{\sigma_0} P_{i+5}^{\sigma_0} P_{i+6}^{\sigma_1} P_{i+7}^{\sigma_1} \ldots,$$

а нижняя граница — при чередовании значений битов, записанной в регистр информации, что дает конфигурацию

вида

$$\ldots P_i^{00} P_{i+1}^{00} P_{i+2}^{01} P_{i+3}^{11} P_{i+4}^{11} P_{i+5}^{10} P_{i+6}^{00} P_{i+7}^{00} P_{i+8}^{01} \ldots$$

Среднее значение пропускной способности равно $5/(12t)$.

Другой вариант плотного конвейерного регистра отличается от предыдущего наличием двух дополнительных входов в каждой ячейке. Это на первый взгляд незначительное усложнение схемы позволило в полтора раза поднять пропускную способность регистра. Ячейка для этого варианта приведена на рис. 11.11.

Существуют и более экономичные схемы плотных конвейерных регистров, например вариант, граф переходов

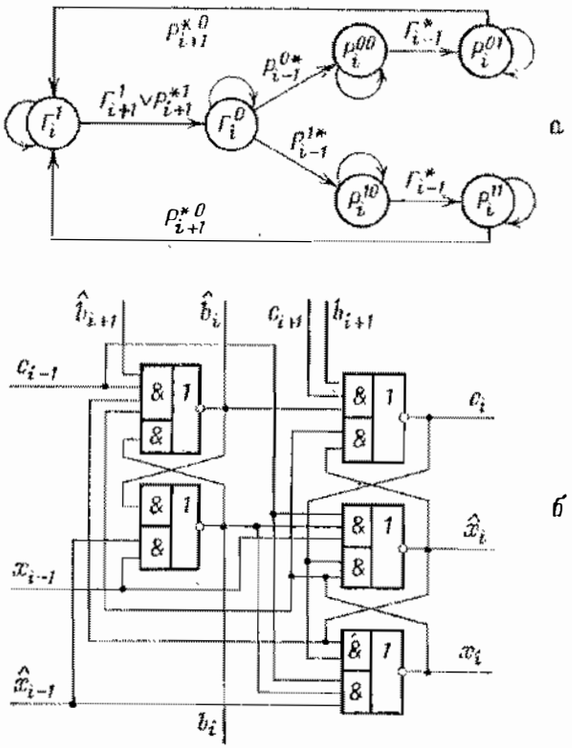


Рис. 11.12. Граф переходов ячейки плотного конвейерного регистра (*а*) и ее реализация (*б*) на триггере с раздельными входами и трехстабильном триггере

и схема ячейки которого приведены на рис. 11.12, *а* и *б* соответственно. Вершины графа переходов этой ячейки соответствуют следующим шести ее состояниям: двум состояниям гашения Γ_i^0 и Γ_i^1 и четырем рабочим состоя-

ниям P_i^{00}, P_i^{10}, P_i^{01} и P_i^{11}. Состояние Γ_i^0 является транзитным, остальные состояния ячейки устойчивые. Обозначения рабочих состояний соответствуют предыдущему графу переходов (рис. 11.10, *а*), верхние индексы состояний гашения соответствуют метке. Как и в предыдущем случае, на графе приняты следующие обозначения: $P_{i-1}^{**} = P_{i-1}^{00} \vee P_{i-1}^{10} \vee P_{i-1}^{01} \vee P_{i-1}^{11}$, $P_{i-1}^{\sigma *} = P_{i-1}^{\sigma 0} \vee P_{i-1}^{\sigma 1}$, $P_{i+1}^{*\sigma} = P_{i+1}^{1\sigma} \vee P_{i+1}^{0\sigma}$, $\Gamma_{i-1}^{*} = \Gamma_{i-1}^{0} \vee \Gamma_{i-1}^{1}$.

Если в регистре отсутствует информация, то все его ячейки находятся в состоянии Γ_i^1. В заполненном регистре состояния ячеек P_i^{01} или P_i^{11}. Ячейка содержит два триггера — один трехстабильный, а другой — *RS*-триггер. Трехстабильный триггер имеет два устойчивых состояния, соответствующих хранению двух различных состояний помещенного в ячейку бита информации, и третье устойчивое состояние, соответствующее состоянию гашения ячейки, т. е. состоянию, когда информация в ней стерта. *RS*-триггер служит для хранения метки. Состояния ячейки закодированы в соответствии с табл. 11.3.

Описанный плотный конвейерный регистр имеет *пропускную способность*, равную $1/(6t)$, где $t = 2T$.

Таблица 11.3

	$\hat{b}_i$	b_i	c_i	$\hat{x}_i$	x_i
Γ_i^1	1	0	0	1	1
Γ_i^0	0	1	0	1	1
P_i^{00}	0	1	1	1	0
P_i^{10}	0	1	1	0	1
P_i^{01}	1	0	1	1	0
P_i^{11}	1	0	1	0	1

11.3.4. Байтный плотный конвейерный регистр. Одним из возможных подходов к построению конвейерного регистра, в котором каждая порция информации представлена одним байтом, заключается в использовании восьми двоичных конвейерных регистров. При этом, очевидно, необходимо синхронизировать работу всех входных головок,

а также всех выходных. Тогда байт информации, принятый в эту совокупность регистров, выйдет из нее без искажения.

Другая возможность заключается в том, чтобы обеспечить синхронизацию в каждом разряде регистра. Используя идею, положенную в основу построения ячейки плотного конвейерного регистра рис. 11.12, можно получить ячейку *плотного байтного регистра*, которая является более экономичной, чем восемь ячеек объединенных регистров. Это достигается за счет того, что схема управления ячейками восьми регистров объединена в одну, этим же достигается синхронизация в каждом разряде. Выделение

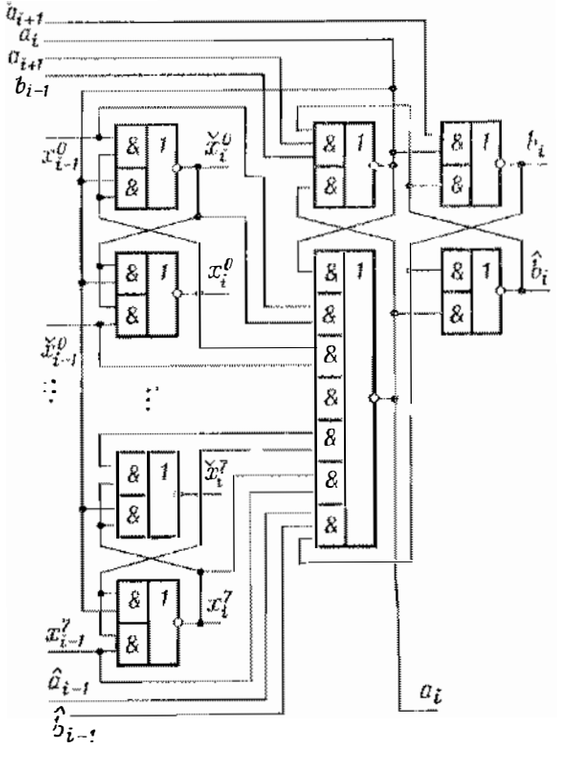


Рис. 11.13. Ячейка байтного плотного конвейерного регистра

схемы управления в ячейке рис. 11.12 можно осуществить, отождествив информационные выходы трехстабильного триггера, входящего в состав ячейки. Тогда трехстабильный триггер преобразуется в *RS*-триггер, а это

эквивалентно отождествлению пар рабочих состояний P_i^{00} и P_i^{10} и P_i^{01} и P_i^{11}. В результате схема управления ячейки будет иметь два рабочих состояния и два состояния гашения, одно из которых является транзитным. Для получения ячейки байтного плотного конвейерного регистра осталось подключить к такой схеме управления восьмиразрядный параллельный регистр, как это сделано, например, на рис. 11.13. *Пропускная способность такого регистра составляет* $1/(12T)$.

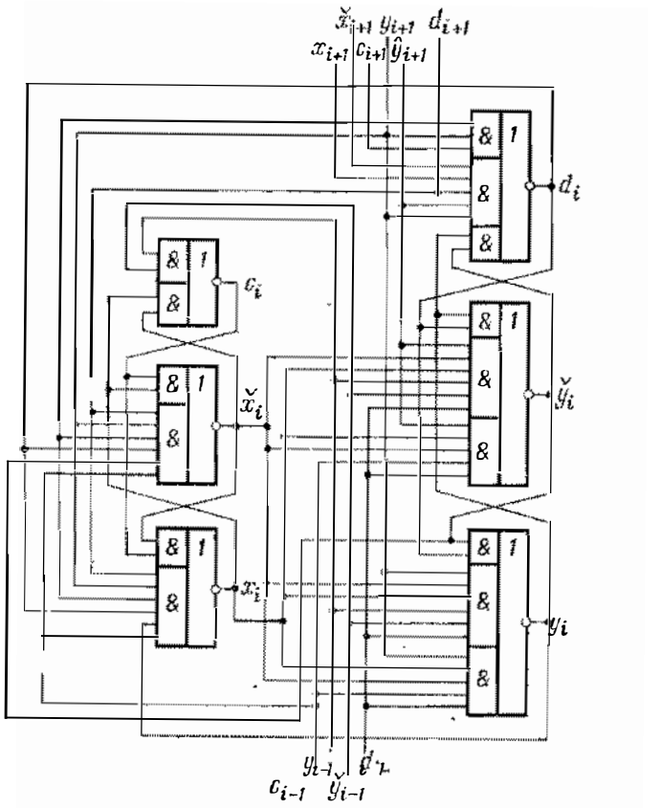


Рис. 11.14. Ячейка конвейерного регистра с параллельным считыванием информации

11.3.5. Конвейерные регистры с параллельным считыванием и записью информации. Стек. В описанных конвейерных регистрах каждая порция информации продвигается от первой ячейки к последней за время, которое зависит как от собственных (реальных) задержек элементов, из которых построен регистр, так и от заполнения

регистра информацией. В результате практически невозможно определить, в какой ячейке регистра находится в данный момент времени тот или иной разряд сдвигаемого кода. С другой стороны, параллельное считывание информации из последовательного регистра (к которым в известной степени относятся и конвейерные регистры) предполагает, что в момент считывания в i-й ячейке регистра находится i-й разряд сдвигаемого кода.

Пусть к источнику информации, испускающему разряды двоичного кода последовательно, начиная со старшего, подключен своей входной головкой конвейерный регистр, позволяющий осуществить плотное заполнение информацией. Тогда i-й разряд кода попадает в i-ю ячейку такого регистра после того, как $(i+1)$-й разряд окажется в $(i+1)$-й ячейке. Этот факт и позволяет выявить моменты, когда, начиная с последней ячейки из конвейерного регистра, можно осуществить *параллельное считывание информации.*

Схема ячейки конвейерного регистра, работающего на этом принципе, приведена на рис. 11.14. Она содержит два трехстабильных триггера — основной и вспомогательный.

Как и в случае простейшего (неплотного) конвейерного регистра (рис. 11.7), одно состояние каждого из трехстабильных триггеров ячейки соответствует отсутствию информации в триггере, а два — соответственно хранению в триггере единицы и нуля. Основные триггеры (верхний на рис. 11.14) и образуют собственно конвейерный регистр, аналогичный регистру из ячеек типа рис. 11.7, *а* с выходов вспомогательного (нижний на рис. 11.14) осуществляется параллельное считывание информации. Пока во вспомогательных триггерах отсутствует информация, регистр работает аналогично неплотному конвейерному регистру. Когда старший разряд сдвигаемого кода попадает в последнюю ячейку регистра, в ней произойдет перепись информации из основного триггера во вспомогательный. Теперь, если в предпоследнюю ячейку регистра будет записана информация, то, очевидно, это будет разряд кода, следующий за старшим, поэтому и в этой ячейке информация из основного триггера переписывается во вспомогательный. При поступлении очередной порции информации в i-ю ячейку признаком того, что эту информацию необходимо переписать из основного триггера во вспомогательный (и тем самым передать на параллельное считывание) является наличие информации во вспомогательном триггере следующей ячейки регистра. Этот процесс будет продолжаться до тех пор, пока во всех вспомогательных триггерах не будет записана информация, после чего можно осуществлять параллельное считывание информации из регистра. Процесс установки регистра в исходное состояние распространяется последовательно от первой ячейки к последней, при этом сна-

чала стирается информация в основном триггере i-й ячейки (условием этого является отсутствие информации в основном триггере $(i-1)$-й ячейке и наличие ее во вспомогательном триггере $(i-1)$-й ячейки), а затем стирается информация во вспомогательном триггере i-й ячейки (условием этого является отсутствие информации в основном триггере $(i+1)$-й ячейки.

Пропускная способность этого регистра в режиме простого сдвига составляет, как для неплотного конвейерного регистра $1/(4t)$. Информационная емкость регистра из n ячеек составляет, как и для плотного конвейерного регистра, n.

Схема ячейки *конвейерного регистра с параллельной записью* информации приведена на рис. 11.15. Как и в предыдущем случае, она содержит основной трехстабильный и вспомогательный триггеры. Отличие в том, что

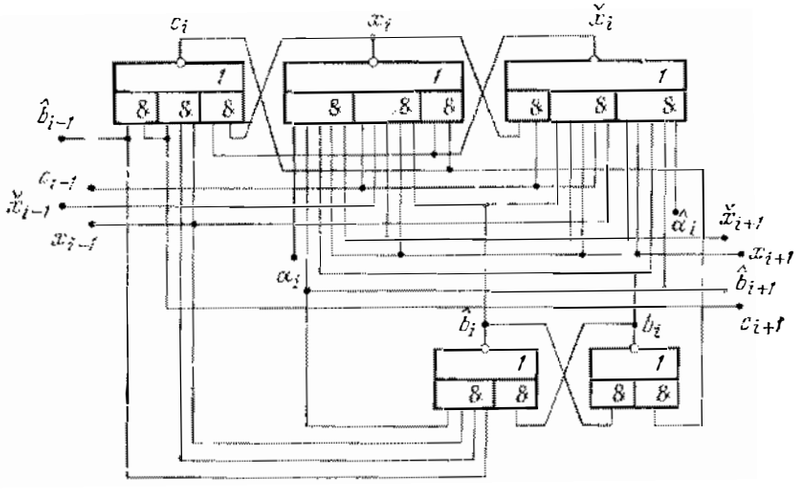


Рис. 11.15. Ячейка конвейерного регистра с параллельной записью информации

вспомогательный RS-триггер предназначен лишь для хранения метки, не дублируя информации основного.

В режиме параллельной записи информация по шинам записи поступает в основной триггер i-й ячейки только после того, как $(i+1)$-я ячейка перейдет из режима записи в режим сдвига и информация в ней будет стерта. После того как будет записана информация в основной триггер i-й ячейки, в ее вспомогательном RS-триггере будет стерта метка — признак параллельной записи, т. е. i-я ячейка перейдет в режим сдвига. Таким образом, процесс перехода регистра в режим сдвига происходит последовательно от последней ячейки к первой. В этом режиме регистр, также

как и предыдущий, работает аналогично простейшему неплотному регистру. В режим параллельной записи i-я ячейка переходит после того, как в этом режиме окажется $(i-1)$-я ячейка, а в i-й ячейке будет стерта информация в основном триггере. Таким образом в режим записи регистр переводится также последовательно, но от первой ячейки к последней. Пропускная способность и информационная емкость у этого регистра такие же, как у простейшего неплотного конвейерного регистра.

