В. Г. Лазарев, Н. П. Маркин, Ю. В. Лазарев.

Проектирование дискретных устройств автоматики

Учеб. пособие для вузов связи

М.: Радио и связь
1985.

168 с., ил. 40 к.

Рассматриваются вопросы проектирования применяемых в связи дискретных устройств систем автоматики и вычислительной техники с помощью методов теории автоматов. Излагаются принципы автоматизированного проектирования с использованием ЭВМ. Большое внимание уделяется проектированию дискретных устройств в современных базисах интегральных микросхем и на основе микропроцессоров. Приводятся примеры, иллюстрирующие отдельные этапы синтеза искретного устройства; рассмотрен пример типового задания па курсовое проектирование и даются рекомендации по его выполнению.

Для студентов электротехнических институтов связи по специальности «Автоматическая электросвязь».

ББК 32.88
6Ф1
046(01)—85

РЕЦЕНЗЕНТЫ: Н. Б. СУТОРИХИН, В. И. ШЛЯПОБЕРСКИЙ,
Р. А. АВАКОВ

Редакция литературы по электросвязи

Владимир Георгиевич Лазарев,
Николай Петрович Маркин,
Юрий Владимирович Лазарев

**ПРОЕКТИРОВАНИЕ
ДИСКРЕТНЫХ УСТРОЙСТВ АВТОМАТИКИ**

Редактор Е. В. Комарова
Художественный редактор Р. А. Клочков
Обложка художника Ю. В. Архангельского
Технический редактор Г. И. Колосова
Корректор Т. В. Дземидович

ИБ № 645

Сдано в набор 30.08.84 Подписано в печать 23.11.84
Т-21186 Формат 60X90/,. Бумага писчая № 1 Гарнитура литературная
Печать высокая Усл. печ. л. 10,5 Усл. кр.-отт. 11,0 Уч.-изд. л. 11,73
Тираж 14000 экз. Изд. № 20371 Зак. № 91 Цена 40 к.
Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Московская типография № 5 ВГО «Союзучетиздат»
101000 Москва, ул. Кирова, д. 40

Предисловие

Дискретные устройства нашли широкое применение в различных системах автоматики и вычислительной техники. Особенно велика доля дискретных устройств в системах автоматической электросвязи. При этом все возрастающий объем функций, возлагаемых на автоматические устройства, в сочетании с использованием современного элементного базиса в виде больших и сверхбольших интегральных микросхем (БИС и СБИС) привели к усложнению дискретных устройств и, как следствие этого, к качественному изменению принципов их построения. Так, на смену координатным узлам коммутации с относительно простыми дискретными управляющими устройствами в виде регистров и маркеров приходит новая система .узлов коммутации с программным управлением. В новой системе узлов коммутации процессами обслуживания заявок на соединения управляют электронные управляющие машины (ЭУМ), которые по принципу построения, работе и возможностям не уступают современным высокопроизводительным электронным вычислительным машинам (ЭВМ).

В связи с этим, как и при создании ЭВМ, для успешной разработки дискретных устройств автоматической электросвязи в приемлемые сроки необходима автоматизация процесса их проектирования с максимальным сокращением доли ручного труда. Любая автоматизированная система проектирования должна основываться на использовании той или иной формальной модели. Для дискретных устройств автоматики формальной моделью служит конечный автомат. На его основе разработан ряд формальных методов синтеза дискретных устройств, широко используемых в различных автоматизированных системах проектирования.

Автоматизация проектирования дискретных устройств на основе формализованных методов теории автоматов позволяет существенно сократить сроки их проектирования и повысить качество разработки. Для освоения этих методов студентами вузов связи учебным планом по специальности 0702 «Автоматическая электросвязь» предусмотрены курс «Основы дискретной автоматики» и вводимые в ближайшие годы в соответствии с новым учебным планом курс «Основы цифровой техники и микропроцессоры» и для специализации «Управляющие системы электросвязи» курс «Теория и проектирование управляющих систем электросвязи».

Необходимо отметить, что при изучении ряда разделов этих курсов, несмотря на наличие учебника и монографий по теории автоматов, встречаются определенные трудности. Особенно тяжело преодолевается «барьер» между теорией автоматов и практикой проектирования дискретных устройств. Для обеспечения преодоления этого «барьера» предусмотрено курсовое проектирование.

Настоящее учебное пособие может быть использовано при курсовом проектировании как по дисциплине «Основы дискретной автоматики», так и по новым дисциплинам «Основы цифровой техники и микропроцессоры» и «Теория и проектирование управляющих систем электросвязи». В данном учебном пособии основное внимание уделяется разбору примеров и использованию для построения применяемых в системах электросвязи дискретных устройств современного элементного базиса.

Первая глава пособия является вводной. Она дает представление о принципах построения дискретных устройств, реализованных на основе различного рода интегральных микросхем и микропроцессоров, об особенностях построения наиболее известных автоматизированных систем проектирования дискретных устройств автоматики и вычислительной техники, позволяет понять задачи, .которые возникают при их проектировании.

Основной материал, необходимый для выполнения курсового проекта, содержится в последующих главах пособия. В приложениях приводится пример типового задания на курсовое проектирование и даются рекомендации о порядке его выполнения. Материал подобран так, чтобы студенты при выполнении курсового проекта могли пользоваться в основном данным учебным пособием. Список литературы приведен в минимальном объеме, в него включена только та литература, в которой содержатся основные доказательства, обоснования описанных в пособии методов и предлагаемых для использования в процессе проектирования решений, а также справочный материал. Эта литература может быть использована для более глубокой проработки вопросов, вошедших в рассматриваемый вариант курсового проекта.

Учебное пособие предназначено для студентов электротехнических институтов связи и может быть полезно для разработчиков дискретных систем и устройств, применяемых в электросвязи и других областях техники.

Предисловие и разд. 3.3 написаны В. Г. Лазаревым и Н. П. Маркиным, гл. 1, 3 и разд. 4.1, 5.5—В. Г. Лазаревым и Ю. В. Лазаревым, разд. 2.1, 4.2 и гл. 6, 7—В. Г. Лазаревым, разд. 2.2—2.6, 5.1—5.4—Н. П. Маркиным, приложения—В. Г. Лазаревым, Ю. В. Лазаревым и Н. П. Маркиным.

Авторы

Глава 1.

ОСНОВЫ ПРОЕКТИРОВАНИЯ ДИСКРЕТНЫХ УСТРОЙСТВ

1.1. Принципы построения дискретных устройств автоматики

Дискретные устройства (ДУ) в автоматической электросвязи [1] используются в качестве управляющих, различного рода кодирующих и декодирующих устройств. Они также применяются в системах контроля параметров линий и узлов связи, современной каналообразующей аппаратуре и т. п.

В любом из этих случаев дискретное устройство реализует (аппаратно или программно-аппаратно) заданный алгоритм функционирования, определяющий точный перечень предписаний о порядке выполнения таким устройством тех или иных операций в зависимости от состояния внешней среды, в которой оно функционирует. Внешней средой может быть, в частности, некоторый объект, в котором протекают определенные процессы, управляемые дискретным устройством. В этом случае ДУ

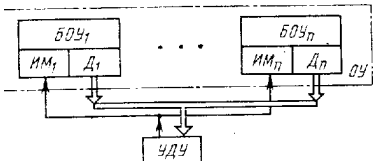
называется *управляющим* (УДУ), а рассматриваемый объект—*объектом управления* (ОУ). На рис. 1.1 двойной стрелкой показан путь передачи от ОУ к УДУ сигналов, характеризующих состояние ОУ (входное воздействие на УДУ), а одинарной стрелкой—путь передачи к ОУ от УДУ сигналов управления (выходное воздействие УДУ), вырабатываемых УДУ в зависимости от поступающего входного воздействия и того алгоритма функционирования, который оно реализует.

Рис. 1.1

ОУ

УДУ

Рис. 1.2



1.2) обычно содержит один или несколько исполнительных механизмов (ИМ), которые обеспечивают прием сигналов управления от УДУ, и датчики Д (сигнализаторы), вырабатывающие в УДУ сигналы о состоянии БОУ. Например, если в качестве ОУ рассматривать многократный координатный соединитель (МК.С), то его блока ми можно считать отдельные выбирающие (ВМ) и удерживающие (УМ) электромагниты с соответствующими контактными группами. Для маркера, используемого в качестве УДУ, исполнительным механизмом в МКС — блоке ОУ, очевидно, будет ВМ (УМ) с выбирающей (удерживающей) рейкой. Датчиком же такого БОУ является контактная группа ВМ (УМ).

Если ДУ применяется как простое контрольное устройство, то внешней средой для него будут, воервых, /га блоков (в частности, один) контролируемого объекта (БКО), от которых поступают сигналы о значениях контролируемых параметров, а вовторых система сигнализации (СС), оповещающая, например, тем или иным способом обслуживающий персонал о несоответствии допустимым нормам одного или нескольких контролируемых. параметров (рис. 1.3).

Аналогичным образом можно всегда выделить внешнюю среду для любого другого ДУ, а также определить характер и состав поступающих на него и вырабатываемых им внешних воздействий, т. е. определить входы и выходы ДУ. В зависимости от сложности (числа компонент, например реле, транзисторов и т. п.) все дискретные устройства по принципу построения можно разделить на две группы: одноблочные и многоблочные. В одноблочных ДУ отдельные компоненты (реле, транзисторы и т. п.) имеют между собой достаточно сильные связи. К одноблочным обычно относятся ДУ с небольшим числом: компонент, например комплекты реле шаговых и декадно-шаговых искателей, содержащих до десяти реле.

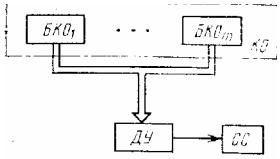


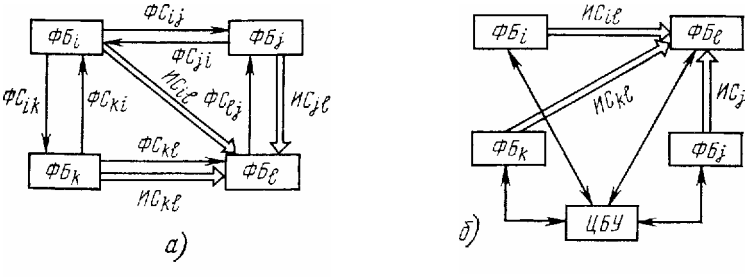
Рис. 1.3

Более сложные дискретные устройства разбиваются на функциональные блоки (ФБ), выполняющие те или иные операции (акты) алгоритма функционирования ДУ. В зависимости от реализуемых операций все ФБ можно разделить на два вида: *логические* (ЛФБ) и *операторные* (ОФБ). Логический ФБ осуществляет проверку какого-либо условия. В качестве ЛФБ могут рассматриваться элементы ДУ, проверяющие состояние различного рода датчиков, например контактов удерживающего или выбирающего электромагнита МКС. В операторных ФБ вырабатываются сигналы управления объектом управления или осуществляется какое-либо преобразование информации. К ОФБ относятся элементы (реле) ДУ, обеспечивающие выработку сигнала включения различных исполнительных механизмов, в том числе удерживающих и выбирающих электромагнитов МКС.

Для того чтобы дискретное устройство могло реализовать тот или иной алгоритм функционирования, между ФБ предусматриваются: функциональные связи ФСij, (связи управления), определяющие в соответствии с заданным алгоритмом порядок работы ФБ и обеспечивающие передачу сигналов взаимодействия ФБ (сигналов включения и выключения ФБ); информационные связи (ИСij), необходимые для передачи от ФБi к ФБj, осведомительной информации о полученном в ФБI ПО окончании его работы результате. Например, если ФБj принял серию импульсов, определяющую цифру номера вызываемого абонента, то от ФБi к Фбij, который должен включиться после этого в работу, по ФСij будет передан сигнал о завершении приема этой серии импульсов, а по ИСij — сама цифра в виде некоторой кодовой комбинации. Следует заметить, что по окончании работы ФБL по *ФСij* передается сигнал, включающий в работу ФБij информация о полученном в ФБ, результате (например, о цифре номера вызываемого абонента) может быть при этом передана в другой ФБl, *l<>j* (накопитель), или совсем не передаваться. Таким образом, структура взаимодействия ФБ по информационным связям может не совпадать со структурой взаимодействия ФБ по функциональным связям.

К многоблочным ДУ относятся общие управляющие устройства (УУ) — такие, как маркеры,

регистры и тем более ЭУМ. В зависимости от принципа взаимосвязи ФБ через ФС многоблочные ДУ делятся на два типа: с рассредоточенными функциональными связями (рис. 1.4,а) и с концентрированными функциональными связями (рис. 1.4,б). К первому типу относятся регистры и маркеры, а ко второму — ЭУМ и все другие устройства программного типа [2, 3].



Как видно из рис. 1.4,б, в многоблочных ДУ второго типа, в отличие от ДУ первого типа (см. рис. 1.4,а), функциональные связи между отдельными ФБ отсутствуют. В ДУ с концентрированными ФС имеется центральный блок управления (ЦБУ), который связан функциональными связями со всеми ФБ. По этим связям от ФБ (ЛФБ) в ЦБУ поступает осведомительная информация, а от ЦБУ к ФБ (ОФБ или ЛФБ) —управляющая информация. В таких ДУ, называемых *программными,* порядок работы ФБ определяется выдаваемой из ЦБУ управляющей информацией в зависимости от поступающей от ЛФБ в ЦБУ осведомительной информации и заложенным в ЦБУ алгоритмом функционирования.

Алгоритм функционирования любого дискретного управляющего устройства может рассматриваться как его алгоритмическое описание. В случае, когда по заданному алгоритму функционирования построена структурная схема дискретного устройства в том или ином элементном базисе, говорят, что алгоритмическоеописание переведено в структурное, представляющее собой структурную модель заданного алгоритма функционирования ДУ.

Альтернативой структурному моделированию алгоритма функционирования является программное моделирование. При программном моделировании алгоритм функционирования ДУ представляется в виде последовательности так называемых команд, каждая из которых определяет операцию над множеством исходных или промежуточных данных (операндов) и номер следующей команды. В качестве операнд могут рассматриваться как отдельные одно- и многоразрядные числа, хранящиеся в запоминающих устройствах (ЗУ), так и наборы значений входных сигналов (в частности, одного сигнала), которые также могут быть предварительно записаны в ЗУ в виде многоразрядных чисел.

Упорядоченная совокупность команд, однозначно описывающая заданный алгоритм функционирования, называется *программой,* а представление алгоритма функционирования в виде программы—*программным моделированием алгоритма.* Для хранения программы может быть использовано запоминающее устройство, в котором для каждой команды обычно предусматривается отдельная ячейка. Следует заметить, что структурой дискретного устройства, аппаратно реализующей его алгоритм функционирования, обеспечивается активное выполнение последнего в соответствии с воздействующими на ДУ входными сигналами.

В отличие от аппаратной реализации при структурном моделировании алгоритма функционирования, программа, реализующая тот же алгоритм при программном моделировании, является пассивной, т. е. она является не устройством, а некоторой записью данного алгоритма. Для чтения этой записи и выполнения команд программы используется универсальное дискретное устройство — *процессор* (Пр). При этом процессор выполняет любую, но в один и тот же момент одну операцию имеющуюся в программе (;зис. 1.5,а,1.

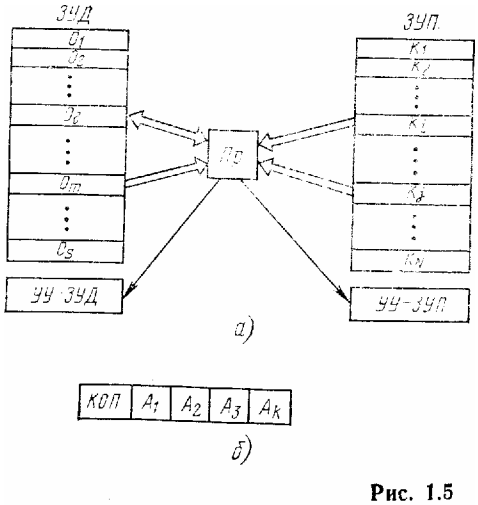


Рис. 1.5

Процессор вырабатывает в УУ запоминающего устройства для хранения программ (УУ-ЗУП) сигнал на считывание из него очередной команды, например *Кг* После того, как команда *Ki,* содержащая (рис. 1.5,б) код операции (КОП), которую необходимо выполнить над операндами, хранящимися в ЗУ данных (ЗУП) в ячейках Oi и Om, адреса которых (А1 и А2 соответственно) указаны в команде, поступит в процессор, он вырабатывает а УУ-ЗУП сигнал считывания этих операнд из ЗУД. Результат выполнения над ними операции, код которой указан в команде *Ki,* отправляется в ячейку (пусть *Oi)* ЗУП по адресу Аз. По окончании выполнения команды *Кг* процессор вырабатывает сигнал в УУ-ЗУП на считывание очередной команды *Ki* адрес An, который указан в реализованной команде. После считывания из ЗУП команды *К;* процессор настраивается на выполнение той операции, код которой указан в этой команде, и т. д.

Таким образом, если при структурном моделировании заданного алгоритма функционирования ДУ аппаратно реализует весь алгоритм, то при программном моделировании процессор обеспечивает реализацию всего алгоритма функционирования путем последовательного выполнения четко разграниченных отдельных операций (актов алгоритма), определяемых хранящейся в ЗУП программой. Легко понять, что при структурном моделировании изменение алгоритма функционирования требует изменения структурной схемы дискретного устройства, тогда как при программном моделировании необходимо соответственно изменить лишь хранящуюся в ЗУП программу. В связи с этим управляющие устройства при программном моделировании алгоритма функционирования являются достаточно универсальными ДУ, тогда как ДУ с аппаратурной реализацией ориентированы на выполнение вполне определенного алгоритма функционирования.

Следует заметить, что при программном моделировании могут быть реализованы только те алгоритмы функционирования, которые удастся представить в виде программы, команды которой содержат операции, реализуемые в процессоре данного программного ДУ. Однако на практике число выполняемых процессором операций выбирается таким, что с их помощью можно описать любой алгоритм функционирования. Такой набор операций называется *функционально полным.* Вместе с тем функционально полные наборы могут заметно отличаться друг от друга составом операций. При этом один набор позволяет более компактно описать один класс алгоритмов, а другой набор — иной класс. В таких случаях говорят, что наборы операций являются проблемно-ориентированными, т. е. специализированными для описания алгоритмов решения тех или иных проблем.

Структурное (аппаратное) и программное моделирования являются крайними принципами реализации алгоритма функционирования ДУ. Промежуточным между ними является принцип реализации структуры ДУ в базисе однородных сред, рассматриваемых в гл. 5.

Успехи в области микроэлектроники привели к созданию дешевых и малогабаритных больших и сверхбольших интегральных схем, которые позволили размещать функциональные блоки ДУ либо непосредственно на самом ОУ, либо в его отдельном блоке **ОУ,** существенно сократив тем самым проводность такого децентрализованного ДУ (рис. 1.6,а). Наличие среди БИС и СБИС высокопроизводительных микропроцессоров обеспечивает возможность построения распределенных ДУ (рис. 1.6,б) без ЦБУ. На этой основе стало целесообразным строить ДУ

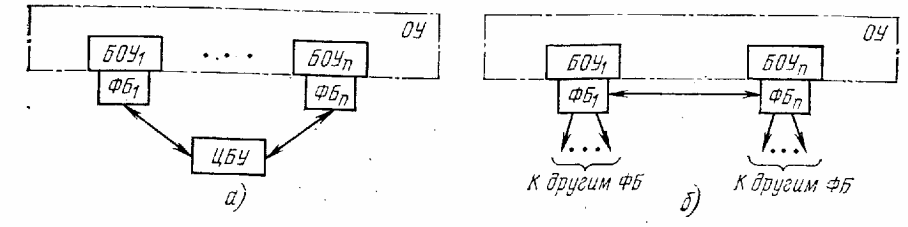


Рис. 1.6

сосредоточенными ФС, фунциональные блоки которого не сосредоточены в одном устройстве, а распределены по БОУ. Такой принцип, называемый распределенным принципом построения ДУ, все шире используется при создании дискретных устройств и систем автоматики [4].

1.2. Этапы проектирования дискретного устройства

Процесс проектирования современных дискретных устройств сложен, поэтому его выполняют в несколько этапов. Как правило, выделяют следующие основные этапы проектирования: системное, программно-логическое, техническое и технологическое.

На этапе *системного проектирования* по заданным тактико-техническим требованиям к проектируемому устройству, обычно сформулированным в виде технического задания (ТЗ), составляется общее алгоритмическое описание дискретного устройства, представляющее собой задание условий его работы (т. е. алгоритм функционирования), по которому разрабатывается блочная структура устройства или, как говорят, архитектура устройства.

Основной исходной информацией для выполнения этапа системного проектирования служат содержащиеся в ТЗ сведения о назначении ДУ и тех функциях, которые оно должно реализовать, а также основные параметры (часто с указанием допустимых пределов отклонения от их средних значений) проектируемого. ДУ — быстродействие, надежность, стоимость, габаритные размеры и др.

Результатами выполнения этапа системного проектирования являются состав блоков, структура их соединений и общее алгоритмическое описание каждого из блоков. Обычно алгоритмическое описание каждого из блоков отличается от алгоритмического описания всего ДУ более детальной формулировкой реализуемых функций. Так, если в качестве отдельных операций в алгоритмическом описании всего ДУ

рассматриваются макрооперации, выполняемые целым блоком, то в алгоритмическом описании соответствующего блока такая макрооперация расшифровывается и представляется уже в виде частного алгоритма выполнения более мелких операций. Заметим, что для достаточно сложных устройств удобно каждый из блоков ДУ представлять в виде совокупности подблоков. Тогда каждая из операций, реализуемая блоком, при алгоритмическом описании подблока представляется в виде частного алгоритма выполнения еще более мелких операций— микроопераций.

Алгоритмическое описание ДУ, блока и подблока может быть представлено как на обычном неформализованном, например русском языке, так и на одном из формализованных языков, используемых в теории автоматов [I, 3]: на языке таблиц переходов, языке логических схем алгоритмов и т. д. При этом очевидно, что для автоматизации процесса проектирования ДУ необходимо использование одного или нескольких формализованных языков.

На этапе системного проектирования выбирается также принцип реализации алгоритма функционирования всего ДУ и/или его блоков — аппаратный, программный с использованием одной из выпускаемых ЭВМ или программно-аппаратный с разработкой специализированного программного устройства (машины).

В течение многих лет задачи, возникающие на этапе системного проектирования, выполнялись проектировщиком в основном интуитивно, на базе его личного опыта, поэтому получаемые при этом решения в значительной степени определялись квалификацией и опытом проектировщика. И сама формализация алгоритма функционирования, т. е. запись условий работы дискретного устройства на языке также осуществлялась проектировщиком на основе опыта и интуиции, что не способствовало получению оптимальных решений на последующих этапах. В последнее время были разработаны формализованные методы решения отдельных задач системного проектирования, что позволило на этапе системного проектирования использовать ЭВМ. Это не только ускорило процесс проектирования, но и повысило качество получаемых решений.

В настоящее время на этапе системного проектирования очень широко используется моделирование процесса функционирования на ЭВМ. Обычно анализ вытекающих из ТЗ особенностей функционирования проектируемого ДУ с целью определения основных принципов и возможных вариантов его построения производится опытным специалистом. Дальнейший процесс системного проектирования может выполняться с применением ЭВМ. При этом по определенным критериям (производительности, надежности, стоимости и т. п.) с помощью ЭВМ из множества вариантов, отобранных для рассмотрения специалистом, может быть выбран один наилучший вариант разбиения ДУ на блоки с указанием схемы организации обмена между блоками и принципа реализации как блоков, так и всего ДУ в целом, а также распределения «функций между аппаратными и программными средствами. На этом же этапе может быть выбран один или несколько элементных базисов.

При оптимизации блочной структуры ДУ может использоваться статистическое моделирование.

Процесс системного проектирования носит, как правило, итерационный характер. Вначале принимаются во внимание лишь самые общие свойства условий работы ДУ, на основе которых определяется первоначальный вариант принципа построения ДУ. Затем постепенно в процессе проектирования вводятся дополнительные условия и осуществляется все большая детализация принципа построения ДУ. В результате получается структура ДУ, соответствующая предъявляемым к нему требованиям. Естественно, число итераций и длительность этапа системного проектирования существенно зависят от сложности проектируемого ДУ. Если для достаточно сложных ДУ типа ЭУМ этап системного проектирования может занимать десятки процентов времени разработки всей ЭУМ, то при проектировании, скажем, счетчика на десять импульсов этап системного проектирования не выделяется, а отдельные задачи, обычно выполняемые на этапе системного проектирования (например, выбор элементного базиса), решаются непосредственно на этапе программно-логического или технологического проектирования.

На *этапе программно-логического проектирования* разрабатываются функциональные схемы каждого из блоков. При этом по общему алгоритмическому описанию, заданному на одном из формализованных языков, производится построение функциональной схемы в заданном базисе логических элементов.

В связи с широким использованием в устройствах автоматики микропроцессоров, ЭВМ и других программных дискретных устройств процесс логического проектирования в том или ином базисе элементов заменяется для программных ДУ процессов программирования алгоритма функционирования в базисе команд, применяемого программного устройства (микропроцессора, микро-ЭВМ и т. п.).

При программно-аппаратной реализации ДУ на данном этапе осуществляются логическое проектирование аппаратной части ДУ в базисе логических элементов и программирование алгоритма функционирования в базисе выбранной системы команд. Учитывая, что во многих современных ДУ в той или иной степени используются микропроцессоры и другие программные устройства в настоящее время удобно говорить об этапе программно-логического проектирования, а не логического, как это было в прошлом, когда принцип построения дискретных устройств автоматики базировался лишь на основе аппаратной реализации алгоритма функционирования ДУ. При этом следует заметить, что в

случае применения для построения ДУ достаточно сложных БИС этап логического проектирования может существенно упроститься, а если для каждого из функциональных блоков предусмотрена одна стандартная БИС, надобность в нем совсем отпадает, так как число и состав БИС, а также их взаимосвязь могут определиться уже на этапе системного проектирования.

При логическом проектировании дискретного устройства возможно использование интуитивных приемов построения функциональной схемы. Однако в теории автоматов разработаны достаточно эффективные формализованные методы логического синтеза функциональных схем дискретных устройств, которые позволяют на этапе логического проектирования широко применять ЭВМ.

При программировании также обычно используются формализованные методы разработки программ, основанные на алгоритмических языках высокого уровня, обеспечивающих автоматизацию процесса программирования.

Полученные на этапе программно-логического проектирования результаты в виде функциональных схем ДУ и программ являются исходным материалом для выполнения следующего этапа — этапа технического проектирования. Целью *этапа технического проектирования* является построение принципиальных и монтажных схем, а для программных ДУ — комплексирование средств вычислительной техники, т. е. выбор состава и типа устройств (процессора, внешних и внутренних запоминающих устройств, вводно-выводных устройств и т. п.), обеспечивающих с заданными характеристиками выполнение алгоритма функционирования проектируемого ДУ.

При техническом проектировании аппаратной части ДУ решаются в основном следующие задачи [5]:

1 На основе функциональной схемы всех блоков и их связей производится разбиение логических элементов по корпусам заданной системы элементов с выбором типов корпусов, корпусов по платам (ячейкам), плат по панелям (кассетам), а панелей по стативам (шкафам) с учетом минимальной межкорпусной, межплатной межпанельной и межстативной проводности монтажа.

Таким образом, после решения этой задачи будет **известно,** какие элементы находятся в каждом используемом для построения дискретного устройства корпусе, какие корпуса **находятся** на определенной плате, какие платы—в определенной кассете и какие кассеты—на определенном стативе.

2 Для получения разбиений производится размещение элементов в корпусе, корпусов на плате, плат в кассете, кассет на стативе Таким образом, с каждым логическим элементом функциональной схемы сопоставляется логический элемент определенного корпуса (а значит, на функциональной схеме могут быть размещены выводы корпусов), для каждого корпуса определяется его месторасположение на плате, для каждой платы — ее месторасположение в кассете, а кассеты — на стативе.

3 Для полученных размещений производятся трассировка печатных соединений на платах, трассировка соединений в жгутах между платами, кассетами и стативами.

В настоящее время разработано большое число алгоритмов выполнения каждой из этих трех задач этапа технического проектирования, которые положены в основу разнообразных и достаточно эффективных автоматизированных систем технического проектирования дискретных устройств. Сейчас этап технического проектирования практически всеми достаточно крупными фирмами выполняется с помощью автоматизированных систем проектирования с использованием ЭВМ.

По окончании этапа технического проектирования должна быть получена полная документация в установленной форме на изготовление дискретного устройства.

Цель *этапа технологического проектирования* состоит в разработке технологических процессов и операций по изготовлению ДУ. В результате выполнения этого этапа должна быть получена в полном объеме документация на создание технологического процесса изготовления ДУ. Особенно важное значение этап технологического проектирования приобретает при создании новых типов БИС, в частности заказных. В этих случаях на данном этапе разрабатывается технология изготовления БИС. Полученная при этом документация должна быть выдана в таком виде, чтобы она могла быть использована непосредственно (без переработки вручную) в качестве исходной документации в системах автоматизированного изготовления БИС.

Последовательность этапов проектирования ДУ может быть представлена в виде схемы рис. 1.7.

После получения технического задания на разработку, содержащего условия работы и основные характеристики проектируемого устройства (быстродействие, допустимые габаритные размеры и массу, надежность и т. п.), проектировщик приступает к системному проектированию. После системного проектирования начинается программно-логическое проектирование каждого из блоков дискретного устройства, выделенных на этапе системного проектирования. При этом в ряде случаев в процессе программно-логического проектирования выясняется нецелесообразность решения, принятого на этапе системного проектирования. Поэтому приходится возвращаться к этапу системного проектирования для коррекции принятых решений по результатам программно-логического проектирования.

В результате такого интерационного процесса строятся функциональные схемы и/или составляются программы для всех выделенных блоков дискретного устройства, которые являются исходными

данными для технического проектирования. В процессе технического проектирования может также возникнуть необходимость в коррекции функциональных схем и даже в коррекции разбиения дискретного устройства на блоки, в коррекции выбранной системы элементов и т. п. Таким образом, приходится вновь возвращаться к логическому или к системному проектированию.

Необходимость коррекции принятых на этапе технического проектирования решений может возникнуть и при технологическом проектировании. При изготовлении, наладке и/или испытании также может выясниться, что требуется коррекция **принятых** решений на любом из этапов проектирования.

Таким образом, процесс создания дискретного устройства представляет собой обычно многоитерационный процесс, в котором чередуются все указанные выше этапы проектирования. Этапы технического проектирования выполняются с учетом допустимых конструктивных решений, обеспечивающих унификацию оборудования и устройств автоматики и вычислительной техники.

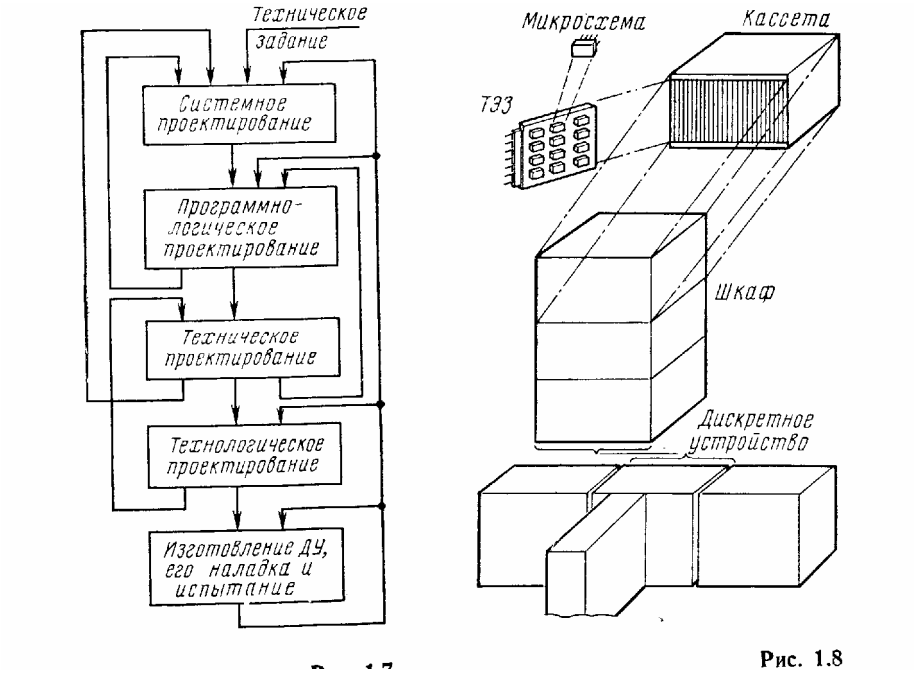


Рис. 1.8

Под конструкцией современного дискретного устройства автоматики понимается комплекс различных по своей природе деталей, объединенных определенным образом электрически и механически друг с другом и призванных выполнять заданные функции в заданных условиях и режимах эксплуатации. Для обеспечения высоких требований к конструкции ДУ, реализованного в базисе интегральных схем, разработаны рекомендации и ГОСТ, регламентирующие виды и типы конструктивных единиц дискретного устройства [б].

Конструкцию микроэлектронного ДУ принято рассматривать как совокупность конструктивных единиц, находящихся в определенной иерархической соподчиненности. Исходная конструктивная единица этой иерархии—корпус интегральной микросхемы. При этом стремятся для построения одного ДУ использовать один тип корпуса, являющегося исходной унифицированной единицей — конструктивным элементом. Кроме этого, унифицируются и другие конструктивные единицы ДУ, так как только в этом случае оно будет технологичным в производстве, надежным в работе, удобным в наладке, ремонте и эксплуатации.

В сложных дискретных устройствах типа ЭВМ выделяют пять структурных уровней (рис. 1.8).

На *нулевом уровне* находится конструктивно неделимый элемент—корпус интегральной микросхемы. Определенное число корпусов микросхем, объединенных на одной плате, образуют конструктивную единицу *первого уровня.* Разработан ряд типов стандартных плат. Платы различаются по типу размещаемых на них корпусов, допустимому числу корпусов на плате и числу выводов с платы. Такая плата представляет собой съемную конструктивную единицу, так как отдельные корпуса на плате делаются несъемными. При наличии повреждения в микросхеме из ДУ изымается вся плата. Плата с неисправной микросхемой подвергается ремонту в специальных мастерских.

Таким образом, простейшей конструктивной единицей, подлежащей замене при наличии повреждения, является плата. В связи с этим такую плату в настоящее время принято называть типовым элементом замены (ТЭЗ).

Ко *второму уровню* относятся конструктивные единицы, предназначенные для механического и электрического объединения конструктивных единиц первого уровня, т. е. ТЭЗов. Часто конструктивные единицы второго уровня, называемые кассетой, субблоком и т. п., в отличие от ТЭЗ,

имеют лицевую панель, на которую выведена индикация о состоянии входящих в данную кассету отдельных функциональных блоков ДУ (например, индикация о состоянии разрядов того пли иного регистра), а также ключи и кнопки управления, гнезда и т. п. Однако, как правило, панель одной кассеты не имеет законченного самостоятельного значения для данного ДУ. Она является или частью общей панели, расположенной на нескольких кассетах, или общей панелью всего дискретного устройства, на которой расположены индикация и приборы управления для всех кассет, образующих устройство.

Кассета может быть выполнена съемной или несъемной. Однако даже при съемном варианте выполнения кассеты при возникновении в ней повреждений заменяются только ТЭЗы. Съем кассеты осуществляется только при очень крупных повреждениях. Съемная конструкция кассет удобна при транспортировке, сборке и пусковой наладке ДУ.

Кассеты собираются на стативе или в шкафу, которые представляют собой *третий уровень* конструктивной иерархии дискретного устройства. Следует заметить, что для ЭУМ и других микроэлектронных дискретных управляющих устройств узлов коммутации, как и для ЭВМ, наиболее характерным является шкафный, а не стативный принцип построения конструктивной единицы третьего уровня. При этом для более компактной компоновки блоков устройства, сокращения длины соединений, что очень важно для дискретных устройств с высоким быстродействием, кассеты в шкафу размещаются на нескольких рамах (двух-трех), которые для удобства эксплуатации (простоты доступа к ТЭЗ) все или крайние делаются поворотными.

Сложное дискретное устройство может размещаться на нескольких стойках или в нескольких шкафах, соединенных кабелем. Совокупность таких стоек или шкафов определяет верхний *четвертый уровень* конструктивной иерархии дискретного устройства.

Многоуровневый (пятиуровневый) метод компановки конструктивных единиц ДУ позволяет решить следующие задачи:

организовать производство по независимым циклам для каждого структурного уровня;

упростить задачи автоматизации на этапах технического и технологического проектирования;

унифицировать стендовую аппаратуру для испытания конструктивных единиц;

сократить период настройки, так как может быть произведена предварительная настройка отдельных конструктивных единиц;

упростить эксплуатационное обслуживание дискретного устройства.

Число уровней конструктивной иерархии может быть изменено в сторону как увеличения, так и уменьшения в зависимости от функциональной сложности ДУ, уровня технологии его изготовления и элементной базы. Например, если устройство достаточно простое, то оно все может быть построено на базе одной кассеты. Если для построения дискретного устройства используются СБИС, то может исключаться первый уровень иерархии, так как СБИС заменяет ТЭЗ. При этом при неисправности может быть осуществлена замена на уровне как СБИС, так и кассеты. В последнем случае кассета представляет собой некоторый микро-ТЭЗ.

Если дискретное устройство, например УДУ, строится по децентрализованному или распределенному принципу, то его отдельные конструктивные единицы, находящиеся на БОУ, могут размещаться непосредственно в конструкции соответствующего БОУ. Связь между разнесенными на расстояние такими конструктивными единицами осуществляется по каналам связи или даже через сеть связи [4].

1.3. Задачи и особенности построения автоматизированных систем проектирования

Для облегчения и ускорения процессов проектирования, а также для получения более качественных решений в течение последних 15—20'лет усиленно ведутся исследования по автоматизации процесса проектирования с использованием ЭВМ. На основе опыта создания систем автоматизации проектирования дискретных устройств автоматики и ЭВМ выделяются две основные разновидности таких систем:

автоматическая система сквозного проектирования;

автоматизированная система проектирования.

В первой системе не предполагается участие человека в процессе проектирования. Такая система требует введения только исходной информации, а само проектирование производится ЭВМ без вмешательства человека. Однако практика разработки таких систем показала, что не только при существующем парке ЭВМ, но при парке ЭВМ в обозримый период времени такая система практически не может быть создана не только для всех этапов проектирования, но и для каждого этапа в отдельности. Примером таких систем является одна из первых систем — система сквозного синтеза АВТОМАТ, разработанная в Сибирском физико-техническом институте. Эта система предназначается для

автоматизации только логического синтеза, при этом с ее помощью могут быть построены только несложные дискретные устройства, условия работы которых хорошо описываются таблицами переходов.

Позднее эта система была преобразована на основе развитого языка программирования логических задач ЛЯПАС в автоматизированную систему [7]. Практический интерес в настоящее время представляет вторая разновидность систем автоматизации проектирования.

В автоматизированной системе проектирования (АСП) разумно сочетаются опыт и интуиция проектировщика при решении сложных многоэкстремальных задач с быстродействием ЭВМ, выполняющей громоздкие вычисления, благодаря чему проектировщик может рассмотреть большое число возможных вариантов. Таким образом, в автоматизированной системе проектирования процесс проектирования основан на диалоге между человеком и машиной.

При разработке принципа построения и функционирования АСП необходимо в первую очередь выбрать принцип взаимодействия заказчика с ЭВМ,'в которой реализована АСП, и язык общения. В системе взаимодействия заказчика с АСП удобно выделить три элемента (рис. 1.9); заказчика, приемщика и проектировщика.

Заказчик формулирует задачу проектирования и по промежуточным результатам, получаемым проектировщиком на отдельных этапах процесса проектирования, осуществляет ее корректиров-

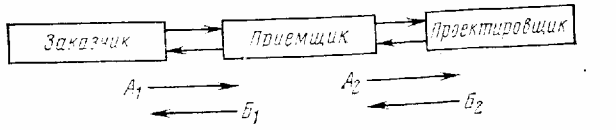


Рис. 1.9

ку и уточняет задание на проектирование очередного этапа или дает команду проектировщику на повторное проектирование выполненных им ранее одного или нескольких этапов проектирования по уточненным или измененным условиям.

Заказчик должен иметь представление об общих возможностях АСП без каких-либо детальных сведений о структуре АСП и общаться с АСП на обычном неформальном русском языке, используя привычные ему термины и понятия, которые могут быть различными для разных заказчиков. Вместе с тем от заказчика требуется, чтобы в процессе формулировки задачи проектирования он использовал условия проектирования одних и тех же терминов для одних и тех же понятий-объектов. Кроме того, заказчик должен выполнять элементарные операции по осуществлению общения с системой (вызов системы или одной из ее подсистем, порядок передачи задания в систему т. п.).

Очевидно, что любая АСП, реализованная на ЭВМ, должна основываться на использовании формальных языков описания условий проектирования и формальных методов проектирования. В АСП не должно существовать различных толкований терминов и обозначений. Поэтому для упрощения взаимодействия заказчика и проектировщика целесообразно использовать приемщика, в задачу которого входит осуществление взаимосвязи заказчика и проектировщика, а именно перевод условий проектирования с неформального языка заказчика (фраз на русском языке, эскизов и т. п.) на формальный язык проектировщика и, наоборот, перевод результатов проектирования с формального языка проектировщика в удобную для заказчика форму их представления (в виде текста на русском языке, графиков, схем и т. п.). Такие функции приемщика по переводу с одного языка на другой принято называть *трансляцией,* а устройство (или программа для ЭВМ), осуществляющее такой перевод, — *транслятором.*

Для упрощения взаимодействия заказчика с приемщиком (транслятором) и приемщика с проектировщиком, а в случае программной или аппаратной реализации транслятора для упрощения программы-транслятора или устройства-транслятора целесообразно в каждой из таких пар взаимодействующих партнеров выделить одного главного (активного) партнера, который мог бы вести диалог, оставив для другого партнера (пассивного) лишь право отвечать на поставленные активным партнером вопросы. При этом, допуская для каждого из партнеров этих двух пар взаимодействующих партнеров активное или пассивное состояние в процессе диалога, можно выделить четыре варианта (см. рис. 1.9):

активный заказчик (Ai) в первой паре «заказчик—приемщик» и активный приемщик *(А2)* во второй паре «приемщик — проектировщик» (обозначим это сочетание через А1А2);

активный приемщик (Б1) в первой паре и активный проектировщик (Б2) во второй паре *(Б1Б2);*

активный заказчик (А1) в первой паре и активный проектировщик (Б2) во второй паре *(А1Б2};*.

активный приемщик (Б1) в первой паре и активный приемщик (А2) во второй паре *(Б1А2).*

В большинстве АСП принят вариант *Б1А2.*.

Учитывая, что заказчиком является человек, а в качестве проектировщика в АСП используется система программ, представленная в ЭВМ, в зависимости от вида реализации приемщика можно выделить три принципа реализации АСП (рис. 1.10).

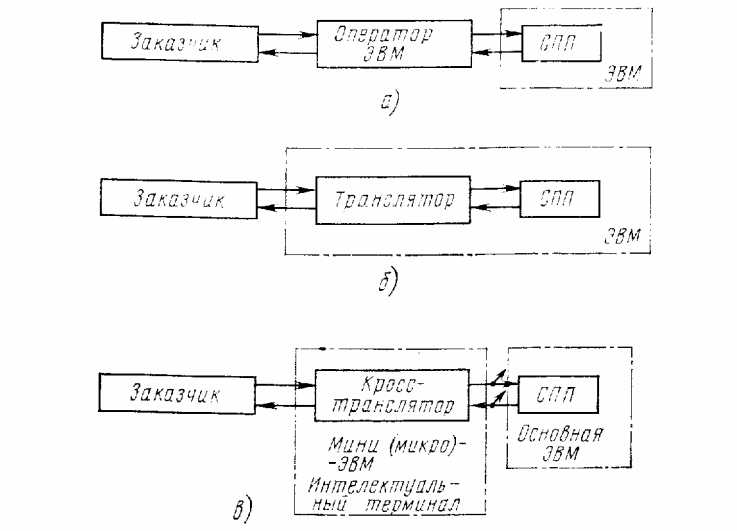


Рис. 1.10

В первом из этих принципов (рис. 1.10,а) приемщиком является оператор ЭВМ, владеющий формализованным языком проектирования. Такой принцип целесообразно применять в том случае, когда система программ проектирования (СПП) реализуется в небольшой ЭВМ, в которой из-за ограничения памяти или производительности невозможно выполнение функции приемщика с достаточно мощным сервисным программным обеспечением, облегчающим взаимодействие человека-заказчика с ЭВМ. При использовании мощных ЭВМ приемщиком может служить программа-транслятор вместе с системой программ проектирования в виде АСП (рис. 1.10,б). С появлением достаточно мощных мини-ЭВМ и особенно микро-ЭВМ с развитым программным обеспечением стало удобно функции приемщика реализовать как кросс-транслятор (рис. 1.10,в), т. е. в виде системы программ мини (микро)-ЭВМ, используемой в качестве интеллектуального терминала, подключенного к основной ЭВМ с реализованной на ней СПП. Такие терминалы могут взаимодействовать с основной ЭВМ в режиме разделения времени. При этом возможна специализация кросс-трансляторов по обслуживаемым различным заказчикам и различным областям использования АСП.

Рассмотрим некоторые из известных АСП. Наиболее развитыми автоматизированными системами проектирования сложных дискретных устройств типа ЭВМ являются системы ПРОЕКТ, разработанная в Институте кибернетики АН УССР под руководством академика В. М. Глушкова [8], и АСП-1, разработанная под руководством члена-корреспондента АН СССР Н. Я. Матюхина и доктора технических наук Е. И. Гурвича [9].

В АСП ПРОЕКТ особо следует выделить операционную часть, структура и организация которой позволяют практически неограниченно расширять систему путем усовершенствования реализуемой методики проектирования и увеличения числа программ. Список директив и семейство разработанных языков делают удобным общение проектировщика с системой.

Система наиболее полно разработана в части логического проектирования, меньшее внимание уделено вопросам технического проектирования. Вместе с тем в ней решается ряд задач технологического проектирования.

В системе АСП-1 полно проработан этап технического проектирования, тогда как задачи логического проектирования решаются в основном методами моделирования.

В Институте проблем управления под руководством члена-корреспондента АН СССР М. А. Гаврилова [10] разработана диалоговая автоматизированная система проектирования дискретных устройств промышленной автоматики (ДАСП), имеющая следующие особенности:

1. Система построена по модульному принципу. Каждая из подсистем, на которые она разбита, может использоваться независимо от других и в свою очередь состоит из отдельных программ (модулей), согласованных по входной и выходной информации в рамках одной подсистемы или всей ДАСП. Модульный принцип позволяет использовать для проектирования отдельные пакеты программ, не дожидаясь полного окончания разработки ДАСП, делает ее гибкой к модификациям и расширению путем создания новых модулей на базе старых или введния тех, которые отсутствуют в системе.

2. Для каждой подсистемы ДАСП (кроме обеспечивающейввод условий работы проектируемого дискретного устройства) исходная информация выбирается из единого банка данных, а результат проведенного ею проектирования отсылается обратно в банк данных и может быть использован любой другой подсистемой для дальнейшего проектирования. Одной из основных форм представления информации об устройстве внутри ЭВМ является система булевых функций в нормальной и скобочной формах, для работы с которой специально разработаны алгоритмы и программы, относящиеся к различным подсистемам.

3. Система имеет трехступенчатую иерархию языков описания условий работы ДУ: первичные,

базовые и автоматные. Первичные языки относятся к проблемно-ориентированными обладают развитыми изобразительными средствами, удобными для описания условий работы ДУ того или иного класса (в зависимости от конкретной области применения устройства). Первичные языки близки к естественным. В отличие от первичных, базовые языки не ориентированы на конкретную область применения; их изобразительные средства более универсальны и, как следствие этого, менее доступны проектировщику, мало знакомому с теорией формальных языков. Базовые языки используются в качестве математической модели для первичных языков. В них уже предусматриваются некоторые преобразования по оптимизации решений в процессе проектирования ДУ. Автоматные (или, как их еще называют, стандартные) языки обладают слаборазвитыми изобразительными средствами, но удобны для различных формальных преобразований по оптимизации проектируемого дискретного устройства.

Система ДАСП позволяет транслировать условия работы дискретного устройства, представленные на первичном языке, в выражения на базовом языке, а затем и автоматном языке, в качестве которого используется система булевых функций.

Иерархия позволяет' довольно просто использовать в ДАСП другие первичные языки, осуществляя их трансляцию в базовый язык.

4. Ввод условий работы ДУ осуществляется в режиме диалога проектировщика и ЭВМ с использованием дисплея, что позволяет оперативно исправлять ошибки при вводе информации и тем самым ускоряет процесс проектирования.

5. Большинство пограмм ДАСП выполнено на языке ФОРТРАН.

В настоящее время Институт проблем передачи Информации АН СССР занимается разработкой автоматизированной системы проектирования ЭУМ узлов коммутации (АСПУМ) [2]. Предполагается, что система будет открытой, допускающей введение новых программ по мере их готовности. Ориентированная на проектирование специализированных дискретных устройств управления узлов коммутации АСПУМ, в отличие от ДАСП, основана на использовании частных, а не универсальных методов синтеза. Изза сложности и длительности процесса ее реализация осуществляется по частям. На первом этапе разработана подсистема АСПУМ-1, обеспечивающая автоматизацию совмещенного этапа системно-логического проектирования микропрограммных управляющих устройств, применяемых на узлах коммутации малой емкости, или отдельных блоков микропрограммного управления ЭУМ

Алгоритмы синтеза этапа системно-логического проектирования микропрограммных управляющих устройств в АСПУМ-1 основаны на использовании языка логических схем алгоритмов (ЛСА).

Из других наиболее известных автоматизированных систем проектирования дискретных устройств отметим также АСП, описание которых дано в[11—13].

Контрольные вопросы

1. Какие существуют основные принципы построения дискретных устройств?
2. В чем состоит принципиальное отличие программного моделирования алгоритма функционирования дискретного устройства от структурного?
3. На какие этапы обычно подразделяют процесс проектирования дискретного устройства?
4. Какие существуют типы конструктивных единиц дискретного устройства?
5. Какие существуют принципы взаимодействия человека с ЭВМ в АСП?

Глава 2.

ДИСКРЕТНЫЕ ЭЛЕМЕНТЫ, ИСПОЛЬЗУЕМЫЕ В УСТРОЙСТВАХ СВЯЗИ

2.1. Элементные базисы дискретного устройства

Для построения дискретного устройства применяются как электромагнитные реле, так и различного рода электронные элементы. Применение того или иного типа элементов зависит от назначения ДУ, требуемого быстродействия, массо-габаритных и стоимостных характеристик, требуемой надежности и т. д. Если к быстродействию проектируемого устройства не предъявляются высокие требования, то для

его построения могут использоваться электромагнитные реле, среди которых имеются как достаточно мощные многоконтактные реле, работающие от постоянного или переменного тока, так и малогабаритные реле, для работы которых требуется небольшой ток. Однако в связи с успехами микроэлектроники для построения ДУ все шире применяются различного рода электронные элементы и в первую очередь интегральные микросхемы.

В области микроминиатюризации электронной аппаратуры можно отметить три основных направления:

создание микромодулей с высокой плотностью упаковки дискретных микрокомпонентов;

интегральная электроника;

молекулярная электроника.

Первое направление, появившееся раньше других, предусматривает сборку миниатюрных дискретных компонентов электронных схем в одном блоке (микромодуле) с высокой степенью упаковки, достигающей трех и более компонентов в 1 $см^3$. Однако микромодульная техника, основанная на применении дискретных микроэлементов, не смогла коренным образом решить проблему микроминиатюризации электронных устройств, так как уменьшение размеров компонентов затрудняло операции сборки и моитажа их в микромодуле. Кроме того, имеющиеся между компонентами внутри модуля и между модулями в блоке соединения, выполненные пайкой или сваркой, являлись источником повреждений (отказов) и не позволяли получать высоконадежные устройства. При этом увеличение плотности монтажа существенно ограничивалось соединительными элементами, опорными конструкциями и прочими устройствами, которые занимали больший объем, чем сами компоненты.

По указанным выше причинам первое направление микромипиатюризации электронной аппаратуры было вытеснено вторым — интегральной электроникой. Используемые интегральные микросхемы представляют собой монолитную структуру, внутри или на поверхности которой в едином технологическом процессе формируются все компоненты схем и соединения между ними. Значительное уменьшение размеров и существенное повышение надежности интегральных микросхем по сравнению с микромодульнымн обусловлено прежде всего сокращением объема, занимаемого проволочным или печатным монтажом, опорными конструкциями и т. п. Высокая надежность и резкое снижение стоимости электронной аппаратуры, основанной на применении интегральных микросхем, объясняются также исключением многочисленных операций, связанных с испытанием, сборкой и монтажом отдельных компонентов схем при ее изготовлении.

В настоящее время интегральная электроника является основным направлением в области микроэлектроники. Поэтому в дальнейшем будет рассматриваться лишь интегральный технологический принцип микроминиатюризации электронной аппаратуры. Вместе с тем следует иметь в виду, что в будущем основным направлением может оказаться третье направление — молекулярная электроника (молектроника), которое в данное время только начинает развиваться. В отличие от интегральных микросхем молектроника предлагает принципиально новый подход к созданию электронных устройств, основанный на физических явлениях в твердом теле. Блоку твердого тела придаются такие свойства, которые требуются для выполнения определенной заданной функции, причем без создания в нем какой-либо привычной для обычной электроники функциональной схемы.

Еще недавно интегральные микросхемы относили к БИС, если на одном кристалле размещалось порядка 100 элементов. Сейчас такие схемы имеют уровень интеграции в 100 тыс. элементов на кристалл. И это не предел. За последние 10 лет степень интеграции элементов на одном кристалле выросла примерно в 10^3 раз и к настоящему времени достигает 10^s элементов на кристалл. К 1984 г. прогнозируется степень интеграции 10^6, **а к 1990** — 10^7 элементов на кристалл.

Высокая и все возрастающая быстрыми темпами степень интеграции элементов на одном кристалле привела к тому, что принцип построения элементной базы, используемый ранее для построения мпкроэлектронных устройств автоматики и вычислительной техники на основе ИС с малой степенью интеграции (малых ИМС), оказался неприемлемым. Создание БИС по такому же принципу, как и малых ИМС, когда на одном кристалле размещаются набор отдельных логических элементов типа И, ИЛИ, НЕ—И, НЕ—ИЛИ и т. п,, привело бы к недопустимо большому числу внешних выводов в корпусе БИС. Неприемлемым для БИС является и принцип построения законченных функциональных схем на олном кристалле, который достаточно успешно применялся при построении ИМС со средней степенью интеграции. Создание БИС, в которых реализовывались бы отдельные блоки или даже целые устройства (так называемых заказных БИС) возможно лишь небольшим тиражом, что обусловливает высокую их стоимость. Исключение представляют отдельные типовые устройства массового применения. Особенно дорогостоящими оказываются заказные БИС для управляющих устройств, так как последние, в отличие от устройств вычислительной техники, часто изготавливаются в одном или очень небольшом числе экземпляров и существенно отличаются друг от друга реализуемыми алгоритмами функционирования.

В настоящее время известны два основных принципа построения БИС, обеспечивающих их универсальность и, как следствие этого, сокращение необходимых для построения разнообразных дискретных устройств числа типов БИС и увеличение тем самым тиража каждого из типов БИС. Это — оздание однородных сред и микропроцессоров. При этом общим для данных принципов построения БИС является свойство настраиваемости БИС на реализацию тех или иных функций создаваемого ДУ.

Таким образом, настраивая одну или несколько однотипных БИС, на основе которых создается ДУ, на реализацию алгоритма функционирования последнего, можно построить любое ДУ в базисе лишь одной универсальной БИС. Практически удобно иметь набор различных типов настраиваемых БИС, позволяющих реализовать любое ДУ, т. е. универсальный базис на основе не одного типа

настраиваемых БИС, а набора небольшого *чис.ла* (обычно не более пяти — шеста) типов настраиваемых БИС.

Первый принцип построения настраиваемого БИС, который возник еще в 60-е годы, — это создание однородных сред [14] из многофункциональных элементов. Настраивая многофункциональные элементы на реализацию той пли иной функции в однородной среде, можно реализовать любое ДУ. Фактически однородная среда имеет структуру, аналогичную структуре оперативной или постоянной памяти, но, в отличие от оперативного ищ постоянного запоминающего устройства (ОЗУ или ПЗУ), каждый разояд однородной среды реализует не две (тождественно ложную—запись нуля и тождественно истинную—запись единицы), а четыре, пять и более логических функций, которые обеспечивают не только хранение, но и переработку информаци, аппаратно реализуя тем самым алгоритм функционирования любого управляющего устройства.

Следует отметить, что в те годы, когда основным элементным базисом были малые ИМС и еще не было полупроводниковой памяти, этот принцип построения ДУ оказался неконкурентоспособным с обычной схемной реализацией ДУ в базисе определенного набора логических элементов. Однако по мере развития микроэлектроники и особенно с созданием БИС и полупроводниковой памяти большой и сверхбольшой емкости можно ожидать, что однородные среды, обеспечивающие универсальную в базисе двух-трех типов БИС аппаратную реализацию алгоритмов функционирования сложных ДУ, найдут практическое применение. При этом, основной недостаток памяти сверхбольшой емкости, являющейся прототипом однородной среды, — относительно низкое быстродействие — будет в значительной степени скомпенсирован аппаратной реализацией алгоритма функционирования ДУ.

Простейший вид однородной среды в виде программируемых логических матриц (ПЛМ) уже нашел широкое признание [15]. Вопросы проектирования ДУ в базисе однородных сред **и ПЛМ** излагаются в гл. 6.

Второй принцип построения настраиваемых БИС — это создание комплексов микропроцессорных БИС и использование их в качестве элементного базиса для программной реализации алгоритмов функционирования ДУ.

Микропроцессоры (МП), появившиеся в начале 70-х годов, удивительно быстро (за 8—9 лет) завоевали широкое признание и проникли буквально во все области техники. За эти немногие годы микропроцессоры благодаря успехам микроэлектроники прошли путь от многокристальных медленнодействующих с ограниченными возможностями, 'небольшим набором команд и малой разрядностью (2—4 разряда) до однокристальных быстродействующих многоразрядных (16—32 разряда) с расширенной системой команд и включенной сверхоперативной регистровой, оперативной и постоянной памятью, фактически представляющих собой однокристальные микро-ЭВМ.

Массовое производство 16-разрядных микро-ЭВМ с ОЗУ и ПЗУ емкостью в несколько килобайт, реализованных на кристалле 7,5Х7,5 мм^2 и содержащих до 10^5 элементов, ожидается в ближайшие год-два. На 1984 г. прогнозируется разработка микро-ЭВМ с памятью емкостью 64К бит при степени интеграции 10^6 элементов на кристалле 12Х12 мм^2.

Рост вычислительной мощности микропроцессоров при существенном сокращении их стоимости позволило рассматривать МП не только как элементную базу для построения вычислительных и управляющих устройств и не только как средство для сокращения их объема, массы, потребления электроэнергии и стоимости, но и как основу для построения распределенных многомикропроцессорных вычислительных и, в первую очередь управляющих систем (см. рис. 1.6), где в качестве ФБ используются МП или микро-ЭВМ. Проектирование распределенных многомикропроцессорных ДУ рассматривается в гл. 7.

2.2. Герконовые реле

Герконовое реле (рис. 2.1,а) содержит ряд герметизированных контактов (герконов), представляющих собой заполненную инертным газом стеклянную ампулу диаметром 3—5 мм и длиной 25—50 мм, в которой находятся две контактные пружины, изготовленные из упругого магнитопроводящего материала. Концы контактных пружин в месте образования контакта при их замыкании (в рабочем зазоре) покрыты тонким слоем золота. Группа герконов размещается внутри катушки. Для уменьшения магнитного сопротивления магнитной системы, состоящей из корпуса и контактных пружин, между корпусом и выводами пружин делают прокладки из магнитной резины, обладающие малым сопротивлением для магнитного потока и большим омическим сопротивлением электрическому току.

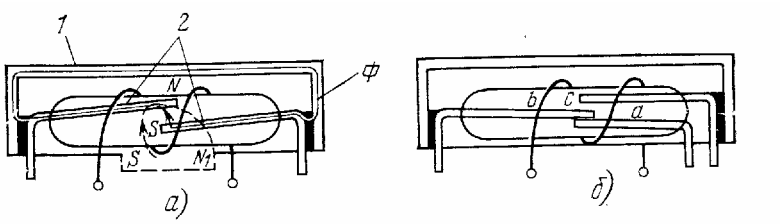


Рис. 2.1

При подаче тока в обмотку реле в магнитной системе возникает магнитный поток Ф, проходящий через корпус *1* и контактные пружины *2.* В результате концы пружин в рабочем зазоре приобретают разную полярность и притягиваются друг к другу, образуя цепь выходного сигнала. При отключении тока от обмотки реле под действием сил упругости контактные пружины размыкаются, разрушая цепь выходного сигнала.

Наряду с герконовыми реле, имеющими контакты на замыкание, существуют реле с контактами на размыкание и переключение У герконозого реле с контактами на размыкание в цепь магнитопровода вводится постоянный магнит (на рис. 2.1,*а* постоянный магнит показан штриховой линией), который удерживает контактные пружины в замкнутом состоянии. Для размыкания контактов геркона обмоткой создается магнитный поток обратного направления, компенсирующий магнитный поток постоянного магнита. В результате пружины геркона под действием сил упругости размыкаются.

В герконовом реле с контактами па переключение (рис. 2.1,6) имеются три пружины, при этом пружина *а* изготавливается из диамагнитного материала, а пружины *Ь* и с—из магнитопроводящего материала. При подаче тока в обмотку реле магнитный поток, проходящий через корпус и контактные пружины *Ь* и *с,* создает разность магнитных потенциалов в рабочем зазоре этих пружин. В результате пружины *а и Ь* размыкаются, а пружины *Ь* и *с* замыкаются. При отключении тока от обмотки пружины геркона возвращаются в исходное состояние.

Основными достоинствами герконового реле являются высокоебыстродействие (порядка 0,5—3 мс), малые габаритные размеры и масса. Герметизация и нейтральная среда, в которой находятся контакты, обеспечивают их независимость от внешней среды и повышенную надежность. Недостаток этого реле — большое время вибрации контактов (порядка 0,5 мс).

2.3. Ферридовые реле

Ферридовое реле, кратко называемое *ферридом,* является элементом с памятью. Оно представляет собой герконовое реле, магнитная система которого изготовлена из материала с прямоугольной петлей гестирезиса, обладающего остаточным намагничиванием, достаточным для срабатывания и удержания контактов геркона в притянутом состоянии.

По принципу построения магнитной системы различают ферриды с последовательной и параллельной структурами магнитной системы, а по принципу выключения — с компенсацией магнитного потока и с размагничиванием материала сердечника.

Феррид с последовательным построением магнитной системы и компенсацией магнитных потоков (рис. 2.2,а) содержит: сердечник, выполненный из материала с прямоугольной петлей гистерезиса; магнитный шунт, представляющий собой массивную металлическую плату с укрепленным на ней ферридом, делящую магнитную систему на две части — верхнюю и нижнюю;

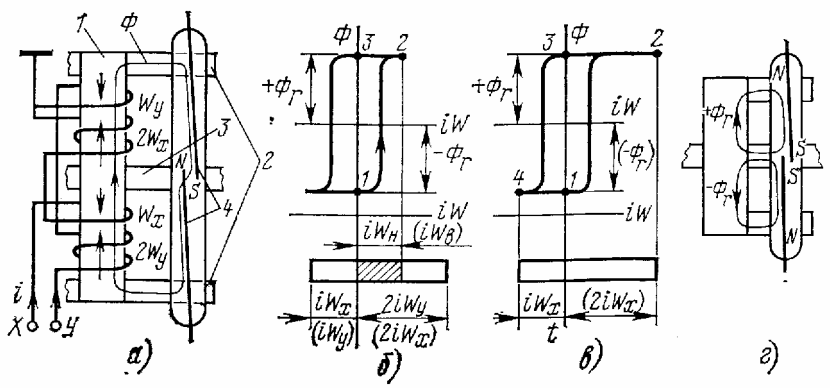


Рис. 2.2

несколько герконов, расположенных вокруг сердечника (на рисунке показан только один геркон), и две

обмотки, одна из которых подключена к входу *X,* а другая — к входу *Y.* Обмотки имеют одинаковое число витков и состоят из двух секций, одна из которых расложена в верхней, а другая — в нижней половине сердечника. Обмотка, подключенная к входу *X в* нижней половине сердечника, имеет *W^,* а в верхней половине—*2Wx* витков. Обмотка, подключенная к входу У, имеет в нижней половине сердечника 2 *Wy* витков, а в верхней — *Wy.* Секции включены навстречу друг другу.

Для включения геркона на входы *X* и *Y* подаются импульсы тока одинаковых величины и полярности. При прохождении импульсов тока по обмоткам создается магнитодвижущая сила (МДС), направление которой для каждой секции показано стрелками на сердечнике (рис. 2.2,а). Возникающая в верхней половине сердечника МДС *iW^2iW^—iWy.* Так как *Wx=Wy,* то *iWs==iWx.* В нижней половине сердечника МДС *iW^=i2Wy —iWx=iWy.* Диаграмма изменения МДС и магнитного потока в нижней и верхней половинах сердечника показана на рис. 2.2,6. Величина МДС для нижней половины сердечника указана без скобок, а для верхней половины — в скобках. Если перед поступлением импульсов в обмотки в верхней половине сердечника магнитный поток равен *+Фг* (точка *3),* а в нижней —*Фг* (точка /), то при одновременном прохождении импульсов по обмоткам под действием МДС *iW»* и *iWy* магнитный поток в обеих половинах сердечника будет иметь значение, отмеченное точкой *2.* По окончании прохождения импульсов магнитный поток в обеих половинах сердечника будет иметь одинаковое направление, а значение его будет равно +Фг (точка i).

Суммарный магнитный поток Ф обеих половин сердечника, проходя через полюсные надставки (см. рис. 2.2,а), замыкается через пружины геркона. Под действием разности магнитных потенциалов в рабочем зазоре пружины геркона притягиваются Друг к другу и остаются в замкнутом состоянии за счет остаточного магнитного потока.

Для выключения геркона на один из входов (например, *X)* необходимо подать импульс. В результате прохождения импульса по обмотке под действием МДС *i2Wy:* сердечник в верхней половине перейдет из состояния, соответствующего точке *3,* в точку *2* (рис. 2.2,б), а в нижней половине под действием МДС *iWx*— из состояния, соответствующего точке *3,* в точку *4.*

По окончании импульсов величина магнитного потока в верхней половине сердечника будет равна *+Фг* (точка *3),* а в нижней—ф,. (точка /). Магнитные потоки в нижней и верхней половинах сердечника / будут иметь одинаковое значение, но противоположное направление. Каждый из них замыкается через по люсную надставку 2, одну из пружин *4* геркона и магнитный шунт *3* (рис. 2.2,г). Концы пружины геркона имеют одинаковую полярность и под действием сил упругости и отталкивания размыкаются, обрывая цепь выходного сигнала.

Аналогичное положение возникает при поступлении импульса на вход *Y.* Отличием является то, что магнитные потоки в верх- ней и нижней половинах сердечника изменят свое направление на обратное по сравнению с указанным на рис. 2.2,г.

Таким образом, при одновременном поступлении импульсов на входы *X* и У пружины геркона замыкаются, а при поступлении импульса на один из входов за счет компенсации магнитных потоков двух половин сердечника пружины размыкаются; следовательно, функция включения данного элемента *Рвкп=x-y,* а функция *Рвыкл=x(Bу.*

Отечественные ферриды данного типа содержат два или четыре геркона и имеют две обмотки, секции которых содержат: одна 40, а другая 20 витков. На входы *X* и *Y* при включении или на один из входов при выключении подается импульс тока 8 А длительностью 0,6 мс. Более подробные сведения по конструкции и основным параметрам ферридов приведены в [33].

Феррид с параллельной магнитной системой, разработанный фирмой LMT (Франция), содержит (рис. 2.3,а): два сердечника. выполненных из материала со прямоугольной петлей гистерезиса

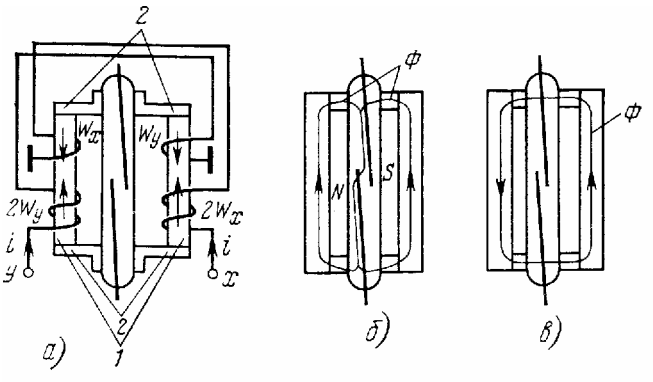


Рис. 2.3

и остаточным намагничиванием, достаточным для срабатывания и удержания герконов в поитянутом состоянии; полюсные надставки: герконы и две обмотки. Как и у ферридов с последовательной магнитной системой, каждая из обмоток делится на две секции. Одна из секций имеет в 2 раза больше витков, чем другая. Схема включения обмоток показана на рис. 2.3, *а.*

При одновременном поступлении импульса тока на входы *X* и У в левом сердечнике возникает МДС

iWn=iWy—iWx=iWy, так как *Wx=Wy,* а в правом сердечнике *iWn=2iWx—iWy=iWx.* По окончании импульса остаточные магнитные потоки имеют одинаковое направление и равны друг другу. Магнитные потоки правого и левого сердечников, проходя через полюсные надставки *2,* замыкаются через пружины геркона (рис. 2.3,6). Под действием разности магнитных потенциалов в рабочем зазоре пружины геркона притягиваются и остаются в замкнутом состоянии за счет остаточного намагничивания сердечников.

Для выключения геркона достаточно подать импульс тока на один из входов (например, *X}.* В этом случае произойдет перемагничивание левого сердечника за счет МДС, равной *iWx.* После скончания импульса остаточные магнитные потоки правого и левого сердечника / будут равны и противоположны друг другу (рис. 2.3,а). Суммарный магнитный поток в рабочем зазоре геркона равен нулю, поэтому пружины геркона размыкаются, обрывая цепь выходного сигнала.

Феррид с параллельно последовательной магнитной системой, разработанный фирмой Bell System (США), основные элементы которого показаны на рис. 2.4,а, состоит из двух сердечников /, расположенных с обеих сторон геркона, полюсных надставок *2* из магнитной резины и двух обмоток, подключенных ко входам *X и Y.*

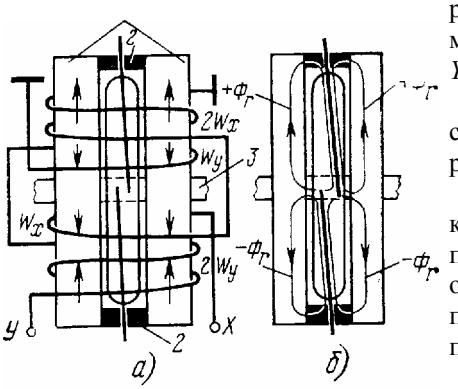


Рис. 2.4

Каждая из обмоток, охватывающих оба сердечника, имеет две секции, включенные встречно. Число витков одной из секций в 2 раза больше, чем другой.