На основе конвейерного регистра с параллельным считыванием информации можно построить регистр, обеспечивающий не только сдвиг и параллельное считывание

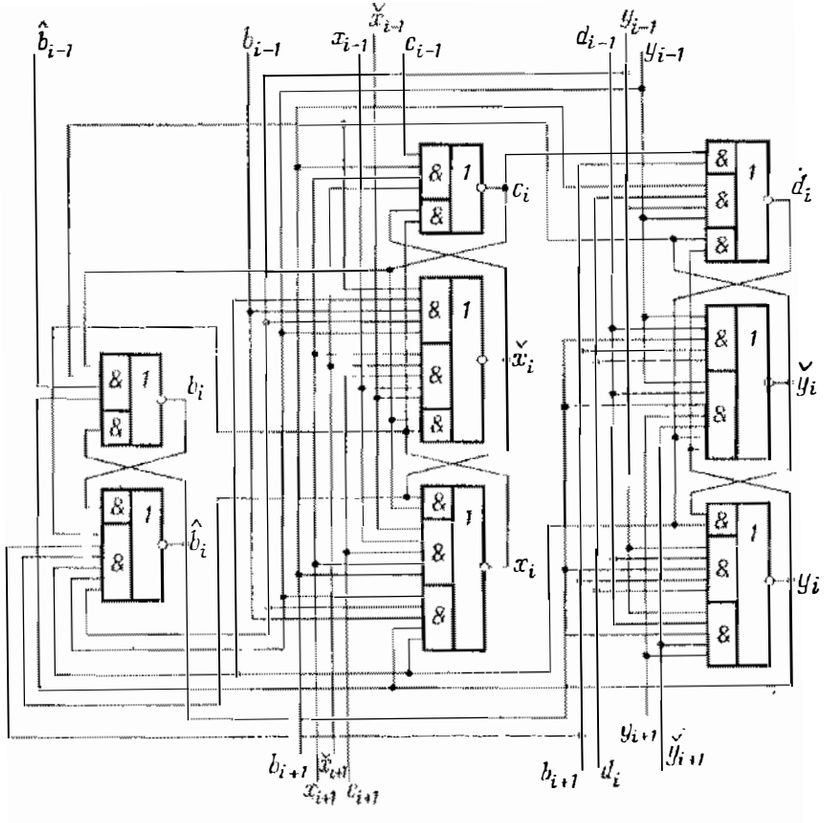


Рис. 11.16. Ячейка стека

информации, но и последовательное считывание в обратном порядке записанной ранее информации, т. е. реализующий *стек*. Ячейка такого регистра приведена на рис. 11.16. Она отличается от прототипа (рис. 11.14) на-

личием дополнительного RS-триггера, служащего для координации режимов работы регистра.

В этом регистре первоначально происходит заполнение информацией основных триггеров ячеек регистра, аналогично тому, как это делается в регистре с параллельным считыванием. При этом по мере заполнения регистра в дополнительные триггеры всех ячеек, начиная с последней, записывается метка. Далее происходит последовательная, начиная с первой ячейки, перепись информации из основных триггеров во вспомогательные, из которых она может быть считана так же, как и в регистре из ячеек типа рис. 11.14. По мере того как во вспомогательные триггеры записывается информация, метка в дополнительных триггерах стирается (перед этим стирается информация в основных триггерах ячеек). При этом из последовательности вспомогательных триггеров может быть последовательно считана информация, но в обратном порядке, так как выходной головкой теперь является вспомогательный триггер первой ячейки. Таким образом, дополнительные триггеры «переключают» два конвейерных регистра, связанных параллельными каналами между собой, при наличии метки работает регистр, образованный основными триггерами,— происходит последовательная запись, а при отсутствии метки — регистр, образованный вспомогательными триггерами, — происходит последовательное считывание информации. Следует заметить, что эти два режима работы регистра могут быть совмещены во времени.

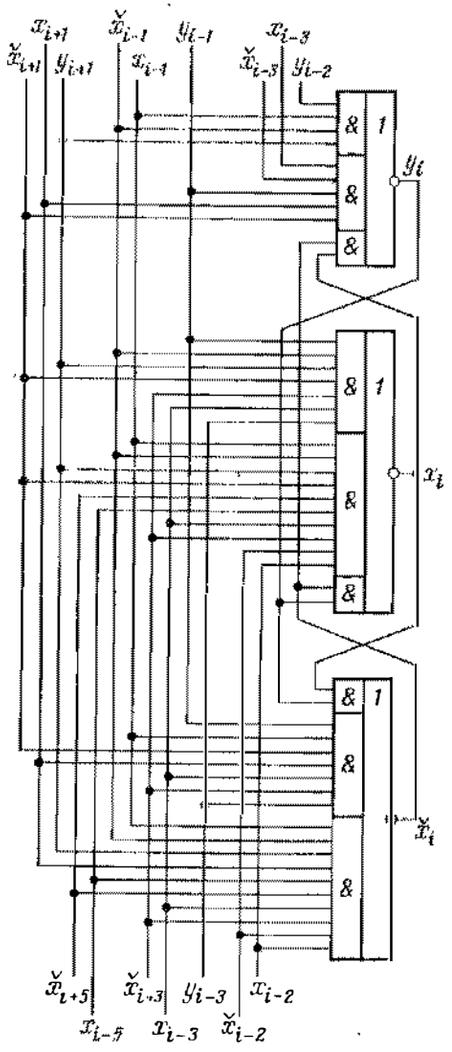


Рис. 11.17. Неплотный реверсивный конвейерный регистр

Описанный регистр имеет такие же характеристики, как и прототип — регистр из ячеек типа рис. 11.14.

11.3.6. Реверсивные конвейерные регистры. Одним из способов расширения функциональных возможностей конвейерных регистров является обеспечение в них реверсивного сдвига информации. При этом конвейерный регистр может быть использован для построения магазинной памяти вычислительных устройств. Схема ячейки *реверсивного конвейерного регистра* приведена на рис. 11.17.

Она содержит трехстабильный триггер, одно из состояний которого, как и раньше, соответствует отсутствию информации в ячейке, а два других — хранению единицы и нуля соответственно. Если две пары ячеек, соседних справа и слева с i-й ячейкой, погашены, то информация в i-й ячейке не сдвигается. Сдвиг информации из i-й ячейки вправо осуществляется, если слева от нее погашена лишь одна соседняя ячейка. Если же слева от i-й погашены три соседние ячейки, то осуществляется сдвиг информации из нее влево. Управление сдвигами информации в этом регистре происходит из крайней левой его ячейки (которая в силу специфики регистра является входной и выходной головками одновременно).

Максимальная информационная емкость конвейерного регистра, построенного из ячеек типа рис. 11.17, составляет $]n/2[$, однако максимальное быстродействие его обеспечивается при заполнении всего на $]n/3[$. Следует отметить, что при использовании описанного регистра в качестве магазинной памяти, он не должен заполняться более чем на $]n/3[$, иначе из него невозможно считать всю ранее записанную информацию.

Сдвиг информации в регистре на один разряд происходит за $6T$ независимо от направления сдвига. Благодаря совмещению выполнения операций в соседних группах разрядов частота выполнения операций равна $1/(3T)$, откуда следует, что пропускная способность этого регистра составляет $1/(9T)$.

Недостатком описанного реверсивного конвейерного регистра является избыточная сложность (низкая информационная емкость) и, как следствие, сравнительно низкое быстродействие. Устранение этого недостатка может быть осуществлено в первую очередь за счет повышения информационной емкости (применения более «плотного» регистра). Примером такого *плотного реверсивного конвейерного регистра* является регистр, схема ячейки которого приведена на рис. 11.18. Его максимальная информационная емкость составляет n, т. е. увеличена по сравнению с предыдущим в два раза, однако емкость, при которой обеспечивается максимальное быстродействие и работа

регистра в режиме магазинной памяти — только в полтора раза и составляет всего $]n/2[$.

Сдвиг на один разряд в этом регистре, так же как и в предыдущем, производится за $6T$, равны в обоих регистрах и величины частоты выполнения операции сдвига.

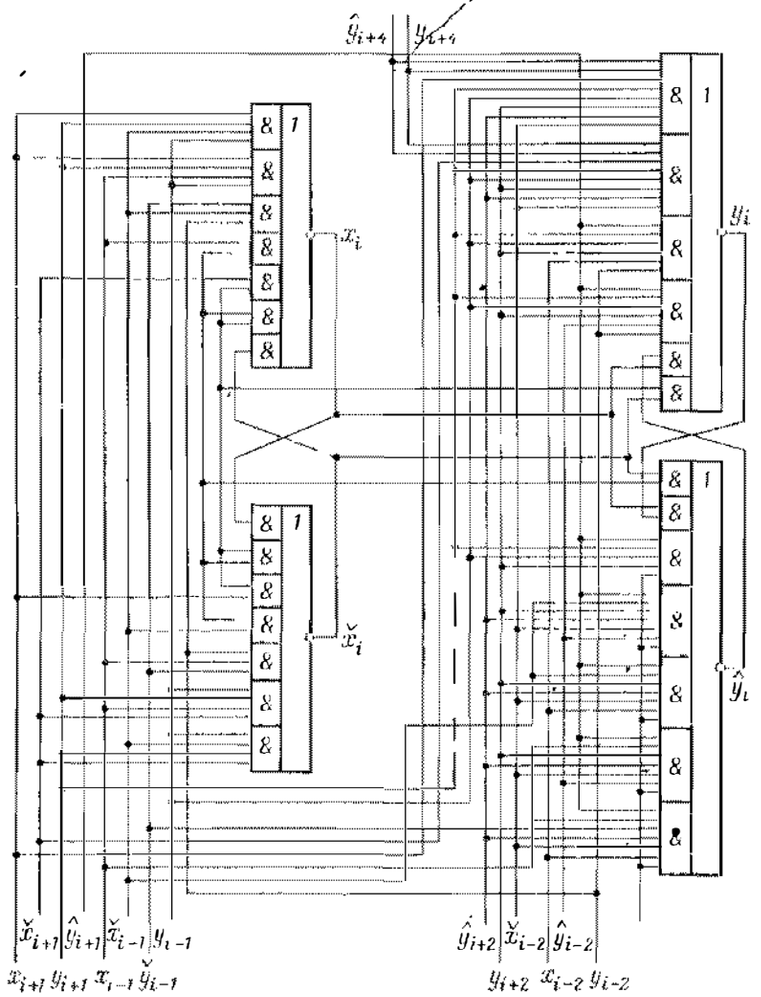


Рис. 11.18. Полуплотный реверсивный конвейерный регистр

Однако, благодаря повышению информационной емкости в этом регистре, по сравнению с предыдущим увеличилась пропускная способность, которая составляет в данном случае $1/(6T)$.

§ 11.4. Преобразование однофазных сигналов в парафазные

11.4.1. Параллельный регистр с однофазными входами. Регистр, приведенный на рис. 11.19, служит для *параллельного преобразования набора однофазных сигналов в парафазные*. С его помощью, например, может быть осуществлена связь апериодического блока с однофазной информационной магистралью.

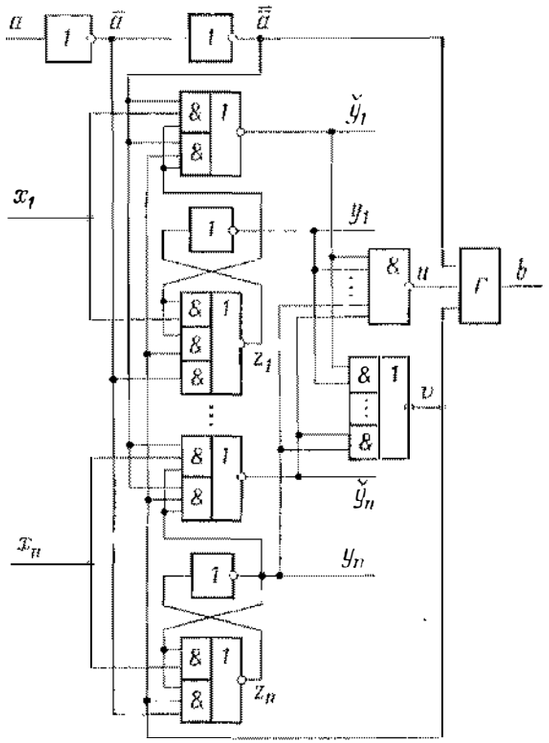


Рис. 11.19. Параллельный регистр с однофазными входами

Магистраль имеет n шин для передачи значений n-разрядного кода, а принимающий регистр, помимо n информационных входов, связанных с соответственными шинами магистрали,— вход a фазового сигнала и выход b сигнала индикации окончания переходных процессов.

При $a=0$ регистр находится в состоянии гашения ($\check{y}_i = y_i = 1$ $z_i = 0$, $1 \leqslant i \leqslant n$, $\bar{a}=1$, $\bar{a}=u=v=b=0$) независимо от состояния информационных входов. Переход 0-1 фазового сигнала вызывает прием информации в разряды регистра. После

того как в результате записи информации во все разряды регистра произойдет переход 0-1 индикаторов u и v, а также сигнала $\bar{a}$, такой же переход произойдет на выходе общего индикатора b, что свидетельствует об окончании переходных процессов в этой фазе работы регистра. Далее, при $b = 1$ возможна смена информации на шинах магистрали, так как регистр хранит принятую ранее информацию до тех пор, пока $a = 1$. Переход 1-0 фазового сигнала вызывает гашение регистра. Эта фаза работы регистра завершается переходом 1-0 сигнала b. После установки на шинах магистрали нового кода может быть начата следующая рабочая фаза.

Описанный прием преобразования однофазных сигналов в парафазные применяется также при построении входных и выходных головок конвейерных регистров.

11.4.2. Входные и выходные головки конвейерных регистров. Конвейерные регистры работают с парафазными кодовыми сигналами, которые в ряде случаев не согласуются с интерфейсом источника и приемника информации. В тех случаях, когда этот интерфейс требует однофазных кодовых сигналов, регистр может быть снабжен специальными входной и выходной головками, схемы которых отличаются от схемы ячеек регистра.

Рассмотрим для примера полуплотный конвейерный регистр. Пусть источник информации связан со входной головкой такого регистра тремя проводами, один из которых является информационным, а два других используются для передачи сигналов управления: фазового сигнала $\bar{a}$ и сигнала индикации b.

Граф переходов *входной головки* представлен на рис. 11.20, *а*. Схема головки изображена на рис. 11.20, *б*. Она содержит семь элементов, четыре из которых образуют пару *RS*-триггеров $\hat{x}_1$ и $\hat{y}_1$. Кодирование состояний головки (и обозначения вершин графа ее переходов) то же, что и для ячейки полуплотного регистра рис. 11.9 (см. табл. 11.1). Заметим, что, в отличие от ячейки регистра, в головке отсутствуют связи между элементами $\hat{x}_1$ и $\hat{y}_1$, которые в ячейке блокируют переход в четвертое (запрещенное) состояние. Дело в том, что во входной головке не возникает ситуация, вызывающая этот нежелательный переход.