При одновременном поступлении импульсов на входы *X* и У каждой из обмоток создаются МДС, направления которых показаны на рис. *2А,а* стрелками. Результирующая МДС в обмотках обоих сердечников, как и у феррида с последовательным
построением магнитной системы, направлена в одну сторону.

После окончания импульсов остаточный магнитный поток в обоих сердечниках имеет одинаковое направление и, как у феррида с параллельной магнитной системой, замыкается через полюсные надставки и пружины геркона (рис. 2.3,6). Под действием разности магнитных потенциалов пружины геркона притягиваются друг к другу и остаются в этом состоянии под действием остаточного намагничивания сердечников.

Для выключения феррида на один из входов (например, *X)* подается импульс тока. При прохождении этого импульса по обмотке в нижней и верхней половинах сердечника создаются МДС противоположного направления. После окончания импульса остаточные магнитные потоки *Фт* в верхней и нижней половинах каждого сердечника будут иметь противоположные направления. Каждый из магнитных потоков замыкается через одну из пружин геркона и магнитный шунт *3.* В рабочих зазорах пружины герконов имеют одинаковую полярность и под действием сил упругости и отталкивания размыкаются.

Феррид с последовательной магнитной системой и размагничиванием магнитной системы (рис. 2.5,а) содержит: сердечник *1* из магнитополутвердого материала, характеризуемого широкой петлей гистерезиса и возможностью размагничивания по частным петлям гистерезиса, герконы, полюсные надставки *2,* демпфирующее кольцо *3* и две обмотки.

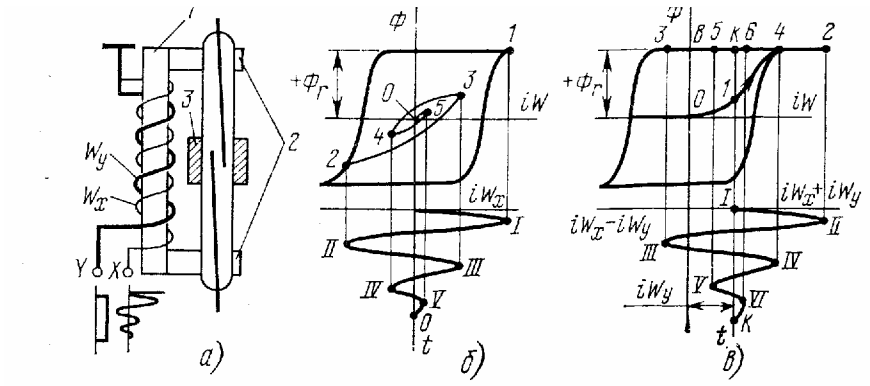


Рис. 2.5

Для включения герконов одновременно подаются импульс тока на обмотку *Wy* и знакопеременные импульсы с затухающей амплитудой на обмотку *Wx-* Если сердечник размагничен, то при поступлении импульса в обмотку *Wy* он переходит из состояния *О* (рис. 2.5,в) в состояние, соответствующее точке /.

При одновременном поступлении импульсов в обмотку *Wy* и первого импульса в обмотку *Wx,* МДС которой характеризуется точкой II, магнитный поток в сердечнике соответствует величине, определяемой точкой *2.* При следующем импульсе обратной полярности (точка ///) величина магнитного

потока определяется точкой *3.* При последующих импульсах величина магнитного потока сердечника соответствует точкам *4, 5,...,* а после прекращения импульсов — точке *к.* После окончания импульса в обмотке *Wy* величина магнитного потока будет соответствовать точке *е,* где остаточный поток будет равен *+Ф2.*

Остаточный магнитный поток +Фг сердечника проходит через полюсные надставки и пружины геркона, создает разность магнитных потенциалов в рабочем зазоре, в результате чего пружины геркона притягиваются друг к другу, замыкая цепь выходного сигнала.

Для выключения геркона на вход *X* подаются знакопеременные импульсы с затухающей амплитудой. Если сердечник находился в состоянии +Фг, характеризуемом точкой е, то при импульсе / обратной полярности он перейдет в состояние, соответствующее точке *1* (рис. *2.5,6),* при //—в состояние, определяемое точкой *2,* при /// — точкой *3,* и т. д. В конечном результате состояние сердечника будет соответствовать точке *0,* где остаточный магнитный поток равен нулю. Из-за отсутствия магнитного потока в рабочем зазоре пружины геркона размыкаются, обрывая цепь выходного сигнала.

Величина импульса тока, поступающая на вход У, выбрана таким образом, что МДС обмотки *Wy* создает напряженность поля соответствующей области упругого намагничивания, которая характеризуется тем, что после окончания импульса намагниченность сердечника будет равна исходной. Например, если сердечник был размагничен (точка *0* на рис. 2.5,в), то при поступлении импульса на вход *Y* магнитный поток сердечника будет соответстствовать точке /, а после окончания импульса — точке *0, т.* е. магнитный поток равен нулю. Если перед поступлением импульса магнитный поток сердечника равен +Фг, то при поступлении импульса его величина будет соответствовать точке *к,* а после его окончания снова станет равной +Фг.

Поскольку при поступлении знакопеременных импульсов величина магнитного потока может превышать Ф,, то возможно ложное кратковременное подрабатывание геркона в случае, если феррид был выключен. Для устранения указанного недостатка на герконы одевается демпфирующее кольцо из электропроводящего материала, которое замедляет нарастание магнитного потока. Действие данного кольца аналогично действию короткозамкнутой обмотки в нейтральных реле.

Таким образом, при поступлении импульса на вход *Y* состояние магнитной системы не меняется. При поступлении знакопеременных импульсов на вход *X* феррид выключается, а при одновременном поступлении импульсов на вход *X* и У — включается. Следовательно, функция включения феррида будет равна *Рвкл=ху,* а функция выключения *Рьыкл=х.*

Отечественное ферридовое реле данного типа имеет две облютки по 100 витков каждая. На вход У подается импульс тока 1,1—2,0 А, амплитудное значение первого импульса, подаваемого на вход *X,* равно 3,5—5,5 *А.* Основным достоинством этого феррида является отсутствие высоких требований совпадения управляю1цих импульсов по времени и амплитуде.

2.4. Гезаконовые реле

Гезаконовое реле, или просто *гезакон* (герметизированный запоминающий контакт), представляет собой ферридовое реле, у которого контактные пружины изготовлены из материала, обладающего достаточной упругостью и имеющего остаточное намагничивание, обеспечивающее срабатывание и удержание контактов и замкнутом состоянии.

Гсзаконовое реле состоит из корпуса, гезакона, магнитного шунта и трех обмоток (рис. 2.6,а). Обмотки / и //, включенные согласованно, используются для включения, а встречно соединенные обмотки / и /// — для выключения реле.

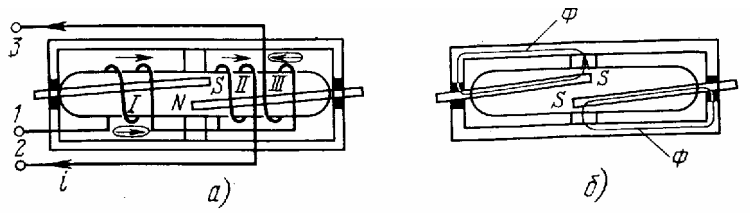


Рис. 2.6

При поступлении импульса тока на входы *1, 2* ток проходит по обмоткам / и //, создавая МДС одинакового направления (показано стрелками над обмотками). Магнитный поток, создаваемый МДС в обмотках / и //, проходит по корпусу и пружинам гезакона. Под действием разности магнитных потенциалов в рабочем зазоре пружины гезакона притягиваются друг к другу и остаются замкнутыми и после окончания импульса за счет остаточного их намагничивания.

Для выключения гезакона подается импульс тока на входы / и *3,* который, проходя по обмоткам / и *III,* создает МДС, направленные навстречу друг другу (на рис. 2.6,а показано стрелками

в кружках). Магнитный поток, создаваемый обмоткой /, замыкается через левую пружину, магнитный шунт и корпус (рис. 2.6,6), а магнитный поток, создаваемый обмоткой *III,* проходит через правую пружину, магнитный шунт и корпус. Концы пружин в рабочем зазоре гезакона имеют одинаковую полярность и под действием сил упругости и отталкивания размыкаются.

Возможен и другой способ управления гезаконом, аналогичный используемому в ферриде с последовательной магнитной системой и компенсацией магнитного потока. В этом случае на гезакон наматываются две обмотки, разделенные на две секции, каждая из которых расположена в разных половинах гезакона. Число витков одной из секций в 2 раза больше, чем другой. Секции включены встречно. Принцип работы такого гезаконового реле аналогичен принципу работы феррида с последовательной магнитной системой.

Время срабатывания гезакона 1—3 мс, а время отпускания 0,5 мс. Магнитодвижущая сила срабатывания 75—100 А, а отпускания 20—50 А.

Основное достоинство гезаконовых реле в сравнении с ферридами — значительное уменьшение габаритных размеров и массы за счет отсутствия в них магнитных сердечников.

2.5. Многократные соединители

Для коммутации различной информации используются многократные соединители (МС), представляющие собой матричную схему, имеющую *M* входов и *N* выходов. В точках пересечения координат матрицы расположены элементы точек коммутации, представляющие собой либо дискретный элемент, либо группу контактов, с помощью которых осуществляется соединение любого входа с любым из выходов.

Многократный соединитель, в котором в качестве элемента точек коммутации используется герконовое реле, получил название *многократного герконового соединителя* (МГС). Если в точках коммутации находятся ферриды с компенсацией магнитных потоков, то такие МС получили название *многократных ферридовых соединителей* (МФС). Наконец, МС, использующие в качестве точек коммутации ферриды с размагничиванием сердечника, называются *многократными интегральными соединителями* (МИС).

В каждой точке коммутации МФС находится феррид с соответствующим числом герконов, контактами которых образуется матрица, используемая для коммутации информации. Схема управления МФС емкостью 2x2 (рис. 2.7, а) содержит четыре феррида, расположенных на пересечении двух вертикалей и горизонталей. В каждом горизонтальном ряду обмотки *Wy* и в каждом вертикальном ряду обмотки *Wx* соединены последовательно.

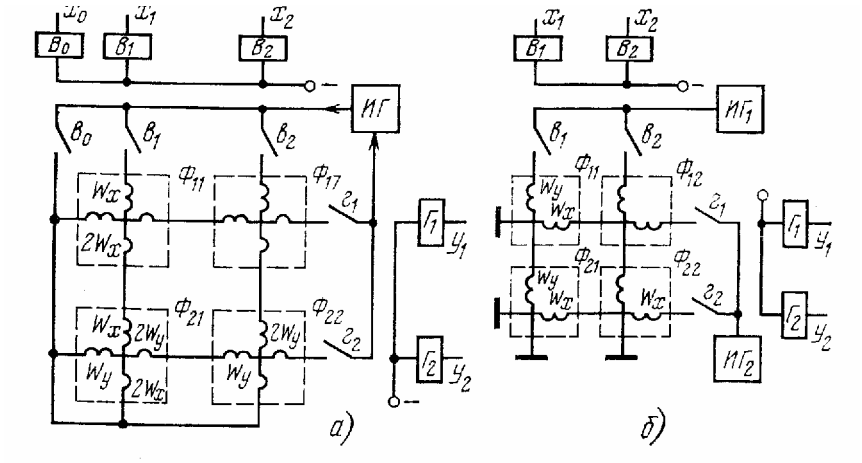


Рис. 2.7

Для выбора соответствующего входа (вертикали) и выхода (горизонтали) в схеме имеются реле *B1, B2* и Г1, Г2, а для включения феррида, расположенного в точке пересечения выбранной вертикали и горизонтали, используется импульсный генератор (ИГ). Реле *Bo* необходимо для выключения феррида при отбое.

Выбор горизонтали и вертикали производится координатными сигналами X1*, X2* и Y1, Y2, которыми включается одно из реле вертикали и одно из реле горизонтали. Через контакты указанных реле импульсом от ИГ включается феррид, расположенный на пересечении выбранных вертикали и горизонтали.

Если в МФС требуется установить соединение входа 1 с выходом *2,* то из управляющего устройства даются координатные сигналы X1 и Y2. От указанных сигналов срабатывают реле *B1* и ***Г2.*** Контактом B1 выход ИГ подключается к первой вертикали, а контактом Г2 вход ИГ подсоединяется к второй горизонтали. Импульс генератора проходит через контакт B1, обмотки *Wx* и 2Wx феррида Ф11,

аналогичные обмотки в Ф21, обмотки *Wy* и 2*Wy* в Ф21 и такие же обмотки в 022, контакт *гг,* вход ИГ и включает Ф21. В ферридах же Ф11 и Ф22 импульс от ИГ проходит только по одной обмотке, поэтому данные ферриды выключаются или подтверждается их выключение. При срабатывании феррида Ф21 контактами его герконов вход / подключается к выходу *2.*

Таким образом, при поступлении импульса на один из входов и один из выходов включается феррид, расположенный на пересечении выбранных вертикали и горизонтали, а остальные ферриды данных горизонтали и вертикали выключаются, исключая возможность двойных соединений.

Для выключения феррида даются координатные сигналы на входы *Хо* и Y2*,* от которых срабатывают реле *Во* и Га. Через контакты указанных реле импульс с выхода ИГ проходит через контакт Во, обмотки 2*Wy* и Wy ферридов Ф21, *Ф22,* контакт на на вход ИГ.

Вследствие того, что импульс проходит по обмоткам *2Wy* и *Wy* феррида 021, последний отпускает, размыкая контакты герконов и отключая вход от выхода. Аналогично строится МС на гезаконах, имеющих две обмотки. Схема управления МИС (рис. 2.7,6) отличается от схемы управления МФС тем, что она содержит два импульсных генератора. Один *из* генераторов (ИГ1) выдает импульсы постоянного тока, а другой (ИГа) — знакопеременные импульсы с затухающей амплитудой.

Если необходимо установить соединение входа / с выходом *2,* то от координатных сигналов на входах X1 и Y2 срабатывают реле *В1* и Г2. Импульс постоянного тока от генератора ИГ1, проходя через контакт В1 и обмотки *Wy* ферридов Ф11 и Ф21, подготавливает их к включению, а знакопеременный импульс с затухающей амплитудой от ИГ2, проходя через контакт Г2 и обмотки *Wx* ферридов Ф22 и Ф21, включает Ф21 и выключает или подтверждает выключение феррида Ф22. Контактами герконов сработавшего феррида Ф21 подключается вход 1 коммутационной системы к выходу *2.*

Для выключения феррида Ф21 дается координатный сигнал на вход Y2, от которого срабатывает реле Г2. Импульс от ИГ2, проходя через замкнутый контакт Г2 и обмотки *Wx* обоих ферридов горизонтали, выключает Ф21 и подтверждает выключение Ф22. Контактами герконов феррида Ф21 вход коммутационной системы отключается от выхода.

Многократные соединители, как правило, имеют емкость 8x8x2, 8x8x4, 8x4x2, где первая цифра определяет число входов, вторая — число выходов и третья — проводность. В СССР выпускаются МФС1 и МФС2 емкостью 8X8X2 и 8X8X4. Данные МФС отличаются только габаритными размерами — за счет более компактного размещения деталей объем МФС2 несколько уменьшен.

Емкость выпускаемых МИС 8X4x2. Объем, приходящийся на одну точку коммутации, у МИС в 2 раза меньше, чем у МФС1 (за счет более компактной конструкции феррида).

2.6. Интегральные микросхемы

Существует большое число серий логических интегральных схем (ИС), отличающихся технологией производства, функциональным назначением, степенью интеграции и т. д. В настоящее время наибольшее распространение получила серия 155, имеющая достаточно разнообразный и универсальный набор логических компонентов с разной степенью интеграции. У полупроводниковых ИС на одном кристалле может располагаться ряд компонентов, содержащих различное число элементов: транзисторов, резисторов, диодов и т. д. В зависимости от числа элементов N в одном компоненте различают ИС малой, средней и большой степеней интеграции.

В серии 155 к малой степени интеграции относятся компоненты с числом элементов на одном компоненте *N<10,* реализующие схемы И—НЕ, ИЛИ—НЕ, И, ИЛИ, НЕ. Схемы И—НЕ серии 155 имеют два, три, четыре или восемь входов. При этом в одном корпусе могут размещаться четыре схемы И—НЕ на два входа каждая, либо три такие схемы на три входа каждая, либо две схемы на четыре входа каждая, либо одна схема на восемь входов, Схемы ИЛИ и И имеют по два входа, и в одном корпусе размещается четыре схемы. Кроме этого, серия 155 содержит ряд схем И—ИЛИ—НЕ на различное число входов.

К средней степени интеграции относятся триггерные схемы типов JK и *D с* числом элементов 10 < N <100. В одном корпусе размещается один триггер *JK* или два триггера *D.* Кроме входов *!, К, D* триггеры имеют вход синхронизации и входы *R, S.* Наличие в триггерах входов *R* и S делает их достаточно универсальными и расширяет область применения.

К большой степени интеграции относятся различные виды счетчиков, сумматоры, преобразователи кодов, постоянные запоминающие устройства и т. д. с числом элементов 100<N<=1000.

Интегральные микросхемы серии 155 позволяют реализовать дискретные автоматы в различных базисах или комбинированные схемы с использованием элементов различных базисов. В качестве основного элемента памяти используется триггер типа *RS,* на основе которого строятся триггеры *JK* и *D.* Схема триггера *RS* в базисе И—НЕ приведена на рис. 2.8,а, а его условное

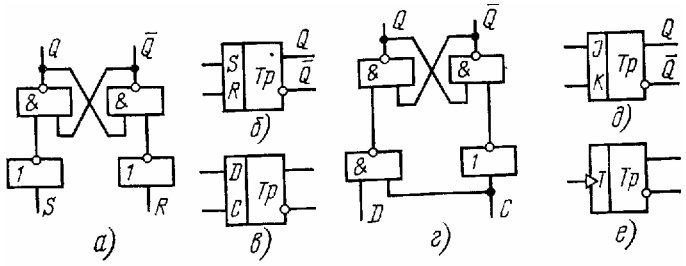


Рис. 2.8

изображение показано на рис. 2.8,6. Схема синхронного триггера *D* и его условное изображение показаны на рис. *2.8,в* и *г_* условное изображение триггеров *JK* и *T* приведено на рис. 2.8 д и *е.* Порядок функционирования триггера *RS* показан в табл. 2.*1*.

Таблица 2.1

	Вход	Со	стоян	ие на	этапе	рабо	ты	
Домент	(выход)	1	2	3	4	5	6	7	8
t	*R*	0	1	0	1	0	0	1	1
	S	0	0	1	1	0	1	0	1
T+1	*Q*	O	0	0	0	1	1	0	I
	Q	0	0	1	1	1	1		н

Из таблицы видно, что триггер в последующий момент *t+l* переключается в состояние 1 (Q=1) при наличии в данный момент *t* сигнала на входе S и отсутствии сигнала на входе R *(S=1, R=0)* и переключается в состояние 0 (Q=0) при наличии сигнала на входе *R* и отсутствии сигнала на входе S (R=1, S=0) одновременное поступление сигналов на входы R и S запрещается в связи с тем, что приводит к неопределенному состоянию триггера.

Таблица 2.2

Момент времени	Вход (выход)	1	Сос 2	тояние 3	на эта 4	пе раб 5	оты 6	7
T	*D*	0	1	1	1	O	0	1
t+l	*c*	0	0	0	1	1	1	0
	Q	0	0	1	1	1	0	1
	o.	0			1	0	0	1

Порядок работы триггера *D* приведен в табл. 2.2.

Триггер *D* (см. табл. 2.2) переключается в состояние 1 (0=1) при наличии сигналов на входах *С и D* С=1, *D=\)* и переключается в состояние О (Q=0) при поступлении сигнала на вход С(С=1). Поступление сигнала только на вход *D* не изменяет состояние триггера.

Порядок функционирования триггеров *T* и *JK* приведен в табл. 2.3 и 2.4.

Таблица 2.3

Момент	Вход	Сост	оян ие	на эт	апе
времени	(выход)	1	2	3 1	4
t	*T*	0	1	0	1
	Q	0	0	1	1
		0	1	1	0
t+1	*Q*				

Таблица 2.4

Момент времени	Вход (выход)	Сос 1	оян раб 2	на от 3	гапе 4
t	*J*	O	0	1	1
	к;	O	1	0	1
t+1	Q	*QW*	0	1	Q(t)

Из табл. 2.3 видно, что триггер *T* изменяет свое состояние на противоположное при каждом поступлении сигнала на его вход. Триггер *JK* переключается в состояние / при наличии сигнала на входе / и ОТСУТСТВИИ сигнала на входе *К* (^=1, *К=0),* а переключение в состояние *0* происходит при отсутствии сигнала на входе / и его наличии на входе *К: [J=0,* Я=1). При одновременном поступлении сигналов на входы *J* и *К* (/=1, ^С=1) триггер переключается в состояние *Q(t).,* обратное тому, которое имело место до поступления входных сигналов.

На основе триггеров строятся счетные схемы (счетчики) серии 155 счетчики на микросхемах

К155ИЕ2, К155ИЕ4 и К155ИЕ5 содержат по четыре триггера и имеют два счетных входа, четыре выхода с триггеров и два входа установки в исходное

Счетчик на микросхеме К155ИЕ2 производит счет в двоичном коде до девяти импульсов. Кроме двух входов установки в исходное состояние схема имеет два входа для переключения триггеров в девятое состояние. Счетчик на микросхеме К155ИЕ4 производит счет до 11 импульсов а счетчик на микросхеме К155ИЕ5, функциональная схема которого показана на рис. 2.9,а,—до 15 импульсов. Один из счетных входов схемы (Ci) переключает первый триггер, а второй (Са) — воздействует на остальные три триггера. Для того чтобы схема могла производить счет до 15 импульсов, необходимо подать нулевой потенциал на один из входов *Ro* и соединить выход Qi с входом *С2.* В этом случае вход *С,* будет счетным входом всей схемы.

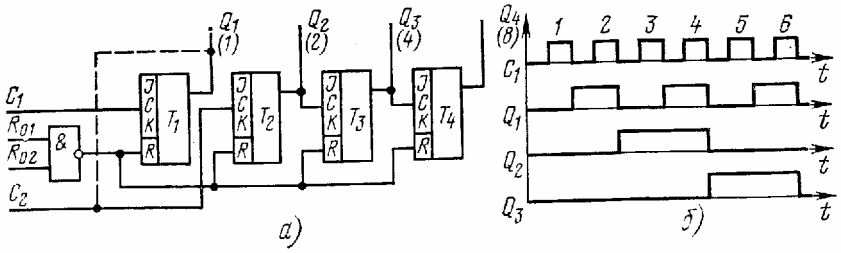


Рис. 2.9

Если необходимо построить счетчик на меньшее число импульсов, то соответствующие выходы триггеров непосредственно или через дополнительную схему подключаются к входам *Roi* и *Ro-г.* Например, для построения счетчика на шесть импульсов выходы Qi и Оз подключаются к входам *Roi* и Ro2. В этом случае после окончания шестого импульса все триггеры счетчика устанавливаются в состояние 0. Диаграмма работы счетчика на шесть импульсов показана на рис. 2.9,6.

Другой разновидностью схем, широко используемых в устроиствах связи, являются дешифраторы, используемые для перехода от двоичного кода на код десятичный. К числу таких дешифраторов относятся микросхемы К155ИД1, К155ИДЗ и К155ИД4.

Дешифратор на микросхеме К155ИД1 (рис. 2.10,а) имеет четыре адресных входа *(J, 2, 4, 8)* и десять выходов. Выходы де шифратора с открытым коллектором предназначены для включения газоразрядных

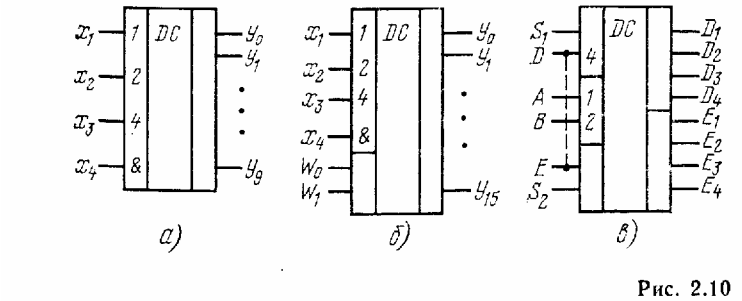


Рис. 2.10

цифровых индикаторов с напряжением питания 200—300 В. Дешифратор на микросхеме К155ДИЗ (рис. 2.10,6) предназначен для совместной работы с микросхемами серии ТТЛ и имеет четыре адресных входа *xi, Хг, Хз, х^,* два входа стробирования *(Wo, Wi)* и 16 выходов *{уо, у\, ...,* t/is).

В случае, когда на обоих входах стробирования имеется логический нуль, на одном из выходов также появляется логический нуль в соответствии с двоичным кодом на адресных входах. При наличии на одном или обоих входов *Wo, Wi* единицы, все входы будут иметь уровень логической единицы.

Если необходимо построить дешифратор большей емкости, например на 32 выхода, то один из входов *Wo* или *Wi* первого дешифратора подключается через схему НЕ к аналогичному входу второго дешифратора. Объединенный'стробирующий вход первого дешифратора будет адресным входом с весом 16, а выходы второго дешифратора будут иметь нумерацию 16—31.

Дешифратор на микросхеме К.155ИД4 состоит из двух дешифраторов на четыре выхода с объединенными адресными входами *А, В* и раздельными входами стробирования Si, *D* для первых четырех выходов *Di* и *Е, S'>* — для вторых четырех выходов *Е{.*

При наличии единицы на входе *D,* нуля на входе *Si* и единицы на входе *Е* или Sg на одном из выходов *Ог* появится потенциал логического нуля, соответствующий двоичному коду на входах *А* и *В.* Потенциалом логического нуля на входах *Е, Sa* и нулем на входе *D* или единицей на входе 5i производится подключение входов *А* и *В к* выходам *Ег.*

Микросхема может использоваться как два дешифратора на четыре выхода или как один на восемь выходов. В последнем случае третьим информационным входом с весом 4 будет вход *D,* объединенный с одним из входов *Е* или *Ss,* а объединенный вход *Si с* другим из входов *Е* или *Ss* будут стробирующим

ходом всего дешифратора.

Кроме рассмотренных микросхем серия 155 содержит ряд мультиплексоров, ОЗУ и ПЗУ, принцип действия которых приводится в [35].

Контрольные вопросы

1. Как производятся включение и выключение герконов феррядовых реле с компенсацией магнитных потоков?
2. Поясните принцип работы ферридового реле с размагничиванием магнитной системы.
3. В чем заключается отличие в принципах работы МФС и МИС?
4. Как строится схема счетчика с уменьшенным числом импульсов счета?
5. Каковы особенности работы различных дешифраторов серии 155?

Глава 3.

ФОРМАЛИЗАЦИЯ ЗАДАНИЯ НА ПРОЕКТИРОВАНИЕ ДИСКРЕТНОГО УСТРОЙСТВА

3.1. Автоматная модель дискретного устройства

Дискретное устройство с логическими и запоминающими функциональными элементами можно представить (рис. 3.1) в виде ориентированного (л, Д)-полюсника с *n* входами *Xi, ... Xn; с а* выходами г,, ..., ^ и с s обратными связями, каждая из которых имеет задержку или, как будем говорить, *элемент памяти* (сШ). Часть устройства, в которой сосредоточены логические элементы (элементы, реализующие операции алгебры логики И, ИЛИ, НЕ и т. п., а также контакты), образующие одноактную схему, принято называть *логическим преобразователем* (ЛП) устройства.

На входы ЭП воздействуют сигналы, снимаемые с дополнительных (внутренних) выходов ЛП 2^ , ..., ^, . С выходов ЭП сигналы воздействуют на дополнительные (внутренние) входы ЛП *Xn+\, ..., Xn+s*- Будем полагать, что каждый ЭП может находиться в одном из двух возможных состояний, а все сигналы — двузначные. Формальной моделью таких дискретных устройств в теории автоматов является конечный автомат [1, 2].

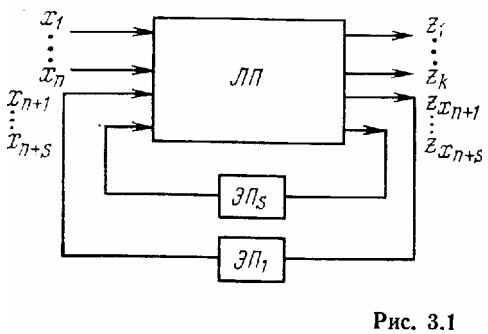


Рис. 3.1

Говорят, что i-й набор значений входных сигналов, воздействующих на основные входы ЛП *X1, ..., Xn,* образует состояние входа pi, причем при двузначных входных сигналах всего может быть 2^k их значений, а значит, и *N=2^n* состояний входа pi, ..., pk. Каждый из 2^k наборов значений двузначных выходных сигналов, снимаемых с основных выходов ЛП z1, ..., zk, образует одно из КС=2^k состояний выхода Л1, ..., *Лк.* Заметим, что Л-n набор состоянии ЭП определяет у-й набор значений сигналов, воздействующих на дополнительные входы ЛП Xn+1, .., *Xn+s.* При s двузначных ЭП число их состояний 5= 2^8.

Конечным автоматом называется устройство, определяемое конечными множествами состояний входа {pi, ..., pn), выхода (Л1, ..., Л*к),* внутренних состояний {Xo, X1, ..., Xs-i}, а также функциями переходов и выходов, определяющих последовательность смены внутренних состояний и состояний выходов автомата в зависимости

от смены состояний входа. Из множества внутренних состояний выделяется некоторое состояние Хо, называемое *начальным внутренним состоянием автомата.*

Конечный автомат может быть представлен в виде рассмотренного выше ориентированного ЛП с s обратными связями, причем состояния входа и выхода ЛП являются соответственно состояниями входа и выхода конечного автомата. Внутренними состояниями конечного автомата являются состояния ЭП. Очевидно, что множества состояний входа и выхода конечны, так как число двузначных входов и выходов конечно. (Рассматриваются только двузначные сигналы.)

Множество внутренних состояний характеризует память автомата. В том случае, когда множество внутренних состояний не является конечным, автомат выходит из класса конечных и нами рассматриваться не будет. Далее описываются только конечные автоматы, поэтому термин «конечный» часто будем опускать.

Предполагается, что автомат функционирует в дискретные моменты времени, т. е. непрерывная шкала времени разделена на интервалы, которые занумерованы целыми положительными числами О, 1, 2, ... *t,* ... и называются *тактами.* На протяжении одного такта сохраняются неизменными состояния входа, выхода и внутреннее состояние автомата. Время, в течение которого состояние входа автомата не изменяется, обозначим через *Т.* В зависимости от того, чем определяется длительность этого интервала времени, различаются два класса автоматов: синхронные и асинхронные.

Синхронные автоматы характеризуются тем, что имеется генератор тактовых (или синхронизирующих) сигналов (ТГ) и входные сигналы могут воздействовать на автомат лишь при наличии сигнала от тактового генератора (рис. 3.2,а). Таким образом, время *Т* строго фиксировано и определяется длительностью сигнала тактового генератора. Автомат может воспринимать новое состояние входа лишь после того, как он перешел в определенное внутреннее состояние. Поэтому частота тактового генератора выбирается такой, чтобы до появления следующего сигнала от тактового генератора автомат уснел перейти в это внутреннее состояние, т. е. необходимо обеспечить соотношение Д>Tmax, где Д=Г+6, 6 — интервал времени между двумя соседними сигналами тактового генератора; Tmax — наибольшее время перехода автомата из одного внутреннего состояния в другое. Обычно длительность этого сигнала выбирают такой, что Т<=Tmax, т. е. внутреннее состояние автомата изменяется лишь в интервалах между двумя соседними сигналами тактового генератора, когда входные сигналы не воздействуют на автомат. Поэтому определенному моменту времени *ti* соответствует определенное внутреннее состояние автомата.

Таким образом, для синхронных автоматов интервал времени *ti* совпадает с тактом работы автомата, поскольку состояние входа и внутреннее состояние автомата на протяжении этого времени остаются неизменными. Обычно в абстрактной теории автоматов не интересуются поведением автомата в интервалах времени 6, а, считая, что переход автомата из одного внутреннего состояния в другое происходит мгновенно, рассматриваются лишь интервалы времени *Т,* каждый из которых сопоставляется с моментом времени *ti.* При этом может оказаться, что два соседних такта, определяемых двумя последовательными моментами времени *ti* ^i+i, будет соответствовать одним и тем же состояниям входа. **Это** также является характерной чертой синхронных автоматов.

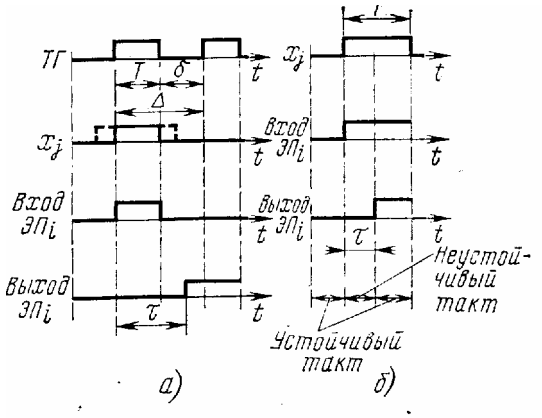


Рис. 3.2

В асинхронных автоматах длительность интервала времени *Т,* в течение которого остается неизменным состояние входа, является величиной переменной и определяется только моментами изменения состояния входа. Поэтому сколь продолжительным не был бы интервал времени, в течение которого сохраняется неизменным состояние входа, он будет восприниматься автоматом как один и тот же интервал, времени. Следовательно, двум последовательным интервалам времени ^ и ^;+i всегда должны соответствовать различные состояния входа.

Обычно считают, что состояние входа асинхронного автомата может измениться лишь после того, как автомат перешел в определенное внутреннее состояние (рис. 3.2,6). Для этого требуется, чтобы выполнялось соотношение 7\пгп»Ттах, где Г^п — наименьший интервал времени, в течение которого сохраняется неизменным состояние входа. В связи с этим изменение внутреннего состояния автомата происходит при неизменном состоянии входа, так что одному и тому же интервалу времени *Т* может соответствовать несколько различных внутренних состояний. Поэтому интервал времени *Т* будет разделен на несколько тактов. Таким образом, переход к новому такту вызывается не только изменением состояния входа, но и изменением внутреннего состояния автомата. Такт, предшествующий моменту изменения состояния входа автомата, называют *устойчивым.* Такт, предшествующий моменту изменения внутреннего состояния автомата называют *неустойчивым.* Длительность неустойчивого такта определяется временем т перехода автомата из одного внутреннего состояния в другое и зависит

от параметров элементов, образующих структуру автомата. В ряде случаев автомат работает так, что на протяжении одного и того же интервала времени *Т* может быть несколько неустойчивых тактов. Поэтому должно выполняться не равенство *Тmгп'ё>Tmax,* чтобы при любом конечном числе *г* неустойчивых ТаКТОВ *ГmaxГ<Tmгп* Функционирование, или поведение автомата при заданных множествах **{рг},** {^'}, **{^г}** и начальном внутреннем состоянии **хо,** полиостью детерминировано и определяется функциями переходов и выходов. Функция переходов устанавливает зависимость внутреннего состояния автомата в следующий момент времени *(t+l)* от состояния входа и внутреннего состояния в настоящий момент времени *t:*

$$\varkappa(t+1) = \varphi[\rho(t);\ \varkappa(t)]. \qquad (3.1)$$

Функция выходов устанавливает зависимость состояния выхода автомата от состояния входа и внутреннего состояния автомата в один и тот же момент времени *t:*

$$\lambda(t) = \Gamma[\rho(t);\ \varkappa(t)]. \qquad (3.2)$$

В выражениях (3.1) и (3.2) *p(t)*—состояние входа автомата в момент (такт) *t,* принимающее одно из значений **{pi, ..., p^};**

X(^) —состояние выхода автомата в момент времени *t,* принимающее одно из значений {^i, ..., *Кк}', •к (t)* —внутреннее состояние автомата в момент времени *t,* принимающее одно из значений {яо, кб ..., xs-i}.

Часто удобно использовать понятие «состояние автомата», которое, в отличие от внутреннего состояния, иногда называют *полным.* Состояние автомата р. в момент времени *t* определяется внутренним состоянием и состоянием входа автомата в этот же момент времени. Заметим, что множество состояний *{\u\,* [iz, ..., **[лд}** (причем *u(t}<={u.m})* конечного автомата всегда конечно, так как *R=NS.*

Таким образом, внутреннее состояние автомата в момент времени ;'+! и состояние выхода автомата в момент *t* определяется состоянием автомата в момент времени *t:*

$$\varkappa(t+1) = \varphi[\mu(t)]; \quad \lambda(t) = \Gamma[\mu(t)]. \qquad (3.3);\ (3.4)$$

Задавая для всевозможных пар р(0 и я (0 то или иное значение х(/+1) и то или иное значение *K(t),* определяем соответственно функции переходов и выходов. Автоматы, для которых функции переходов и выходов определены на всех парах [р(it); я (О], называют *полностью определенными* или *полными.* Автоматы, для которых функции выходов или переходов определены не на всех этих парах, называют *недоопределенными.* При этом автоматы, у которых для какой-либо **пары** [р(0; х(0] функция выходов определена, а функция переходов не определена, обычно не рассматриваются.

Состояние *и. (t)* недоопределенного автомата, соответствующее паре *[(>(t); K(t}],* на которой функция переходов не определена, называют *неиспользуемым.* Остальные состояния будем называть *используемыми.* Если на каком-либо используемом состоянии автомата функция выходов не определена, то будем говорить, что ему соответствует безразличное состояние выхода *.

Поведение асинхронного автомата определяют следующими уравнениями:

$$\varkappa(t+1) = \psi[\rho(t+1);\ \varkappa(t)]; \qquad (3.5)$$

$$\lambda(t+1) = \varphi[\rho(t+1);\ \varkappa(t+1)]. \qquad (3.6)$$

Первое из этих уравнений показывает, что внутреннее состояние автомата в момент времени *t+1* определяется состоянием входа автомата в этот же момент и внутренним его состоянием в предыдущий момент. Заметим, что в синхронных автоматах внутреннее состояние определялось состоянием входа в предшествующий момент времени [см. выражение 3.1)].

В классе асинхронных автоматов также рассматриваются полные и недоопределенные автоматы.

Кроме разделения автоматов на синхронные и асинхронные все автоматы могут быть разделены на автоматы с памятью и без памяти. *Автоматом без памяти,* или *комбинационным,* называется автомат, имеющий одно внутреннее состояние. Состояние выхода такого автомата однозначно определяется состоянием его входа и функционирование автомата однозначно определяется функцией выходов

$$\lambda(t) = \Gamma^*[\rho(t)]. \qquad (3.7)$$

Заметим, что в частном случае может быть безразличным те состояние выхода, а лишь один **или** несколько выходных сигналов, набор значении которых определяет данное состояние **выхода.**

Время *t* при этом часто опускается.

Следует отметить еще один частный тип автоматов, а именно таких автоматов, для которых **{pi},** т. е. множество состояний входа, состоит лишь из одного элемента pi. Этот случай возможен тогда, когда автомат не имеет входов или, что то же самое, состояние его входа имеет неизменное значение pi. Такие автоматы называют *автономными.* Все остальные автоматы относятся к *неавтономным.* Поскольку далее рассматриваются только эти автоматы, слово «неавтономный» опускается.

Все автоматы можно разделить на автоматы общего вида (универсальные) и автоматы того или

иного частного вида (специализированные, проблемно-ориентированные и т. п.). В отличие от автоматов общего вида, специализированные автоматы обладают какой-либо характерной особенностью реализуемых или алгоритмов функционирования применяемых элементов и т. д. Примером одного из таких частных классов автоматов может служить получивший широкое распространение в автоматике и вычислительной технике микропрограммный автомат. Другим примером специализированного вида автоматов является автомат с распределенной задержкой.

Микропрограммный автомат (МА) может рассматриваться в качестве формальной модели ЦБУ дискретного устройства с концентрированными функциональными связями (см. рис. 1.4,6). Общая структура микропрограммного автомата изображена на рис. 3.3.

Микропрограммный автомат вырабатывает последовательность воздействий 2л; и Zp/ для включения операторных функциональных блоков (ОФБг) и логических функциональных блоков (ЛФБд) соответственно.

При воздействии на ЛФБ, вырабатывается одно из двух возможных значений проверяемого параметра *p,,* которое поступает на вход *pj* микропрограммного автомата. Следовательно, на вход *pj* воздействие может поступить лишь в том случае, если имеется воздействие на выходе 2р/. Поэтому все входы МА разделяются на две группы: внешние /-i, ..., *Гп* и внутренние *pi, ..., pm*- Выходы ^рг •••' ^{Z}Pm иззовем внутренними, а ^{ZA}^, •••> ^{Z}Att—внешними.

Зависимость поступления воздействий на внутренние входы микропрограммного автомата от его выходных воздействии — одна из основных особенностей микропрограммного автомата. Второй основной особенностью МА является то, что при неизменном состоянии внешнего входа автомат вырабатывает последовательность сигналов на основных выходах.

Следует заметить, что в частном случае в микропрограммном автомате могут отсутствовать ЛФБ. Тогда микропрограммный автомат при неизменном состоянии внешнего входа вырабатывает «жесткую» последовательность воздействий на ОФБ, а при отсутствии внешних входов — лишь одну из таких «жестких» последовательностей.

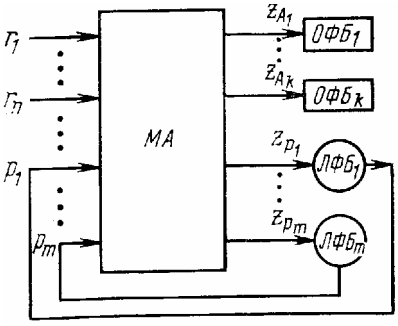


Рис. 3.3

Набор значений сигналов на входах т-i, ..., *Гп* образует состояние внешнего входа *Ri<=:{Ri, ..., Ri},* а на входах *pi, ...,* pm — состояние внутреннего входа *P]^{N, Pi, ..., Рм},* где *N* — состояние внутреннего входа при отсутствии воздействий на всех внутренних выходах *Zp, ..., гр*^. Пара *[Ri; Pj]* образует состояние (полное) входа микропрограммного автомата — pv- При L=1 микропрограммный автомат называется автономным.

Состоянием внешнего выхода является набор значений сигналов на выходах *гд, ..., ZA^,* а состоянием внутреннего выхода — набор значений сигналов на выходах 2р,,.., *Zp*^.

Микропрограммный автомат может иметь S внутренних состояний (S^sl). При 5=1 микропрограммный автомат, очевидно, становится комбинационным. При 5>1 среди внутренних состояний МА кроме начального состояния х(0)=хо выделяют некоторое число конечных состояний.

Конечным яд называют такое внутреннее состояние, из которого автомат не может перейти в ругое при неизменном состоянии внешнего входа. Обычно предполагается, что изменение состояния внешнего входа возможно только тогда, когда МА находится в начальном или конечном внутреннем состоянии.

Если класс МА определяется особенностью его алгоритма функционирования при взаимодействии с функциональными блоками, то класс автоматов с распределенной задержкой — особенностью применяемого элементного базиса. В автомате с распределенной задержкой каждый элемент логического преобразователя (ЛП) кроме реализации логической функции / обеспечивает также задержку т (рис. 3.4,а). Из модели данного элемента, изображенной на рис. 0.4,6, видно, что сигнал на входе элемента задержки т формируется только одним логическим элементом /, который иногда называют *модулем.* Таким образом, при представлении автомата с распределенной задержкой в виде модели с ЛП и элементом памяти (ЭП) в виде задержки т получим схему, изображенную на рис. 3.5. Таким образом, характерной особенностью ЛП модели автомата с распределенной задержкой является то, что в нем имеется только один каскад логических элементов, а объем памяти автомата определяется совокупностью задержек всех логических элементов ЛП. Автоматы с распределенной задержкой получают все большее распространение в связи с применением интегральных схем для по строения быстродействующих дискретных устройств.

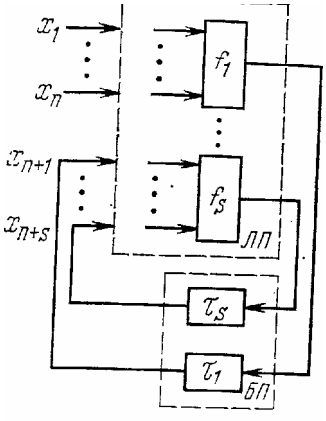


Рис. 3.5

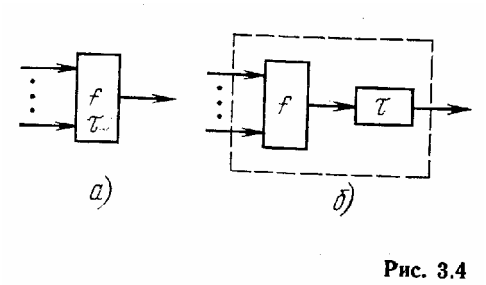


Рис. 3.4

3.2. Автоматные языки описания условий работы проектируемого дискретного устройства

Для задания автомата существуют различные способы или как говорят, языки. Рассмотрим лишь некоторые из них, получившие в последнее время наибольшее применение.

Для того чтобы задать алгоритм функционирования автомата необходимо задать его функции переходов и выходов. Задание функции переходов означает задание определенного перехода из внутреннего состояния -*Лг* во внутреннее состояние *щ* (не исключается случай 1=/') при воздействии состояния входа ру. Задание функции выходов означает, что каждой паре (ру; к<) сопоставлено состояние выхода *Ki* автомата.

Часто говорят, что задание функций переходов и выходов означает задание однозначных отображений множества пар {[р„; *Кг]}* в множества ***{к,}*** **и {Хг}. Я**зыки, позволяющие таким образом задать функции переходов и выходов автомата, будем называть *стандартными.* К стандартным языкам относятся, например, таблицы переходов, матрицы переходов, диаграммы переходов, таблицы включений и т. д. Из всех известных стандартных языков в последнее время в качестве языка задания автомата общего вида основное применение нашли таблицы переходов.

Наряду со стандартными существуют и другие языки задания алгоритма функционирования релейного устройства, с помощью которых могут быть заданы отображения последовательностей состояний входа (входных последовательностей) в последовательности состояний выхода (в выходные последовательности), т. е. отображения множества слов во входном алфавите в множестве слов в выходном алфавите. Функция же переходов с помощью таких языков в явном виде не задается.

Вначале алгоритм функционирования автомата часто удобнее записать именно на таком языке, условно называемом *начальным.* К начальным языкам относятся, например, первоначальные таблицы включений и логические схемы алгоритмов. В этом случае автомат рассматривается как «черный ящик», когда из условий взаимодействия с другими устройствами известно лишь о входных и соответствующих им выходных последовательностях.

Для задания автоматов общих классов начальные языки не получили практического применения. Для задания одного из частных классов автоматов, а именно микропрограммных автоматов, оказался удобным язык логических схем алгоритмов.

Все известные начальные языки предназначены для описания в каком-либо виде условий работы автомата, заданных множеством пар последовательностей (в том числе и бесконечных) состояний входа и выхода автомата, т. е. множеством последовательностей «вход — выход». Каждая пара такого множества включает входную последовательность, т. е. входное слово, буквами которого являются состояния входа автомата, и однозначно сопоставленную ей выходную последовательность, т. е. выходное слово, буквами которого являются состояния выхода автомата. Например, условиями работы задана следующая последовательность «вход—выход»:

P^Ki, **р.2^2> Pl~> рз^з-**

Тогда если множеством состояний выхода автомата является множество *{Ki, Кч,* U, то построенный по этим условиям автомат будет реализовать одну из следующих трех последовательностей:

l^l' P2^2> Pl^l, Рз^з;
Pl^.1,

P2^2' Pl^2> рз^-3'

Pl^li ?2^2» Pl^3> рз^з-

При задании условий работы автомата, служащего моделью реального устройства, часто удобно говорить не о состояниях входа и выхода, а о последовательности смены значения сигналов на входах и выходах автомата. Тогда вместо последовательности «вход — выход» будет задаваться последовательность значений сигналов на входах и выходах автомата. Для задания такой последовательности значений обычно используются первоначальные таблицы включений [I].

Пример первоначальной таблицы включений для счетчика на пять импульсов приведен на рис. 3.6.

Как видно из рисунка, сигнал на входе *x* принимает значение «I» в тактах *2, 4, 6, 8, 10,* а в остальных тактах—значение «О». После такта *10* вновь наступает такт 7, т. е. цикл состоит из десяти тактов. В момент появления на входе *x* значения «I» в пятый раз (такт *10}* на выходе *z* возникает значение «I», сигнализирующее о том, что в счетчик поступило пять импульсов.

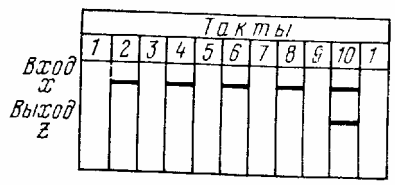


Рис. 3.6

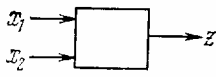


Рис. 3.

Пример 3.1. Необходимо составить таблицу включений для автомата с двумя входами *Xi,* хз и одним выходом z (рис. 3.7), условия работы которого состоят в следующем. Вначале появляется значение «I» сигнала на входе *Xi,* а затем на .входе *x-г.* (В дальнейшем для простоты будем говорить, что на входе или выходе появился сигнал.) В момент появления сигнала на входе *x* автомат вырабатывает значение «I» сигнала на выходе *г,* которое остается и после пропадания сигнала на входе ^ до тех пор, пока не исчезнет сигнал на входе *Xi.*

Решение. Очевидно, в таблице включении должны быть две строки для входов A-I и *x:* и одна строка для выхода z. Первый такт является начальным. Поэтому в первой строке наличие сигнала на входе *x* будем указывать, начиная со второго такта (рис. 3.8,а). В третьем такте появляется сигнал на входе *Xt,* а сигнал на .входе *x* сохраняет свое единичное значение. При этом возникает сигнал на выходе z (рис. 3.8,6). В четвертом такте исчезает сигнал на входе *Хг,* а сигналы на входе *Xi* и выходе *г* сохраняют свое предыдущее значение (рис. 3.8,б). В следующий момеят времени (пятый такт) исчезает сигнал на входе *xi,* что приводит к исчезновению сигнала на выходе *г* (рис. 3.8,г). 1ак как после четвертого такта автомат приходит в исходное (начальное) состояние, то .вместо пятого такта целесообразно указывать первый такт (такт 1 на рис. 3.8,г указан в скобках). На этом процесс построения таблицы включений заканчивается. Таким образом, первоначальная таблица включений имеет четыре такта.

Пример 3.2. Необходимо задать условия работы автомата (см. рис. 3.7), который должен реализовать две последовательности «вход-выход». Пусть одна последовательность останется той же, что и в условиях примера 3.1, а вторая последовательность является следующей: первым появляется сигнал на входе ***x2,*** а затем на входе *x1.* При этом в момент наличия сигналов на входах X1 и X2 допустимо (но не обязательно) возникновение сигнала на выходе z.

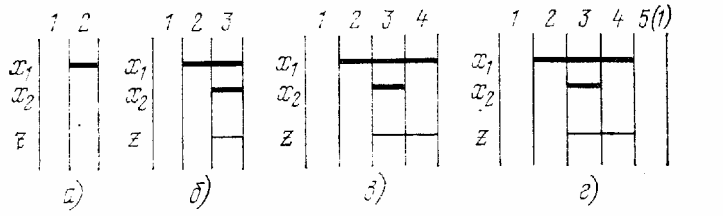


Рис. 3.8

Решение. Для двух последовательностей необходимо иметь уже две первоначальных таблицы включений, одна из которых была составлена ранее (см. рис 38 е) Процесс составления второй таблицы включении ясен из рис. 3.9,а-г, где штриховой линией обозначено безразличное значение сигнала на выходе. Таким образом, в данном случае автомат задается двумя таблицами включении, приведенными на рис. 3.8,г и рис. 3.9,г.

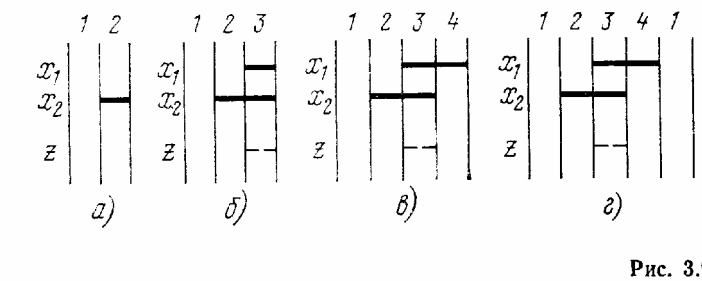


Рис. 3.9

Очевидно, что при большом числе последовательностей язык таблиц включений становится крайне громоздким, поэтому он не нашел широкого применения, хотя для некоторых классов автоматов типа счетчиков он оказывается очень удобным. Для автоматов, условия работы которых задаются относительно большим числом последовательностей, может быть использован язык таблиц переходов, который уже относится к классу стандартных языков. Вначале рассмотрим таблицу переходов для синхронного автомата, а затем остановимся на особенностях задания асинхронных автоматов.

Таблица переходов задает функцию переходов автомата. Каждая строка этой таблицы соответствует определенному внутреннему состоянию автомата, каждый столбец - определенному

состоянию входа автомата. Клетке таблицы переходов соответствует состояние автомата, определяющее внутреннее состояние, в которое автомат должен перейти в следующий момент времени. Если в автомате какое-либо состояние не определено, т. е. является неиспользуемым*, то в соответствующей клетке ставится черточка. Например, в табл. 3.1, задающей недоопределенный синхронный автомат, состояния (А^, (А'З, (^4 являются неиспользуемыми. (Через ^*г* будем обозначать состояние автомата, соответствующее внутреннему состоянию Хг и состоянию входа PJ.)**

Таблица 3.1

	ρ_2
$\varkappa_1$	1, λ_1 | 2, λ_2
$\varkappa_2$	3, λ_3 | —
$\varkappa_3$	— | 4, λ_4
$\varkappa_4$	1, λ_1 | —

Из табл. 3.1 видно, что автомат имеет два состояния входа (pi и рг) и четыре внутренних состояния (xi, кг, *кз* и К4). Если автомат находится во внутреннем состоянии •*n.i,* то при воздействии на него состояния входа pi он не изменит своего внутреннего состояния, а при воздействии состояния входа ра он перейдет во внутреннее состояние кг.

Неиспользуемые состояния в автомате обычно возникают из-за ограничения входных последовательностей. Например, в табл. 3.1 не могут быть два одинаковых состояния входа, следующих один за другим. Однако возможны и другие причины возникновения неиспользуемых состояний входа, например когда безразлично, в какое внутреннее состояние перейдет автомат.

* Иногда такое такое состояние называют запрещенным

** Здесь и в дальнейшем для простоты в клетках таблицы переходов вместо символа внутреннего состояния x, будем указывать только один его номер i.

Функция выходов автомата также может быть задана в виде таблицы, при этом в таблице задается соответствие состояния выхода состоянию автомата. В случае совмещения таблицы выходов с таблицей переходов автомата получим одну таблицу, задающую одновременно как функцию переходов, так и функцию выходов соответствующего автомата (см., например, табл. 3.1).

В таблице переходов вместо состояний входа и выхода часто указывают значения кодирующих их переменных (значения отдельных входных и выходных сигналов).

В том случае, когда выходной сигнал в одном из состояний может иметь любое значение, в соответствующей клетке вместо нулевого или единичного значения этого сигнала ставится тильда'.

Состояния входа синхронного автомата, указанные в таблице переходов, соответствуют тем моментам, в которые на входе автомата присутствует сигнал от тактового генератора. В таблице переходов с целью упрощения данный сигнал обычно не указывается и только предполагается (табл. 3.2), что имеется еще столбец *N* который соответствует нулевому значению этого сигнала. Напомним, что при нулевом значении синхронизирующего сигнала синхронный автомат не воспринимает состояния входа, поэтому они все равны нулю. Заметим, что в столбце *N* все выходные сигналы могут иметь любые значения, так как предполагается, что в синхронном автомате они появляются лишь в моменты присутствия на входе автомата синхронизирующего сигнала.

Рассмотрим теперь особенности таблицы переходов, задающей асинхронные автоматы различного вида. Каждая строка таблицы соответствует определенному внутреннему состоянию автомата, каждый столбец — определенному состоянию входа автомата. Клетка таблицы, определяемая состоянием входа и внутренним состоянием автомата, соответствует полному состоянию автомата. Устойчивое состояние автомата, т. е. состояние, соответствующее устойчивому такту, обозначается цифрой в круглых скобках, а неустойчивое состояние (соответствующее неустойчивому такту), предшествующее данному устойчивому состоянию, — той же цифрой без скобок. Переход автомата из одного внутреннего состояния в другое всегда связан с переходом его в неустойчивое состояние.

Таблица 3.2

N	ρ_1	ρ_2
1, ⌣⌣	1,00	2,10
2, ⌣⌣	3,01	—
3, ⌣⌣	—	4,11
4, ⌣⌣	1,00	—

Если состояние входа *рк* вызывает переход автомата из внутреннего состояния к, во внутреннее состояние *к,,* то автомат перейдет сначала в неустойчивое состояние ^*г,* соответствующее внутреннему состоянию **и,** и состоянию входа *рк,* и только после этого он может перейти в устойчивое состояние соответствующее внутреннему состоянию *к,* и тому же состоянию входа ря.

В табл. 3.3 приведен пример таблицы переходов, задающей функции переходов и выходов асинхронного автомата. Если, напри мер, автомат находится в состоянии (2) и состояние входа его меняется с ра на рз, то он перейдет в состояние (4) и внутреннее состояние его при этом не изменится. Если автомат находится в том же состоянии (2), но состояние входа его меняется с ра на pi, то он вначале перейдет в неустойчивое состояние 1, а затем в устойчивое состояние (1), при этом внутреннее состояние его изменится с иг на Hi.

Таблица 3.3

Внутреннее состояние автомата	Состояние входа			
	ρ_1	ρ_2	ρ_3	ρ_4
1	(1), 11	2, ∽0	3,01	—
2	1,11	(2), ∽0	(4), 00	5,1 ∽
3	6,00	—	(3), 01	(5), 1 ∽
4	(6), 00	2, ∽0	(7), 10	—

Таким образом, характерной чертой асинхронных автоматов является то, что изменение внутреннего состояния автомата всегда связано с переходом его через неустойчивое состояние: переход автомата из одного внутреннего состояния в другое через устойчивое состояние невозможен.

Таблицы переходов и выходов, задающие функции переходов и выходов асинхронного автомата, полностью определяют его функционирование.

Обычно с неустойчивым состоянием автомата сопоставляется то же состояние выхода, что и с соответствующим ему устойчивым состоянием. Поэтому в таблице переходов состояния выходов в неустойчивых состояниях часто не указывают, если специально не оговорено, что с неустойчивым состоянием сопоставляется иное состояние выходов, чем с соответствующим устойчивым состоянием.

Если состояние выхода автомата сопоставляется не с полным, а с внутренним состоянием, то таблица переходов будет такой, как указано в табл. 3.4.

Таблица 3.4

Внутреннее состояние автомата		Состояв	ие входа		FСостоят	не выхода
	Pi	P,	P,	P<	2i	*z,*
	(1)	2	3		1	*0*
2	1	(2)	(4)	5	0	*1*
3	6		(3)	(5)	«-»	1
4	(6)	2	(7)		0	^

В связи с тем, что таблицы переходов относятся к классу стандартных языков, при первоначальной формулировке условий работы приходится иметь дело с внутренними состояниями автомата. Однако при первоначальной формулировке условий работы автомата на языке таблиц переходов с целью упрощения процесса ее построения не стремятся использовать минимальное число внутренних состояний. Наоборот, при составлении таблицы переходов каждое изменение состояния входа сопровождается введением нового внутреннего состояния. При этом будет получена так называемая первоначальная таблица переходов, в каждой строке которой будет только одно устойчивое состояние.

Пример 3.3. Условия работы автомата сформулированы в примере 3.1.

Решение. Поскольку имеются два входа (два входных сигнала): *x* я *x-г,* — таблица переходов, вообще говоря, должна содержать 22=4 столбца—по числу возможных состояний входа. Однако из условии работы следует, что из числа возможных состояний входа pi—00, рг—Ю, рз—11, р4—01) состояние р4—01, когда только на одном входе *x<г* имеется сигнал, недопустимо (это же видно из рис. 3.8,г). Поэтому для рассматриваемого примера таблица переходов имеет три столбца. Число строк заранее неизвестно.

Вначале вводим одно внутреннее состояние, в котором указываем начальное устойчивое полное состояние (1) автомата с указанием состояния 0 аа выходе *г* (табл 35) Далее из условий работы следует, что после состояния входа pi наступает состояние входа ра (появляется входной сигнал *x,}.* Поэтому вводим новое внутреннее состояние (яа), причем в столбце ра в строке и, (т е во внутреннем состоянии Xi) указываем неустойчивое состояние 2, а в строке Яа -устойчивое состояние (2) с указанием соответствующего значения выходного сигнала z (табл. 3.6).

Дальнейший процесс построения первоначальной таблицы переходов легко проследить по табл. 3.7-3.9, которые даются в упрощенном виде.

Таблица 39 является первоначальной таблицей переходов, описывающей заданные условия .работы автомата. В ней четыре строки, соответствуют четырем ВНУТрОННИМ СОСТОЯНИЯМ Я]—Х4.

Пример 3.4. Необходимо построить первоначальную таблицу переходов по условиям работы автомата, указанным в примере 3.2.

Решение Вначале таблицу переходов построим исходя только из одно» первой последовательности (т. е. получим табл. 3.9), атем добавим переходы, соответствующие второй последовательности. При этом таблица переходов будет иметь уже четыре столбца. Процесс дальнейшего построения первоначаль ной таблицы переходов для данного примера легко проследить из рассмотрения табл. 3.10—3.13.

Таблица 3.5

Внутреннее состояние Автомата	Состояние входа P1 X^X_2 0 0	P2 X^X_2 1 0	P3 X_tX_t 1 1
1	(1),0		

Таблица 3.0

Внутреннее состояние	Состояние входа Pi $X_iX.i$ 0 0	P_2 X_tX_t 1 O	P_s X^X_s 1 1
1	(1), 0	2,0	
2		(2). 0	

Таблица 3.7

X_iX2; z			
1	(1),0	2,0	
2		(2),0	3, 1
3			(3),!

Таблица 3.8

X_iX_t; z	O O 0 0	1 0	1 1
1	(1),0	2,0	
2		(2),0	3, 1
3		4, 1	(3),!
4		(4), 1	

Таблица 3.9

	1 1		10	11
1	(1),0	2,0	—	
2	—	(2),0	3, 1	
3	—	4, 1	(3),!	
4	1,0	(4).!	—	

Таблица 3.10

x_1x_2; z	00	10	11	01
1	(1), 0	2,0		5,0
2		(2), 0	3,1	
3		4,1	(3), 1	
4	1,0	(4), 1		
5				(5), 0

Таблица 3.11

x_1x_2; z	00	10	11	01
1	(1), 0	2,0		5,0
2		(2), 0	3,1	
3		4,1	(3), 1	
4	1,0	(4), 1		
5			6, ∽	(5), 0
6			(6), ∽	

Таблица 3.12

x_1x_2; z	00	10	11	01
1	(1), 0	2,0		5,0
2		(2), 0	3,1	
3		4,1	(3), 1	
4	1,0	(4), 1		
5			6, ∽	(5), 0
6		7,0	(6), ∽	
7		(7), 0		

Таблица 3.13

x_1x_2; z	00	10	11	01
1	(1), 0	2,0	—	5,0
2	—	(2), 0	3,1	—
3	—	4,1	(3), 1	—
4	1,0	(4), 1	—	—
5	—	—	6, ∽	(5), 0
6	—	7,0	(6), ∽	—
7	1,0	(7), 0	—	—

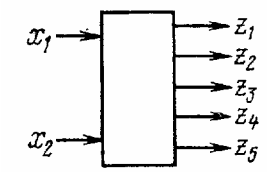


Рис. 3.10

Пример 3.5. Необходимо построить первоначальную таблицу переходов для реверсивного счетчика на пять импульсов (рис. 3.10). Реверсивный счетчик имеет два входа *x* и *Хг* и пять выходов Zi—25, единичное значение каждого из которых соответствует положению реверсивного счетчика, т. е. числу сосчитанных импульсов. При поступлении сигнала на вход *xi в* счетчике добавляется единица, а на вход *xs*—вычитается единица. При положении счетчика в нулевом (исходном^ состоянии ни на одном из пяти его выходов сигнала нет. При одновременном появлении сигналов на входах *Xi* и л-z счетчик, очевидно, своего состояния не меняет.

Решение. Вначале составим первоначальную таблицу переходов только для одной последовательности входных сигналов, когда на вход *Xi* последовательно поступают пять импульсов (табл. 3.14). Заметим, чта эта последовательность аналогична последовательности, представленной в таблице включений, изображенной на рис. 3.6 (за исключением того, что в нашем случае указаны внутренние состояния и имеется не один, а пять выходов). При этом будем сопоставлять состояния выходов с внутренними состояниями автомата.

Теперь рассмотрим другие возможные последовательности значений входных сигналов. Пусть после поступления сигнала на вход *x* поступит сигнал на вход *Хч.* Тогда, очевидно, счетчик должен перейти в предыдущее состояние (табл. 3.15). Например, после двух импульсов, поступивших на вход *x^,* пусть появится импульс на входе *хч.* Тогда автомат (т. е. реверсивный счетчик) должен перейти из устойчивого состояния (5) в неустойчивое состояние 13, а затем в устойчивое состояние (13) во вновь введенном внутреннем состоянии я». С внутренним состоянием *к»* сопоставлено состояние выхода Zi==l, указывающее на то, что в счетчике после вычитания единицы осталась еще одна единица. После исчезновения сигнала на входе *x-г* автомат должен перейти в неустойчивое состояние 3 в столбце 00.

Таблица 3.14

	x_1x_2 00	10	11	01	$z_1z_2z_3z_4z_5$
1	(1)	2			0 0 0 0 0
2	3	(2)			1 0 0 0 0
3	(3)	4			1 0 0 0 0
4	5	(4)			0 1 0 0 0
5	(5)	6			0 1 0 0 0
6	7	(6)			0 0 1 0 0
7	(7)	8			0 0 1 0 0
8	9	(8)			0 0 0 1 0
9	(9)	10			0 0 0 1 0
10	11	(10)			0 0 0 0 1
11	(11)	12			0 0 0 0 1
12	1	(12)			0 0 0 0 0

Таблица 3.15

	x_1x_2 00	10	11	01	$z_1z_2z_3z_4z_5$
1	(1)	2			0 0 0 0 0
2	3	(2)			1 0 0 0 0
3	(3)	4			1 0 0 0 0
4	5	(4)			0 1 0 0 0
5	(5)	6		13	0 1 0 0 0
6	7	(6)			0 0 1 0 0
7	(7)	8			0 0 1 0 0
8	9	(8)			0 0 0 1 0
9	(9)	10			0 0 0 1 0
10	11	(10)			0 0 0 0 1
11	(11)	12			0 0 0 0 1
12	1	(12)			0 0 0 0 0
13	3			(13)	1 0 0 0 0

Легко заметить, что из этого неустойчивого состояния автомат должен перейти в устойчивое, находящееся в такой строке таблицы, с которой сопоставлено то же состояние выхода Zi=l, что и со строкой »<is. Таким устойчивым состоянием будет **(3) во** внутреннем состоянии *Кз.* Поэтому в столбце 00 и строке Xi3 указано неустойчивое состояние 3. Аналогичным образом можно определить переходы после появления сигнала на входе *Хц* и для других состояний счетчика. В результате получим табл. 3.16.

В табл. 3.16 указаны также переходы при одновременном появлении сигналов на двух входах *x* и *x^.* При этом, поскольку состояние счетчика не меняется, новые внутренние состояния не вводятся. Таким образом, табл. 3.16 уже не является первоначальной таблицей переходов, так как в ее строках имеются по два устойчивых состояния, хотя объединение внутренних состояний произведено неявно непосредственно на этапе составления таблицы переходов.

Вместо таблиц переходов можно использовать матрицы переходов [1,3]. Например, вместо таблицы переходов

(3.8)

в примере 3.1 (см. табл. 3.9) можно составить матрицу переходов

	$\varkappa_1$	$\varkappa_2$	$\varkappa_3$	$\varkappa_4$
$\varkappa_1$	[00,0]	10,0	—	—
$\varkappa_2$	—	[10,0]	11,1	—
$\varkappa_3$	—	—	[11,1]	10,1
$\varkappa_4$	00,0	—	—	[10,1]

Таблица 3.16

01	*г^г,г^-„*				
1	(1)	2	(1)	(1)	00000
2	3	(2)			10000
3	(3)	4	(3)	14	10000
4	5	(4)			01000
5	(5)	6	(5)	13	01000
6	7	(6)			00100
7	(7)	8	(7)	15	00100
8	9	(8)		.	00010
9	(9)	10	(9)	16	00010

00 10

10	11	(10)			00001
11	(11)	12	(11)	17	00001
12		(12)			00000
13	3	—	—	(13)	10000
14	1	—	—	(14)	00000
15	5	—	—	(15)	01000
lo	7	—	—	(16)	00100
17	9	—	—	(17)	00010

^Для наглядности элементы этой матрицы, соответствующие устойчивым состояниям автомата, указаны в квадратных скобках.

Как таблицы, так и матрицы переходов удобно использовать при большом числе последовательностей «вход — выход». Если же чисел последовательностей немного, а автомат является сильно недоопределенным, то в таблице или матрице переходов многие клетки остаются незаполненными. Поэтому в таких случаях данные языки являются неэффективными с точки зрения экономности записи условий работы автомата. Особенно неэффективны таблицы и матрицы переходов для описания условий работы микропрограммного автомата, где обычно имеется большое число входов, а сам автомат является сильно недоопределенным.

3.3. Задание условий работы микропрограммного автомата

Условия работы микропрограммного автомата, как и любого другого конечного автомата, могут быть заданы в виде таблицы переходов. Однако более удобной формой задания условий работы МА является язык логических схем алгоритмов (ЛСА), который в наибольшей степени учитывает особенности алгоритма функционирования автомата [31.

При использовании ЛСА предполагается, что каждый его оператор Л,, реализуется в соответствующем ОФБ,, а проверка логического условия р, осуществляется в ЛФБ,, тогда как ЛСА описывает последовательность воздействий МА на такие ФБ. При этом выполнение оператора Л, означает, что на выходе *гд* МА должен быть сигнал, включающий в работу ОФБ,-. Выполнение в ЛСА условия *p,* соответствует сигналу на выходе *z-p,* а результат проверки условия *pj* в ЛФБ, - воспринимается МА как состояние его внутреннего входа *pj*.

Условия работы неавтоматных микропрограммных автоматов с L состояниями внешних выходов могут быть описаны в виде L логических схем алгоритмов, обычно называемых частными. Рассмотрим следующий простейший пример.

Пример 3.6. Пусть задай коммутатор с n вертикалями и m горизонталями (рис. 3.11). К каждой вертикали может подключаться измерительный прибор. а к каждой горизонтали—объект, подлежащий измерению. Прибор позволяет измерять только один из параметров. Электрическое соединение его с объектом осуществляется с помощью коммутатора. При этом могут оыть два режима работы ДУ, управляющего таким коммутатором. При первом режиме после освобождения какого-либо измерительного прибора и его проверки в ДУ подается сигнал вызова о возможности подключения данного прибора к объекту. При этом возможны три случая: к объекту подключен другой измерительный прибор;

другой измерительный прибор не подключен, но данный прибор не может измерять параметр объекта;

объект свободен и прибор может измерять парамегр объекта.