Головка работает следующим образом. При $\bar{a} = 1$, $u = v = 0$ головка отключена от информационной шины $x_{вх}$. Если ячейка находится в состоянии гашения, т. е. в триггерах $\hat{x}_1$ и $\hat{y}_1$, в соответствии с табл. 11.1 записан код 0110, то $b = 0$, и головка готова к приему очередного кода. После установки значения $x_{вх}$ переходом 1-0 начинается рабочая фаза головки. Эта фаза завершается

переходом 0-1 сигнала b, после чего можно начинать смену входной информации и переходить в фазу гашения головки, которая инициируется переходом 0-1 сигнала $\bar{a}$. Теперь происходит ожидание переписи информации из головки во вторую ячейку регистра (напомним, что головка является его первой ячейкой). После переписи обычным порядком происходит переход головки в состояние гашения, который в свою очередь вызывает переход 1-0 сигнала b, чем завершается фаза гашения головки.

Граф переходов и схема *выходной головки* полуплотного конвейерного регистра приведены на рис. 11.21, *а* и

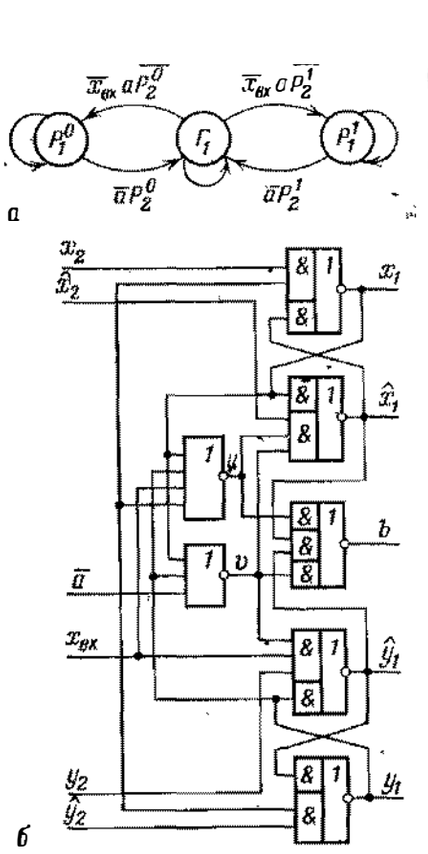


Рис. 11.20. Граф переходов входной головки полуплотного конвейерного регистра (*а*) и ее реализация (*б*)

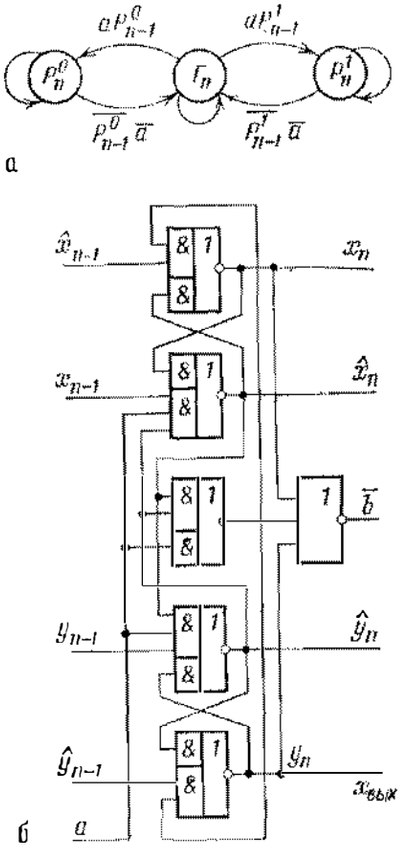


Рис. 11.21. Граф переходов выходной головки полуплотного конвейерного регистра (*а*) и ее реализация (*б*)

б соответственно. Так же, как и входная, эта головка связана с приемником информации тремя проводами — одним информационным и двумя управляющими (для передачи

фазового сигнала a и сигнала индикации $\bar{b}$). Кодирование состояний головки и обозначение на ее графе переходов опять совпадают с таковыми для ячейки полуплотного регистра. Значение сигнала $a=1$ соответствует тому, что приемник не готов к приему информации, а сигнал $\bar{b}=1$ означает, что головка находится в состоянии гашения, т. е. также не готова к выдаче информации.

При поступлении информации из предыдущей ячейки регистра один из триггеров головки переключается и происходит переход 1-0 сигнала $\bar{b}$, что говорит приемнику о наличии информации в головке. Если приемник готов к приему информации, то он списывает ее с выхода y_n головки, а затем инициирует в ней фазу гашения переходом 1-0 сигнала a. При $a=0$, если предыдущая ячейка находится в состоянии гашения или хранит бит информации, значение которого противоположно значению бита, находящегося в головке, то головка переходит в состояние гашения,— фаза гашения головки завершается переходом 0-1 сигнала $\bar{b}$. По сигналу $\bar{b}=1$ приемник осуществляет переход 0-1 фазового сигнала, инициирующий следующую фазу работы выходной головки, и т. д.

Начальная установка полуплотного регистра может быть произведена посредством замыкания выхода $\bar{b}$ выходной головки на ее вход a, после чего при неработающем источнике информации все ячейки полуплотного конвейерного регистра будут погашены.

§ 11.5. Счетчики

На рис. 11.22 представлена схема *апериодического счетчика*, построенного на триггерах типа рис. 4.12, *б*; использовано условное обозначение Т-триггера (без индикатора), введенное на рис. 4.12, *в*. Счетчик представляет

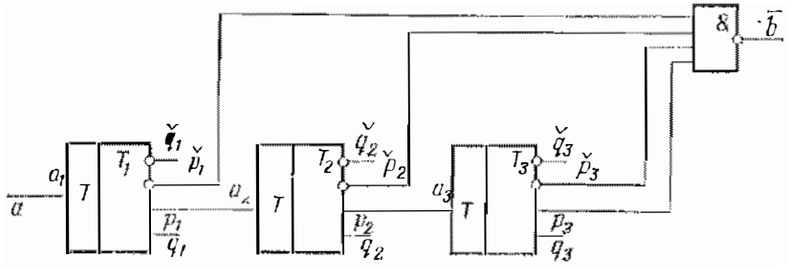


Рис. 11.22. Апериодический счетчик

собой в этом варианте систему последовательно включенных Т-триггеров с общим индикатором. Коэффициент

пересчета k-разрядного счетчика равен 2^k. Роль переноса в j-м триггере играет сигнал с выхода вентиля u_j, поскольку переходный процесс в каждом Т-триггере завершается изменением состояний вентилей u_j и $\check{u}_j$, общий индикатор, очевидно, имеет собственную функцию $\bar{b}=\check{u}_1\ldots\check{u}_{n-1}\check{u}_n u_n$.

Оценим среднюю длительность переноса в апериодическом счетчике. В половине случаев длина переноса равна нулю, в четверти случаев — единице, в одной восьмой случаев — двум и т. д. Следовательно, при $u\to\infty$ средняя длина переноса определяется суммой ряда $G=(1/4)\cdot 1+(1/8)\cdot 2+(1/16)\cdot 3+\ldots$ Очевидные преобразования дают $G=1/2+(1/2)\cdot G$, откуда $G=1$, т. е. в среднем за цикл работы счетчика срабатывают два разряда. Аналогичная оценка для сумматора равна $\log_2(0{,}8n)$.

Таким образом, для приведенного счетчика средняя длительность цикла равна $12T+2\tau_{инд}$, минимальная — $6T+2\tau_{инд}$ (в половине всех случаев), а максимальная — $6nT+2\tau_{инд}$ (всего один случай за полный цикл работы счетчика).

На основе счетчика рис. 11.22 могут быть получены схемы с коэффициентом пересчета, отличным от 2^k. *Пересчет на пять*, например, осуществляет схема рис. 11.23.

В общем случае после переполнения счетчик переводится не в состояние, соответствующее нулю, а в состояние, соответствующее 2^k-l, чем и обеспечивается пересчет с коэффициентом l. Коэффициент пересчета может задаваться внешним двоичным парафазным кодом $\check{l}_j$. Тогда триггер $\check{y}_j$ реализуется в соответствии с уравнениями $y_j=\check{y}_j\check{u}_j u_{j-1}u_n\vee u_{j-1}u_n\check{l}_j$, $\check{y}_j=y_j u_j u_{j-1}\vee u_{j-1}u_n l_j$, $1\leqslant j\leqslant n$, $u_0=a$, а собственная функция индикатора имеет вид $\bar{b}=\check{u}_1\check{u}_2\ldots\check{u}_n\vee u_n$.

На рис. 11.24 приведена схема *десятичного счетчика*, также построенного на основе триггеров типа рис. 4.12, *б*, *в*. Этот счетчик работает в коде 1125 и является частным случаем, когда в приведенные выше уравнения подставлены конкретные значения коэффициента пересчета.

Соединив выход индикатора счетчика с коэффициентом пересчета l с его счетным входом, получим автономную схему, в которой l-кратному изменению сигнала на выходе индикатора (на счетном входе) соответствует однократное изменение сигнала на выходе любого элемен-

та старшего разряда. Разомкнув шину, соединяющую выход этого элемента со входами всех других элементов

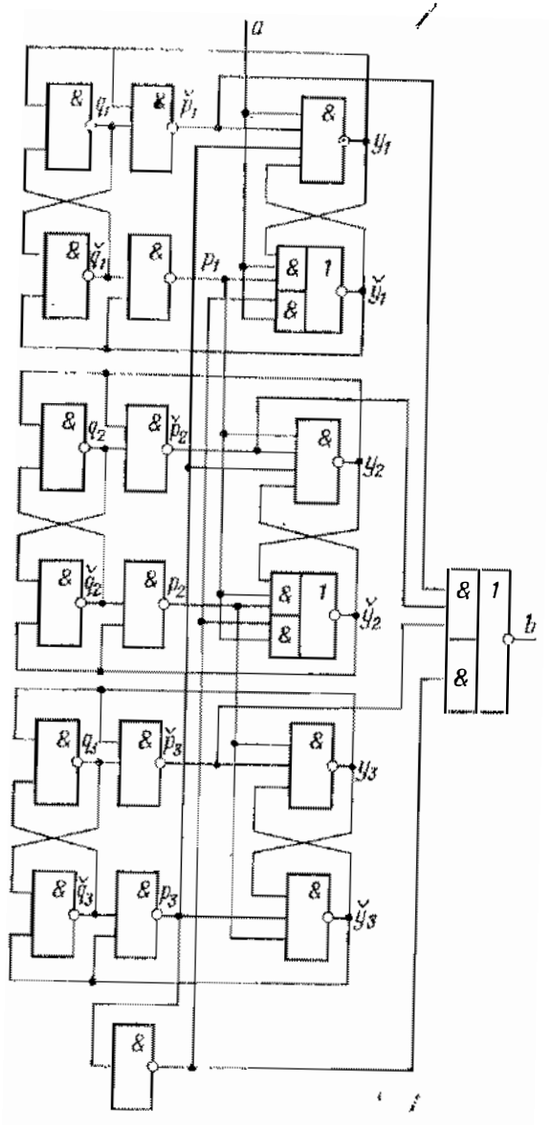


Рис. 11.23. Счетчик с пересчетом на пять

счетчика, получим схему «умножителя частоты»: однократному изменению сигнала на ее входе соответствует l-кратное изменение сигнала на выходе индикатора. Если, в соответствии с теоремой соединения, включить такую

схему в разрыв шины, соединяющей выход элемента старшего разряда другого счетчика, например с коэффициентом пересчета m, со входами всех его других элементов, то получим *пересчетное устройство* с коэффициентом m/l, т. е. любым рациональным коэффициентом пересчета.

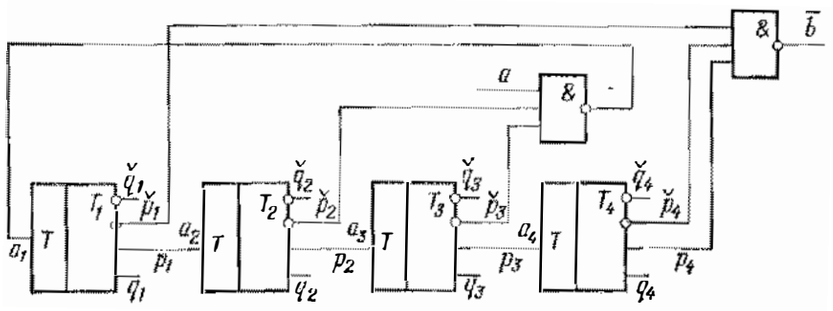


Рис. 11.24. Десятичный счетчик

Апериодический счетчик с установкой в нулевое состояние приведен на рис. 11.25. В этом счетчике вход a_1 счетный, $\bar{b}_1$ — выход индикатора окончания переходных процессов в режиме счета, a_2 — вход установки счетчика в нулевое состояние, $\bar{b}_2$ — выход индикатора окончания переходных процессов в режиме установки. При $a_1 = a_2 = 1$ счетчик находится в режиме хранения информации. Перевод в режим счета или установки осуществляется переходом 1-0 соответствующего фазового сигнала a_1 или a_2. Комбинация $a_1 = a_2 = 0$ является запрещенной. Процесс установки завершается переходом 0-1 сигнала $\bar{b}_2$. После этого можно произвести переход 0-1 сигнала a_2, который, не меняя состояния счетчика, вызовет переход 1-0 сигнала индикации $\bar{b}_2$, что свидетельствует о завершении второй фазы режима установки. Максимальное время цикла установки счетчика равно $6T$. В режиме счета этот счетчик работает так же, как счетчик на рис. 11.22.

Индикацию окончания переходных процессов в последовательном счетчике можно упростить, используя теорему Маллера о соединении полумодулярных схем (см. § 6.2). Напомним, что в качестве примера применения этой теоремы на рис. 6.3 был изображен последовательный счетчик, в котором окончание переходных процессов во всех разрядах индицируется через элементы первого разряда. Счетчик, приведенный на рис. 11.26,

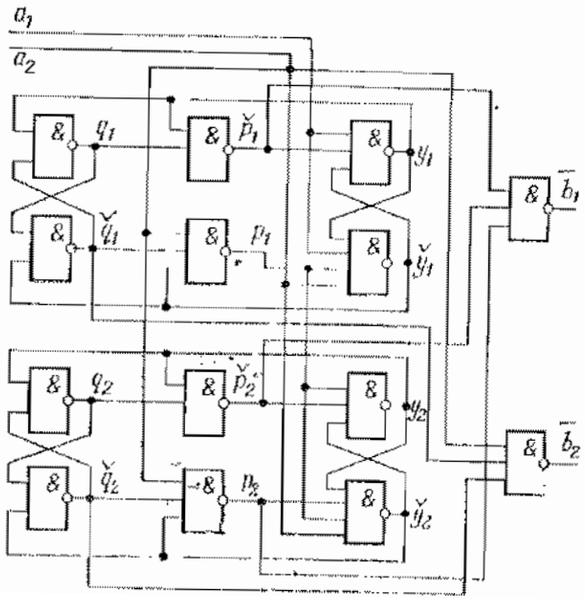


Рис. 11.25. Счетчик с начальной установкой

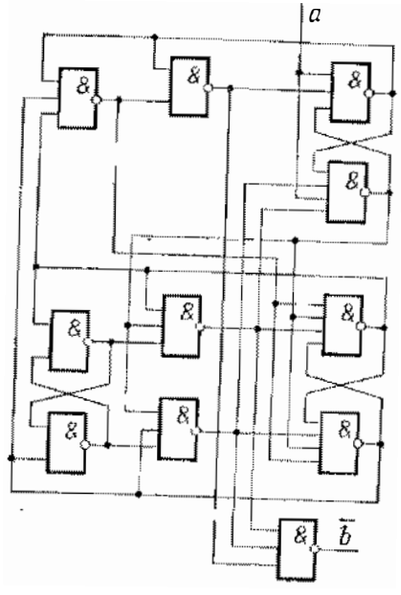


Рис. 11.26. Счетчик, полученный применением теоремы Маллера

использует тот же принцип индикации, однако обладает более высоким быстродействием за счет того, что в каждом разряде, кроме последнего, удалены два элемента. Функцию этих элементов выполняют (благодаря дополнительным входам) элементы следующего разряда.