При подключении измерительного прибора (что возможно лишь в третьем случае) в прибор посылается сигнал «Готово», по которому он начинает процесс измерения. В том случае, когда пет объектов, к которым можно подключать прибор, последнему посылается сигнал «Занято». По этому сигналу прибор переходит в состояние ожидания и возобновляет ПОПЫТКУ подключения через время т.

Таким образом, при первом режиме работы ДУ инициаторами вызовов являются измерительные приборы; объекты пассивны. Второй режим работы ДУ соответствует случаю, когда инициаторами вызова являются объекты, а измерительные приборы пассивны. При этом соблюдаются аналогичные первому режиму условия подключения объекта к измерительному прибору.

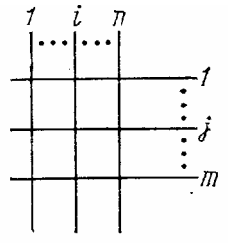


Рис. 3.11

Введем следующие операторы и логические условия:

F{ — прибавить к *i* единицу ('параметр i указывает номер входа);

^—прибавить к / единицу (параметр *j* указывает номер выхода);

Ai— включить точку соединения на пересечении 1-го входа и *!-ТО* вы

Аг—послать сигнал «Ответ» источнику вызова (прибору или объекту);

Лз—передать источнику вызова сигнал «Занято»;

Ah (точка) —оператор конца, который выполняет также следующую функцию: он приравнивает параметры *i* и ; нулю, т. е. после его выполнения »-и, *1-0;*

_ l 1, если кга;

$\overline{p_1}$- I 0, если *i>n;*

^ (1, если возможно измерение;

•$^{2—}$ I 0, если измерение невозможно;

_ f 1, если на t-ю вертикаль поступил вызов;

^{ps}~ \ 0, если на i-ю вертикаль вызов не поступил;

_ *{ 1,* если *!<гп;*

$^{P4-}$ _ I 0, если *j>m;*

_ f 1, если свободен объект, подключенный к ;'-й горизонтали;

$^{P5—}$ *[0* в противном случае;
_ J 1, если на ;-ю горизонталь поступил вызов;
$^{^6—}$ \ 0 в противном случае;
^ **(1,** если свободен прибор, подключенный к 1-й вертикали;
$^{^7—}$ \ 0 в противном случае, сп
Используя эти операторы и логические условия, составим следующие ЛСА:

а) для первого режима

$$^1 = i^3\ Fi\ pi\ fi\ P_3T^{^4}\ Pt\ T^3\ Pa\ T^4\ /'2\ T^4\ _{AI}\ A^{\wedge}\ coT^2\ ,^3\ As\ cot^2\ I^1\ ; \quad (3.9)$$

б) для второго режима

$$^ll = I^1\ FjPt\ \Gamma\text{-}Ps\ T^1\ i^{\wedge}Fi\ p\text{-},\ T^3/?,\ T^4\ ^i\ /4<o\ T^1\ i^3\ ^3\ a\ T^1 1^2. \quad (3.10)$$

Очевидно, микропрограммный автомат, реализующий условия, заданные ЛСА (3.9) и (3.1.0), должен иметь один внешний вход, одно из состояний которого. пусть г=0. сопоставим с ЛСА 2Ii, а состоян.ие г=1 — с ЛСА 21ц.

Если на внешнем входе /-=0, то выполняется первый режим работы микропрограммного автомата в соответствии с ЛСА Яг; если г=1—то второй в соответствии с ЛСА Мп.

При составлении ЛСА для сложного алгоритма функционирования возникают значительные трудности. Поэтому в таких случаях процесс работы ДУ расчленяют на этапы (части) и для каждого этапа составляют свою частную ЛСА, а затем уже определяют алгоритм выполнения таких частных ЛСА.

Иногда процесс работы ДУ приходится представлять в виде иерархии частных ЛСА. Например, ДУ работает в двух режимах, первый из которых удобно представить в виде трех этапов, а второй — двух. После выполнения первого режима ДУ переходит к выполнению второго, затем вновь первого и т. д. При этом первый этап первого режима работы ДУ совпадает с первым этапом второго режима, а на остальных этапах алгоритмы функционирования ДУ различаются.

Вначале составляем ЛСА для каждого из этапов: ЛСА 2Ii, описывающую алгоритм функционирования ДУ на первом этапе как первого, так и второго режимов работы ДУ; ЛСА *Щ* и 2tg на втором и третьем этапах первого режима и 2^ на втором этапе второго режима. Далее задаем последовательность выполнения частных ЛСА для каждого из режимов. Пусть at^StiSIat/$t^{13^{\wedge}1}$ Для первого режима и 2[„ =2^21:4 для второго режима. Затем составим общий алгоритм функционирования ДУ в виде ЛСА

$$ЭД = ^\ 9Ii\ ЭД_{„}\ (ofi = ^\ ^\ ^\ q\ ]^2\ Яд\ ^\ SIi\ ^\ cof^1. \quad (3.11)$$

Таким образом, общая ЛСА составлена за три шага: вначале были составлены частные ЛСА этапов, затем — частные ЛСА режимов и только после этого—общая ЛСА, задающая условия работы микропрограммного автомата.

Легко видеть, что в ЛСА (3.11) частная ЛСА 21, встречается дважды, так как Я, встречается как в Я;,так и в Sin. Однако нетрудно построить общую ЛСА с одним вхождением ЛСА 2I,. Для этого, как и в предыдущем примере, введем дополнительное логическое условие *г,* одно из значений которого (г=0) припишем ЛСА 2Ii, а второе (г== 1) — ЛСА 2[„.

Поскольку как ЛСА 2t„ так и ЛСА 2Г„ начинаются с ЛСА а, общая ЛСА также будет начинаться с ЛСА 2Ii. Однако после выполнения ЛСА 21, в первом режиме следует выполнять ЛСА 2^, а во втором — ЛСА S^. Поэтому по окончании 21, необходимо приписать логическое условие *г,* значение которого и укажет на необходимость выполнения одной из этих двух ЛСА, т. е. общая логическая схема должна иметь вид

$$Я = ^\ 2l^f\ a^q\ f2\ 21^2\ щ\ ^^з^ю+i. \quad \textbf{(3.12)}$$

Из ЛСА (3.12) видно, что при г=0 выполняется ЛСА SI„ а при г=l—ЛСА^п. Однако, поскольку в данном случае, в отличие от предыдущего примера, режимы чередуются, в процессе работы МА необходимо менять значение логического условия г. Так, после выполнения ЛСА 2ti значение логического условия *г* должно измениться с нуля на единицу, а после выполнения ЛСА 2I„ — с единицы на нуль. Очевидно, изменение значения *г* здесь уже не может производиться извне, а поэтому в ЛСА необходимо ввести специальные операторы, обычно обозначаемые через $F\{$, которые и будут устанавливать необходимые значения логического условия *г.*

Пусть Fi устанавливает значение *г=0, а.* Рц—значение *г*=\. Тогда окончательно общая ЛСА может быть записана в виде

$$Я^^ЭД^зэДз^^ду^2^!!"^{32}^®!^4- \quad (3.13)$$

Следует заметить, что функции операторов $F\{$ могут выполнять операторы частных ЛСА.

3.4. Составление логических схем алгоритмов

Условия работы МА не всегда легко записывать непосредственно на языке ЛСА. Оказывается, более удобно алгоритм функционирования вначале записать на так называемых вспомогательных языках, а затем от них уже перейти к ЛСА. В качестве таких вспомогательных языков рассмотрим два: язык формул переходов (ФП) и язык секвенций [3].

Алгоритм функционирования автомата, в том числе и МА, представим в виде причинно-следственных связей операторов (актов алгоритма) или ФБ, в которых эти операторы реализуются. В таких причинно-следственных связях выделим два типа операторов: операторы-предшественники, т. е. такие, которые к рассматриваемому моменту времени закончили свою работу, и операторы-последователи, т. е. такие, которые должны выполняться после завершения работы операторов-предшественников. Такие причинно-следственные связи между *k*+\ операторами Ло,..., *A.k* заданного алгоритма функционирования можно представить в виде *матричной схемы алгоритма* (МСА) [3]

ЭД==Л,||4||, (3.14)

где каждая строка сопоставлена с оператором-предшественником *Ai, i==0,l,... ,k*—1, а столбец—с оператором-последователем Л;, /=1,... , *k.* Элемент а,, МСА представляет собой логическую функцию от логических переменных (логических условий) pi, ..., p,,, т. е.

^{v}•*iS=f(Pь* •-, *Pn),* (3.15)

определяющих условия выполнения оператора *A,* после окончания работы оператора *Ai.*

Заметим, что МСА и матрица переходов автомата (3.8) имеют много общего. Так, если операторы сопоставим с внутренними состояниями автомата, а вместо функции а,, выпишем те наборы значений всех логических условий pi, ..., *pn* (т. е. всех входных переменных), на которых аг,==1, то МСА превратится в матрицу переходов.

Условие выполнения оператора *A]* после любого оператора МСА может быть представлено в виде дизъюнкции отмеченных функций ciij столбца *A,* МСА, т. е. в виде

(•==0r ..... *k*—1 V "„л,ь^, (3.16)

называемой секвенцией.

Для простоты в секвенции функции *aijAi,* где а^-ssO, не выписываются. Очевидно, если будут выписаны секвенции для каждого из *k* столбцов МСА, то полученное секвенциальное описание автомата будет задавать алгоритм его функционирования.

Так как в секвенциях не выписываются функции *a^==0,* то секвенциальное описание, очевидно, задает алгоритм функционирования автомата значительно более компактнее, чем МСА, а следовательно, и матрица, и таблица переходов.

Теперь составим дизъюнкцию отмеченных функций *щ,* одной 1-й строки. Такая дизъюнкция будет определять условия выполнения любого оператора после реализации оператора *A*\. Эту зависимость представим в виде следующего выражения:

$$A1 \rightarrow \bigvee_{j=1,..,k} aij\ Aj \qquad (3.17)$$

называемую формой переходов (ФП), или формулой Дьяченко.

В ФП, как и в секвенции, функции а<з=0 не выписываются. Таким образом, если секвенция (3.16) определяет подмножество у,-=*{Aji,...,Ajrj}, 0<:rj-^k,* операторов-предшественников из множества *M={Ao, Ai,..,Ak},* после выполнения которых при условиях, определяемых функциями (3.15), должен выполняться оператор-последователь *A,* (рис. 3.12,а), то формула Дьяченко (3.17) определяет подмножество рг= *{Aц,...* ,Агг, }, *0<ri*$^{<}$*S:k,* операторов-последователей из множества операторов *M* (рис. 3.12,6).

Легко видеть, что компактность описания алгоритма функционирования секвенциями и формулами Дьяченко одна и та же. Поэтому при задании условий работы автомата определяющим фактором при выборе одного из языков (секвенций или ФП, т. е. формул Дьяченко) служит не компактность описания, а удобство задания причинно-следственных связей между операторами. В зависимости от того, как легче сформулировать причинно-следственные связи между операторами, проектировщик может использовать для представления алгоритма функционирования автомата секвенциальное описание или

систему ФП либо смешанное описание, сочетающее в себе секвенции и ФП. Последнее является следствием того что на отдельных этапах функционирования ДУ легче определить операторы-предшественники для оператора-последователя, а на других наоборот—один оператор-предшественник с соответствующими операторами-последователями. В том случае, когда составляется смешанное описание, целесоооразно затем перейти к одному из языков, причем, как будет следовать из дальнейшего, наиболее предпочтительным оказывается язык ФП.

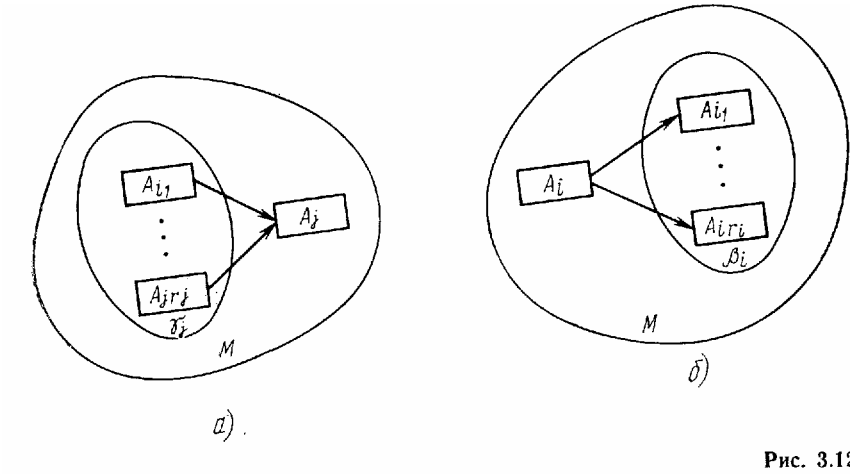


Рис. 3.1

В теории автоматов разработан достаточно эффективный метод перехода от секвенций к ФП, изложенный в [3]. Однако мы рассмотрим более громоздкий, но значительно более понятный метод. Громоздкость метода вытекает из того, что необходимо строить МСА которая, как указывалось выше, не обеспечивает компактности описания алгоритма функционирования МА.

Пример 37 Пусть проектировщиком составлено смешанное описание алгоритм •^нкционйрованияМА, включающее две формулы переходов:

$$A_0 \rightarrow p_1 A_1 \vee \bar{p}_1 A_2 ; \quad A_2 \rightarrow p_2 A_1 \vee \bar{p}_2 A_2$$

и четыре секвенции:

$$\bar{p}_1 A_1 \vee p_1 \bar{p}_2 A_3 \vdash A_3 ;$$

$$p_1 A_0 \vee p_1 p_2 A_1 \vee p_2 A_2 \vee \bar{p}_1 A_3 \vdash A_1;$$

$$\bar{p}_1 A_0 \vee \bar{p}_2 A_2 \vee p_1 p_2 A_3 \vdash A_2 ;$$

$$p_1 \bar{p}_2 \vdash A_k ,$$

ляютсГ^ми ^и^ с=^ор^, -^^~^

^ ?Se STL^aT^n^M^{0}^^^

для операторов Ло и *Аъ* при этом получим

63

$$= \begin{array}{c} \\ A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \left\| \begin{array}{cccc} A_1 & A_2 & A_3 & A_k \\ p_1 & \bar{p}_1 & & \\ & & & \\ p_2 & \bar{p}_2 & & \end{array} \right\| . \qquad (3.18)$$

Рассмотрим теперь первую секвенцию (для оператора *Аз).* Так как секвенц.ия соответствует столбцу МСА, то по .ней заполним столбец *Аз* матрицы и получим

$$\mathfrak{A} = \begin{array}{c} \\ A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \left\| \begin{array}{cccc} A_1 & A_2 & A_3 & A_k \\ p_1 & \bar{p}_1 & & \\ & & \bar{p}_1 & \\ p_2 & \bar{p}_2 & & \\ & & & p_1\bar{p}_2 \end{array} \right\| . \qquad (3.19)$$

Затем заполним столбец *Ai* по второй секвенции. При этом заметим, что элементы Ctoi и 02i уже заполнены по ФП для операторов *А»* и Ла. Поэтому к тем элементам *ai,* МСА, которые были заполнены ранее, логически (дизъюнктивно) прибавляем соответствующие логические функции из рассматриваемой секвенции и при возможности упрощаем их. В результате получаем

$$\mathfrak{A} = \begin{array}{c} \\ A_0 \\ A_1 \\ A_2 \\ A_3 \end{array} \left\| \begin{array}{cccc} A_1 & A_2 & A_3 & A_k \\ p_1 \vee p_1 = p_1 & \bar{p}_1 & & \\ p_1 p_2 & & \bar{p}_1 & \\ p_2 \vee p_2 = p_2 & \bar{p}_2 & & \\ \bar{p}_1 & & p_1\bar{p}_2 & \end{array} \right\| . \qquad (3.20)$$

Аналогично заполняем второй и четвертый столбец МСА по третьей и четвертой секвенциям соответственно.

Окончательно получаем

$$\mathfrak{A} = \begin{array}{c|cccc} & A_1 & A_2 & A_3 & A_k \\ A_0 & p_1 & \overline{p_1} & & \\ A_1 & p_1p_2 & & \overline{p_1} & p_1\overline{p_2} \\ A_2 & p_2 & \overline{p_2} & & \\ A_3 & \overline{p_1} & p_1p_2 & p_1\overline{p_2} & \end{array}. \qquad (3.21)$$

Из МСА (3.21) легко получить систему ФП:

$$\begin{cases} A_0 \rightarrow p_1 A_1 \vee \overline{p_1} A_2; \\ A_1 \rightarrow p_1 p_2 A_1 \vee \overline{p_1} A_3 \vee p_1\overline{p_2} A_k; \\ A_2 \rightarrow p_2 A_1 \vee \overline{p_2} A_2; \\ A_3 \rightarrow \overline{p_1} A_1 \vee p_1 p_2 A_2 \vee p_1 \overline{p_2} A_3. \end{cases}$$

Как известно [3], МСА и ФП обладают двумя следующими свойствами:

$$1) \bigvee_{j=1,\dots k} \alpha_{ij} \equiv 1; \quad 2) \ \alpha_{ij}\alpha_{il} \equiv 0, \quad j \neq l.$$

iicpouc ^iiiiiво указывает на полноту алгоритма, а второе—на его непротиворечивость. При этом формально считается, что алгоритм функционирования составлен правильна c^in »n ^.^ ,. ..^ ?^^ивооечив Таким образом, для того чтобы проверить, правильно ли со^^ функционирования. Достаточно прорить выполнение указанных выше двух свойств для всех сцюк

К с^жа^ник^секвенции этими свойствами не обладают, поэтому по нТмп^Тоту и непротиворечивость алгоритма проверить значительно сложнее. Вот почему желательно от секвенции перейти к формулам Дьяченко или к МСА.

Следует заметить, что полнота алгоритма функционирования таким образом, проверяется лишь с формальной точки зрения без ^-чета смыслового содержания алгоритма. Полнота алгоритма фунционирования с содержательной точки зрения условии работы проектируемого ДУ должна проверяться непосредственно проекти^П^о^ьку при использовании секвенции проверка па полноту и непротиворечивость условий работы автомата затруднительна, проектировщик может'не учесть каких-либо переходов (т. е. составить алгоритм функционирования не полно) или ошибочно включить в условия работы противоречивые переходы. Неполнота или (и) противоречивость алгоритма функционирования проектируемого ДУ проверяется при составлении МСА или **Ф11.**

Пример 3.8. Пусть проектировщик составил условия работы в виде смешанного описания, состоящего из трех ФП

$A_0 \rightarrow A_1$,

$A_2 \rightarrow \overline{p}_1 A_1 \vee p_1 A_3$;

$A_3 \rightarrow \overline{p}_1 A_1 \vee p_1 A_4$

и четырех секвенций

$p_1 p_2 A_1 \vee p_1 p_3 A_4 \vdash A_2$;

$\overline{p}_1 p_2 A_1 \vee p_1 A_2 \vee p_1 \overline{p}_3 A_1 \vdash A_3$;

$p_1 A_3 \vdash A_4$;

$p_1 A_4 \vdash A_k$.

Решение. Составим матричную схему алгоритма

$$\mathfrak{A} = \begin{array}{c|ccccc} & A_1 & A_2 & A_3 & A_4 & A_k \\ A_0 & 1 & & & & \\ A_1 & & p_1p_2 & \overline{p_1}p_2 & & \\ A_2 & \overline{p_1} & & p_1 & & \\ A_3 & \overline{p_1} & & & p_1 & \\ A_4 & & p_1p_3 & p_1\overline{p_3} & & p_1 \end{array}. \qquad (3.22)$$

Из рассмотрения (3.22) видно, что составленные условия раооты незлые, так как не выполняется первое условие для строки оператора Л,, и противоречивые, так как не выполняется второе условие для строки оператора *А,*

При выявлении неполноты и (или) противоречивости условий работа ?е составляемого алгоритма функционирования, последние должны быть скорректированы. В том случае, когда человек, формулирующий условия работы, выступает в качестве заказчика, а формализация условий работы в виде алгоритма функционирования, представляемого на одном из формальных языков (в нашем случае на языках ФП и секвенциях), с последующим проектированием ДУ осуществляется проектировщиком, в частности ЭВМ, корректировка условий работы осуществляется в процессе диалога «проектировщик—заказчик». В таком диалоге активным является проектировщик.

Продолжим рассмотрение нашего примера.

Устраняем вначале неполноту условий работы. Для этого проектировщик спрашивает заказчика: «Что следует за оператором *Ai,* если логическое условие p2==0?». Если от заказчика ответа не последует, то это означает, что он не знает условий работы проектируемого ДУ. В противном случае он должен ответить (например, что после оператора *Ai* следует выполнить оператор *Ai.).* Следовательно, проектировщик в МСА (3.22) добавят элемент aii=p2.

Теперь устраняем противоречие. Здесь может быть несколько вариантов вопросов. Однако наиболее естественным вопросом является следующий: «Какой оператор должен выполняться после оператора *Ai,,* если ЛУ /?i=0?». Этот вопрос возникает вследствие того, что ни в одном из элементов нет pi, из-за чего и появилось противоречие. При pi=0 ответов может быть несколько:

1. После оператора Л* следует оператор *Ah.*

2. После оператора *А^* следуют операторы *А-г* и *Аз.*

3. После оператора *Ai,* следует оператор *А^.*

4. После оператора *Ai,* следует оператор Л.ч.

Если заказчик дает первый или второй варианты ответа, в результате коррекции строки *At* для нее будут выполняться первое и второе условия.

Скорректированная МСА будет описывать полный и непротиворечивый алгоритм функционирования проектируемого ДУ. В том случае, когда заказчик даст ответ по третьему или четвертому варианту, диалог между проектировщиком 11 заказчиком несколько удлиняется. Рассмотрим продолжение диалога при ответе по третьему варианту.

В данном случае по-прежнему не выполняется второе условие, так как №4зи4^=p1pзp1=p1pз=г^0. Поэтому естествен вопрос проектировщика: «Не будет ли выполняться оператор *Аз* после оператора *Ai.,* если pi=0?». Если ответ будет отрицательным, то заказчик сам не знает условий работы проектируемого ДУ и поэтому процесс проектирования прекращается. Проектировщик предлагает заказчику уточнить условия работы автомата. Если ответ будет положительным, то строка *Ад* корректируется, после чего для нее будут выполняться первое и второе условия и диалог между проектировщиком и заказчиком оканчивается, а заказчику выдается для контроля составленная МСА (ФП или секвенпии). Условия работы могут быть выданы и на неформализованном (русском) языке. В нашем случае для первого ответа скорректированная МСА будет иметь, следующий вид:

$$\mathfrak{A} = \begin{array}{c|cccccc} & A_1 & A_2 & A_3 & A_4 & A_h \\ A_0 & 1 & & & & \\ A_1 & & p_1p_2 & \overline{p_1}p_2 & \overline{p_2} & \\ A_2 & \overline{p_1} & & p_1 & & \\ A_3 & \overline{p_1} & & & p_1 & \\ A_4 & & p_1p_3 & p_1\overline{p_3} & & \overline{p_1} \end{array} . \qquad (3.23)$$

Диалог между проектировщиком и заказчиком возможен не только при корректировке условий работы, но и в процессе состав ления секвенций и ФП.

После получения скорректированной МСА легко построить систему ФП.

Пример 3.9. Необходимо по МСА (3.23) построить систему ФП.

Решение. Функции *ац* отмечаем операторами Л, и получаем следующую систему ФП:

f Д-Ai;
| ^i—PiP2-4aV ~*P1PгА»*<~*Рч*^\

\ Aa—PiAiV/'l^a;

А3 — Pi Ai V *Pi* Ai;_
[Ai—PlPs^V^lPs^VPi^ft-

Рассмотрим теперь процесс построения ЛСА по ФП [31. Пусть задана система ФП, полученная в примере 3.7. От ФП к ЛСА удобно переходить не непосредственно, а через так называемые скобочные ФП (иначе скобочные формулы Дьяченко), которые получаются из ФП следующим образом. В каждой ФП обязательно имеется такое одно ЛУ *pj,* которое входит во все ее члены. При этом говорят, что ФП приведена по ЛУ *pj.* Если таких ЛУ найдется несколько, то выбирают одно из них. В ФП выносят за скобки ЛУ без отрицания, т. е. *pj,* и с отрицанием, т. е. *pj.* Например, в ФП оператора А) получим *Ai~*[7] *(ргА^р^А^Ур^Аз.* Это и есть скобочная ФП. Для нашего примера получим следующую систему скобочных ФП:

Ло— *p^A^p^A.:,*
^i~[7] *Pl(pг*[A]*l\/P,*[A]*ь}\/~Pl*[A]*s,*

A^-7p^A-^\j~p^A^,

At ~~[7] *Pi (Pa* ^2 V й^з) V *Pi* [AI]-

Из сравнения этой системы с исходной видно, что ФП для операторов *Аа* и А] остались без изменения. Если после вынесения за скобки переменной *pj* в каждом из заключенных в скобки выражений формулы переходов содержатся два и более ЛУ, то процедуру вынесения за скобки следует продолжить применительно к этим выражениям ФП. Например, пусть получена следующая ФП:

Pi (Рг Рз [AI] V *Pt Рз А-г* V *Рз Ал) \/Pi* [АЯ]-

В части, заключенной в скобки, вновь находим переменную, которую можно вынести за скобки. Таким образом, получаем

Pi (Рз (р2 А V^а А) *\/~Рз А,)\/pi* [A]*а•*

От скобочных ФП легко перейти к ЛСА. Вначале выписываем оператор *Ао.* После него выписываем ЛУ (со стрелкой), вынесенное за первую скобку. В нашем случае это *pi,* т. е. получаем AoPif. Затем выписываем оператор, выполняемый после *Ац* при *p\=\,* т. е:
получаем *Аорi}Ai.* После выписки оператора Л, (в нашем случае *А^* обращаемся к ФП оператора Л,, т. е. к ФП оператора Л,, н проделываем ту же процедуру. В результате получаем

[A]*» Pi^i* Л^аМг (3.24)

3* *Ц.*

Как видно из скобочной ФП оператора А], после *pi* стоит не оператор, а начинается скобка. Поэтому за *pi* выписывается *p^,* а затем уже оператор А]. Однако в том случае, когда оператор *А,:* уже выписан, его не повторяют в ЛСА, а вводят тождественно ложное ЛУ со стрелкой, оканчивающейся перед имеющимся оператором *А{.* В нашем случае оператор *А* выписан ранее. Поэтому вместо (3.24) получаем

^{A}» *PiH*^{l}4 /"if/c^M^{1}.

В том случае, когда таким образом прерывается выписывание нового оператора, необходимо обратиться к первому слева логическому условию в выписанной части ЛСА, у которого не отмечена номером стрелка. В нашем случае это логическое условие *p\,* стоящее после оператора *Ац.* Стрелку у *pi* пронумеруем и продолжим выписывание скобочной ФП оператора *Ао* (так как *pi* выполняется после *Ао)* при условии, что *pi=0.* В результате получаем

^оЛ^Лй^M^{1}!^{2}^.

Продолжая процесс построения ЛСА таким же образом, получаем последовательно:

$A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow p_2\uparrow\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow\omega\uparrow^1;$

$A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow^3 p_2\uparrow\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow\omega\uparrow^1\downarrow^3 A_3;$

$A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow^3 p_2\uparrow\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow\omega\uparrow^1\downarrow^3 A_3\ p_1\uparrow p_2\uparrow\omega\uparrow^2;$

$A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow^3 p_2\uparrow^4\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow\omega\uparrow^1\downarrow^3 A_3\ p_1\uparrow p_2\uparrow\omega\uparrow^2\downarrow^4 A_k;$

$A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow^3 p_2\uparrow^4\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow^2\omega\uparrow^1\downarrow^3 A_3\ p_1\uparrow^1 p_2\uparrow\omega\uparrow^2\downarrow^4 A_k$

и окончательно

$2\mathfrak{l} = A_0\ p_1\uparrow^2\downarrow^1 A_1\ p_1\uparrow^3 p_2\uparrow^4\omega\uparrow^1\downarrow^2 A_2\ p_2\uparrow^2\omega\uparrow^1\downarrow^3 A_3\ p_1\uparrow^1 p_2\uparrow^3\omega\uparrow^2\downarrow^4 A_k.$

Пример 3.10. Пусть задана система ФП, полученная в примере 2.9. Необходимо построить ЛСА.

Решение. По заданным ФП получаем скобочные ФП:

$$\begin{cases} A_0 \rightarrow A_1; \\ A_1 \rightarrow p_2 (p_1 A_2 \vee \overline{p}_1 A_3) \vee \overline{p}_2 A_4; \\ A_2 \rightarrow p_1 A_3 \vee \overline{p}_1 A_1; \\ A_3 \rightarrow p_1 A_4 \vee \overline{p}_1 A_1; \\ A_4 \rightarrow p_1 (p_3 A_2 \vee \overline{p}_3 A_3) \vee \overline{p}_1 A_h. \end{cases}$$

Затем строим ЛСА:

$A_0\ A_1\ p_2\uparrow p_1\uparrow\downarrow^1 A_2\ p_1\uparrow A_3\ p_1\uparrow A_4\ p_1\uparrow p_3\uparrow\omega\uparrow^1;$

$A_0\ A_1\ p_2\uparrow^2 p_1\uparrow\downarrow^1 A_2\ p_1\uparrow A_3\ p_1\uparrow\downarrow^2 A_4\ p_1\uparrow p_3\uparrow\omega\uparrow^1;$

$A_0\ A_1\ p_2\uparrow^2 p_1\uparrow^3\downarrow^1 A_2\ p_1\uparrow\downarrow^3 A_3\ p_1\uparrow\downarrow^2 A_4\ p_1\uparrow p_3\uparrow\omega\uparrow^1;$

$A_0\downarrow^4 A_1\ p_2\uparrow^2 p_1\uparrow^3\downarrow^1 A_2\ p_1\uparrow^4\downarrow^3 A_3\ p_1\uparrow\downarrow^2 A_4\ p_1\uparrow p_3\uparrow\omega\uparrow^1;$

$A_0\downarrow^4 A_1\ p_2\uparrow^2 p_1\uparrow^3\downarrow^1 A_2\ p_1\uparrow^4\downarrow^3 A_3\ p_1\uparrow^4\downarrow^2 A_4\ p_1\uparrow p_3\uparrow\omega\uparrow^1;$

68

<4о i^{4} *Aip,* J2 ^ ^ ^ *A*^*pt* T^{4} *i*^{3} *As Pi* t^{4}1^{2} *A, Pi* T^{5} *Ps* T **<о** T^{1}1^{6} *Ak.*

И, наконец, получаем ЛСА

SI = Ло i^{4} ^ Ра t^{2} Pi T^{3}1^{1} ^2 Pi T^{4} i^{3} Лз *Pi* t^{4} ^ ^4 Pi T^{6} *Рз* T^{3} ш T^{1} *f*^{s} *Ah.*

3.5. Объединение логических схем алгоритмов

Как было отмечено в разд. 3.3, для удобства составления ЛСА весь алгоритм функционирования дискретного устройства представляют в виде алгоритма выполнения частных ЛСА, описывающих отдельные этапы работы ДУ. Однако в каждую из таких частных ЛСА могут входить одни и те же операторы и логические условия. Поэтому, когда вместо символов частных ЛСА будут подставлены соответствующие выражения из операторов и логических условий, общая ЛСА, описывающая весь алгоритм функционирования ДУ, может оказаться далеко не минимальной. При этом в такой ЛСА могут повторяться не только логические условия, но и операторы.

Для минимизации ЛСА с повторяющимися операторами и логическими условиями может быть использован так называемый метод объединения логических схем алгоритмов. Рассмотрим пример объединения двух ЛСА. Общий случай одновременного объединения нескольких ЛСА описан в [3].

Пример 3.11. Пусть необходимо объединить две частные ЛСА, полученные в примере 3.6 [см. (3.9) и (3.10)]. При этом общая ЛСА имеет вид

-ai^itiatJcuT'^ain.i" (з.25)

Решение. На основе логических схем (3.9) и (3.10) составим матричные схе,мы алгооитма:

$$
\text{I} = \begin{array}{c|cccccc}
 & A_1 & A_2\ A_3 & F_i & F_j & A_k \\
A_0 & & & 1 & & \\
A_1 & & 1 & & & \\
A_2 & & & 1 & & \\
A_3 & & & 1 & & \\
F_i & & & p_1\bar{p}_3 & p_1p_3 & \bar{p}_1 \\
F_j & p_4p_5p_2 & \bar{p}_4 & & p_4\bar{p}_5 \vee p_4p_5\bar{p}_2 &
\end{array};
$$

$$
\mathfrak{A}_{\text{II}} = \begin{array}{c|cccccc}
 & A_1 & A_2\ A_3 & F_i & F_j & A_k \\
A_0 & & & & 1 & \\
A_1 & & 1 & & & \\
A_2 & & & & 1 & \\
A_3 & & & & 1 & \\
F_i & p_1p_7 & \bar{p}_1 & p_1\bar{p}_7 & & \\
F_j & & & p_4p_6 \vee p_4\bar{p}_6 = p_4 & \bar{p}_4 &
\end{array}.
$$

Если в (3.25) подставить ЛСА (3.9) и (3.10), то получим ЛСА с 22 членами, включая *Ау* и не считая тождественно ложных логических условий <в,в которой каждый из операторов встречается дважды. Объединим ЛСА (3.9) я (3.10) так, чтобы в объединенной логической системе алгоритма каждый из операторов был бы выполнен только 1 раз.

Процесс построения объединенной ЛСА, в которой операторы не повторяются, состоит в следующем.

По матричным схемам алгоритма двух частных логических схем алгоритма составляется МСА объединенной ЛСА, каждый элемент которой

$$\alpha_{ij} = \bigvee_{l=1,2} \alpha_{ij}^l \beta_i^l,$$

где P^i — определяющая функция, равная функции от логических условий p'i=/(/-i,... ,rm), принимающей единичное значение на наборе значений переменных *г\,...,Гт,* который сопоставлен с ЛСА St. Если оператор *А,* не входит в ЛСА 21,, то в функцию р^ соответствующий набор значений *г\,..., Гт* войдет в качестве условного.

В нашем случае для всех операторов логической схемы алгоритма Яисх *Уг=г,* а для всех операторов ^в^\=*г,* так как в 2Ii и 21,, входят все операторы.