В качестве основы во всех приведенных счетчиках были использованы Т-триггеры типа рис. 4.12, *б*. Однако возможно использование и апериодических Т-триггеров других типов, например типа рис. 4.12, *а*. Схема счетчика, построенного из таких триггеров, приведена на рис. 11.27. Собственная функция индикатора для этого счетчика имеет вид $b = \check{q}_k y_n \vee \bigvee_{i=1}^{n} q_i y_i$. Цикл одного разряда в этом счетчике составляет $4T$, а поскольку в нем, как и в предыдущем счетчике, в среднем за цикл срабатывают два разряда, средняя длительность цикла этого счетчика составляет $8T + 2\tau_{инд}$.

Апериодические счетчики могут обладать конвейерными свойствами в том смысле, что переходные процессы в различных разрядах счетчика протекают не последовательно, как в уже рассмотренных примерах, а параллельно.

Работу *конвейерного счетчика* кратко можно задать следующим образом: количество срабатываний его i-го разряда должно быть вдвое меньше, чем $(i-1)$-го. Это требование удовлетворяется, если каждый разряд имеет два рабочих состояния — P_i^0 и P_i^1, i-й разряд из состояния гашения попадает поочередно в каждое из них, и его переход в рабочее состояние совершается только при одном из рабочих состояний $(i-1)$-го разряда, например при P_{i-1}^1. Для запоминания предыдущего рабочего состояния разряд должен иметь два состояния гашения — Γ_i^0 и Γ_i^1. Граф переходов i-го разряда конвейерного счетчика приведен на рис. 11.28, *а*, его реализация — на рис. 11.28, *б*. Значения выходов x_i, $\bar{x}_i$, y_i, $\bar{y}_i$ соответствуют: 0111 состоянию Γ_i^0, 1011 состоянию Γ_i^1, 1101 состоянию P_i^0, 1110 состоянию P_i^1.

Конвейерный счетчик, построенный на основе простейшего последовательного счетчика, приведен на рис. 11.29.

В заключение отметим, что основным достоинством конвейерных устройств, определяющим перспективность их использования, является высокая производительность.

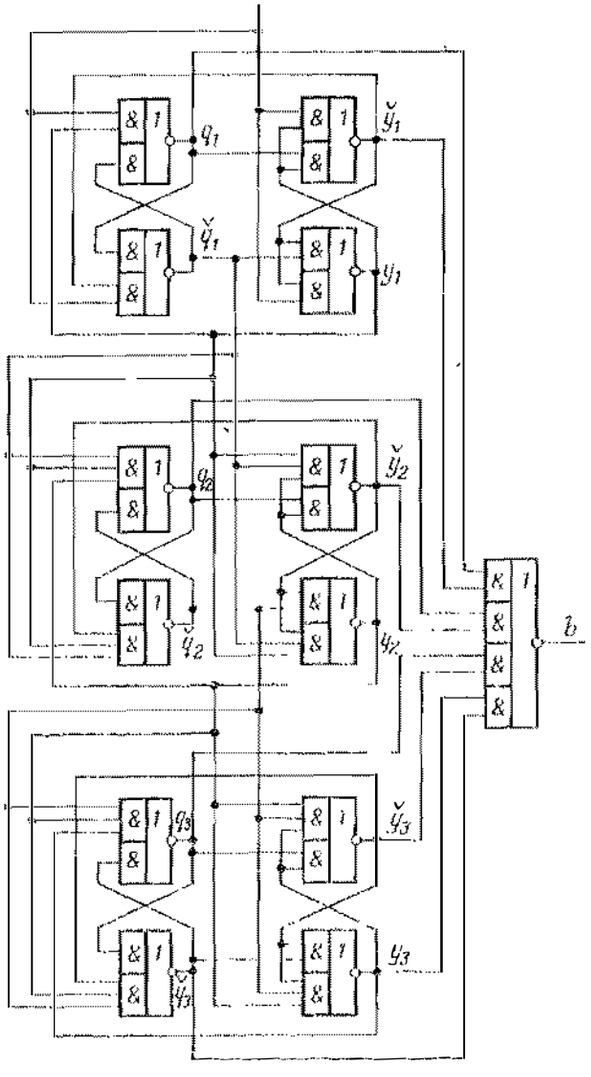


Рис. 11.27. Счетчик на основе триггеров типа рис. 4.12, *б*

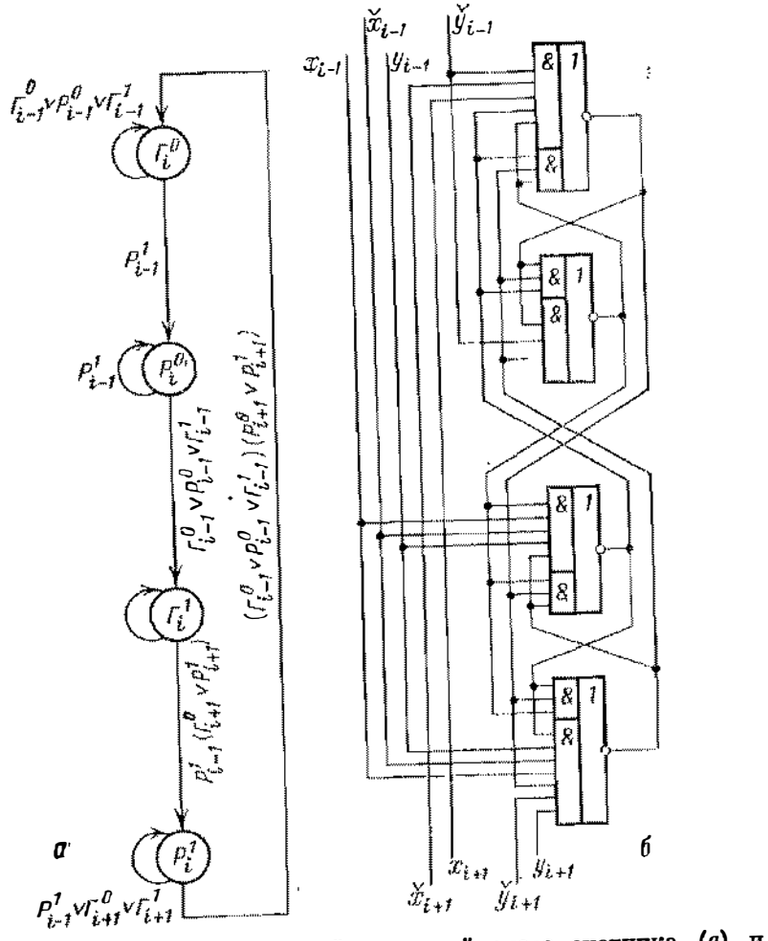


Рис. 11.28. Граф переходов ячейки конвейерного счетчика (*а*) и ее реализация (*б*)

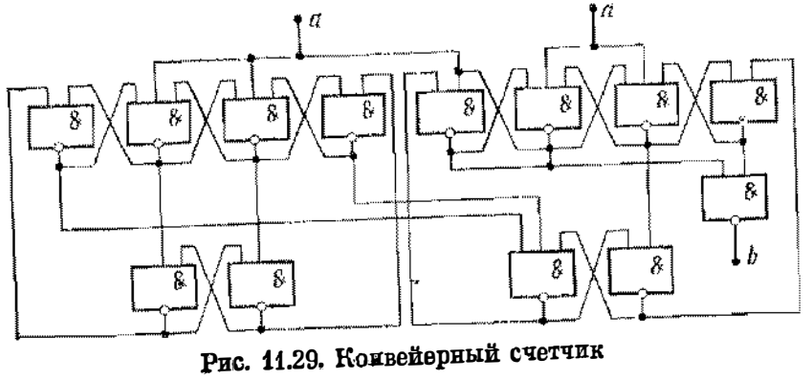


Рис. 11.29. Конвейерный счетчик

Приведенные в гл. 11 многочисленные примеры таких устройств можно считать синтезированными на основе описанных ранее моделирующих схем.

§ 11.6. Замечания по библиографии

Все схемы, приведенные в этой главе, являются оригинальными разработками авторов. Устройства базовой схемотехники, сконструированные до 1975 г., описаны в монографии [6]. Большинство остальных схем защищено авторскими свидетельствами, к описанию которых может обратиться заинтересованный читатель [13—44].

Оценки среднего числа переносов в апериодических счетчиках и сумматорах выведены в [59].

Не знаю, является ли профессор Н. великим писателем, но тот, кто до конца прочел его книгу, безусловно, великий читатель.

Из рецензии

ПОСЛЕСЛОВИЕ РЕДАКТОРА

Дорогой читатель!

Если ты дошел до этого послесловия, значит, ты прочел нашу книгу. Благодарим тебя за это. Но не обольщайся — хорошие схемы ты делать не умеешь. К этому убеждению мы пришли в результате более чем двухлетней работы после написания этой книги. Сказанное не означает, что книга бесполезна. Именно рассмотренные в ней теоретические вопросы обеспечивают возможность построения «хороших» схем, обладающих теми достоинствами, которые были анонсированы во введении и обосновывались в тексте. Материал книги обеспечивает универсальные методы анализа и синтеза схем, поведение которых инвариантно к задержкам элементов. Однако всегда надо помнить, что универсальные методы именно в силу своей универсальности хороши во всех случаях и плохи в каждом конкретном. Проектирование же есть творческий процесс, компонентами которого являются: теория, опыт проектирования и изобретательность проектировщика. Низкий уровень любой из этих компонент делает проектирование бессмысленным. Сказанное побудило нас завершить книгу этими заключительными соображениями.

«Классические» методы логического синтеза схем, как известно, принципиально ограничены «проклятием размерности». Известные асимптотические оценки, казалось бы, делают невозможным создание схем, зависящих более чем от двух-трех десятков переменных. Но мы постоянно имеем дело с устройствами, функционирование которых определяется сотнями переменных. Почему же здесь не работает теория? А потому, что в нашей практической деятельности мы имеем дело не со схемами вообще, а лишь с очень узким классом схем, который мы не можем определить иначе, чем словами «практически встречающиеся схемы». «Проклятие размерности» в классическом смысле распространяется и на способ задания функционирования

схемы. А коль скоро мы сумели на каком-то языке задать функционирование схемы, то можно ожидать, что ее реализация будет по сложности «в силу сложности исходного задания». Это общеметодологическое соображение приводит нас к идее блочного синтеза как к очевидному следствию блочного задания. Понятие «практически встречающихся схем» есть очевидное следствие «практически встречающихся заданий», а такие задания всегда имеют блочную структуру. Языки описания совместного поведения и координации асинхронных процессов есть языки, отражающие алгоритмы совместного функционирования и взаимодействия блоков. Естественно — для случая согласованных блоков. Создание таких языков, изучение их свойств, разработка методов анализа описаний на корректность есть задача теории. Эффективность и компактность описания есть результат опыта проектирования и изобретательности проектировщика. Теоретические методы оптимизации описаний, в силу того, что они, как правило, являются универсальными переборными задачами, эффективны опять же только во взаимодействии с опытным и изобретательным проектировщиком.

Следующим шагом проектирования является переход от описания на языке достаточно высокого (для схем!) уровня к схемной реализации. На уровне блочного описания это структурные методы синтеза, которые можно интерпретировать как процесс трансляции с языка описания взаимодействия процессов на язык управляющей схемы. Для асинхронных согласованных процессов — это прямая трансляция с языка описания на язык моделирующей схемы. Такие методы в книге рассмотрены. Установлено соответствие между операторами языка и представляющими их схемами, причем собственно представляющие схемы есть результат изобретательности. Как это ни странно, но в книге мы не обратили внимания на очевидный факт: сложность моделирующей схемы линейно зависит от сложности описания в языке задания. Этот факт позволил нам в последний год развить весьма эффективные методы синтеза с языка сигнальных графов, игнорирующие необходимость введения дополнительных внутренних переменных. Последнее утверждение, казалось бы, противоречит здравому смыслу — устройство должно обладать внутренней памятью. Но она присутствует в схеме в неявном виде, как внутренняя память ячеек, представляющих операторы языка! Дальнейшее развитие этой идеи привело

нас к понятию автокорректной реализации. Поясним этот термин примером. Пусть поведение системы задано небезопасной сетью Петри. Введение одного дополнительного провода между соседними ячейками моделирующими сеть Петри, т. е. введение одного (двух в КМОП-технологии) транзистора, конвейеризирующего схему, делает поведение небезопасной по описанию сети Петри безопасным. Таким образом, специальный выбор базовых ячеек позволяет снять ряд некорректностей исходного описания без анализа этого описания на корректность и без изменения этого описания, увеличивающего, иногда существенно, его сложность.

Существует, однако, один тип некорректности, который в ряде случаев не может быть снят ни реализацией, ни изменением задания, так как он является следствием природы взаимодействия устройства с внешней средой. Речь идет об арбитраже. За последние два года и в литературе и в группе, которую представляют авторы книги, появилось много нового в понимании физической природы электронного арбитража. При этом установлена возможность возникновения колебательных аномалий не только на логическом, но и на физическом уровне. И если с метастабильной аномалией можно бороться на схемном уровне, то ситуация с колебательной аномалией оказывается принципиально более сложной. Борьба за снижение вероятности попадания схемы в колебательный аномальный режим должна вестись на топологическом уровне, и эта проблема требует дальнейшего глубокого исследования. Но одно утверждение может быть сделано: *переход от синхронных схем к схемам, не зависящим от задержек, существенно снижает общую вероятность возникновения в устройстве арбитражных ситуаций.*

Несмотря на сказанное, читатель не найдет в книге ответа на вопрос, который его безусловно заинтересует: следует ли безоговорочно отдать предпочтение самосинхронным схемам или схемам, не зависящим от задержек, перед синхронными? «Scribtur ad narrandum, non ad probandum» («Пишут для того, чтобы рассказать, а не для того, чтобы доказать» — Квинтилиан. Обучение оратора). Мы не ставили перед собой задачу ответить на этот вопрос, мы стремились по мере возможности достаточно полно осветить состояние и возможности теории схем, не зависящих от задержек, и теории апериодических автоматов применительно к задаче управления асинхронными

процессами или, если хотите, задаче блочного синтеза в классе схем, не зависящих от задержек. Однако в заключение, по-видимому, имеет смысл привести ряд соображений относительно сравнения различных схемотехнических принципов.

Как уже отмечалось в самом начале книги, в последние годы резко повысился интерес к исследованию и использованию самосинхронных схем. Проблема самосинхронизации включена практически во все крупные западные научно-технические программы по созданию нового поколения больших интегральных схем. Однако дискуссия о перспективах использования различных принципов синхронизации не прекращается. Каков же ведущий аргумент против использования самосинхронных схем? Этот аргумент — более высокая сложность самосинхронных схем по сравнению с синхронными. По существующим сегодня оценкам переход от синхронных схем к самосинхронным требует приблизительно двукратного увеличения оборудования. К сожалению, бытовавший одно время среди разработчиков БИС лозунг «Кремния не жалеть!» оказался несостоятельным. Действительно, неумолимым стражем на пути увеличения степени интеграции стоит выход годных схем, который, грубо говоря, экспоненциально зависит от площади, занимаемой схемой. Для схем предельной сложности выход годных может падать до 0,5%. Так вот, если в этой ситуации мы уменьшим сложность схемы всего на 5%, то выход годных увеличится на 30%, а при уменьшении сложности на 10% выход годных увеличится на 70%. Что же говорить об удвоении сложности?