Таким образом, получаем следующую объединенную матричную схему алгоритма:

$$
\begin{array}{c|cccccc}
 & A_1 & A_2 & A_3\ A_4 & F_i & F_j & A_k \\
A_0 & & & & r & \bar{r} & \\
A_1 & & r \vee \bar{r} = 1 & & & & \\
A_2 & & & & r & \bar{r} & \\
A_3 & & & & r & \bar{r} & \\
F_i & p_1\,p_7\,\bar{r} & & \bar{p}_1\,\bar{r} & p_1\,\bar{p}_3\,r \vee p_1\,\bar{p}_7\,\bar{r} & p_1\,p_3\,r & \bar{p}_1\,r \\
F_j & p_4\,p_5\,p_2\,r & & \bar{p}_4\,r & p_4\,\bar{r} & p_4\,\bar{p}_5\,r \vee p_4\,p_5\,\bar{p}_2\,r & \bar{p}_4\,\bar{r}
\end{array}.
$$

От матричной схемы алгоритма 21 перейдем к системе формул перехода, после чего преобразуем последнюю в систему скобочных формул перехода, в которой объединены общие выражения:

$$
\begin{cases}
A_0 \to r\,F_i \vee r\,F_j = \downarrow^1 r\,F_i \vee \bar{r}\,F_j; \\
A_1 \to A_2; \\
A_2 \to r\,F_i \vee \bar{r}\,F_j = \omega\uparrow^1; \\
A_3 \to F_i; \\
F_i \to p_1\,p_7\,\bar{r}\,A_1 \vee \bar{p}_1\,\bar{r}\,A_3 \vee p_1\,\bar{p}_3\,r\,F_i \vee p_1\,\bar{p}_7\,\bar{r}\,F_i \vee p_1\,p_3\,r\,F_j \vee \\
\quad \vee \bar{p}_1\,r\,A_k = p_1\,[r\,(p_3\,\bar{F}_j \vee \bar{p}_3\,F_i) \vee \bar{r}\,(p_7\,A_1 \vee \bar{p}_7\,F_i)] \vee \\
\quad \vee \bar{p}_1\,(r\,A_k \vee \bar{r}\,A_3); \\
E_j \to p_4\,p_5\,p_2\,r A_1 \vee \bar{p}_4\,r A_3 \vee p_4\,\bar{r}\,F_i \vee p_4\,\bar{p}_5\,r F_j \vee p_4\,p_5\,\bar{p}_2\,r\,F_j \vee \\
\quad \bar{p}_4\,\bar{r}\,A_k = p_4\,\{r\,[p_5\,(p_2\,A_1 \vee \bar{p}_2\,F_j) \vee \bar{p}_5\,F_j] \vee \bar{r}\,F_i\} \vee \\
\vee \bar{p}_4\,(r A_3 \vee \bar{r}\,A_k).
\end{cases}
$$

Теперь нетрудно построить объединенную ЛСА

$$\mathfrak{A} = \downarrow^{1} r\uparrow^{2}\downarrow^{7} F_i\, p_1 \uparrow^{3} r \uparrow^{5} p_3\uparrow^{7}\downarrow^{2} F_j\, p_4\uparrow^{8} r\uparrow^{7} p_5\uparrow^{2} p_2\uparrow^{2}\downarrow^{6} A_1\, A_2\, \omega\uparrow^{1}\downarrow^{3} r\uparrow^{9} \omega\uparrow^{4}\downarrow^{5} p_7$$
$$\uparrow^{7}\omega\uparrow^{6}\downarrow^{8} r\uparrow^{4}\downarrow^{9} A_3\, \omega\uparrow^{7}\downarrow^{4}., \qquad (3.26)$$

в которой вместе с оператором Л о имеется 18 членов без учета тождественно-ложных ЛУ.

Контрольные вопросы

1. Какими параметрами характеризуется конечный автомат?
2. Назовите разновидности конечных автоматов.
3. В чем состоит принципиальное отличие начального языка описания условий работы конечного автомата от стандартного?
4. Укажите достоинства и недостатки языка таблиц переходов по сравнению с языком таблиц включении.
5. Почему микропрограммный автомат удобнее задавать на языке ЛСА?

Глава 4.

МЕТОДЫ АБСТРАКТНОГО СИНТЕЗА АВТОМАТОВ

4.1. Минимизация числа внутренних состояний автомата

Как отмечалось в разд. 3.2, условиями работы автомата называется множество пар последовательностей (в том числе и бесконечных) состояний его входа и выхода (иначе множество последовательностей состояний «вход— выход»). Будем говорить, что автомат реализует заданные условия работы (т. е. является реализующим), если при поступлении на его вход последовательности на выходе автомата выдается такая последовательность, которая не противоречит выходной последовательности, заданной условиями работы автомата. Будем также говорить, что автомат не реализует заданных условий работы (т. е. является нереализующим), если найдется хотя бы одна последовательность, при поступлении которой на вход автомата на выходе его выдается последовательность, противоречивая выходной последовательности, заданной условиями ра боты автомата.

Два автомата, реализующие одни и те же заданные условия работы, называются *эквивалентными.* Очевидно, у двух эквивалентных недоопределенных автоматов выходные последовательности, соответствующие одной и той же входной последовательности, могут отличаться одна от другой лишь теми состояниями выхода, которые соответствуют безразличным его состояниям в выходной последовательности, заданной условиями работы для той же последовательности входов.

Одни и те же условия работы в ряде случаев реализуются несколькими эквивалентными автоматами, которые могут иметь различное число внутренних состояний. Задача минимизации состоит в выявлении среди этих эквивалентных автоматов такого, который имеет минимальное число внутренних состояний.

Существующие методы минимизации числа внутренних состояний автомата позволяют получить минимальный автомат в том случае, когда задается полный автомат. Регулярного алгоритмизируемого метода минимизации числа внутренних состояний нсдоопредеденных автоматов нет, поэтому получение минимального недоопределенного автомата связано с трудностями. Простой же перебор всех возможных решений с целью выбора из них минимального автомата становится практически невозможным даже при относительно небольшом числе внутренних состояний автоматов.

В настоящее время в области минимизации числа внутренних состояний недоопределенных автоматов наметились следующие две тенденции:

разработка приближенных, но алгоритмизируемых методов, которые не гарантируют построения минимальных автоматов, но позволяют запрограммировать задачи минимизации на ЭВМ;

разработка методов минимизации внутренних состояний отдельных (частных) классов недоопределенных автоматов, для которых они дают возможность построить минимальный автомат.

Среди методов минимизации числа внутренних состояний целесообразно выделить две следующие группы. Для методов первой группы характерно то, что вначале берется автомат с одним внутренним состоянием, а затем их число увеличивается до тех пор, пока автомат не станет реализующим. Полученное при этом число внутренних состояний будет минимальным. Во второй группе методов, наоборот, берется

заведомо реализующий автомат с каким-то числом внутренних состояний, которое затем сокращается (минимизируется) до такого *Smin,* что при *Smin*—1 автомат становится нереализующим.

Существуют и промежуточные методы, при которых вначале берется автомат не с одним, а с несколькими внутренними состояниями, а затем, как и при методах первой группы, число их доводится до такого, при котором автомат становится реализующим.

Методы минимизации первой группы применяются при заданий автомата с использованием начальных языков (таблицы включений, ЛСА), а второй — с использованием стандартных языков (таблицы и матрицы переходов).

Методы построения таблицы включений с минимальным числом внутренних состояний достаточно просты. Поэтому ограничимся рассмотрением только отдельных примеров. Подробно эти методы изложены в [16].

При построении микропрограммного автомата с минимальным числом внутренних состояний используется метод первой группы, состоящий в формировании минимального числа микрокоманд. Этот метод будет рассмотрен в следующем разделе.

Один из методов второй группы применительно к таблицам переходов достаточно подробно изложен в [I], а также в [3]. Поэтому здесь также ограничимся рассмотрением отдельных примеров.

Пример 4.1. Пусть задана первоначальная таблица включений, полученная е .примере 3..1 (см. рис. 3.'8,г). Необходимо по .ней построить таблицу включений реализуемого авгомата.

Решение. Как видно из рис. 3.8,г, в тактах *2* и *4* имеется одно и то же состояние входа (входной сигнал JCi=l, входной сигнал -<2="_т. е. отсутств> ет^, а состояния выхода различны: в такте 2 выходной сигнал г-и, а в такте* »1. Следовательно, необходимо вводить внутренние состояния автомата, т е. элементы памяти Э,П, сигналы на входе которых (т. е. ма выходе логического преобразователя ЛП) обозначим через У), ... *Y,,* а на выходе *(^* е. входе ЛП) -через (/„ .... у.. При переводе нереализуемой первоначальной таблиды включений в реализуемую вначале делается попытка построить ее с использованием одного элемента памяти. В данном случае этот ЭП можно включить между совпадающими тактами *2* и *4,* в которых при одном и том же состоянии входа имеются различные состояния выхода), т. е. только в такте *и.* "ыкло_ чить же его можно в такте *4* (рис. 4..1,а). Поскольку SHi имеет задержку т (см. рис. 3.2), сигнал на его выходе, а значит на входе ЛП (см. J.I), появится через время т, а исчезнет после снятия сигнала на входе aiii tr. е. на выходе ЛП) также через время т.

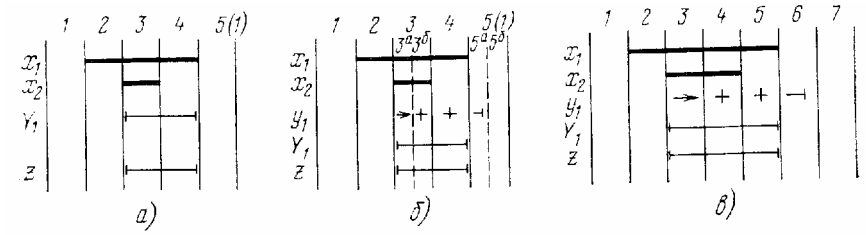


Рис. 4.1

.Каждый из тактов *3* я 5 *(1)* первоначальной таблицы включении делятся на две части (рис. 4.1,6). Состояния ЭП, при которых с его выхода снимается сигнал у=1, в таблице включений показаны знаком «+». Кроме того, для наглядности такт, в котором на вход ЭП подан сигнал Уг=1, а на его выходе еще нет сигнала *уг=0* (т. е. неустойчивое состояние ЭП), в строке *у{* таблицы отмечается стрелкой. Такт, при котором Уг=0, а у,=1, часто вместо знака «+» отмечается знаком « ч »•

Таблица включений с указанными обозначениями, а также с перенумерованными тактами изображена на рис. 4.1,е. Такты *3 6,* где ЭП находится в неустойчивом состоянии, являются неустойчивыми. Как видно, эта таблица задает реализующий автомат.

Пример 4.2. Пусть условия работы автомата заданы первоначальной таб.лицей включения, изображенной на рис. 3.6. Необходимо привести эти условия :в реализуемые.

Решение. Введем вначале один ЭП, и попытаемся включить его в такте *9,* который находится между совпадающими тактами *8* и *10* (рис. 4.2,я). Из рис. 4.2,а видно, что этого сделать нельзя, так как сразу же возникают новые совпадающие такты (такт *9а* является совпадающим с тактами *1,* rf, о и 7). Попытка включить ЭП[в предыдущих тактах вплоть до такта *3* также при. •зодит к возникновению новых совпадающих тактов. Оказывается, что ЭП можно включить только в такте *2* (рис. 4.2,6). Однако в данном случае такты 8 я *10* по-прежнему являются совпадающими. Кроме того, появляются новые .совпадающие такты *1а* и *9.* Это означает, что одного элемента памяти недостаточно. Попытаемся ввести еще один элемент памяти — ЭПз. Рассуждая анаявгичным образом, получаем вновь неудовлетворительное решение—по-прежнему имеются совпадающие такты '(рас. 4J2,е). Теперь, прежде чем ввести третий ЭП, попытаемся эффективно использовать четвертую комбинацию двух ЭП—;/1=0, й=1 (рис. 4.2,е).

Последовательно вводя новые ЭП и используя при этом все 2" возможные комбинации состояний включенных элементов памятя *п,* можно добиться перевода нереализуемых условий в реализуемые. В нашем случае таблица включении становится реализуемой при четырех ЭП (рис. 4.2,<Э).

Пример 4.3. Пусть условия работы автомата заданы двумя первоначальными таблицами включений, полученными в примере 3.2 (см. рис. 3.8 *г* и 39г). Необходимо перевести эти условия в реализуемые.

Решение. Из анализа таблиц включений находим, что такт *4* первой таблицы включений (см. рис. 3.8,г) совпадает с тактом *2* этой же таблицы *к* с тактом *4* второй таблицы включений (см. рис. 3.9,г). Поэтому вначале включаем dlli в первой таблице. Однако, поскольку во второй таблице в такте *3* имеется то же состояние входа, что в такте *3* первой таблицы, этот элемент должен быть включен и во второй таблице (рис. 4.3,<г и б). При этом на рис. 4.3,6 такты во второй таблице перенумерованы, так как они принадлежат второй последовательности, задающей тот же автомат. Введением одного ЭП не удается перевести условия работы в реализуемые - остаются совпадающие такты *4* и 7. Поэтому необходимо ввести еще один ЭП. Он должен быть включен во вторую таблицу так, как показано на рис. 4.d,e.

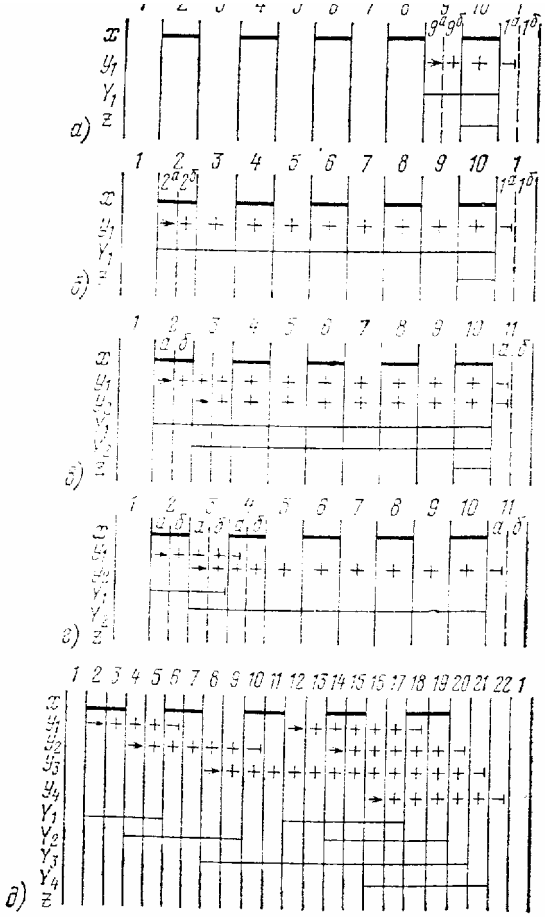


Рис. 4.2

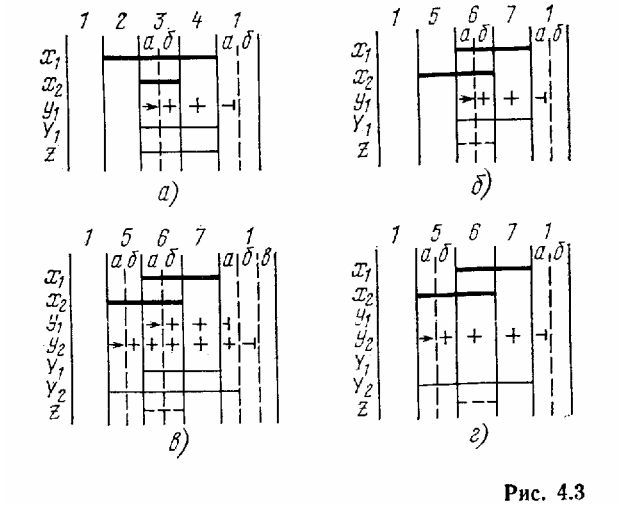


Рис. 4.3

Следует заметить, что во многих случаях существует множество вариантов введения ЭП. Так, в данном примере во второй таблице можно было бы и не включать ЭП1. При этом реализуемые условия проектируемого автомата задаются двумя таблицами включений, изображенными на рис. 4.3,д и 4.3,г.

Для таблицы переходов минимизация числа внутренних состояний автомата означает сокращение числа ее строк, т. е. ее сжатие. При этом будем считать, что перед выполнением этого этапа каким-либо способом была построена так называемая первоначальная таблица переходов.

Как отмечалось в разд. 3.2, особенностью первоначальной таблицы переходов является то, что в каждой ее строке имеется не более чем одно устойчивое состояние. Первоначальной таблицей переходов является, например, табл. 4.1. Из нее видно, что при каждом изменении состояния входа автомат меняет свое внутреннее состояние.

Таким образом, первоначальная таблица переходов описывает работу автомата, в котором каждое устойчивое состояние находиться в различных внутренних состояниях. Нетрудно понять, что такой автомат всегда является реализующим, хотя, быть может, и с избыточным числом внутренних состояний. Поэтому удобно сначала составлять первоначальную таблицу, гарантирующую задание реализующего

автомата.

Метод сжатия таблицы переходов асинхронного автомата основан на выявлении эквивалентных и псевдоэквивалентных состояний, а также выявлении и объединении совместных внутренних состояний автомата (строк таблицы переходов) [1,3]. *Эквивалентными состояниями* асинхронного автомата называются такие два устойчивые состояния, которые удовлетворяют следующим условиям:

1) им соответствует одно и то же состояние входа автомата (т. е. они находятся в одном и том же столбце таблицы);

2) им соответствует одно и то же состояние выхода автомата;

3) любой последовательности состояний входа автомата соответствует одна и та же последовательность состояний его выхода независимо от того, какое из этих рассматриваемых устойчивых состояний автомата взято за исходное.

Таблица 4.1

Pi	**P2**	**Pз**	**Pi**
(1),0	7	2	6
1	7	(2).0	3
1	4	5	(3), 1
	(4),0	2	3
1	4	(5).0	3
1	4	5	(6).0
1	(7).0	5	8
1	4	5	(8), 1

Таблица 4.2

Pi	**Pa**	**Pэ**	**P4**
(I),0	4	2	6
1	4	(2),0	5
1	4	2	(3). 1
1	(4),0	2	3
1	4	2	(6),0

Пусть, например, задана первоначальная таблица переходов (см. табл. 4.1). В соответствии с первым условием эквивалентными состояниями могут быть только те устойчивые состояния, которые находятся в одном столбце таблицы переходов, т. с. в нашем случае это состояния (4) и (7); (2) и (5); (3) и (6); (3) и (8); (6) и (8). Теперь из этих пар надо выбрать состояния, удовлетво ряющие второму условию, т. е. такие, которым соответствует одно и то же состояние выхода. Так как устойчивому состоянию (6) соответствует состояние выхода, противоречивое состояниям выхода, соответствующим устойчивым состояниям (3) и (8), то устойчивые состояния (3) и (6); (6) и (8) не могут быть эквивалентными. Из оставшихся пар устойчивых состояний (4) и (7); (2) к (5); (3) и (8) эквивалентными будут такие состояния, которые удовлетворяют еще и третьему условию эквивалентности.

Для того чтобы два устойчивых состояния удовлетворяли третьему условию эквивалентности, в одном и том же столбце соответствующих им строк должны стоять или одни и те же цифры, или различные цифры, но определяющие эквивалентные состояния. В нашем случае эквивалентными будут устойчивые состояния (3) и (8); (2) и (5); (4) и (7).

После того, как эквивалентные состояния выявлены, они объединяются. При объединении эквивалентных состояний производятся:

приписывание каждой группе эквивалентных состояний одного номера, например наименьшего номера состояния, входящего в эту группу;

замена всех строк, в которые входят эквивалентные состояния одной группы, одной строкой.

Для рассматриваемого примера (см. табл. 4.1) устойчивое состояние (8) заменяем на (3). Аналогично состояния (7) и (5) заменяем соответственно на эквивалентные им состояния (4) и (2). Заменяем также и неустойчивые состояния 8, 7 и 5 соответственно на 3, 4 и 2. После этого получаем таблицу, в которой строки, соответствующие внутренним состояниям яз и *v.&,* ^4 и И7; *у.-г* и э<5, являются попарно одинаковыми. Строки таблицы переходов, которым соответствуют эквивалентные устойчивые состояния, объединяем и получаем таблицу переходов с пятью строками (табл. 4.2).

Выше были произведены выявление и объединение эквивалентных состояний для полного автомата. Для недоопределенного автомата кроме понятия «эквивалентности» состояний введено понятие «пссвдоэквивалентности» состояний [1, 3]. *Псевдоэквивалентными* называются такие два устойчивые состояния асинхронного автомата, которые удовлетворяют следующим условиям:

1) им соответствует одно и то же состояние входа автомата;

2) им соответствуют непротиворечивые состояния выходов автомата;

3) любой последовательности состояний входа автомата соответствуют непротиворечивые последовательности состояний **его** выхода независимо от того, какое из этих рассматриваемых устойчивых состояний автомата взято за исходное. Другими словами, среди пар последовательностей состояний входа и выхода автомата, начинающихся с этих устойчивых состояний, не должно быть противоречивых.

После объединения эквивалентных и псевдоэквивалснтных состояний всегда получается таблица, каждой строке которой, как и в первоначальной таблице, соответствует только одно устойчивое состояние. Сокращение числа строк таблицы переходов при объединении эквивалентных и пссвдоэквивалентных состояний происходит за счет уменьшения числа устойчивых состояний. Однако после объединения устойчивых состояний можно произвести дальнейшее сжатие таблицы, объединив совместные внутренние состояния автомата (совместимые строки).

Совместимыми внутренними состояниями автомата называются состояния, при объединении которых будет получен автомат, эквивалентный исходному. Под совместимыми внутренними состояниями можно понимать таких два внутренних состояния, которым соответствуют строки с непротиворечивым размещением цифр в них, т. е. такие строки, в одном и том же столбце которых должны быть одинаковые цифры или в одной строке — цифра, а в другой — прочерк. Например, для табл. 4.2 внутренние состояния X1 и X5; *к-2,* хз и X4 являются совместимыми.

Строки таблицы переходов, соответствующие совместимым внутренним состояниям, могут быть объединены в одну (табл. 4.3). В объединенной строке должны быть устойчивые состояния во всех тех столбцах, в которых были устойчивые состояния в какой-либо из объединяемых строк. Таким образом, при объединении совместимых внутренних состояний получаем таблицу, в каждой строке которой может быть уже несколько устойчивых состояний. Объединением всех совместимых внутренних состояний заканчивается процесс минимизации числа внутренних состояний асинхронного автомата.

При минимизации числа внутренних состояний синхронного автомата необходимо учитывать особенности его таблицы переходов. Эта особенность состоит в том, что в таблице переходов синхронного автомата отсутствует деление состояний автомата на устойчивые и неустойчивые. В связи с этим нельзя говорить об эквивалентности или псевдоэквивалентности полных состояний автомата. Здесь, в отличие от асинхронного автомата, можно рассматривать лишь совместимость внутренних состояний. При этом совместимость внутренних состояний синхронного автомата определяется аналогично совместимости внутренних состояний асинхронного автомата.

Таблица 4.3

	ρ_1	ρ_2	ρ_3	ρ_4
1	(1), 0	4	2	(6), 0
2	1	(4), 0	(2), 0	(3), 1

Пример 4.4. Необходимо минимизировать первоначальную таблицу переходов, полученную в примере 3.3 (см. табл. 3.9).

Решение. Вначале попытаемся выявить эквивалентные или псесдоэкннвалентные состояния. Ими могли бы быть только устойчивые состояния (2) и (4), так как они находятся в одном и том же столбце. Однако этим состояниям соответствуют различные состояния выхода. Поэтому в табл. 3.9 эквивалентных и псевдоэквпвалентных состояний нет. Попытаемся теперь найти совместные внутренние состояния, используя метод, изложенный в [1, 3]. Для этого составим треугольную таблицу (табл. 4.4).

Таблица 4.4

x_2	V		
x_3	×	×	
x_4	×	×	V
	x_1	x_2	x_3

Таблица 4.5

	x_1x_2; z 0 0	1 0	1 1
x'_1	(1), 0	(2), 0	3, 1
x'_2	1, 0	(4), 1	(3), 1

Исходя яз этой таблицы найдем максимальные группы совместных внутренних состояний: 1, 2; 3 4.

Таким образом, можно объединить внутренние состояния xi и *MS,* а также *у.з* и Я4. После объединения этих внутренних состоянии получим минимизированную таблицу переходов (табл. 4.5) с двумя внутренними состояниями, т. е. требуется один элемент памяти.

Пример 4.5. Необходимо минимизировать первоначальную таблицу переходов, полученную в примере 3.4 (см. табл. 3.13).

Решение. Анализ табл. 3.13 показывает, что эквивалентных состояний в ней иет. Однако имеются псевдоэквивалентные состояния (2) и (7). После их объединения и замены состояния (7) на состояние (2) получим табл. 4.6.

Таблица 4.6

	x_1x_2; z 00	10	11	01
1	(1),0	2,0	—	5,0
2	1,0	(2),0	3,1	—
3	—	4,1	(3),1	—
4	1,0	(4),1	—	—
5	—	—	6, ∽	(5), 0
6	—	2,0	(6), ∽	—

Таблица 4.7

2	V				
3	×	×			
4	×	×	V		
5	V	3,6×	3,6×	V	
6	V	3,6×	×	×	V
	1	2	3	4	5

Выявим теперь совместимые внутренние состояния, для чего построим треугольную таблицу (табл. 4.7).

В табл. 4.7 в клетках* (2, 5), (2, *6)* и (3, 5) указана пара внутренних состояний яз и Хб. Это означает, что внутренние состояния ха и *x.s; v.i* и Хо; xs ii *у.5* могут быть объединены (т. е. будут совместимыми), если совместимы внутренние состояния Хз и хв. Однако внутренние состояния Хз и *л^* несовместимы—крест в клетке (3, 6). Значит, пары внутренних состояний Хг и *у.ъ', у--!,* и -х.з; у.з .ii Х5 несовместимы. Поэтому в клетках (2,5), С2, 6) и (3,5) треугольной таблицы указаны кресты.

Найдем теперь группы совместимых внутренних состояний:

1	2	3	4	5	6
1, 2	3,4		4,5		5,6
1, 5					
1, 6					
1,5	,6				

Из них максимальными группами совместимых внутренних состоянии будут следующие: а=1,2; 6=1, 5, 6; с=3, 4; d=4, 5, три пары из которых (группы а и &, *Ь* и *d)* являются .пересекающимися. Остальные группы (;1, 5, 1, 6 5, 6, 1, 2, 3, 4, 5, 6) входят в максимальные группы и поэтому не являются максимальными. Теперь, строя таблицу покрытий (табл. 4.8), необходимо найти минимальное число максимальных групп совместимых внутренних состояний.

Таблица 4.8

	1	2	3	4	5	6
a	V	V				
b	V				V	V
c			V	V		
d				V	V	

Таблица 4.9

	x_1x_2; z 00	10	11	01
a	(*a*),0	(*a*),0	*c*,1	*b*,0
b	(*b*),0	*a*,0	(*b*),ъ	(*b*),0
c	*a*,0	(*c*),1	(*c*),1	(*c*),0

По табл. 4.8 составим функцию Петрика [3]:

$F = (a \vee b)\, ac\, (c \vee d)\, (b \vee d)\, b.$

Преобразуем ее в дизъюнктивную нормальную форму (ДНФ):

$F = (a \vee ab)\, (c \vee cd)\, (b \vee bd) = acb.$

* В клетке (i, *j)* номер *I* соответствует столбцу, а i — строке треугольной таблицы.

Таким образом, все максим ал ь;;ые группы, кроме *d,* должны быть использованы при построении минимальчои таблицы переходов (табл. 4.9) с тремя внутренними состояниями. При этом состояние автомата обозначаем буквой (номером) того внутреннего состояния, в котором оно устойчиво.

Следует заметить, что выявлять и объединять эквивалентные и псевдоэквивалентные состояния перед выявлением и объединением совместимых внутренних состояний нет необходимости. Можно сразу начать выявление совместимых внутренних состояний с последующим их объединением. При этом объединятся, если они есть, эквивалентные и псевдоэквивалентные состояния автомата. К тому же в ряде случаев первоначальное объединение псевдоэквивалентных состояний недоопределенного автомата может ухудшить окончательное решение, т. е. будет получен автомат с большим числом внутренних состояний по сравнению с автоматом, для которого сразу выявлялись совместимые внутренние состояния.

Пример 4.6. Необходимо минимизировать таблицу переходов (см. табл. 3.16), полученную в примере 3.5.

Таблица 4.10

2	×													
3	×	2,4×												
4	×	×	×											
5	×	×	×	4,6×										
6	×	×	×	×	×									
7	×	×	×	×	×	6,8×								
8	×	×	×	×	×	×	×							
9	×	×	×	×	×	×	×	8,10×						
10	×	×	×	×	×	×	×	×	×					
11	×	×	×	×	×	×	×	×	×	10,12×				
12	2,12×	×	×	×	×	×	×	×	×	×	×			
13	×	V	13,14×	×	×	×	×	×	×	×	×	×		
14	×	×	×	×	×	×	×	×	×	×	×	V	×	
15	×	×	×	V	13,15×	×	×	×	×	×	×	×	×	×
16	×	×	×	×	×	V	15,16×	×	×	×	×	×	×	×	×
17	×	×	×	×	×	×	×	V	16,17×	×	×	×	×	×	×	×
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Решение. Выявим совместимые внутренние состояния без предварительного выявления и объединения эквивалентных и псевдоэквивалентных состояний. Для этого построим треугольную табляцу (см. табл. 4.10).

По табл. 4.10 получаем следующие группы совместимых внутренних состояний:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

\- 2,13 — 4,15 — 6,16 — 8,17 — 12,14 — — — — —

Таким образом, максимальным группами совместимых внутренних состояний_ являются: *1=1;* 2=2, 13; 3=3; 4=4, 15; 5=5; 6=6, 16; 7=7; 8=8, 17; 9=9; 10=10; 11=11; 12=12, 14. Группы все непересекающиеся, а поэтому таблицу покрытий строить нет необходимости. Следовательно, получаем минимальный автомат с 12 внутренними состояниями (табл. 4.11).

Таблица 4.11

	x_1x_2 0 0	1 0	1 1	0 1	$z_1z_2z_3z_4z_5$
1	(1)	2	(1)	(1)	0 0 0 0 0
2	3	(2)	—	(2)	1 0 0 0 0
3	(3)	4	(3)	12	1 0 0 0 0
4	5	(4)	—	(4)	0 1 0 0 0
5	(5)	6	(5)	2	0 1 0 0 0
6	7	(6)	—	(6)	0 0 1 0 0
7	(7)	8	(7)	4	0 0 1 0 0
8	9	(8)	—	(8)	0 0 0 1 0
9	(9)	10	(9)	6	0 0 0 1 0
10	11	(10)	—	—	0 0 0 0 1
11	(11)	12	(11)	8	0 0 0 0 1
12	1	(12)	—	(12)	0 0 0 0 0

4.2. Формирование микрокоманд микропрограммного автомата

В разд. 3.4 было отмечено сходство матричной схемы алгоритма и матрицы переходов, причем сходство отмечалось при условии сопоставления с каждым оператором МСА отдельного внутреннего состояния автомата. Поэтому МСА можно рассматривать как язык задания автомата при условии, что оператор сопоставляется с выходом автомата, соответствующим не полному, а внутреннему состоянию автомата (т. е. автомату Мура [1, 3]). При такой автоматной интерпретации матричной схемы алгоритма задача минимизации числа внутренних состояний автомата состоит в объединении совместимых внутренних состояний автомата, имеющих непротиворечивые состояния выходов, т. е. непротиворечивые операторы.

Из сказанного вытекает, как следствие, то, что если условия работы микропрограммного автомата (МА) заданы не МСА, а ЛСА, то для минимизации числа внутренних состояний МА необходимо иметь метод, обеспечивающий непосредственно по ЛСА допустимое объединение операторов в группы, с каждой из которых могло бы сопоставиться внутреннее состояние МА. Такой метод получил название метода формирования микрокоманд [2, 3]. При этом, когда говорят о МА, обычно члены ЛСА (операторы и ЛУ) называют *микрооперациями* (операторы — внешние, ЛУ — внутренние микрооперации), а внутренние состояния МА — *макрокомандами [2,* 3]. Упорядоченная совокупность микрокоманд,. обеспечивающая реализацию автоматом заданных условии работы, называется *микропрограммой [2,* 3]. Таким образом, минимизация числа внутренних состояний МА—это такое объединение микроопераций в группы (т. е. в микрокоманды), при котором образуется минимальное число микрокоманд.

Рассмотрим метод формирования микрокоманды, включаюшен максимально допустимое число одновременно выполняемых микроопераций. Объединение нескольких микроопераций в одну микрокоманду дает возможность сократить общее число микрокоманд в микропрограмме, что приводит к уменьшению числа внутренних состояний МА, а значит и к уменьшению сложности последнего. Прежде чем изложить способ формирования микрокоманд, введем несколько определений.

Говорят, что логическое условие *рг* входит в распределение сдвигов оператора Л,, если последний может изменить значение *pi,* или оно может независимо измениться при выполнении данного оператора. Например, *pi* в ЛСА (3.9) проверяет значение номера входа (не превысило ли значение f числа имеющихся входов, т. е. *i^n),* а оператор *Fi* после прибавления к *i* единицы может изменить значение ЛУ *p*\. В частности, если перед выполнением оператора *Fj* значение *1=n,* а поэтому pi=l, то после реализации оператора *Fi* значение pi изменится, т. е. pi=0, так как *i>n.* Поэтому ЛУ *pi* входит в распределение сдвигов оператора Fi.

Выявив, таким образом, все те ЛУ (например, pj1, *pj2, ...,* pjl)
l<n, где n- число всех ЛУ в ЛСА), значения которых могут измениться после выполнения оператора Aj, получим распределение сдвигов для оператора *A],* которое запишем в виде *Aj={pi, pi2,...,pil}.*
Определив распределения сдвигов для каждого оператора ЛСА получим распределение сдвигов для всех

операторов ЛСА (полное распределение сдвигов). Из распределений сдвигов для операторов выделяют универсальное распределение, когда l=n, т. е. *A,—{pi, pi, ..., pn},* и пустое распределение сдвигов, когда оператор *A,* не изменяет значения ни одного из *n* ЛУ, *m.* е. *AJ—*{—}Более подробно о распределении сдвигов можно наити в [3].

Два члена ЛСА *Xi* и *XJ* являются противоречивыми, если:

1) они не могут быть выполнены одновременно;

2) *Xi* — логическое условие, а *xj* — оператор, причем ЛУ xj входит в распределение сдвигов оператора xj.

Член *Xi,* который может быть выполнен непосредственно перед членом *x,,* называется *предшественником x,,* а соответственно член *Xj — последователем* члена *Xj.* Очевидно, у любого члена ЛСА может быть несколько предшественников, у оператора может быть только один последователь, а у логического условия—два последователя, за счет чего при выполнении ЛСА образуются разветвления. Группа членов ЛСА образует ветвь, если каждый (И-1)-й член этой ветви является последователем 1-го члена.

Один и тот же член ЛСЛ может входить в несколько различных ветвей.

Группу членов ЛСА назовем *совместимой,* если в каждой ветви, образуемой членами этой группы, не содержится противоречивых членов. Совместимую группу членов ЛСА назовем *максимальной,* если добавление в нее любого другого члена ЛСА превращает ее в несовместимую. Каждая совместимая группа, в том числе и максимальная, может интерпретироваться как отдельная микрокоманда. Такая микрокоманда может состоять из набора как внутренних (логических условий), так и внешних (операторов) микроопераций, но при определенном наборе значений логических условий одновременно выполняются микрооперации, входящие лишь в одну ветвь микрокоманды. Для получения совместимой группы *Mxi* (микрокоманды), т.е. группы, первым членом которой является *Xi* выписывается член *Xj* ЛСА. Если у *Xi* имеется один последователь (т. е. Xj,—оператор), то этот последователь приписывается справа от *Xj.* При наличии двух последователей образуется разветвление и каждый из них выписывается справа от л--; на отдельной ветви. Этот процесс повторяется для всех вновь приписанных членов группы в каждой ветви. Ветвь группы Mxj, обрывается, если в нее вошел последний оператор ЛСА, либо если в нее должен быть включен член ЛСА, который является противоречивым с одним членом этой ветви, либо если после *x,* необходимо выписать *Xi,* который уже вошел в другую ветвь группы Mxi. В последнем случае необходимо указать стрелку от *Xj* к *Xi.* Формирование группы заканчивается после того, как обрываются все ее ветви. Если в первую группу вошли не все члены ЛСА, образуется вторая группа, начиная с наименьшего по порядку члена ЛСА, не вошедшего в предыдущую группу. Так процесс повторяется до тех пор, пока каждый член ЛСА не войдет хотя бы в одну группу.

Следует заметить, что если последователь *Xг* последнего члена ветви этой группы не был включен хотя бы в одну группу (например, из-за того, что *Xг* является противоречивым с одним из членов ветви), то обязательно должна быть группа *MXI.* Кроме того, отметим, что если имеется группа (микрокоманда) Mxi, то в остальных группах все ветви, в которые входит член xj, МОГУТ быть оборваны на предшественнике этого члена *Xi.*

Процесс формирования микрокоманд рассмотрим на следующих примерах.

Пример 47 Пусть задана ЛСА (3.9). Исходя из содержательного рассмотрения операторов этой ЛСА (см. пример 3.6), выявим операторы, которые можно выполнить одновременно. Информацию о возможности одновременного выполнения операторов (т. е. непротиворечивых операторов) представим в треугольной таблице (табл. 4.12), аналогичной той, которая использовалась ггря минимизации числа внутренних состояний автомата (см., например, табл. 4.11).