Однако именно здесь за последние годы и особенно за последние два года ситуация начала принципиально изменяться. Прежде всего обратимся к рис. 11.26 (с. 362). На этом рисунке приведена схема счетчика, полученного из хорошо известной «гарвардской схемы» использованием теории схем, не зависящих от задержек. При использовании этой схемы в стандартной ТТЛ-Ш вентильной матрице (вентиль четырехвходовый И-НЕ) мы получаем 20%-ю экономию оборудования даже в обычной синхронной реализации. Пример уникальный? Да. Но в то же время и обнадеживающий.

Наша группа занимается вопросами теории апериодических автоматов и схем, не зависящих от задержек, более 15 лет, и все эти годы нам удается постоянно снижать сложность реализации схем (базовых блоков). Это должно

быть заметно из сравнения схем, приведенных в книге «Апериодические автоматы» (1976 г. [6]) и в настоящей книге. Однако последние два года были в этом направлении уникальными. К сожалению, результаты этих исследований не успели войти в книгу. В значительной мере прогресс был обеспечен переходом с уровня вентиля на уровень транзистора. Что я имею в виду, говоря о прогрессе в области построения базовых схем?

Во-первых, это реализация логических функций. Как ясно из текста книги, в схемах, не зависящих от задержек, требуется парафазное (в меньшем числе случаев — самосинхронное) представление сигналов, что, собственно, в значительной мере и определяло необходимость удвоения оборудования. Однако удалось показать, что в случае использования двухфазной дисциплины со спейсером парафазная реализация в КМОП-технологии не приводит к увеличению числа транзисторов по сравнению с тактируемой однофазной логикой. Этот результат связан с тем, что КМОП-схемы и в случае однофазной реализации содержат каналы прямой и инверсной проводимостей.

Во-вторых, это реализация индикаторов. Апериодические схемы содержат каналы индикации моментов окончания переходных процессов. Эти каналы строятся на базе Г-триггеров. В силу того, что Г-триггер содержит компоненту И, число входов на него ограничено, а необходимость использования либо пирамид из ограниченных Г-триггеров, либо системы параллельного сжатия приводит к росту объема оборудования и задержки в схеме индикации, что может резко снизить эффект повышения быстродействия за счет работы по реальным задержкам. Использование свойства двусторонней проводимости МОП-транзистора позволило построить схему двухкаскадного индикатора с практически неограниченным числом входов и расходом оборудования 4 транзистора на индицируемый вход.

В-третьих, это обнаружение классов устройств, которые могут быть представлены в апериодической реализации с пренебрежимо малым увеличением оборудования по сравнению с синхронной реализацией. Для счетчиков это было известно еще в 1974 г. Сейчас удалось найти решения такого типа для памяти.

Апериодические схемы обладают рядом очевидных преимуществ, а, как известно, за преимущества надо платить. В нашем случае — платить сложностью схем и труд-

ностями проектирования. Последние результаты позволяют надеяться, что эта плата не будет чрезмерной или, по крайней мере, будет соизмеримой с получающимися преимуществами.

Следует заметить, что именно отсутствие развитых методов эффективного проектирования привело американских исследователей в области самосинхронизации к необходимости компромисса. То, что сегодня описано в качестве самосинхронных схем в американской литературе (например, в книге «Введение в СБИС» Мида и Конвей) или в докладах на ежегодных конференциях по СБИС в Калифорнийском технологическом институте, схемами, не зависящими от задержек, не является. Это схемы, правильно функционирующие и самосинхронизирующиеся при заданном соотношении задержек. Безусловно, идея построения схем с ограниченной независимостью от задержек является плодотворной и в ряде случаев приводит к упрощению реализации (в действительности не принципиальной); эта идея требует специального исследования. Однако, отказываясь от полной независимости от задержек, мы отказываемся от весьма важного, возможно самого важного, свойства схем, не зависящих от задержек,— их полной самопроверяемости относительно константных неисправностей.

Задача локализации неисправностей и их автоматического парирования, т. е. задача саморемонта, является в настоящее время ключевой не только с точки зрения необходимости повышения надежности схем, но и с точки зрения увеличения степени интеграции путем автоматического парирования технологических дефектов и увеличения тем самым выхода годных.

В настоящее время уже интенсивно ведутся работы по созданию интегральных схем супервысокой степени интеграции (Вафер-БИС) на пластинах размером до 100 см^2. Описанные тактики изготовления таких схем в основном базируются на внешнем тестировании пластины, выявлении неисправных подсхем, лазерном выжигании и нанесении новых связей. Легко понять трудоемкость и сложность такой технологии. Саморемонтирующиеся схемы открывают альтернативный путь, при котором парирование дефектов может быть продолжено на весь жизненный цикл схемы, т. е. продолжено в область повышения эксплуатационной надежности. Такой подход, однако, имеет смысл только тогда, когда собственная сложность схем

локализации неисправностей и коммутации резерва не является чрезмерной. Можно надеяться, что использование схем, не зависящих от задержек, и принцип скользящего резерва открывают здесь заманчивые перспективы. В этом направлении также получены в последнее время обнадеживающие результаты.

Следующим важным вопросом является проблема интерфейсов и длинных внутрикристальных связей. Заметим, что провод длиной 1 мм занимает площадь, эквивалентную нескольким вентилям, и парафазная связь становится слишком накладной. Привлекательной здесь является идея перехода на трехстабильную передачу с квитированием сигнала по тому же проводу.

Вся сказанное должно явиться предметом рассмотрения в следующей книге, над которой мы уже начали работать и которая будет посвящена проблеме проектирования схем, не зависящих от задержек. Мы надеемся, что работа над рукописью и обычный издательский цикл не выйдут за пределы пяти лет.

Мы также надеемся, что знакомство с материалом этой книги инициирует у читателя интерес к схемам, не зависящим от задержек, и апериодическим автоматам, а также интенсивную работу как в области теории, так и практического создания таких схем.

Ленинград, февраль 1986 г. *В. Варшавский*

Статьи ведут себя по тем же нормам, что и население, за исключением того, что им, видимо, нужно собраться вдесятером, чтобы произвести на свет еще одну статью, тогда как у людей хватает пары.

Д. Прайс

СПИСОК ЛИТЕРАТУРЫ

1. Авиженис А. Отказоустойчивость — свойство, обеспечивающее постоянную работоспособность цифровых систем. Пер. с англ.— Труды ИИЭР, 1978, т. 66, № 10, с. 5—25.
2. Аксенова Г. Н., Согомонян Е. С. Построение самопроверяемых схем встроенного контроля для автоматов с памятью.— Автоматика и телемеханика, 1975, № 7, с. 132—142.
3. Амбарцумян А. А., Потехин А. И. Стандартные реализации асинхронного автомата.— Автоматика и телемеханика, 1977, № 10, с. 122—131.
4. Анализ асинхронных логических схем. I. Проблема достижимости и схемы, не зависящие от скорости. II. Достижимость рабочих состояний и влияние задержек в проводах./В. И. Варшавский, М. А. Кишиневский, А. Р. Таубин, Б. С. Цирлин.— Изв. АН СССР. Техническая кибернетика, 1982, № 3, с. 137—149, № 4, с. 84—97.
5. Апгер С. Асинхронные последовательностные схемы.— М.: Наука, 1977.— 400 с.
6. Апериодические автоматы/Под ред. В. И. Варшавского.— М.: Наука, 1976.— 424 с.
7. Артюхов В. Л., Копейкин Г. А., Шалыто А. А. Настраиваемые модули для управляющих логических устройств.— Л.: Энергоиздат, 1981.— 168 с.
8. Асинхронные интерфейсы: кодирование информации, организация/В. И. Варшавский, В. Б. Мараховский, В. А. Песчанский и др.— Препринт, Л.: ИСЭП АН СССР, 1981.— 40 с.
9. Асинхронные процессы. I. Определение и интерпретация. II. Композиция и согласование/В. И. Варшавский, В. Б. Мараховский, В. А. Песчанский, Л. Я. Розенблюм.— Изв. АН СССР. Техническая кибернетика, 1980, № 4, с. 137—142, № 5, с. 138—143.
10. Аскеров Ч. И., Гамидов В. В. Эквивалентные представления дискретных устройств.— М.: Энергия, 1978.— 120 с.
11. Астановский А. Г. Апериодические вычислительные устройства.— Автореферат дис. канд. техн. наук.— Л.: ЛЭТИ, 1975.— 17 с.
12. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов.— М.: Мир, 1979.— 536 с.
13. А. с. 425318 (СССР). Триггер с индикацией окончания переходных процессов/В. И. Варшавский и др.— Опубл. в Б. И., 1974, № 15.

14. А. с. 491949 (СССР). Асинхронный сумматор/В. И. Варшавский и др.— Опубл. в Б. И., 1975, № 42.
15. А. с. 528612 (СССР). Асинхронный регистр сдвига/А. Г. Астановский и др.— Опубл. в Б. И., 1976, № 34.
16. А. с. 532963 (СССР). Асинхронный счетчик/А. Г. Астановский и др.— Опубл. в Б. И., 1976, № 39.
17. А. с. 548892 (СССР). Регистр сдвига/А. Г. Астановский и др.— Опубл. в Б. И., 1978, № 8.
18. А. с. 558380 (СССР). Счетный триггер/А. Г. Астановский и др.— Опубл. в Б. И., 1977, № 18.
19. А. с. 561182 (СССР). Универсальный логический модуль /В. И. Варшавский и др.— Опубл. в Б. И., 1977, № 21.
20. А. с. 561298 (СССР). Последовательный счетчик/В. И. Варшавский и др.— Опубл. в Б. И., 1977, № 2.
21. А. с. 583480 (СССР). Параллельный однофазный регистр /А. И. Бухштаб, В. И. Варшавский и др.— Опубл. в Б. И., 1977, № 45.
22. А. с. 598081 (СССР). Устройство для контроля переходных процессов в асинхронных логических блоках/А. И. Бухштаб, В. И. Варшавский и др.— Опубл. в Б. И., 1978, № 10.
23. А. с. 618853 (СССР). Последовательный счетчик/В. И. Варшавский и др.— Опубл. в Б. И., 1978, № 29.
24. А. с. 658561 (СССР). Устройство для контроля переходных процессов в логических блоках/В. И. Варшавский и др.— Опубл. в Б. И., 1979, № 15.
25. А. с. 661606 (СССР). Ячейка памяти для буферного регистра /А. И. Бухштаб, В. И. Варшавский и др.— Опубл. в Б. И., 1979, № 17.
26. А. с. 706934 (СССР). Последовательный счетчик/В. И. Варшавский и др.— Опубл. в Б. И., 1979, № 48.
27. А. с. 718940 (СССР). Ячейка асинхронного распределителя /А. И. Бухштаб, В. И. Варшавский и др.— Опубл. в Б. И., 1980, № 8.
28. А. с. 723683 (СССР). Однотактный регистр сдвига/В. И. Варшавский и др.— Опубл. в Б. И., 1980, № 11.
29. А. с. 728161 (СССР). Асинхронный регистр сдвига/В. И. Варшавский и др.— Опубл. в Б. И., 1980, № 14.
30. А. с. 756479 (СССР). Ячейка буферной памяти/В. И. Варшавский и др.— Опубл. в Б. И., 1980, № 30.
31. А. с. 780045 (СССР). Реверсивный буферный регистр сдвига /В. И. Варшавский и др.— Опубл. в Б. И., 1980, № 42.
32. А. с. 799009 (СССР). Регистр сдвига/В. И. Варшавский и др.— Опубл. в Б. И., 1981, № 3.
33. А. с. 799010 (СССР). Ячейка памяти для буферного регистра /Б. С. Цирлин.— Опубл. в Б. И., 1981, № 3.
34. А. с. 800992 (СССР). Комбинационный сумматор/Л. В. Дербунович, В. В. Шатилло.— Опубл. в Б. И., 1981, № 4.
35. А. с. 807307 (СССР). Устройство для контроля согласованного автомата/В. И. Варшавский и др.— Опубл. в Б. И., 1981, № 7.
36. А. с. 807490 (СССР). Триггерное устройство/Г. С. Брайловский.— Опубл. в Б. И., 1981, № 7.
37. А. с. 834929 (СССР). Пересчетное устройство/В. И. Варшавский и др.— Опубл. в Б. И., 1981, № 20.

38. А. с. 841050 (СССР). Реверсивный буферный регистр сдвига /Б. С. Цирлин.— Опубл. в Б. И., 1981, № 23.
39. А. с. 913604 (СССР). Счетчик/В. И. Варшавский и др.— Опубл. в Б. И., 1982, № 10.
40. А. с. 924899 (СССР). Ячейка асинхронного распределителя /В. И. Варшавский и др.— Опубл. в Б. И., 1982, № 16.
41. А. с. 928417 (СССР). Ячейка памяти для буферного регистра /Б. С. Цирлин.— Опубл. в Б. И., 1982, № 16.
42. А. с. 940283 (СССР). Апериодическое импульсное устройство /Г. С. Брайловский.— Опубл. в Б. И., 1982, № 24.
43. А. с. 945960 (СССР).— Г-триггер/Г. С. Брайловский.— Опубл. в Б. И., 1982, № 27.
44. А. с. 953737 (СССР). Последовательный счетчик/В. И. Варшавский и др.— Опубл. в Б. И., 1982, № 31.
45. Бандман О. Л. Синтез асинхронного микропрограммного управления параллельными процессами.— Кибернетика, 1980, № 1, с. 42—47.
46. Баранов С. И. Синтез микропрограммных автоматов (граф-схемы и автоматы).— Л.: Энергия, 1979.— 232 с.
47. Биркгоф Г. Теория структур.— М.: ИЛ, 1952.— 408 с.
48. Блох А. Ш. Синтез переключательных схем.— Минск: Наука и техника, 1966, 200 с.
49. Букреев И. Н., Мансуров Б. М., Горячев В. И. Микроэлектронные схемы цифровых устройств.— М.: Сов. радио, 1975.— 368 с.
50. Бутаков Е. А. Методы синтеза релейных устройств из пороговых элементов.— М.: Энергия, 1970.— 328 с.
51. Бутрименко А. В. Разработка и эксплуатация сетей ЭВМ.— М.: Финансы и статистика, 1981.— 256 с.
52. Варшавский В. И. Апериодические автоматы с самосинхронизацией.— В кн.: Дискретные системы: Труды Междунар. симп. ИФАК — Рига: Зинатне, 1974, т. 1, с. 9—25.
53. Варшавский В. И. Блочный синтез в классе апериодических схем.— В кн.: Теория дискретных управляющих устройств.— М.: Наука, 1982, с. 152—159.
54. Варшавский В. И. Коллективное поведение автоматов.— М.: Наука, 1973.— 408 с.
55. Варшавский В. И., Кишиневский М. А. Аномальное поведение логических схем и проблема арбитража.— Автоматика и телемеханика, 1982, № 1, с. 123—131.
56. Варшавский В. И., Розенблюм Л. Я. Динамический стереотип терминальных устройств и апериодические автоматы.— В сб.: Робототехника.— Л.: Машиностроение, 1979, с. 18—22.
57. Варшавский В. И., Розенблюм Л. Я. Методы устранения состязаний в асинхронных схемах: Учебное пособие.— Л.: ЛЭТИ, 1978.— 73 с.
58. Варшавский В. И., Розенблюм Л. Я. О минимизации пирамидальных схем из мажоритарных элементов.— Изв. АН СССР. Техническая кибернетика, 1964, № 3, с. 24—29.
59. Варшавский В. И., Розенблюм Л. Я., Стародубцев Н. А. О среднем времени формирования сигналов переноса в апериодических схемах счетчика и сумматора.— Автоматика и вычислительная техника, 1975, № 3, с. 88—90.

60. Варшавский В. И., Розенблюм Л. Я., Таубин А. Р. Полностью самопроверяемые асинхронные комбинационные схемы и свойство индицируемости.— Автоматика и телемеханика, 1982, № 5, с. 138—146.
61. Варшавский В. И., Розенблюм Л. Я., Тимохин Е. И. Модель протокола обмена в интерактивных системах.— В кн.: Интерактивные системы: Докл. и тез. докл. и сообщ. третьей школы-семинара.— Тбилиси: Мецниереба, 1981, кн. 1, с. 8—16.
62. Варшавский В. И., Розенблюм Л. Я., Цирлин Б. С. Апериодическая реализация операторных схем алгоритмов.— В кн.: Оптимизация в проектировании дискретных устройств: Материалы семинара.— Л.: ЛДНТП, 1976, с. 34—42.
63. Варшавский В. И., Розенблюм Л. Я., Цирлин Б. С. О композиции апериодических схем.— Изв. АН СССР. Техническая кибернетика, 1980, № 1, с. 206—210.
64. Варшавский В. И., Розенблюм Л. Я., Цирлин Б. С. Функциональная полнота в классе схем, не зависящих от скорости. I.— Кибернетика, 1981, № 2, с. 12—15.
65. Варшавский В. И., Розенблюм Л. Я., Цирлин Б. С. Функциональная полнота в классе схем, не зависящих от скорости. II.— Кибернетика, 1982, № 1, с. 86—88.
66. Визирев И. С. Самосинхронизирующиеся асинхронные цифровые схемы.— Изв. ВМЕИ «Ленин», 1976, т. 34, № 11, с. 419—425 (на болгарском языке).
67. Волков А. Ф., Ведешенков В. А., Зенкин В. Д. Автоматический поиск неисправностей в ЦВМ.— М.: Сов. радио, 1965.— 151 с.
68. Вычислительные и управляющие схемы на базе апериодических автоматов с самосинхронизацией/А. Г. Астановский, Л. Я. Розенблюм, Н. А. Стародубцев и др.— В сб.: Дискретные системы: Труды междунар. симп. ИФАК.— Рига: Зинатне, 1974, т. 3, с. 69—78.
69. Гаврилов М. А. Построение релейных устройств и конечных автоматов из блоков.— Изв. АН СССР. Техническая кибернетика, 1963, № 3, с. 13—27.
70. Гаврилов М. А. Современные проблемы развития теории дискретных устройств.— В кн.: Автоматизированное проектирование дискретных управляющих устройств.— М.: Наука, 1980, с. 3—30.
71. Гаврилов М. А. Теория релейно-контактных схем. Анализ и синтез структуры релейно-контактных схем.— М.; Л.: Изд-во АН СССР, 1950.— 202 с.
72. Гаврилов М. А., Девятков В. В., Пупырев Е. И. Логическое проектирование дискретных автоматов.— М.: Наука, 1977.— 352 с.
73. Гильберт Э. Н. Синхронизация двоичных сообщений.— Кибернетический сборник, 1962, № 5, с. 42—59.
74. Глушков В. М. Синтез цифровых автоматов.— М.: Физматгиз, 1962.— 476 с.
75. Глушков В. М., Цеглин Г. Е., Ющенко Е. Л. Алгебра, языки, программирование.— Киев: Наукова думка, 1974.— 328 с.
76. Головкин Б. А. Методы и средства параллельной обработ-

ки информации.— В кн.: Теория вероятностей. Математическая статистика. Теоретическая кибернетика.— М.: ВИНИТИ, 1979, т. 17, с. 85—193.
77. Головкин Б. А. Параллельные вычислительные системы.— М.: Наука, 1980.— 520 с.
78. Горбатов В. А. Семантическая теория проектирования автоматов.— М.: Энергия, 1979.— 264 с.
79. Горожин А. Д., Крайнов К. С. Построение полностью самопроверяемых комбинационных устройств с использованием полиномиальных форм.— Автоматика и телемеханика, 1979, № 12, с. 159—166.
80. Горяшко А. П. Логические схемы и реальные ограничения.— М.: Энергоатомиздат, 1982.— 184 с.
81. Гретцер Г. Общая теория решеток.— М.: Мир, 1982.— 456 с.
82. Гуртовцев А. Л., Петренко А. Ф., Чапенко В. П. Логическое проектирование устройств автоматики.— Рига: Зинатне, 1978.— 212 с.
83. Деннис Дж. Е., Фоссин Дж. Б., Линдерман Дж. П. Схемы потоков данных.— В кн.: Теория программирования. Ч. II.— Новосибирск: ВЦ СО АН СССР, 1972, с. 7—43.
84. Дискретная математика и математические вопросы кибернетики. Т. I/Под ред. С. В. Яблонского и О. Б. Лупанова.— М.: Наука, 1974.— 312 с.
85. Доманицкий С. М. Построение надежных логических устройств.— М.: Энергия, 1971.— 279 с.
86. Дьяченко В. Ф., Лазарев В. Г., Саввин Г. Г. Управление на сетях связи.— М.: Наука, 1967.— 224 с.
87. Евреинов Э. В. Однородные вычислительные системы, структуры и среды.— М.: Радио и связь, 1981.— 208 с.
88. Закревский А. Д. Алгоритмы синтеза дискретных автоматов.— М.: Наука, 1971.— 512 с.
89. Закревский А. Д. Логический синтез каскадных схем.— М.: Наука, 1981.— 416 с.
90. Захаров В. Н. Автоматы с распределенной памятью.— М.: Энергия, 1975.— 136 с.
91. Захаров В. Н., Поспелов Д. А., Хазацкий В. Е. Системы управления. Задание. Проектирование. Реализация.— М.: Энергия, 1982.— 344 с.
92. Златанов П. С. Возможность реализации конечного автомата, не зависящего от скорости переключения элементов.— В кн.: Электроника и моделирование, вып. 13.— Киев: Наукова думка, 1976, с. 40—45.
93. Интерфейс для программируемых приборов в системах автоматизации и эксперимента/Гореликов Н. И. и др.— М.: Наука, 1981.— 264 с.
94. Калачев В. А., Кравченко А. В. К вопросу оптимизации схем парафазной логики.— В кн.: Однородные цифровые вычислительные и интегрирующие структуры, вып. 9.— Таганрог: 1978, с. 68—69.
95. Калачев В. А., Кравченко А. В. Проблема контроля вычислительных структур и парафазная логика.— Прикладные аспекты теории автоматов: Труды V Междунар. сем., т. 2.— Варна: 1979, с. 556—562.
96. Карп Р. М., Миллер Р. Е. Параллельные схемы программ.— Кибернетический сборник, новая серия, вып. 13.— М.: Мир, 1976, с. 5—61.
97. Кишиневский М. А. Реализация и анализ апериодических схем.— Автореферат дис.— Л.: ЛЭТИ, 1982.— 18 с.
98. Кишиневский М. А., Таубин А. Р., Цирлин Б. С. Модельные методы анализа диалогового взаимодействия.— В кн.: Интерактивные системы: Тез. докл. IV школы-семинара, кн. I.— Тбилиси, 1982, с. 293—295.
99. Кишиневский М. А., Таубин А. Р., Цирлин Б. С. Сети Петри и анализ переключательных схем.— Кибернетика, 1982, № 4, с. 114—117.
100. Кобринский Н. Е., Трахтенброт Б. А. Введение в теорию конечных автоматов.— М.: Физматгиз, 1962.— 404 с.
101. Котов В. Е. Алгебра регулярных сетей Петри.— Кибернетика, 1980, № 5, с. 10—18.
102. Котов В. Е. Сети Петри.— М.: Наука, 1984.— 160 с.
103. Котов В. Е. Теория параллельного программирования: прикладные аспекты.— Кибернетика, 1974, № 1, с. 1—16, № 2, с. 1—18.
104. Котов В. Е., Нариньяни А. С. Асинхронные вычислительные процессы над памятью.— Кибернетика, 1966, № 3, с. 64—71.
105. Кузнецов О. П. Об асинхронных логических сетях.— Проблемы передачи информации, 1961, вып. 9, с. 103—115.
106. Кузнецов О. П., Адельсон-Вельский Г. М. Дискретная математика для инженера.— М.: Энергия, 1980.— 344 с.
107. Лазарев В. Г., Пийль Е. И. Синтез асинхронных конечных автоматов.— М.: Наука, 1964.— 260 с.
108. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов.— М.: Энергия, 1978.— 408 с.
109. Логика, автоматы, алгоритмы/М. А. Айзерман, Л. А. Гусев, Л. И. Розоноэр и др.— М.: Физматгиз, 1963.— 556 с.
110. Ляпунов А. А. О логических схемах программ.— Проблемы кибернетики, 1958, вып. 1, с. 46—74.
111. Мазнев В. И. О синтезе самотестируемых 1/p-тестеров.— Автоматика и телемеханика, 1978, № 9, с. 142—145.
112. Мазнев В. И. Синтез полностью самопроверяемых последовательных схем.— Автоматика и телемеханика, 1977, № 6, с. 167—175.
113. Макроконвейерные вычисления функций над структурами данных/В. М. Глушков, Ю. В. Капитонова, А. А. Летичевский и др.— Кибернетика, 1981, № 4, с. 13—21.
114. Малые ЭВМ и их применение/Под ред. Б. Н. Наумова.— М.: Статистика, 1980, 231 с.
115. Марчук Г. И., Котов В. Е. Модульная асинхронная развиваемая система (концепция). Ч. 1. Предпосылки и направления развития архитектуры ВС; ч. 2. Основные принципы и особенности.— Новосибирск: ВЦ СО АН СССР.— Препринты 86, 87, 1978.— 49 с., 52 с.
116. Мейзда Ф. Интегральные схемы. Технология и применения.— М.: Мир, 1981.— 280 с.
117. Мелихов А. Н. Ориентированные графы и конечные автоматы.— М.: Наука, 1971.— 416 с.
118. Методы введения избыточности для вычислительных систем.— М.: Сов. радио, 1966.— 456 с.

119. Методы параллельного микропрограммирования/Под ред. О. Л. Бандман.— Новосибирск: Наука, 1981.— 180 с.
120. Миллер Р. Теория переключательных схем. Т. 2.— М.: Наука, 1971.— 304 с.
121. Минский М. Вычисления и автоматы.— М.: Мир, 1971.— 364 с.
122. Михеев В. М. О множествах, содержащих наибольшее число пар попарно несравнимых булевых векторов.— Проблемы кибернетики, 1959, вып. 2, с. 69—71.
123. Модель асинхронного автомата с прямыми переходами входных и выходных наборов/В. И. Варшавский, В. Б. Мараховский, В. А. Песчанский, Л. Я. Розенблюм.— Автоматика и телемеханика, 1980, № 11, с. 117—123.
124. Мультипроцессорные системы и параллельные вычисления /Под ред. Ф. Г. Энслоу.— М.: Мир, 1976.— 384 с.
125. Нариньяни А. С. Теория параллельного программирования. Формальные модели.— Кибернетика, 1974, № 3, с. 1—15, № 5, с. 1—14.
126. Николенко В. Н. О преобразованиях асинхронных логических схем.— Кибернетика, 1978, № 5, с. 6—8.
127. Николенко В. Н., Чеботарев А. Н. О реализации недетерминированных автоматов.— В кн.: Методы проектирования схемного и программного оборудования.— Киев: 1979, с. 3—10.
128. Нильсон Н. Искусственный интеллект.— М.: Мир, 1973.— 270 с.
129. О возможности реализации асинхронного интерфейса, использующего самосинхронизирующийся код с идентификатором /В. И. Варшавский, В. Б. Мараховский, В. А. Песчанский, Л. Я. Розенблюм.— Автоматика и вычислительная техника, 1981, № 5, с. 84—88.
130. Пархоменко П. П., Согомонян Е. С. Основы технической диагностики: оптимизация алгоритмов диагностирования, аппаратурные средства.— М.: Энергия, 1981.— 320 с.
131. Питерсон Дж. Теория сетей Петри и моделирование систем.— М.: Мир, 1984.— 264 с.
132. Плакс Т. П. Синтез параллельных программ на вычислительных моделях.— Программирование, 1977, № 4, с. 55—63.
133. Попов Э. В. Общение с ЭВМ на естественном языке.— М.: Наука, 1982.— 360 с.
134. Попов Э. В., Фирдман Г. Р. Алгоритмические основы интеллектуальных роботов и искусственного интеллекта.— М.: Наука, 1976.— 456 с.
135. Поспелов Г. С., Поспелов Д. А. Искусственный интеллект.— Вестник АН СССР, 1978, № 10, с. 26—36.
136. Поспелов Д. А. Введение в теорию вычислительных систем.— М.: Сов. радио, 1972.— 280 с.
137. Поспелов Д. А. Логические методы анализа и синтеза схем.— М.: Энергия, 1974.— 368 с.
138. Поспелов Д. А. Логико-лингвистические модели в системах управления.— М.: Энергоиздат, 1981.— 232 с.
139. Прангишвили И. В., Стецюра Г. Г. Микропроцессорные системы.— М.: Наука, 1980.— 326 с.