Несмотря на то, что в ЛСА (3.9) не указаны в явном виде операторы начала *Ao* и конца *Ak,* при формировании микрокоманд, они должны учитываться. При этом предполагается, что *Ak* переводит автомат в исходное состояние, а поэтому считается, что он всегда совместим с Ло. Вместе с тем операторы *Ao* и *Ak* не должны быть совместимы ни с одним другим оператором ЛСА.

Из табл. 4.12 видно, что кроме операторов *Ay* и *Ak* совместимыми являются также пары операторов *F{* и *F]; A* и *Ач; A* и Лз. Распределения сдвигов выявим также из содержательного рассмотрения операторов и ЛУ ЛСА (3.8). При этом для операторов *Ац я Ak* всегда берется универсальное распределение сдвигов:

$$A_0 — \{p_1, p_2, p_3, p_4, p_5\}\ ; \quad A_k — \{p_1, p_2, p_3, p_4, p_5\}\ ;$$

$$F_i — \{p_1, p_2, p_3\},\ F_j — \{p_2\ p_4, p_5\}\ ; \quad A_1 — \{p_5\}\ ; \quad A_2 — \{—\}\ ; \quad A_3 — \{p_3\}\ .$$

Как видно, для оператора *As* получено пустое распределение сдвигов.

Для примера рассмотрим составление распределения сдвигов для оператора *A1,* при этом выясним, чго он не может изменить значения следующих ЛУ:

1) *pi,* поскольку о.н не меняет значения i; 2) p2, так как возможность изменения определяется только типом измерительного прибора, подключенного к *i-v.* вертикали, а оператор *A1* не меняет значения i; 3) p3, так как гоступленяе вызова от него не зависит, а номера *i* он не меняет; 4) *pj,* поскольку значение / изменяется только оператором *F].*

Таким образом, в распределение сдвигов для оператора *A* логические условия *pi, ps, pз* и /?4 не входят. Оно будет содержать только одкэ ЛУ *ps,* так как после реализации оператора *Ai* объект, подключенный к у-ц горизонтали, станет занятым л *ps* сменит свое значение с «I» на «О».

Образуем максимальные совместимые группы членов ЛСА (3.8). Для этого возьмем оператор *Ац* и

припишем справа его последователь *Ay->[Pг]*. Так как *Fi* не совместим с *Ay* (см. табл. 4.12), то ветвь обрывается на предыдущем члене (т. е. Ло), а оператор *F,* указывается в квадратных скобках. Поскольку у Ао имеется только один последователь, группа *MAo* состоит из одного оператора Ао, т. е. МАо={Ао-> [Fi]}.

Условимся, что члены, указанные в квадратных скобках, не входят з данную группу, а являются начальными членами других групп. Образуем остальные максимальные совместимые группы, при этом стрелка со знаком «+» после *pi* означает, что *pi=1,* а со знаком «—» *pi=0;*

$$M_{F_i} = \{F_i \to [p_1]\}\ ;\ M_{p_1} = \left\{ \begin{array}{l} p_1 \\ \quad +\nearrow F_j \to [p_4] \\ \quad -\searrow [A_k] \end{array} \right\} ;$$

$$M_{p_4} = \left\{ \begin{array}{l} p_4 \\ \quad +\nearrow [A_1] \\ \quad -\searrow [F_i] \\ p_2 \\ \quad +\nearrow p_5 \\ \quad -\searrow [F_j] \\ p_5 \\ \quad +\nearrow [F_j] \\ \quad -\searrow [F_i] \end{array} \right\} ;\quad M_{A_1} = \{A_1 \to A_2 \to [F_i]\}\ ;$$

$$M_{F_j} = \{F_j \to [p_4]\}.$$

Таблица 4.12

	A_0	F_i	F_j	A_1	A_2	A_3
F_i	×					
F_j	×	∨				
A_1	×	×	×			
A_2	×	×	×	∨		
A_3	×	×	×	∨	×	
A_k	∨	×	×	×	×	×

Таблица 4.13

	A_0	A_1	A_2	A_3	A_4	A_5	A_6
A_1	×						
A_2	×	×					
A_3	×	×	∨				
A_4	×	∨	∨	∨			
A_5	×	×	∨	∨	∨		
A_6	×	∨	∨	×	∨	∨	
A_7	×	×	∨	×	∨	∨	×

Как было ранее отмечено, операторы Ао и *Ak* объединяются в одну микрокоманду, т. е.

$$M_{A_0 \cdot A_k} = \{A_0\,(A_k) \to [F_i]\}.$$

Таким образом, получено шесть микрокоманд, т. е. шесть внутренних состояний МА.

Пример 4.8. Пусть задана ЛСА (4.1), в которой различные вхождения одного и того же ЛУ пронумерованы различными верхними индексами:

$$\mathfrak{A} = \uparrow^1 p_1 \uparrow^3 p_2 \uparrow^2 A_1 p_3^1 \uparrow^2 A_3\,\omega \uparrow^3 \downarrow^2 A_2 \downarrow^3 p_4 \uparrow^1 A_4\, p_6 \uparrow^4 p_3^2 \uparrow^4 p_5 \uparrow^4 A_6\, A_7\,\omega \uparrow^1 \downarrow^4 A_5\,\omega \uparrow^1 . \tag{4.1}$$

Исходя из анализа операторов ЛСА (4.1), рассмотренный в [2], выявлены операторы, которые можно реализовать одновременно. Информацию о возможности одновременного выполнения операторов представим в виде табл. 4.13.

Из рассмотрения функции, выполняемых операторами и ЛУ ЛСА (4.1), выявлены такие распределения сдвигов:

В группе *МА,* можно было бы продолжить составление группы после А5, так как *p1* не входит в распределения сдвигов операторов А4 и *А5.* Однако здесь выписывание этой группы прерывается, так как уже имеется группа *Mp1.*

Таким образом, в данном примере получено восемь групп, а следовательно, восемь микрокоманд, причем для каждого последователя последних членов любой ветви этих микрокоманд имеется микрокоманда, в которой он будет первым членом. Поскольку все члены ЛСА (4.1) выписаны в этих восьми группах, полученная система микрокоманд является полной.

$A_0 - \{p_1, p_2, p_3, p_4, p_5, p_6\}$; $A_1 - \{p_1\}$; $A_2 - \{p_1\}$;
$A_3 - \{p_1 p_2, p_3, p_6\}$; $A_4 - \{p_4\}$; $A_5 - \{-\}$;
$A_6 - \{p_3, p_5\}$; $A_7 - \{p_3, p_5, p_6\}$.

Образуем максимальные совместимые группы членов ЛСА (4.1):

$A_0 \to [p_1]$; $M_{A_0} = \{A_0 \to [p_1]\}$;

$$M_{p_1} = \left\{ \begin{array}{l} +[A_1 \\ p_2 \\ +\ -[A_2] \\ p_1 \\ -\ +\ A_4] \\ p_4 \\ -[p_1] \end{array} \right\} ; \quad M_{A_1} = \left\{ \begin{array}{l} +[A_3] \\ A_1 \to p_3^1 \\ -[A_2] \end{array} \right\} ;$$

$$M_{A_2} = \left\{ \begin{array}{l} +[A_4] \\ A_2 \to p_4 \\ -[p_1] \end{array} \right\} ; \qquad M_{A_4} = \left\{ \begin{array}{l} +[A_6] \\ p_5 \\ +\ -A_5 \to [p_1] \\ p_3^2 \\ +\ -A_5 \to [p_1] \\ A_4 \to p_6 \\ -A_5 \to [p_1] \end{array} \right\} ;$$

$$M_{A_3} = \left\{ \begin{array}{l} +[A_4] \\ A_3 \to p_4 \\ -[p_1] \end{array} \right\} ; \quad M_{A_6} = \{A_6 \to [A_7]\} ; \quad M_{A_7} = \{A_7 \to [p_1]\}.$$

4.3. Кодирование внутренних состояний автомата

В том случае, когда автомат задается первоначальной таблицей включений, при выборе минимального (или близкого к минимальному) числа элементов памяти одновременно производится включение ЭП в тех или иных тактах, т. е. осуществляется сопоставление с каждым тактом того или иного набора состояний ЭП. Иначе говоря, при включении ЭП в явном виде с внутренним состоянием автомата сопоставляется определенный набор состояний ЭП.

Если автомат задается таблицей переходов или ЛСА, то после минимизации числа внутренних состояний автомата (или сформирования микрокоманд) внутренним состояниям (или микрокомандам) порежнему не приписываются состояния ЭП. Это объясняется тем, что на данном этапе нет необходимости рассматривать каждое внутреннее состояние как определенный набор состояний ЭП. Достаточно иметь только номер внутреннего состояния без указания состояний ЭП в этом состоянии. Для построения же структурной схемы автомата, т. е. построения цепей включения ЭП и выходных цепей, необходимо знать набор состояний ЭП, который сопоставляется с тем или иным внутренним состоянием. При этом оказывается, что в зависимости от того, как будут приписаны эти наборы состояний ЭП каждому из внутренних. состояний автомата, могут в значительной степени измениться сложность структуры автомата, его устойчивость в работе и т. п.

Процесс приписывания каждому внутреннему состоянию набора состояний ЭП в теории автоматов носит название *кодирования внутренних состояний автомата.* Вообще говоря, такие наборы значений двоичных переменных надо приписать не только внутренним состояниям автомата, но и каждому состоянию его входа и выхода. Однако во многих практических случаях состояния входа и выхода оказываются уже закодированными, так как они являются соответственно выходными и входными состояниями других блоков и устройств.

Пусть с каждым внутренним состоянием сопоставлен набор состояний ЭП, т. е. кодовая комбинация. Если такое сопоставление осуществляется произвольным образом, то может оказаться,. что четырем внутренним состояниям автомата приписаны кодовые комбинации 000, Oil, 010 и 001 (рис. 4.4,а). Пусть, кроме того, по условию работы автомат при появлении на его входе состояния pi должен перейти из своего внутреннего состояния, которому приписана кодовая комбинация 000, во внутреннее состояние 011. (Этот переход на рис. 4.4,а показан сплошной линией с указанием состояния входа pi, вызывающим этот переход.) При этом условиями работы не предусматривается переход автомата из внутреннего состояния 000 в остальные два внутренние состояния, которым приписаны кодовые

комбинации 001 и 010.

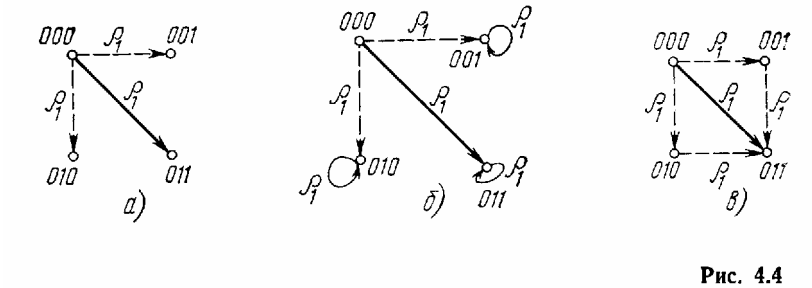


Рис. 4.4

Таким образом, при переходе от состояния 000 к состоянию 011 в автомате должны одновременно сработать второй и третий ЭП. Однако из-за разброса времени срабатывания ЭП (в качестве ЭП могут быть использованы электромагнитные реле, триггеры и другие двухстабильные элементы) может сработать вначале второй или третий ЭП. Из-за этого автомат попадает в одно из Двух непредусмотренных в данном переходе внутренних состояний, которым приписаны кодовые комбинации 010 и 001 соответственно (на рис. 4.4,а указанные переходы обозначены штриховыми линиями). Аналогичное явление может произойти и при отпускании ЭП.

Говорят, в автомате, в котором внутренние состояния закодированы так, что в заданных условиями работы переходах найдется хотя бы один переход, при котором требуется одновременно сработать (отпустить) двум и более ЭП, имеются состязания элементов памяти. Среди состязаний ЭП различают критические и некритические.

К *критическим* относятся такие состязания, которые могут исказить алгоритм функционирования автомата, т. е. его функции переходов и выходов. Например, если в рассмотренном выше примере автомат из состояний 001 и 010 при состоянии входа pi никуда не должен переходить (рис. 4.4,6), то из-за состязаний ЭП он может lie перейти в состояние 011, что предусмотрено условиями его работы. Это есть пример критических состязаний ЭП, из-за которых искажается функция переходов автомата.

К критическим относятся также и такие состязания ЭП, которые искажают функцию выходов. Например, из рис. 4.4,в видно, что функция переходов автомата не искажается. Действительно, даже если автомат из-за состязаний ЭП перейдет в состояние 001 или 010, то при том же состоянии входа он из этих состояний затем перейдет в заданное условиями работы состояние 011. Однако если с внутренними состояниями 000 и 011 сопоставлено значение выходного сигнала z=l, а с внутренними состояниями 010 и 001 — значение выходного сигнала *z=0,* то, например, при переходе автомата 000^-010-^-011 возникает кратковременное изменение значения выходного сигнала с *z=* на z=0, которого не было бы при непосредственном переходе автомата 000—-011. Состязания ЭП, вызывающие такое искажение функции выходов автомата, также называются критическими.

К *некритическим* относятся состязания ЭП, которые не приводят к искажениям ни функции переходов, ни функции выходов. Например, если бы на рис. 4.4,в с внутренними состояниями 001 и 010 было сопоставлено состояние выхода z=l или 2=~, то имеющиеся в автомате состязания были бы некритическими.

Легко понять, что состязания вообще не возникнут в том автомате, в котором при переходе из одного внутреннего состояния в другое изменяет свое состояние только один ЭП.

В теории автоматов [3] разработаны методы кодирования внутренних состояний, которые позволяют так приписывать кодовые комбинации внутренним состояниям, что в автомате на всех
заданных условиями работы переходах не возникают критические состязания ЭП. Не имея возможности подробно осветить в данном пособии все современные методы кодирования внутренних состоянии автомата, рассмотрим на примере один из них, который является наиболее простым для понимания этого процесса, но который, к сожалению, пригоден лишь для автоматов с небольшим числом внутренних состояний. Метод кодирования внутренних состояний на основе разбиений изложен в [1, 3].

Пример 49 Пусть задан автомат (счетчик импульсов) таблицей переходов (табл. 4.14). Требуется закодировать его внутренние состояния.

Таблица 4.14

	ρ_1	ρ_2	$z_1\ z_2$	Кодовые комбинации
x_1	(1)	2	0 0	0 0
x_2	3	(2)	1 0	0 1
x_3	(3)	4	1 1	1 1
x_4	1	(4)	0 1	1 0

Для этого вначале составим диаграмму его возможных переходов при состояниях входа р, и ра (рис. 4.5). Затем возьмем развертку л-мерного единичного куба, указывающую связь его вершин с ребрами. Например, на рис. 4.6 изображена развертка трехмерного куба.

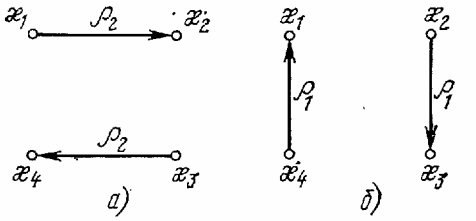


Рис. 4.5

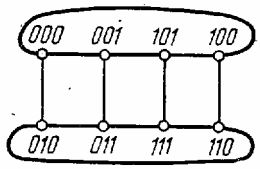


Рис. 4.6

С вершинами развертки п-мерного единичного куба сопоставлены наборы значений двоичных переменных, соответствующих значениям *n* координат каждой вершины единичного куба, одна из вершин которого совмещена с началом системы *n* координат, а его *n* ребер совпадают по направлению с соответствующими *n* координатами. Очевидно, в такой развертке соседним вершинам (т. е. вершинам, .непосредственно связанным ребром) приписаны соседние кодовые комбинации (т. е. кодовые комбинации, отличающиеся значением в одном разряде).

Как было отмечено ранее, в автомате не будет критических состязаний, если при его переходах будет изменяться только один ЭП. Тогда легко понять, что если совместить каждую из диаграмм переходов автомата (см. рис. 4.5) с разверткой п-мерного единичного куба (см. рис. 4.6), то тогда в автомате не будет критических состязаний. При этом надо стремиться использовать куб с наименьшим числом л. Тогда кодирование автомата будет осуществляться с минимальным числом ЭП.

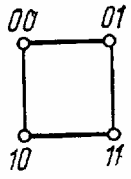
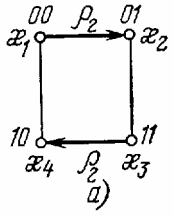


Рис. 4.7



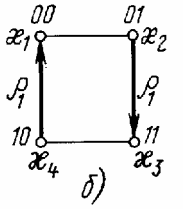


Рис. 4.8

В нашем случае можно использовать развертку двумерного куба (рис. 4.7). Совмещая диаграмму переходов автомата при рг (см. рис. 4.5,о) с разверткой рис. 4.7, получаем рис. 4.8,а, а диаграмму переходов при pi (см. рис. 4.5,6) с той же разверткой рис. 4.7, получаем рис. 4.8,6. При этом очевидно, что с одними и теми же внутренними состояниями автомата в диаграммах переходов при различных состояниях входа должны сопоставляться одни и те же вершины развертки га-мерного куба.

В табл. 4.14 (последняя колонка) указаны выбранные выше кодовые комбинации (состояния ЭП).

Заметим, что в некоторых случаях такое совмещение для *n,* равного минимальному числу ЭП, невозможно. Тогда берут куб с большим *n,* причем для любого автомата такое *n* всегда найдется. В микропрограммном автомате, который относится к синхронным автоматам, состязания ЭП устраняются использованием так называемой «двойной памяти», когда используются два регистра микрокоманд. В связи с этим микрокомандам, представляющим собой внутренние состояния МА, можно приписывать кодовые комбинации, т. е. набор состояний ЭП, вообще говоря, произвольно. Например, микрокомандам, полученным в примере 4.8, припишем кодовые комбинации ЭП так, как показано в табл. 4.15.

Таблица 4.15

Микрокоманды	Кодовые комбинации
M_{A_0}	0 0 0
M_{p_1}	1 0 0
M_{A_1}	1 1 0
M_{A_2}	1 1 1
M_{A_4}	0 1 1
M_{A_3}	0 1 0
M_{A_6}	1 0 1
M_{A_7}	0 0 1

Контрольные вопросы

1. Какие автоматы являются эквивалентными?
2. Могут ли эквивалентные недоопределенные автоматы иметь различное число внутренних состояний?
3. Какие члены ЛСА являются противоречивыми?
4. Какие существуют методы устранения критических состязании ЭП и ДУ?

Глава 5.

СТРУКТУРНЫЙ СИНТЕЗ ДИСКРЕТНОГО УСТРОЙСТВА

5.1. Таблица состояний элементов памяти

Таблица состояний элементов памяти, которую иногда называют *структурной таблицей переходов,* составляется по минимизированной таблице переходов, где каждой строке таблицы задано определенное состояние элементов памяти ЭП.

Таблица состояний элементов памяти, которую в дальнейшем будем называть таблицей ЭП, содержит число столбцов, равное числу состояний входов, и число строк, равное числу внутренних состояний автомата. Единица в клетке матрицы свидетельствует о наличии сигнала на входе ЭП, нуль—о его отсутствии и тильда (~) — о безразличном состоянии (0 или 1).

Способ построения таблицы зависит от свойств и возможностей применяемого элемента. Если в качестве ЭП используется нейтральное электромагнитное реле, то на его вход должен подаваться сигнал в следующих случаях:

1. Во всех устойчивых состояниях строк таблицы переходов, у которых значение элемента памяти равно 1.

2. Во всех неустойчивых состояниях таблицы переходов, соответствующих номерам устойчивых состояний, указанных в предыдущем пункте.

Физически первый случай определяет удержание реле в рабочем состоянии, а второй случай соответствует срабатыванию реле. Поясним сказанное на примере. Пусть работа автомата задана таблицей переходов 5.1. Таблица имеет четыре строки, соответствующие четырем внутренним состояниям автомата. Следовательно, для кодирования четырех состояний требуются два ЭП. Зададимся кодом работы реле У] н *Y^,* указанным в табл. 5.1.

Таблица 5.1

y_1, y_2	$x_1 x_2$ 00	10	11	01
00	(1)	2	3	~
10	1	(2)	(4)	5
01	6	~	(3)	(5)
11	(6)	2	(7)	~

Таблица 5.2

y_1, y_2	$x_1 x_2$ 00	10	11	01
00	(1)	2	3	~
10	1	(2)	(4)	5
11	(6)	2	(7)	~
01	6	~	(3)	(5)

Из табл. 5.1 видно, что i/i=l во второй и четвертой строках, а г/2=1 в третьей и четвертой строках. В соответствии с первым случаем на вход t/i сигнал должен подаваться в устойчивых состояниях (2), (4) второй строки и (6), (7) четвертой строки, а в соответствии со случаем 2 — во всех неустойчивых состояниях, соответствующих номерам устойчивых состояний (2), (4), (6), (7). Такими неустойчивыми состояниями являются состояния 2 в первой и четвертой строках и 6 в третьей строке. Аналогично на вход *У2* сигнал должен подаваться в устойчивых состояниях (3), (5), (6) и (7) и неустойчивых состояниях 3, 5, 6. Соответственно в первой, второй и третьей строках пустые клетки таблицы соответствуют безразличным состояниям, в которых ЭП может принимать значение «I» или «О».

Для удобства перехода к таблице ЭП и ее последующего упрощения поменяем местами третью и четвертую строки. Получим табл. 5.2. Записав условия работы ЭП, получим таблицы для реле *yi* (табл. 5.3) и для реле *уг* (табл. 5.4), где обязательные состояния отмечаются 1, запрещенные 0 и безразличные ~.

При использовании в качестве ЭП триггера типа *RS,* который переключается в состояние 1 при поступлении сигнала на вход S и отсутствии сигнала на входе *R,* а в состояние 0 при наличии сигнала на входе ./? и отсутствии сигнала на входе 5, в таблице ЭП необходимо указывать, на какой из входов подается сигнал при .включении или выключении триггера.

Таблица 5.3

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0	1	0	~
10	0	1	1	0
11	1	1	1	~
01	1	~	0	0

Y_1

Таблица 5.4

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0	0	1	~
10	0	0	0	1
11	1	0	1	~
01	1	~	1	1

Y_2

Из табл. 5.2. видно, что переключение триггера *Yi* из состояния и в состояние 1 производится при переходе из неустойчивых состоянии *2.* и 6 в устойчивые состояния с теми же номерами а переключение из состояния 1 в состояние О—при переходе из неустойчивых состояний 1 и 5 в устойчивые состояния с теми же номерами. Поэтому в неустойчивых состояниях 2 и 6 должен подаваться сигнал на входе S и отсутствовать сигнал на входе *R.* Исключение составляет переход от устойчивого состояния (6) или (7) через неустойчивое состояние 2 к устойчивому состоянию (3) где состояние триггера не меняется.

В табл. 5.5 в клетках, соответствующих состояниям 2 и 6 табл. 5.*2,* записывается 10, а в клетках, соответствующих состояниям 1 и 5, записывается 0,1 где первая цифра определяет состояние входа 1 а вторая — состояние входа *R.* Во второй и третьей строках, где y1=1 в устойчивых состоядиях *(2),* (4), (6), (7) и неустойчивом состоянии 2, сигнал не должен подаваться на вход R, а состояние входа S безразлично. Поэтому в клетках таблицы ЭП, соответствующих указанным состояниям, записывается ~0. В первой и четвертой строках, где y1=0 в устойчивых состояниях *(1),* (3), (5) и неустойчивом состоянии 3, сигнал не должен подаваться на вход S, а состояние входа R безразлично. В клетках таблицы ЭП соответствующих данным состояниям, записывается 0 ~ (табл. 5.5). Аналогично заполняется табл. 5.6 для триггера Y2.

При использовании в качестве ЭП асинхронного триггера типа D, который переключается в

Таблица 5.5

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0~	10	0~	~
10	01	~0	~0	01
11	~0	~0	~0	~
01	10	~	0~	0~

$S_1 R_1$

Таблица 5.6

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0~	0~	10	~
10	0~	0~	0~	10
11	~0	01	~0	~
01	~0	0	~0	~0

$S_2 R_2$

состояние 1 при наличии сигнала на входе *и* и возвращается в исходное состояние 0 при исчезновении сигнала на входе D, таблица ЭП будет соответствовать табл. 5.*3* и 5.4, поскольку действие такого асинхронного триггера аналогично функционированию нейтрального электромагнитного

Если в качестве ЭМ используется триггер типа Т (см. табл. 2.3), который переключается при поступлении каждого импульса на его вход, то в нашем примере первый триггер Y1 должен
из состояния 0 в состояние 1 в неустойчивых состояниях 2 и 6 и переключаться из состояния 1 в состояние. О в неустойчивых состояниях 1 и 5. Поэтому в таблице ЭП в клетках, соответствующих указанным неустойчивым состояпиям, записывается 1, а в остальных клетках, кроме тех, где указано безраличное состояние автомата, записывается 0. Аналогичным способом составляется таблица ЭП для триггера У2.

Последовательность выдачи сигналов на вход триггеров *Y1* и У2 показаны соответственно в табл. 5.7 и 5.8.

Таблица 5.7

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0	1	0	~
10	1	0	0	1
11	0	0	0	~
01	1	~	0	0

Y_1

Таблица 5.8

$y_1 y_2$ \ $x_1 x_2$	00	10	11	01
00	0	0	1	~
10	0	0	0	1
11	0	1	0	~
01	0	~	0	0

Y_2

Элемент памяти на триггере типа /А" переключается из состояния 0 в 1 при наличии сигнала на входе /

и безразличном состоянии входа *К.* и переключается из состояния 0 в 1 при наличии сигнала на входе /< и безразличном состоянии входа /. Поэтому в таблице ЭП в клетках, где указаны неустойчивые состояния, характеризующие переход триггера из состояния 0 в 1, ставится (1~), а в клетках с неустойчивыми состояниями, определяющими переход триггера из состояния 1 в 0, ставится (—1), где первое значение характеризует состояние входа /, а второе—входа *К.*

Для нашего примера (см. табл. 5.2) триггер Y1 переключается из состояния 0 в 1 в неустойчивом состоянии 2 первой строки и в неустойчивом состоянии 6, поэтому в клетках табл. 5.9, соответствующих состояниям 2 и 6, записывается 1~. Переключение триггера из состояния 1 в 0 осуществляется в неустойчивых состо яниях 1 и 5 табл. 5.2, поэтому в соответствующих клетках табл. 5.9 записывается ~1.

Таблица 5.9

$y_1 y_2$	00	10	11	01	00	10	11	01
	J_1	K_1	Y_1		J_2	K_2	Y_2	
00	0⌣	1⌣	0⌣	⌣	0~	0⌣	1⌣	⌣
10	⌣1	⌣0	⌣0	⌣1	0⌣	0⌣	0⌣	1⌣
11	⌣0	⌣0	⌣0	⌣	⌣0	⌣1	⌣0	⌣
01	1⌣	⌣	0⌣	0⌣	⌣0	⌣	⌣0	⌣0

В строках таблицы переходов, у которых значение ЭП равно 1 или 0 во всех устойчивых и неустойчивых состояниях, характеризующих переход к строке с тем же значением ЭП, се состояние не должно изменяться. В нашем примере в первой и четвертой строке, где y1=0, на местах устойчивых состояний (1), (3), (5) и неустойчивого состояния 3 табл. 5.2 в соответствующих клетках табл. 5.9 записывается (0~), а для второй и четвертой строки, где y1=1, на местах устойчивых состояний (2), (4), (6), (7) и неустойчивого состояния 2 записывается (~0). Аналогично для триггера У 2 в третьей и четвертой строках, где Y2=1, на местах устойчивых состояний (6), (7), (3), (5) и неустойчивого состояния 6 записывается в табл. 5.9 (~0), а для первой и второй строк, где У2=0, на местах устойчивых состояний (1), (2), (4) и неустойчивого состояния 2 записывается в табл. 5.9 (0~).

5.2. Таблица состояний выходов

Таблица состояний выходов имеет такое же состояние входов (столбцов) и такое же кодирование строк, как и минимизированная таблица переходов. Для построения таблицы состояний выходов (или кратко таблицы выходов) должны быть заполнены ее пустые клетки.

При заполнении пустых клеток необходимо учитывать возможность появления состязаний в результате возникновения кратковременных неправильных состояний выходов при переходе из одного состояния в другое. Если постоянная времени устройств, на которые воздействуют выходные сигналы больше, чем время появления кратковременных нарушений в последовательности появления сигналов, то можно не обращать внимания на появление состязаний. Однако если имеется возможность устранить состязания, то ею всегда следует воспользоваться при условии, что это не приведет к значительному усложнению всей схемы.

При разработке схемы автомата возникают определенные требования, обусловленные исходным заданием. К их числу относятся:

обеспечение наиболее простой структуры выходных цепей;

обеспечение заданного времени появления сигнала на выходе;

отсутствие состязаний на выходах.

Рассмотрим на примере автомата, условия функционирования которого заданы табл. 5.2, как удовлетворяются эти требования. Автомат имеет два выхода Zi и *zs,* сигналы на которых появляются Б устойчивых состояниях, как показано в табл. 5.10.

Таблица 5.10

$y_1 y_2$	$x_1 x_2$ 00	10	11	01
00	(1), 00	2	3	⌣
10	1	(2), 01	(4), 01	5
11	(6), 01	2	(7), 11	⌣
01	6	⌣	(3), 10	(5), 00

Таблица 5.11

$y_1 y_2$	$x_1 x_2$ 00	10	11	01
00	00			
10		01	01	
11	01		11	
01			10	00

$z_1 z_2$

По таблице переходов 5.10 составлена таблица выходов 5.11, в которую перенесены значения выходных сигналов для каждого устойчивого состояния автомата. Первая цифра в клетках табл. 5.11 указывает на состояние выхода *Z1,* а вторая—на состояние выхода z2.

Если основным требованием является наибольшая простота структуры выходных цепей, то незаполненные клетки таблицы будут соответствовать безразличным состояниям, что **обеспечивает** более, свободный выбор вариантов минимизации схемы.

Относительно времени появления сигналов на выходе различают требования, когда все изменения состояний выходов должны происходить как можно скорее или как можно позднее или время появления каждого из состояний задается отдельно.

Изменение входных сигналов происходит быстрее, если с неустойчивым состоянием автомата сопоставлять такое состояние выхода, которое имеется у соответствующего устойчивого состояния. Для нашего примера при переходе из состояния (1), в котором выходные сигналы имеют значение 00, в состояние (2) с выходными сигналами 01 необходимо, чтобы в неустойчивом состоящий 2 выходные сигналы имели значение 01. Аналогичным способом заполнены остальные клетки таблицы выходов 5.12,а.

Если требуется, чтобы выходные сигналы появлялись как можно позже, во всех клетках таблицы выходов, соответствующих неустойчивым состояниям таблицы переходов, проставляются состояния выходов, от которых производится переход. Например, при переходе от состояния (1) к состоянию (3) выходные сигналы изменяют свое состояние с 00 на 10. Поэтому в клетке, соответствующей состоянию 3, проставляются 00. Однако в этом случае имеются некоторые ограничения. Если в одной строке находится несколько устойчивых состояний автомата с различными состояниями выходов, то во избежание состязаний на месте неустойчивого состояния ставится состояние выхода, к которому производится переход. Например, при переходе от (3), 10 или от (5), 00 к (6), 01 на месте состояния 6 следует записать 01. Если состязания выходов не учитывать, то на месте неустойчивого состояния ставится значение выходов одного из устойчивых состояний данной строки (в рассматриваемом случае 00 или 10). Аналогичное положение будет при переходе от (6) и (7) к (2). Состояние выходов с учетом состязаний показано в табл. 5.12,6.

Таблица 5.12

y_1y_2 \ x_1x_2	00	10	11	01	00	10	11	01	00	10	11	01
00	00	01	10	—	00	00	10	—	00	0—	—0	—
10	00	01	01	00	01	01	01	01	0—	01	01	0—
11	01	01	11	—	01	01	11	—	01	01	11	—
01	01	—	10	00	01	—	10	00	01	—	10	00
	а)				z_1	z_2	б)		в)			

Если основным требованием является устранение состязаний выходов и время их появления не учитывается, то при переходе от одного состояния выходов к другому в ряде случаев нет необходимости задавать на переходах состояние всех выходов, можно записывать только значение выхода, которое не изменяется. Состояние же второго выхода будет безразличным. Например, при переходе от (2) или (4) к (5) состояние выходов изменяется с 01 к 00. Единственный выход, который изменяет свое состояние, — это $z<t$. Поэтому в клетке, соответствующей переходу 5, можно записать 01 или 00, т. е. состояние выхода Zi должно быть равно О, а состояние *г-г.* безразлично. Аналогично заполняются остальные клетки табл. 5.12,в.

Исключение составляют переходы в табл. 5.2 через неустойчивое состояние 6 и через состояние 2 второй строки, при котором изменяются состояния обоих входов. В этих случаях в неустойчивых состояниях ставится такое значение выходов, которое соответствует устойчивому состоянию. Так, при переходе от (3),10 и от (5),00 ко (6),01 для неустойчивого состояния 6 указывается состояние выхода 01.

5.3. Минимизация функций

Минимизация — это процесс определения функций, содержащих минимальное число операций и переменных. Каждая переменная или операция функции соответствует в принципиальной схеме определенному элементу или их соединению. Поэтому одним из основных вопросов синтеза автомата является процесс упрощения функций. Для этой цели разработан ряд методов, основанных на законах алгебры Буля. К таким методам относятся алгебраический, табличный, существенных переменных и т. д.

При небольшом числе переменных (порядка шести) наиболее эффективным является табличный метод. Данный метод предполагает координатное представление функции, где каждый конституент (минитерм), представленный в десятичном эквиваленте, занимает одну из клеток таблицы. Каждый из конституентов, расположенных рядом по строке или столбцу, является соседним, т. е. отличается значением одной переменной. Это свойство таблицы позволяет проводить минимизацию функций путем объединения

(склеивания) соседних конституентов.

Такая таблица, используемая для минимизации четырех переменных, содержит четыре строки и четыре столбца. Если столбцам придать значение переменных, имеющих вес 1 и 2, а строкам — значения переменных с весом 4 и 8, то расположение но меров десятичных эквивалентов конституентов будет таким, как показано в табл. 5.13,а.

Таблица 5.13

y_1y_2 \ x_1x_2	00	10	11	01
00	0	1	3	2
10	4	5	7	6
11	12	13	15	14
01	8	9	11	10

а)

y_1 \ x_1x_2	00	10	11	01
0	0	1	3	2
1	4	5	7	6

б)

$y_1y_2y_3$ \ x_1x_2	00	10	11	01
000	0	1	3	2
100	4	5	7	6
110	12	13	15	14
010	8	9	11	10
001	16	17	19	18
101	20	21	23	22
111	28	29	31	30
011	24	25	27	26

в)

В табл. 5.13,а соседними будут не только номера конституентов, расположенных внутри таблицы, но также и номера, расположенные на концах каждого столбца или строки. Это значит, что соседними являются номера нижних клеток в любом столбце и верхних клеток того же столбца, а также номера и правых клеток одной и той же строки. Это обусловлено тем, что таблица представляет собой развертку на плоскости сложной пространственной фигуры гиперкуба, вершинами которого являются номера состояний таблицы.

Если функция содержит только три переменные, то табл. 5.13,6 будет содержать две строки, соответствующие переменной y1, имеющей вес 4. Для минимизации функций'с пятью переменными используются две таблицы для четырех переменных (табл. 5.13,в), в одной из которых переменная уз с весом 16 имеет значение 0, а в другой— 1. Каждая клетка одной таблицы является соседней с клеткой другой таблицы. Например, клетка, где записан 0, будет соседней с 16, 1 с 17, 8 с 24 и т. д.