140. Проектирование микропроцессорных цифровых устройств/Под ред. С. А. Майорова.— М.: Сов. радио, 1977.— 272 с.
141. Путинцев Н. Д. Аппаратный контроль управляющих цифровых вычислительных машин.— М.: Сов. радио, 1966.— 424 с.
142. Риордан Дж. Введение в комбинаторный анализ.— М.: ИЛ, 1963.— 287 с.
143. Рогинский В. Н. Основы дискретной автоматики: статика и динамика дискретных автоматов.— М.: Связь, 1975.— 432 с.
144. Розенблюм Л. Я. Сети Петри.— Изв. АН СССР. Техническая кибернетика, 1983, № 5, с. 12—40.
145. Розенблюм Л. Я., Цирлин Б. С. О реализации асинхронных согласованных схем.— В кн.: Вопросы теории проектирования ЭЦВМ и систем обработки информации.— Киев: 1976, с. 52—59.
146. Розенблюм Л. Я., Яковлев А. В. Инструментальная модель диалога.— В кн.: Диалог в автоматизированных системах.— М.: МДНТП, 1981, с. 118—125.
147. Розенблюм Л. Я., Яковлев А. В. Модель управляемого протокола.— В кн.: Вычислительные сети коммутации пакетов. Т. I: Тез. докл. II Всес. конф.— Рига: 1981, с. 62—66.
148. Ротанов С. В. Об одном подходе к анализу корректности протоколов.— Автоматика и вычислительная техника, 1982, № 4, с. 17—19.
149. Сагалович Ю. А. Кодирование состояний и надежность автоматов.— М.: Связь, 1975.— 208 с.
150. Сагалович Ю. Б. Полностью разделяющие системы.— Проблемы передачи информации, 1982, вып. 2, с. 74—82.
151. Сапожников В. В., Сапожников В. В. Синтез полностью самотестируемых асинхронных автоматов.— Автоматика и телемеханика, 1978, № 1, с. 154—166.
152. Селлерс Ф. Методы обнаружения ошибок в работе ЭЦВМ.— М.: Мир, 1972.— 310 с.
153. Синтез асинхронных автоматов на ЭВМ/Под ред. А. Д. Закревского.— Минск: Наука и техника, 1975.— 184 с.
154. Сипсер Р. Архитектура связи в распределенных системах, 1, 2.— М.: Мир, 1981.— 744 с.
155. Скарлетт Дж. ТТЛ интегральные схемы и их применение.— М.: Мир, 1974.— 288 с.
156. Слабаков Е. В. Синтез полностью самопроверяемых устройств с использованием метода разбиения входов на независимые группы.— Электронное моделирование, 1980, № 4, с. 39—43.
157. Слабаков Е. В., Согомонян Е. С. Самопроверяемые вычислительные устройства и системы.— Автоматика и телемеханика, 1981, № 11, с. 147—167.
158. Слэйгл Дж. Искусственный интеллект.— М.: Мир, 1973.— 319 с.
159. Согомонян Е. С. Построение одновыходных самопроверяемых схем контроля.— Электронное моделирование, 1980, № 4, с. 26—31.
160. Согомонян Е. С. Построение самопроверяемых схем встроенного контроля для комбинационных устройств.— Автоматика и телемеханика, 1974, № 2, с. 121—143.

161. Стародубцев Н. А. Автономные антитонные последовательные схемы. I. Определения и интерпретация. II. Циклограммы и их свойства. III. Минимизация. IV. Оценки сложности.— Изв. АН СССР. Техническая кибернетика, 1981, № 4, с. 155—162, № 5, с. 87—93, № 6, с. 82—86, 1982, № 1, с. 124—130.
162. Стародубцев Н. А. Вопросы организации управления в апериодических вычислительных устройствах.— Автореферат дис.— Л.: ЛЭТИ, 1975.— 16 с.
163. Стародубцев Н. А. Синтез схем управления параллельных вычислительных систем.— Л.: Наука, Ленингр. отд-ние, 1984.— 192 с.
164. Таль А. А., Юдицкий С. И. Иерархия и параллелизм в сетях Петри. I. Сложные сети Петри. II. Сложные автоматные сети Петри с параллелизмом.— Автоматика и телемеханика, 1982, № 7, с. 113—122, № 9, с. 83—89.
165. Таубин А. Р. Анализ апериодических схем. Автореферат дис.— Л.: ЛЭТИ, 1981.— 15 с.
166. Трахтенброт Б. А., Барздинь Я. М. Конечные автоматы. (Поведение и синтез).— М.: Наука, 1970.— 400 с.
167. Тыугу Э. Х. Решение задач на вычислительных моделях.— Журн. вычисл. мат. и мат. физики, т. 10, 1970, № 3, с. 716—733.
168. Уинстон П. Искусственный интеллект.— М.: Мир, 1980.— 520 с.
169. Уткин А. А. Анализ логических сетей и техника булевых вычислений.— Минск: Наука и техника, 1979.— 152 с.
170. Фет Я. И. Параллельные процессоры для управляющих систем.— М.: Энергоиздат, 1981.— 160 с.
171. Финкельштейн Р. Л. Схемотехнические аспекты построения апериодических вычислительных и управляющих устройств.— Автореферат дис.— Л.: ЛИТМО, 1975.— 17 с.
172. Фор Р., Кофман А., Дени-Папен М. Современная математика.— М.: Мир, 1966.— 272 с.
173. Фридман А., Менон П. Теория и проектирование переключательных схем.— М.: Мир, 1978.— 580 с.
174. Хазанов Б. И. Интерфейсы измерительных систем.— М.: Энергия, 1979.— 120 с.
175. Цеманек Г. Последовательностная асинхронная логика.— В кн.: Теория конечных и вероятностных автоматов: Тр. Междунар. симпоз. ИФАК.— М.: Наука, 1965, с. 232—245.
176. Цирлин Б. С. Алгебра асинхронных логических схем.— Кибернетика, 1984, № 1, с. 16—20.
177. Цирлин Б. С. Алгебра и анализ асинхронных логических схем: Препринт.— Л.: ИСЭП АН СССР, 1981.— 40 с.
178. Цирлин Б. С. Базисы реализации схем, не зависящих от скорости.— Изв. АН СССР. Техническая кибернетика, 1981, № 3, с. 121—126.
179. Цирлин Б. С. Вопросы синтеза апериодических схем.— Автореферат дис.— Л.: ЛИАП, 1976.— 16 с.
180. Цирлин Б. С. О реализации асинхронных логических схем.— Кибернетика, 1981, № 5, с. 136.
181. Чеботарев А. Н. Анализ асинхронных логических схем.— Кибернетика, 1980, № 6, с. 14—23.
182. Чеботарев А. Н. Декомпозиция асинхронных логических схем. I, II.— Кибернетика, 1978, № 2, с. 1—9; № 4, с. 6—9.

183. Чеботарев А. Н. Риск в асинхронных логических схемах.— Кибернетика, № 4, 1976, с. 8—11.
184. Чеботарев А. Н. Схемы и автоматы, I, II.— Кибернетика, 1976, № 5, с. 5—9; 1979, № 5, с. 9—14.
185. Чеботарев А. Н., Николенко В. Н. Простые декомпозиции асинхронных схем.— Кибернетика, 1979, № 6, с. 51—55.
186. Чжен Г., Мэннинг Е., Метц Г. Диагностика отказов цифровых вычислительных систем.— М.: Мир, 1972.— 232 с.
187. Шоломов Л. А. Основы теории дискретных логических и вычислительных устройств.— М.: Наука, 1980.— 400 с.
188. Щербаков Н. С. Самокорректирующиеся дискретные устройства.— М.: Машиностроение, 1975.— 214 с.
189. Юдицкий С. А., Тагаевская А. А., Ефремова Т. К. Проектирование дискретных систем автоматики.— М.: Машиностроение, 1980.— 232 с.
190. Якубайтис Э. А. Архитектура вычислительных сетей.— Статистика, 1980.— 279 с.
191. Якубайтис Э. А. Синтез асинхронных конечных автоматов.— Рига: Зинатне, 1970.— 326 с.
192. Яковлев А. В. Проектирование и реализация протоколов асинхронного обмена информацией в межмодульном интерфейсе.— Автореферат дис. Л.: ЛЭТИ, 1982.— 17 с.
193. Akers S. B. A truth table method for the synthesis of combinational logic.— IRE Trans. Electr. Comput., 1961, EC-10, № 4, p. 604—615.
194. Anderson D. A., Metze G. Design of totally selfchecking check circuits for m-out-of-n codes.— IEEE Trans. on Computers, 1973, v. C. 22, p. 263—269.
195. Armstrong D. B., Friedman A. D., Menon P. R. Desing of asynchronous circuits assuming unbounded gate delay.— IEEE Trans. Comput., 1969, v. C-18, № 12, p. 1110—1120.
196. Ashjace M., Reddy S. M. On totally — self-checking checkers for separable codes.— In: Intern. Symp. on Fault-Tolerant computing. S. 1, 1976, p. 151—156.
197. Automatismes a sequences/M. Blanchard, J. C. Cavarroc, J. Gillon et al.— Rapport final du contract DGRST H. 7. 2912/DERA. Tuillet, 1973.— 420 p.
198. Baer J. L. A survey of some theoretical aspects of multiprocessing.— Computing Surveys, 1973, v. 5, № 1, p. 31—80.
199. Bartky W. C., Muller D. E. An Illiac Program of simulating the behaviour of asynchronous logical circuits and detecting undersirable race conditions.— Report of Univ. of Illinois, 1957, № 221.
200. Berger J. M. A note on error detection code for assymmetric channels.— Inf. and Control, v. 4, 1961, № 1, p. 68—73. Рус. пер.: В кн.: Теория кодирования/Под ред. Э. Л. Блоха, М.: Мир, 1964, с. 107—115.
201. Bochmann G. V. A general transition model for protocols and communication services.— IEEE Trans. Comm., 1980, COM-28, № 4, p. 643—650.
202. Bochmann G. V. Finite state description of communications protocols.— Proc. Comp. Network Protocols Symp., Liege, Belgium, 1978, p. F3—1—F3—11.

203. Bochmann G. V., Sunshine C. A. Formal methods in communication protocol design.— IEEE Trans. Comm., 1980, COM-28, p. 624—631.
204. Brand D., Zafiropulo P. Synthesis of protocols for an unlimited number of processes.— Proc. Trends and applications: 1980, Computer Network Protocols, Gaithersburg, Md., 1980, p. 29—40.
205. Bredson J. G., Hulina P. T. Generation of a clock pulse for asynthronous sequential machines to eliminate critical races.— IEEE Trans. Comput., 1971, C-20, № 2, p. 187—188.
206. Bruno J., Altman S. M. A theory of asynchronous control networks.— IEEE Trans. Comput., 1971, v. C-20, № 6, p. 629—638.
207. Bryant R. E. Report on the workshop on self-timed systems. MIT Lab. Comput. Sci. Techn. Memo. № 166.— Cambridge: MIT, 1980.— 21 p.
208. Brzozowski I. A., Yoeli M. Digital networks.— Englewood Cliffs, N. Y.: Prentice-Hall, 1976.— 388 p.
209. Carter W. C., Schneider P. R. Design of dinamically checked computers.— In: Information processing. 1968. Proc. of IFIP Congress 68, v. 2.— Edinburgh, 1968, p. 878—883.
210. Catt I. Time loss through gating of asynchronous logic signal pulses.— IEEE Trans. Electr. Comput., 1966, v. EC-15, № 1, p. 108—111.
211. Cavarroc J. C., Blanchard M., Gillon J. An approach to the modular design of industrial switching systems.— In: Proc. of Intern. Symp. on Discrete Systems, 1974, v. 3, Riga, 1974, p. 93—102.
212. Chaney T. J. Comments on «A note on synchroniser and interlock maloperation».— IEEE Trans. Comput., 1979, v. C-28, № 10, p. 802—804.
213. Chaney T. J. The synchronizer «glitch» problem. Computer Systems Laboratory, TR № 47/Washington Univ., St. Louis, 1974.— 54 p.
214. Chaney T., Molnar C. Anomalous behavior of synchronizer and arbiter circuits.— IEEE Trans. Comput., 1973, v. C-22, № 4, p. 421—422.
215. Chaney T. J., Ornstein S. M., Littlifield W. M. Beware the synchronizer.— In: COMPCON-72: Proc. of the IEEE Comput. Conf. San Francisco, 1972.
216. Chaney T. J., Rosenberger F. U. Characterization and sealing of MOS flip-flop performance in synchronizer applications.— In: Proc. of the Caltech Conference on Very Large Scale Integration, Pasadena, 1979, p. 357—374.
217. Corsini P. *n*-user asynchronous arbiter.— Electronics Letters, 1975, v. 11, № 1, p. 1—2.
218. Corsini P. Self-synchronising asynchronous arbiter.— Digital Processes, 1975, v. 1, № 1, p. 67—73.
219. Courans G. R., Wann D. F. Theoretical and experimental behaviour of asynchronous operating in metastable region.— IEEE Trans. Comput., 1975, v. C-24, № 6, p. 604—616.
220. Dantine A. A. S. Protocol representation with finite-state models.— IEEE Trans. Comm., 1980, COM-28, № 4, p. 632—643.
221. David R. Thotally self-checking 1-out-3 checker.— IEEE Trans. Comput., 1978, C-27, p. 570—572.

222. David R., Thevenod-Fosse P. Design of totally self-checking asynchronous modular circuits.— J. of Design Automation and Fault-Tolerant Computing, 1978, v. 4, p. 271—287.
223. Day J., Sunshine C. A bibliography on the formal specification and verification of computer network protocols.— Comput. Comm. Rev., 1979, v. 9, № 4, p. 23—39.
224. Dennis J. B. First version of a data flow procedure language.— Lect. Notes in Comput. Sci., 1974, № 19, p. 362—376.
225. Dennis J. B. Modular asynchronous control structures for a high performance computer.— In: Rec. of the Project MAC Cong. on Concurrent Systems and Parallel Computation. ACM. N. Y., 1970, p. 55—80.
226. Desing of self-checking microprogram control/R. W. Cook, W. H. Sission, T. F. Storey, W. H. Toy.— IEEE Trans. on Comput. 1973, v. C-22, p. 255—263.
227. Diaz M. Design of totally self-checking and fail safe sequential machines.— In: Intern. symp. on Fault-Tolerant computing.— 1974.— Urbana, 1974, p. 3.19—3.24.
228. Elinean G., Werner W. A new J-K flip-flop for synchronizers.— IEEE Trans. Comput., 1977, v. C-26, № 12, p. 1277—1279.
229. Frazer W. D., Muller D. E. A method for factoring and action of asynchronous circuits.— In: Proc. of the First Annual AIEE Symp. on Switching Circuit Theory and Logical Design. Chicago, 1960, N. Y., 1961, p. 246—249.
230. Friedman A. D. Feedback in asynchronous sequential circuits.— IEEE Trans. Electr. Comput., 1966, v. EC-15, № 5, p. 740—749.
231. Friedman A. D., Menon P. R. Synthesis of asynchronous sequential circuits with multiple-input changes.— IEEE Trans. Comput., 1968, C-17, № 6, p. 559—566.
232. Friedman A. D., Menon P. R. Systems of asynchronously operating modules.— IEEE Trans. Comput., 1971, C-20, № 1, p. 100—104.
233. Garcia M. H. Hardware implementation of communication protocols: A formal approach.— 7th Annual symposium on computer architecture, 1980, p. 253—263.
234. Geffroy J., Diaz M. Unified approach to the study of self-checking systems.— Digital Processes, 1977, № 3, p. 289—306.
235. Gilbert N. E. Lattice theoretic properties of frontal switching functions.— J. of Math. and Physics, 1954, v. 33, № 1, p. 57—67.
236. Gioffi G. Autotesting speed-independent sequential circuits. IEEE Trans. Comput., 1978, C-27, № 1, p. 90—94.
237. Grabowski J. On the analysis of switching circuits by means of Petri nets.— Elektronische Informations-verarbeitung und Kybernetik, 1978, Bd. 14, S. 611—617.
238. Hack M. Analysis of production schemata by Petri nets. Computer Structure Group, TR-94. Project MAC/M. I. T.— Cambridge, 1972.— 119 p.
239. Hack M. Decidability questions for Petri nets. Computer Structure Group, TR-161. Project MAC/M. I. T.— Cambridge, 1976.— 197 p.
240. Hackbart R. R., Dietmeyer D. L. The avoidance and elimination of function hazards in asynchronous sequential circuits.— IEEE Trans. Comput., 1971, C-20, № 2, p. 184—189.

241. Haraugozo J. Protocol definition with formal grammars.— Proc. Comp. network protocols, Liege, 1978, p. F6-1 — F6-10.
242. Hojberg K. S. An asynchronous arbiter resolves resource allocation conflicts on a random priority basis.— Computer Design, 1977, v. 16, № 8, p. 120—123.
243. Holt A. W., Commoner F. Events and Conditions.— In: Record of Project MAC Conf. Concurrent Systems and Parallel Computation, N. Y., 1970, p. 3—52.
244. IEEE Transaction of Communication. Special issue in computer network architectures and protocols, 1980, COM-28, № 4, p. 624—661.
245. Jones N. D., Landweber L. H., Lien Y. E. Complexity of some problems on Petri nets.— Theoretical Computer Sci., 1977, v. 4, № 3, p. 277—299.
246. Jotwani N. D., Jump J. R. Top-down design in the context of parallel program.— Inf. and Contr., v. 40, № 3, 1979, p. 241—257.
247. Jump J. R., Thiagarajan P. S. On the interconnection of asynchronous control structures.— J. of ACM, 1975, v. 22, № 4, p. 596—612.
248. Keller R. M. A fundamental theorem of asynchronous parallel computation.— Lecture Notes in Computer Schience, 1975, v. 24, p. 102—112.
249. Keller R. M. Toward a theory on universal speed-independent modules.— IEEE Trans. Comput., 1974, v. C-23, № 1, p. 21—23.
250. Kinniment D. J., Edwards D. B. G. Circuit technology in a large computer system.— The Radio and Electronic Engineer, 1973, v. 43, № 7, p. 435—441.
251. Kinniment D. J., Woods J. V. Synchronisation and arbitration circuits in digital systems.— Proc. IEE, 1976, v. 123, № 10, p. 961—966.
252. Ko D. C., Brener M. A. Self-checking of multi-output combinatorial circuits using extended prity technique.— J. Design. Automation and Fault-Tolerant Computing, 1978, v. 2, № 1, p. 29—62.
253. Kwong Y. S. On reduction of asynchronous systems.— Theor. Comp. Sci., 1977, v. 5, p. 25—50.
254. Landweber L. H., Robertson E. L. Properties of conflict-free and persistent Petri nets.— J. of ACM, 1978, v. 25, № 3, p. 253—304.
255. Le Moli G. A theory of colloguies.— Alta Frequenza, 1976, v. XLII, № 10, p. 223E — 230E.
256. Littlefield W. M., Chaney T. J. The glitch phenomenon. Computer Systems Laboratory.— Techn. Memo № 10/Washington Univ., St. Louis, 1966.— 16 p.
257. Mago G. Realization methods for asynchronous sequential circuits.— IEEE Trans. Comput., 1971, C-20, № 3, p. 290—297.
258. Marino L. R. General theory of metastable operation.— IEEE Trans. Comput., 1981, C-30, № 2, p. 107—116.
259. Marino L. R. The effect of asynchronous inputs on sequential network reliability.— IEEE Trans. Comput., 1977, v. C-26, № 11, p. 1082—1090.
260. Marked directed graphs/F. Commoner, A. W. Holt, S. Even, A. Pnueli.— J. of Comput. and System Sci., 1971, v. 5, № 5, p. 511—523.

261. Marouf M. A., Friedman A. D. Efficient design of self-checking checker for any m-out-of-n code.— IEEE Trans. on Comput., 1978, v. C-27, № 6, p. 482—490.
262. Mayne D., Moore R. Minimize computer «crashes».— Electronic Design, 1974, v. 22, № 9, p. 168—172.
263. Mayr E. An effective representation of the reachability set of persistent Petri nets.— Lab. Comput. Sci., Techn. Memo, № 188/M. I. T., Cambridge, 1981.— 17 p.
264. Mayr E. Persistance of vector replacement systems in decidable.— Lab. Comput. Sci., Tech. Memo, № 189/M. I. T., Cambridge, 1981.— 18 p. (Acta Informatica, 1981, v. 15, № 3, p. 308—318).
265. Merlin P. M. A methodology for the design and implementation of communication protocols.— IEEE Trans. Commput., 1976, COM-24, № 6, p. 614—621.
266. Merlin P. M. Specification and validation of protocols.— IEEE Trans. Commput., 1979, COM-27, № 11, p. 1671—1680.
267. Merlin P. M., Farber D. J. Recoverability of communication protocols. Implication of a theoretical study.— IEEE Trans. Commun., 1976, COM-24, № 9, p. 1036—1046.
268. Mikami Y. Glitchess TTL arbiter selects first of two inputs.— Electronics, 1977, v. 50, № 12, p. 136.
269. Misunas D. Petri nets speed independent design.— Comm. ACM, 1973, v. 16, № 8, p. 474—481.
270. Muller D. E. Asynchronous logics and application to information processing.— In: Proc Symp. on Application of Switching Theory in Space Technology. Stanford, 1962, p. 289—297.
271. Muller D. E. Lecture Notes on asynchronous circuits theory.— Urbana: Univ. of Illinois, 1961.
272. Muller D. E., Bartky W. C. A theory of asynchronous circuits. Annals of computation laboratory of Harvard University, 1959, v. 29, p. 204—243.
273. Muller D. E., Bartky W. C. A theory of asynchronous circuits. I, II.— Report of Univ. of Illinois, 1956, № 75, 1957, № 78.
274. Nakumura T., Utsunomiya K. On a universal design procedure to realise the semimodular state transition graph.— Digital Processes, 1977, v. 3, № 3, p. 237—257.
275. Net theory and applications.— Lecture Notes in Computer Science.— 1980, v. 84, Berlin, N. Y.: Springer-Verlag, 537 p.
276. Nozaki A., Hazard analysis of asynchronous circuits in Muller-Bartky's sense.— J. of Computer and Systems Sci., 1976, v. 13, № 2, p. 161—171.
277. Ozguner F. Design of totally self-checking asynchronous and synchronous sequential machines.— In: Intern. Symp. on Fault-Tolerant Computing, 1977.— Los Angeles, 1977.— p. 89—97.
278. Patil S. S. An asynchronous logic array. Comp. Structure Group, Memo-62. Project MAC/M. I. T., Cambridge, 1975.— 30 p.
279. Patil S. S. Circuit implementation of Petri nets. Comp. Structure Group, Memo-73. Project MA/M. I. T., Cambridge, 1972.— 14 p.
280. Patil S. S., Dennis J. B. The description and realization of digital systems.— RAIRO, 1973, № 1, p. 55—69.

281. Pearce R. C., Field J. A., Little W. D. Asynchronous arbiter module.— IEEE Trans. Comput., 1975, v. C-24, № 9, p. 931—932.
282. Pechoucek M. Anomalous response times of input synchronizers.— IEEE Trans. Comput., 1976, v. C-25, № 2, p. 133—139.
283. Peterson J. L. Petri nets and the modelling of systems.— Prentice-Hall, N. Y., 1981. (Рус. пер.: Питерсон Дж. Теория сетей Петри и моделирование систем.— М.: Мир, 1984.— 264 с.)
284. Peterson J. L. Petri Nets.— Computing Surveys, 1977, v. 9, № 3, p. 223—252.
285. Petri C. A. Kommunication mit Automaten.— Schriften für des Rheinich-Westfalischen Inst. für Instrumentalle Mathematik. Univ. Bonn.— Bonn, 1962.
286. Petriu E. N-channel asynchronous arbiter resolves resource allocation conflicts.— Computer Design, 1980, v. 19, № 8, p. 126—132.
287. Plummer W. W. Asynchronous arbiters.— IEEE Trans. Comput., 1972, v. C-21, № 1, p. 37—42.
288. Priese L. An automata theoretical approach to concurrency.— Dig. Syst. Lab., Helsinki Univ. of Techn. Oraniemi, Ser. B, 1980, № 12.— 51 p.
289. Reddy S. M. A not on self-checking checkers.— IEEE Trans. on Comput., 1974, v. C-23, № 10, p. 1100—1102.
290. Sechovsky H., Jura S. Asynchronous S-1 arbiter in a form of a hardware control module.— In: AFIPS Conf. Proc. V—45, Nat. Comput. Conf. N. Y., 1976, p. 777—782.
291. Seitz C. L. Self-timed VLSI Systems.— In: Proc. of the Caltech Conference on Very Large Scale Integration. Pasadena, 1979, p. 345—355.
292. Semantics of concurrent computation. Proc. Int. Symp., Evian, 1979.— Lecture Notes Comput. Sci., 1979, v. 70.— 368 p.
293. Sifakis J. A unified approach for studying the properties of transition system.— Theor. Comput. Sci., 1982, v. 18, № 3, p. 227—258.
294. Smith J. E., Metze G. Strongly fault secure logic networks.— IEEE Trans. Comput., 1978, v. C-27, № 6, p. 491—499.
295. Starke P. M. Petri-Netze.— Veb. Dentscher Verlag der Wissenschaften, 1980, Berlin.
296. Strom B. I. Proof of the equivalent realizability of a time-bounded arbiter and a runt-free inertial delay.— In: Proc. of the 6-th Annual IEEE Symp. on Comput. Architecture. N. Y., 1979, p. 178—181.
297. Stucki M. J., Cox J. R. Synchronization strategies.— In: Proc. of the Caltech Conference on Very Large Scale Integration. Pasadena, 1979, p. 375—393.
298. Sunchine C. A. Survey of protocol. Definition and verification techniques.— Proc. Comp. network protocols symp., Liege, Belgium, 1978, p. F1-1 — F1-4.
299. Survey of french research and applications based on Petri nets./ C. Andre, M. Diaz, G. Girault, J. Sifakis.— Lecture Notes Comp. Sci., 1980, v. 84, p. 321—345.
300. Thayse A., Davio M. Boolean differential calculus and its application to switching theory.— IEEE Trans. Comput., 1973, v. C-22, № 4, p. 409—420.

301. The simulation of a switching system's requirements/Gotterez G. Blanchard M., Gillon J. et al.— Proc. of the IFAC Int. Symp. on Descrete Systems, 1974, Riga, v. 3, p. 103—112.
302. Tohma Y., Ohyama Y., Sokai R. Realization of fail-safe sequential machines by using a R-out-of-H code.— IEEE Trans. on Computers, 1971, v. C-20, № 11, p. 1270—1275.
303. Towards analizing and synthesizing protocols/Zafiropulo P. C. H. West, H. Rudin, D. Cowan.— IEEE Trans. Comm., 1980, COM-28, № 4, p. 651—661.
304. Tracey J. H. Internal state assignment for asynchronous sequential machines.— IEEE Trans. Electr. Comput., 1966, EC-15, № 4, p. 551—560.
305. Valette R. Analysis of Petri nets by stepwise refinements. J. of Comput. and Systems Sci., 1979, v. 18, № 1, p. 35—46.
306. Valette R., Diaz M. Top-down formal specification and verification of parallel control systems, Digital Processes, 1978, № 4, p. 181—199.
307. Varschavsky V. I., Rosenblum L. Ya. Dead-beat automata and asynchronous parallel processes control.— In: Preprints of the 1-st IFAC.— IFIP Symp., SOCOCO — 76. Tallin, 1976, p. 161—164.
308. Warshall S. A theorem on boolean matrices.— J. of ACM, 1962, v. 9, № 9, p. 11—12.
309. West C. H. An automated technique of communications protocol validation.— IEEE Trans. Comm., 1978, COM-26, № 8, p. 1271—1275.
310. Wormald E. G. A note on synchronizer or interlock maloperation.— IEEE Trans. Comput., 1977, v. C-26, № 3, p. 317—318.
311. Wormald E. G. Support for Chaney's «Comments on a note on synchronizer or interlock maloperation».— IEEE Trans. Comput., 1979, v. C-28, № 10, p. 804.
312. Yoeli M., Rinou S. Applications of ternary algebra to study of static hazards.— J. of ACM, 1964, v. 11, № 1, p. 84—97.
313. Zafiropulo P. Protocol validation by duologue-matrix analysis.— IEEE Trans. Comm., 1978, COM-26, № 8, p. 1187—1194.
314. Zafiropulo P., Rudin H., Cowan D. Towards synthesizing asynchronius two-process interactions. In. Proc.: Comput. Networking Symp., Gaithersburg, Md., 1979, N. Y., p. 169—175.
315. Zissos D., Duncan F. G. Microprocessor interfaces.— Electr. Letters, 1976, v. 12, № 23, p. 624—625.

Наши слова — великаны, когда они во вред нам, и карлики, когда мы ждем от них пользы.

Ч. Коллинз

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ОГЛАВЛЕНИЕ

Виктор Ильич Варшавский
Михаил Александрович Кишиневский
Вячеслав Борисович Мараховский
Валерий Анатольевич Песчанский
Леонид Яковлевич Розенблюм
Александр Рафаилович Таубин
Борис Соломонович Цирлин

АВТОМАТНОЕ УПРАВЛЕНИЕ АСИНХРОННЫМИ ПРОЦЕССАМИ В ЭВМ И ДИСКРЕТНЫХ СИСТЕМАХ

Под редакцией *В. И. Варшавского*

Редактор *С. В. Петров*
Художественный редактор *Т. Н. Кольченко*
Технический редактор *С. Я. Шкляр*
Корректоры *О. А. Сигал, Н. Д. Дорохова*

ИБ № 12648

Сдано в набор 20.05.85. Подписано к печати 30.04.86. Т-11012. Формат 84×108$^1/_{32}$. Бумага тип. № 1. Гарнитура обыкновенная. Печать высокая. Усл. печ. л. 21. Усл. кр.-отт. 21. Уч.-изд. л. 22,98. Тираж 4700 экз. Заказ 751. Цена 3 р. 20 к.

Ордена Трудового Красного Знамени
издательство «Наука».
Главная редакция
физико-математической литературы
117071 Москва В-71, Ленинский проспект, 15

4-я типография издательства «Наука»
630077 г. Новосибирск 77, Станиславского, 25

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ

Серия «Проблемы науки и технического прогресса»

ГОТОВЯТСЯ К ИЗДАНИЮ В 1987 г.

Гаазе-Рапопорт М. Г., Поспелов Д. А. **От амебы до робота: модели поведения.**

Есть ли общее в поведении животных и человека? Каковы основные процедуры, которые организуют то, что мы называем целесообразным поведением и разумным поведением? Можно ли создать некоторую схему, которая, подобно дереву эволюции, отражала бы постепенное усложнение форм поведения? Насколько наука о поведении живых организмов, включая человека, может быть полезна при создании искусственных систем, наделенных элементами разума? Читатель, не имеющий специальной подготовки в области этологии (науки о поведении) или теории систем искусственного интеллекта, найдет в книге ответы на эти вопросы. Авторы впервые с единых позиций излагают принципы построения моделей поведения, определяемых как чисто физиологическими потребностями животного, так и потребностями человека, живущего в обществе.

Для всех, кто интересуется теорией поведения и проблемами искусственного интеллекта.

Горелов И. Н. **Разговор с компьютером: Психолингвистические аспекты проблемы.**/ С послесловием Д. А. Поспелова.

Специалисты разных стран заняты разработкой ЭВМ пятого поколения. Наступит черед и следующим поколениям компьютеров. Компьютеры будущего будут общаться с человеком не только на своем, машинном, но и на естественном языке человека, будут уметь ориентироваться в среде и в ситуации общения. Чтобы создать такие машины, нужно понять особенности мышления человека, связь речи с мышлением, понять, как происходит понимание текста. Обо всем этом, а также о многом другом (например, о рассудочной деятельности животных, об опытах по обучению обезьян языкам знаков и жестов) рассказывается в книге, написанной строго научно и вместе с тем доступно и увлекательно.

Для всех, кто интересуется современной наукой о языке и мышлении, проблемами искусственного интеллекта и информатики.