Для функции с шестью переменными необходимо иметь четыре таблицы для четырех переменных, каждая из клеток которых является соседней с клетками двух других таблиц.

При минимизации для каждой функции составляется своя таблица, в которой проставляются ее значения: 0, 1 и '--. Минимизация функции для четырех переменных заключается в объединении номеров соседних клеток. Возможны следующие варианты объединения:

1. При объединении двух клеток, расположенных рядом по строке или столбцу, из данного объединения исключается одна переменная. В результате образуется конъюнкция, содержащая три переменные.

2. В случае объединения четырех клеток, расположенных в одной строке, столбце или образующих квадрат, устраняются две переменные и общая конъюнкция содержит две переменные.

3. При объединении двух соседних строк или столбцов (восьми номеров) устраняются три переменных и полученная конъюнкция состоит из одной переменной.

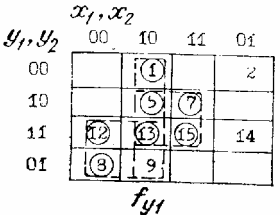
4. При объединении 16 номеров устраняются четыре переменные и вся функция будет равна 1.

Для иллюстрации рассмотрим минимизацию функции *Y1* табл. 5.3, заданной следующим набором обязательных **номеров** конституентов, при которых функция равна 1, и безразличных номеров (заключены в круглые скобки):

$$f_{Y_1} = \{1,\ 5,\ 7,\ 8,\ 12,\ 13,\ 15\,(2,\ 9,\ 14)\}.$$

В соответствии с расположением номеров табл.5.13,а запишем номера конституентов в таблицу, обозначив обязательные номера цифрами в кружках и безразличные—цифрами без кружков; запрещенные номера, при которых функция равна 0, будут соответствовать пустым клеткам (табл. **5.14).**

Таблица 5.14



Минимизацию начинаем с номеров, имеющих наименьшее число соседних номеров конституентов. Такими номерами являются 1, 7, 8 и 12, так как каждый из них имеет два соседних номера. Номер 1 соседний с номерами 5 и 9, номер 7— с 5 и 15, номер - 8 — с 12 и 9 и номер 12 — с 13 и 8. Если начать объединение с номера 7, то его **можно** объединить с номером 5 или 15. **Объедия**я номера 5 и 7, видим, что соседней с ними будет пара номеров 13 и 15. Следовательно, максимальной группой, образующей квадрат, является объединение, состоящее из номеров 5, 7, 13, 15. Номер 1 объединяется с номерами 5, 13, 9, образующими группу, расположенную в одном столбце; номера 12, 13, 8, 9 образуют квадрат.

У первого объединения пара номеров 5 и 7 расположена в одной строке, где *y1=1* и *y2=0,* но каждый из номеров находится в разных столбцах. Номер 7 находится в столбце, где ***x1==1, x2***=1, а номер 5—в столбце, где *xi=1* и *X2=0.* Следовательно, при сложении 5 и 7 исчезнет переменная *X2,* имеющая разное значение.

Пара номеров 13 и 15 расположена в строке, где *y1=1* и *y2=1,* и в тех же столбцах, что и номера 5 и 7 Следовательно, при объединении номеров 13 и 15 также исчезнет переменная *x2.*

Эти пары, расположенные в разных строках, отличаются значением Y2. Поэтому эта переменная не войдет в общую конъюнкцию. Последняя будет состоять из переменных, общих для обеих пар. Такими переменными являются *X1* и *y1.* Общая конъюнкция представляет собой логическое произведение X1Y1.

Проделаем операцию объединения номеров в алгебраическом виде. Для этого запишем каждый из номеров конституентов в совершенной нормальной форме. Номер 5 расположен во втором столбце и второй строке, где X1==1, *X2=0, y1=1* и Y2=0. Поэтому конъюнкция, соответствующая этому номеру, будет равна *x1x2y1y2.*
Аналогичным образом запишем остальные конституенты.

Объединение номеров 5, 7, 13, 15 запишем как логическую сумму конъюнкций отдельных номеров. Используя законы алгебры Буля, произведем упрощение (минимизацию) функции

$$f'_{Y_1} = x_1\bar{x}_2\,y_1\,\bar{y}_2 \vee x_1\,x_2\,y_1\,\bar{y}_2 \vee x_1\,\bar{x}_2\,y_1\,y_2 \vee x_1\,x_2\,y_1\,y_2 =$$
$$= x_1\,y_1\,\bar{y}_2\,(x_2 \vee \bar{x}_2) \vee x_1\,y_1\,y_2\,(x_2 \vee \bar{x}_2) = x_1\,y_1\,\bar{y}_2 \vee x_1\,y_1\,y_2 =$$
$$= x_1\,y_1\,(y_2 \vee \bar{y}_2) = x_1\,y_1.$$

Объединяя номера 8, 9, 12, 13, получим *x^yг;* объединение номеров 1, 5, 9, 13 даст — *XiXz.* Следовательно, минимизированная функция будет иметь следующий вид:

fy1 == ***x1*** **x1** V x2Y2 V x1x2 ($^{15.1}$)

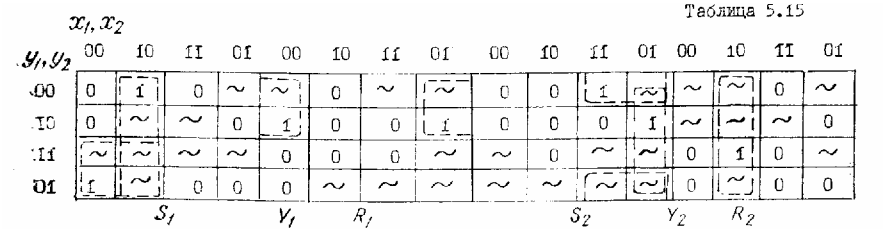
Аналогичным образом по табл. 5.4 получим функцию элемента памяти Уд

$$f_{Y_2} = \bar{x}_1\,x_2 \vee x_2\,\bar{y}_1 \vee x_2\,y_2 \vee \bar{x}_1\,y_2 = \bar{x}_1\,(x_2 \vee y_2) \vee x_2\,(\bar{y}_1 \vee y_2). \qquad (5.2)$$

Порядок расположения номеров в таблице зависит от принятого расположения столбцов, строк и весов элементов. Например, свойства таблицы не изменятся, если поменять местами первый и третий столбцы или строки, или второй и четвертый столбцы и такие же строки, или изменить веса элементов. Поэтому может существовать ряд таблиц, имеющих разное расположение номеров, **но** обладающих одинаковыми свойствами. Чтобы не зависеть от принятых весов элементов и расположения строк и столбцов, в таблицу заносят не номера конституентов, а их значения, т. е. 1, О или —.

При использовании в качестве ЭП триггеров типа *RS* для минимизации функций составляют таблицы для каждого из входов *R* и 5, как показано в табл. 5.15, составленной на основе табл. 5.5 и 5.6.

Таблица 5.15



В табл. 5.15 штрихом показаны объединения номеров состояний каждой из таблиц, приводящие к наиболее простой форме функций:

$$F_{S_1} = x_1\,\bar{x}_2 \vee \bar{x}_2\,y_2; \quad F_{R_1} = \bar{x}_1\,\bar{y}_2; \qquad (5.3)$$

$$F_{S_2} = x_2\,\bar{y}_1 \vee \bar{x}_1\,x_2; \quad F_{R_2} = x_1\,\bar{x}_2. \qquad (5.4)$$

Минимальная .форма функции /"л, получена за счет объединения двух пар номеров, расположенных в крайних столбцах, а функция FS2— при объединении двух пар номеров крайних строк и столбца.

Для автомата, использующего в качестве ЭП триггеры типа **Т,** минимизацию можно проводить непосредственно по таблице состояний ЭП. Из табл. 5.7 и 5.8 получим минимальную форму функций *Yi* и *Y^:*

$$F_{Y_1} = x_1 \bar{x}_2 \bar{y}_1 \vee \bar{x}_1 y_1 \bar{y}_2 \vee \bar{x}_2 \bar{y}_1 y_2; \quad (5.5)$$

$$F_{Y_2} = x_2 \bar{y}_1 \bar{y}_2 \vee \bar{x}_1 x_2 \bar{y}_2 \vee x_1 \bar{x}_2 y_2. \quad (5.6)$$

′ Для автомата, использующего триггер типа *D,* функции определяются уравнениями, аналогичными (5.1) и (5.2). Функции **для** элементов памяти на триггерах типа *JK* в нашем примере соответствуют уравнениям (5.3) и (5.4), хотя в ряде случаев они 'могут быть несколько проще за счет большего числа безразличных состояний.

Если состояние выходов задано табл. 5.12,в, то по ней заполняется табл. 5.16 для выходов Zi и *гч.*

Из этой таблицы получаем функции выходов

(5.7)

$$F_{z_1} = x_1 x_2 y_2; \quad F_{z_1} = x_2 y_2 \vee y_1.$$

Таблица 5.16

y_1y_2	x_1x_2 00	10	11	01	00	10	11	01
00	0	0	∽	∽	0	∽	0	∽
10	0	0	0	0	∽	1	1	∽
11	0	0	1	∽	1	1	1	∽
01	0	∽	1	0	1	∽	0	0
			z_1				z_2	

Рассмотренный метод минимизации позволяет получить функции в базисе И, ИЛИ, НЕ. Для перехода к другим базисам существуют определенные методы, позволяющие получить функции в любом **базисе.**

5.4. Синтез структур автомата в различных базисах

Схема автомата может быть построена на различных элементах: реле, диодах, транзисторах, интегральных схемах и т. д. У контактных элементов, к числу которых относятся нейтральные электромагнитные реле, каждая из переменных в функции соответствует контакту, а операция характеризует соединение контактов. Контакт на замыкание соответствует в функции переменной без инверсии, а контакт на размыкание — переменной с инверсией. Конъюнкция, или операция логического умножения, определяет последовательное включение, а операция дизъюнкции, или логического сложения, — параллельное соединение контактов. Элементы памяти образуются за счет обратных связей обмоток и контактов реле.

Если функции ЭП заданы уравнениями (5.1) и (5.2), а состояние выходов—уравнением (5.7), то принципиальная схема автомата будет иметь вид, показанный на рис. 5.1. Автомат состоит из приемной части, воспринимающей входное воздействие на входы л-, и *Хг,* промежуточной части, реализующей совместно с входными реле порядок работы автомата, и исполнительной части, формирующей сигналы на выходах *z1* и *Z2.*

На бесконтактных элементах, к которым относятся диоды, транзисторы, интегральные схемы, в зависимости от их свойств и возможностей различным образом может быть реализована схема автомата.

Сложность схемы автомата определяется числом операций, а значит и числом элементов, а также числом переменных, от которого зависит число входов элементов. При построении схемы у большинства бесконтактных элементов не допускается запараллеливание выходов, однако возможно запараллеливание входов в пределах возможностей нагрузочной способности элемента, подключенного к данным входам.

В схеме автомата, основными элементами которого являются И, ИЛИ, НЕ, операции дизъюнкции реализуются элементами ИЛИ, конъюнкции — элементами И и инверсии — НЕ.

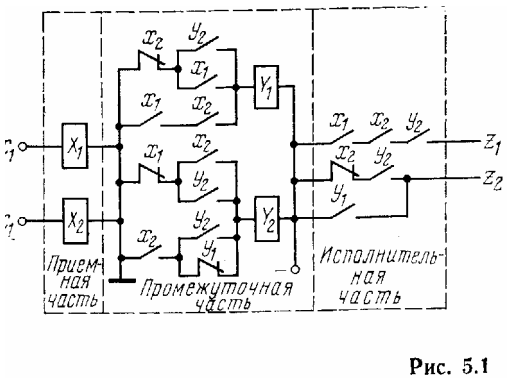


Рис. 5.1

Структурная схема автомата, построенного в соответствии с уравнениями (5.3), (5.4) и (5.7), показана на рис. 5.2.

Если основным элементом схемы автомата является элемент типа И-НЕ, то функции необходимо преобразовать к такому виду, чтобы в них отсутствовали операции ИЛИ и имелись только операции НЕ и

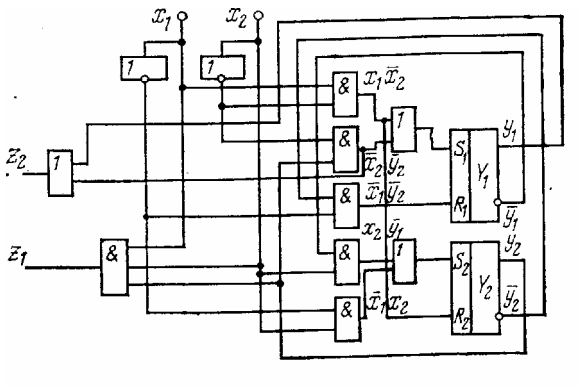


Рис. 5.2

И. Переход от базиса И, ИЛИ, НЕ к базису И-НЕ производится за счет двойной инверсии функции. Одну из операций инверсии используют для замены операции ИЛИ на И, а вторую — для инверсии всей функции.

Рассмотрим пример преобразования функции, заданной уравнением (5.5):

$$F_{Y_1} = \overline{\overline{F}_{Y_1}} = \overline{x_1 \bar{x}_2 \bar{y}_1 \vee \bar{x}_1 y_1 \bar{y}_2 \vee \bar{x}_2 \bar{y}_1 y_2} =$$

$$= \overline{(x_1 \bar{x}_2 \bar{y}_1)\,(\bar{x}_1 y_1 \bar{y}_2)\,(\bar{x}_2 \bar{y}_1 y_2)}.$$

Схема, реализующая данную функцию, приведена на рис. 5.3,а. Из рисунка видно, что первая группа схем И—НЕ реализует операцию логического умножения и инверсию каждой из конъюнкций, а вторая — логическое умножение конъюнкций и их инверсию.

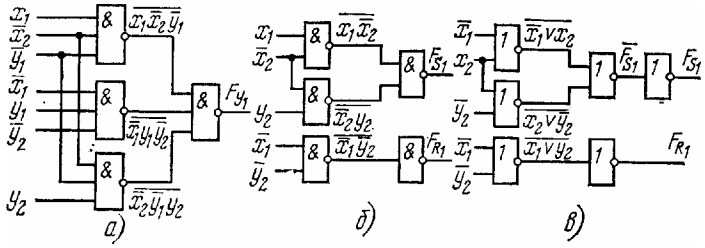


Рис. 5.3

Рассмотрим второй пример автомата, у которого функция задана уравнением (5.3). Произведя преобразование функций, получим:

$$F_{S_1} = \overline{\overline{F}_{S_1}} = x_1 \bar{x}_2 \vee \bar{x}_2 y_2 = \overline{\overline{(x_1 \bar{x}_2)\,(\bar{x}_2 y_2)}}; \quad \overline{\overline{F}_{R_1}} = \overline{\overline{x_1 \bar{y}_2}}.$$

Схема, соответствующая данным уравнениям, приведена на рис. 5.3,6. Из рис. 5.3,6 видно, что схема, соответствующая функции, содержит такое же число элементов, как и схема рис. 5.2, а схема, реализующая функцию Рд,, имеет еще дополнительный элемент НЕ.

Если основным элементом, используемым для реализации схемы автомата, является схема ИЛИ—НЕ, то для перехода от базиса И, ИЛИ, НЕ к базису ИЛИ—НЕ необходимо исключить все операции И, а функция должна содержать только операции ИЛИ и НЕ. Для этой цели, если функция задана в минимальной нормальной форме, производят двойную инверсию каждой конъюнкЦиг! и двойную инверсию всей функции. Одной из операций инверсии заменяют операцию И на ИЛИ, а второй — производят инверсию каждой конъюнкции. Двойная инверсия всей функции необходима для осуществления операции ИЛИ—НЕ между конъюнкциями.

В качестве примера рассмотрим преобразование (ФУНКЦИЙ (5.3):

$$F_{S_1} = \overline{\overline{x_1 \bar{x}_2 \vee \bar{x}_2 y_2}} = \overline{(\bar{x}_1 \vee x_2) \vee (x_2 \vee \bar{y}_2)}; \quad F_{R_1} = \overline{\overline{x_1 \bar{y}_2}} = \overline{x_1 \vee y_2}.$$

Схема, соответствующая данной функции, показана на рис. 5.3,в.

На рис. 5.3,в два элемента ИЛИ—НЕ в схеме F*s,* реализуют операцию логического сложения и инверсии каждой дизъюнкции, а третий элемент ИЛИ—НЕ — операцию логического сложения и инверсию обеих дизъюнкций. На выходе последнего элемента ИЛИ—НЕ получаем инверсное значение функции, а на выходе схемы НЕ — ее истинное значение.

5.5. Синтез структурной схемы микропрограммного дискретного устройства

При логическом проектировании микропрограммного ДУ наибольшие трудности возникают при синтезе центрального блока управления ЦБУ, так как функциональные блоки обычно довольно просты, а в ряде случаев являются стандартными. Например, в качестве логических функциональных блоков (ЛФБ) могут рассматриваться контакты удерживающих электромагнитов МКС, отмечающих занятость той'или иной вертикали. В качестве операторных ФБ (ОФБ) могут использоваться элементарные логические схемы И (клапаны), обмотки электромагнитов и т. п. Для нестандартного и неэлементарного ФБ (например, регистра или кодового приемопередатчика) могут быть применены рассмотренные выше общие методы синтеза.

Основная трудность синтеза ЦБУ состоит в том, что этот блок имеет большое число входов, достигающее в ряде случаев десятков и даже сотен, а практически невозможно использовать как формализованные языки типа таблиц переходов, так и общие ме тоды синтеза. Исследования показали, что ЦБУ целесообразно синтезировать в виде микропрограммного автомата, структурную схему

которого впервые предложили Уилкс (схема Уилкса) [3], а в качестве языка задания условий его работы использовать ЛСА.

В [3] показана взаимосвязь ЛСА и таблицы переходов, при этом дан способ перехода от ЛСА к таблице перехода. Следовательно, для ЦБУ принципиально возможно использовать общие методы синтеза, но, как указывалось выше, практическое их применение связано с большими трудностями. Поэтому рассмотрим ЦБУ, построенный по схеме микропрограммного автомата Уилкса (непосредственно по ЛСА).

Упрощенная структурная схема микропрограммного автомата Уилкса представлена на рис. 5.4. Схема состоит из регистра микрокоманд (РМК), дешифратора (Дш), матрицы внешних микроопераций (Mi), матрицы внутренних микроопераций (Мг) и матрицы формирования кода следующей микрокоманды (Мз).

Каждая внешняя микрооперация *zAi* является управляющим сигналом для ОФБ,, а внутренняя гр/ — управляющим сигналом для ЛФБ;, проверяющего логическое условие *p,.* Значение *p:* зависит от выполнения (pj=l; отмечено знаком «+») или певынолнения (pi=0; отмечено знаком «—») условия. На рис. 5.4 кружочком обозначены схемы, изображенные на рис. 5.5, которые служат для запроса от ЛФБ, значения проверяемого им логического условия *pj.*

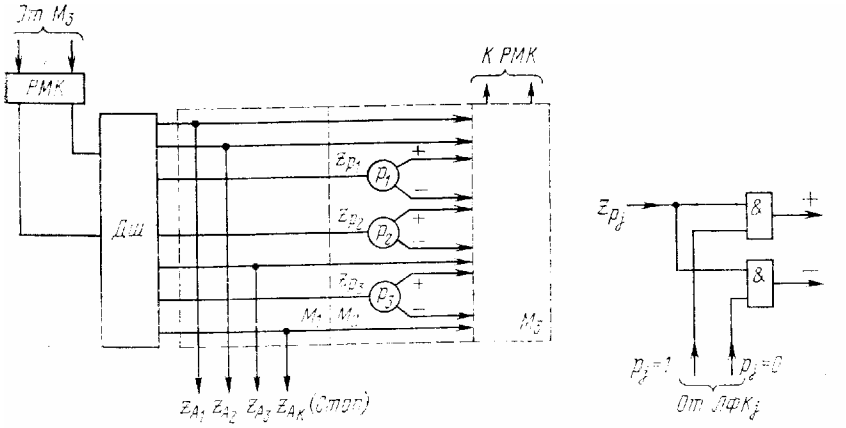


Рис. 5.4

Рис. 5.5

Выход гешифратора Дш сопоставляется с микрокомандой, выполняемой за один такт работы автомата. Код каждой микрокоманды хранится в РМК. Очевидно, число разрядов РМК опрепеляется числом микрокоманд, т. е. числом внутренних состояний МА После выполнения микрокоманды в Мз формируется код следующей микрокоманды, который передается в РМК.

Последовательность выполнения микрооперации удобно описывается на языке ЛСА, причем с оператором Л, сопоставляется внешняя микрооперация *гд,,* а с логическим условием р, — внутренняя микрооперация Zp/. Например, условия работы микропрограммного автомата, изображенного на рис. 5.4, описываются ЛСА:

U=v2A1A2p1^1p2^1A3p32v1 (5.8)

Легко видеть что если в каждую микрокоманду включить по одной микрооперации, то число микрокоманд будет равно числу чпенов ЛСА Однако в большинстве практических случаев за один микротакт может быть выполнено несколько микрооперации, т. е. в одну микрокоманду можно включить несколько микроопераций Метод формирования микрокоманд изложен в разд. 4.2. Рассмотрим процесс синтеза структуры МА по схеме Уилкса.

Пример. Пусть имеется система микрокоманд, полученная в примере 4.8. Необходимо построить структурную схему МА.

Решение Вначале определим разрядность РМК. Поскольку в системе имеется восемь микрокоманд, то РМК должен иметь три ЭП, в качестве которых используем RS-триггеры. Кодовые комбинации, определяющие состояния триггеров Г1—Гз, соответствующие этим микрокомандам, примем такими, как указано в табл. 4.15.

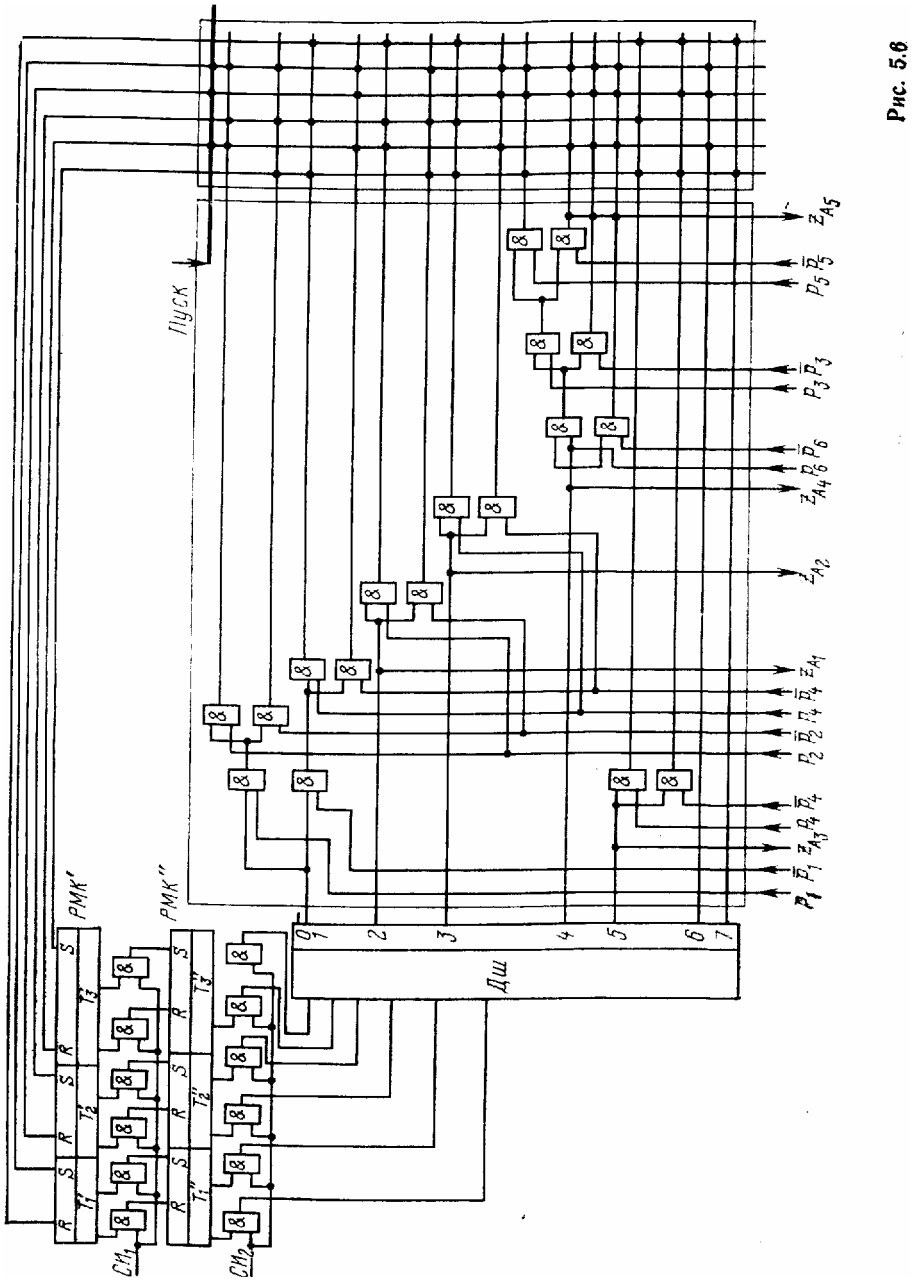


Рис. 5.6

Теперь указано в табл. 4.15. Теперь нетрудно построить принципиальную схему МА (рис. 5.6).

Сигналом, поступающим на вход Si, триггер *T,* переводится в состояние 1, а сигналом на входе *R*— в состояние 0. Для запуска ЦБУ необходимо включить Г;.

Таким образом, непосредственно по ЛСА можно легко построить схему ЦБУ. При этом видно, что чем меньше членов ЛСА, тем проще будет принципиальная схема ЦБУ. Следовательно, ЛСА необходимо предварительно минимизировать рассмотренными выше способами.

На рис. 5.6 показаны два регистра РМК/ и РМК", между которыми установлены схемы И обеспечивающие передачу информации в РМК, накопленной в коде следующей микрокоманды в РМК" во время выполнения текущей микрокоманды. Синхронизирующие импульсы СИ] и СИз находятся в противофазе, т. е. при наличии импульса СНi осуществляется передача информации из РМК' в РМК", а при наличии импульса СИг выполняются микрокоманды и формируется в *РМ.К.'* код следующей микрокоманды.

Контрольные вопросы

1. Каковы принципы составления таблиц ЭП для электромагнитных реле?
2. В чем особенности синтеза дискретного устройства, в котором в качестве ЭП используются триггерные схемы?
3. Каковы особенности синтеза схем выходов дискретного устройства?
4. Чем может быть обеспечено отсутствие состязаний выходных сигналов?
5. В чем заключается табличный способ минимизации функций?

6. Как осуществляется переход от базиса И, ИЛИ, НЕ к другим базисам?
7. В чем состоят особенности синтеза микропрограммного автомата?

Глава 6.

СИНТЕЗ МИКРОЭЛЕКТРОННОГО ДИСКРЕТНОГО УСТРОЙСТВА В БАЗИСЕ БОЛЬШИХ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

6.1. Общие сведения

Как было отмечено в гл. 1, большие успехи микроэлектроники привели к созданию больших и сверхбольших интегральных микросхем (БИС и СБИС). При этом высокая и все возрастающая быстрыми темпами степень интеграции элементов на одном кристалле привела к необходимости использования при создании дискретных устройств настраиваемых (программируемых) БИС и СБИС: однородных сред и микропроцессоров*[* В связи с тем, что БИС и СБИС отличаются лишь количественными значениями создаваемых элементов в одном кристалле, в дальнейшем будем говорить только о БИС.]. К настоящему времени разработано достаточно большое число разновидностей однородных сред и микропроцессоров. Однако здесь описываются лишь те из них, которые являются наиболее характерными представителями имеющихся или перспективных однородных сред и микропроцессоров.

Среди однородных сред можно выделить три основных направления создания настраиваемых БИС: универсальные ячейки, программируемые логические матрицы и матричные однородные среды.

В данной главе рассматриваются особенности проектирования (синтеза) ДУ в базисе однородных сред, а в следующей — в базисе микропроцессоров.

В теории автоматов [3] в основном уделяется внимание методам реализации условий работы автомата в виде дискретного устройства (т. е. методам синтеза ДУ), при которых последнее реализуется непосредственно по структурному описанию автомата (см. гл. 1), т. е. по системе булевых функций [I]. Если же автомат задан в виде алгоритмического описания, то эти методы перед реализацией ДУ предполагают выполнение этапа логического синтеза, на котором заданное алгоритмическое описание автомата (например, в виде таблицы переходов, логической схемы алгоритма и т. п.) транслируется в структурное описание, т. е. В систему булевых функций, представленных в том или ином базисе логических элементов.

Будем называть методы реализации ДУ (аппаратные или программные) по структурному описанию автомата с выполнением этапа логического синтеза (если условия работы ДУ заданы в виде алгоритмического описания) *методами программно-структурного синтеза автоматов.* Эти методы получили широкое распространение, поскольку благодаря возможности применения в процессе синтеза ЭВМ позволяют получать в приемлемые сроки достаточно хорошие решения. Однако с ростом сложности дискретных устройств реализация этих методов даже при использовании ЭВМ становится все более затруднительной. В связи с этим представляет интерес возникший в последнее время в теории автоматов новый принцип построения дискретного устройства непосредственно по алгоритмическому описанию без трансляции в структурное описание. Такие методы будем называть *методами алгоритмического синтеза.* Эти методы, рассмотренные в ряде работ, дают возможность существенно упростить процесс проектирования дискретного устройства. Вместе с тем они учитывают особенность их реализации в базисе БИС, когда особое значение приобретает однородность структуры устройства, а не ее избыточность.

6.2. Синтез дискретного устройства в базисе универсальной ячейки

В ряде работ предлагается реализовать автомат, задаваемый на языке таблиц переходов, в виде дискретного устройства в базисе одной или двух типов универсальных ячеек (УЯ), сопоставляемых с одним состоянием автомата. При этом, когда используются две УЯ, одна из них сопоставляется с устойчивым, а вторая — с неустойчивым состояниями автомата, а переход от одного устойчивого состояния к другому через неустойчивое состояние моделируется непосредственным соединением соответствующих УЯ. Достаточно подробное изложение методов синтеза ДУ в базисе универсальных ячеек дано в [17]. Здесь рассмотрим метод алгоритмического синтеза автомата в базисе одного из типов УЯ [18], когда алгоритмическое описание автомата задано на языке ЛСА, а УЯ сопоставляется с внутренним состоянием автомата. Функциональная схема используемой ячейки представлена на **рис. 6.1.**

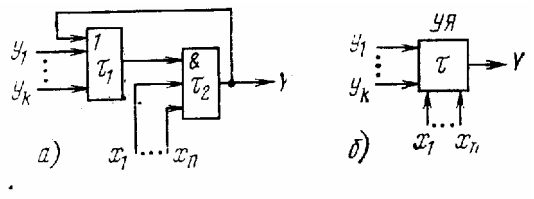


Рис. 6.1

Универсальная ячейка построена на логических элементах ИЛИ, И (рис. 6.1,а), имеющих естественные задержки Ti и Т2. Общая задержка сигнала с входа на выход УЯ составляет T=Ti+T2. Универсальная ячейка УЯг имеет два типа входов: *x^, ...,* JC'n, сопоставляемые с логическим условием p[1]], ..., *p^n,* и </[1]], ..., *y^,* сопоставляемые с выходами *Y^~\ ...,* V/i[1"1] предыдущих УЯJ-1. Каждая УЯ сопоставляется с микрокомандой, т. е. с внутренним состоянием МА. При этом если к микрокоманде Л4,, с которой сопоставлена УЯц есть переход от *M,, /*=1, ..., *k, с* которыми сопоставлены УЯj, j=1, ..., *k,* то выход У УЯ.) соединяется со входом *y]* УЯг. Если в микрокоманду *M,,* сопоставленную с УЯг, входит оператор Л;, то сигнал на выход za *i* дискретного устройства поступает с выхода УЯj, а с входами *x1, ..., Xi, l<=n,* УЯ сопоставляются переменные *pj, ..., pjl,* если оператор Ai выполняется при значениях ЛУ pi1, =1, ..., pil =1.

Поскольку при задании автомата на языке ЛСА каждому внутреннему состоянию соответствует микрокоманда, синтез автомата будем осуществлять непосредственно по системе микрокоманд. Примем, что в каждой микрокоманде имеется не более одного» оператора. Универсальные ячейки будем сопоставлять с микрокомандами. Определим условия, при которых с одной микрокомандой может быть сопоставлена одна УЯ.

Для удобства назовем оператор Ai, входящий в микрокоманду *Mi, начальным,* а оператор [Aj], к которому необходимо перейти после выполнения микрокоманды *Mi — конечным.* Так **как** с входами *Xi, ..., Xn* универсальной ячейки должна быть сопоставлена конъюнкция переменных *pi, ..., pn,* то автомат по системе микрокоманд может быть реализован в базисе УЯ тогда, когда выполняются следующие условия [18]:

1. В каждой микрокоманде имеется одна и только одна ветвь, заканчивающаяся начальным оператором.
2. В одной микрокоманде не должно быть более одной ветви с одним и тем же конечным оператором.
3. В микрокоманде оператор может быть или начальным, или конечным. В середине микрокоманды он не должен стоять.
4. Если оператор Л, является в микрокомандах конечным (т. е. [Aj]), то

а) $\bigvee_{i=1,\dots,k} \alpha_{ij} A_j = \tilde{p}_{r_1} \dots \tilde{p}_{r_s} A_j, \quad s \leqslant n,$

если *n* — число ЛУ в ЛСА;

б) для микрокоманды. Mi, в которой начальным оператором является *Ai,* должно быть выполнено равенство

$$\tilde{p}_{r_1} \dots \tilde{p}_{r_s} \cdot \tilde{p}_{l_1} \dots \tilde{p}_{l_l} = \tilde{p}_{i_1} \dots \tilde{p}_{i_m};$$

где pj1, ... *pr* получено в соответствии с п. а); *pil .... pli* соответствует ветви в Mj, заканчивающейся начальным оператором *Aj;*

m<=s+l.

Пример 6.1. Пусть заданы логическая схема алгоритма и система микрокоманд:

$$\mathfrak{A} = \downarrow^1 A_1 p_1 \uparrow^1 p_2 \uparrow^2 \downarrow^3 A_2 \bar{p}_2 \uparrow^3 \downarrow^2 A_3 p_2 \uparrow^2 p_1 \uparrow^1 \omega \uparrow^3 \qquad (6.1)$$

$$M_{A_1} = \left\{ \begin{array}{l} \\ A_1 \to p_1 \begin{array}{l} \overset{+}{\nearrow}[A_2] \\ p_2 \\ \underset{+}{\nearrow} \quad \underset{-}{\searrow}[A_3] \\ \underset{-}{\searrow}[A_1] \end{array} \end{array} \right\}; \quad M_{A_2} = \left\{ A_2 \to p_2 \begin{array}{l} \overset{+}{\nearrow}[A_2] \\ \underset{-}{\searrow}[A_3] \end{array} \right\};$$

$$M_{A_3} = \left\{ A_3 \to p_2 \begin{array}{l} \overset{+}{\nearrow}[A_2] \\ p_1 \\ \underset{+}{\nearrow} \quad \underset{-}{\searrow}[A_1] \\ \underset{-}{\searrow}[A_3] \end{array} \right\}.$$

Из системы микрокоманд видно, что первые три условия выполняются для всех микрокоманд. Легко убедиться, что выполняется и четвертое условие. Вначале проверим выполнение условия 4а: для конечного оператора *Ai*

$\bar{p}_1 A_1 \vee 0\, A_1 \vee p_2 \bar{p}_1 A_1 = \bar{p}_1 A_1 (1 \vee 0 \vee p_2) = \bar{p}_1 A_1$;

для конечного оператора A_2

$p_1 p_2 A_2 \vee p_2 A_2 \vee p_2 p_1 A_2 = p_2 A_2 (p_1 \vee 1 \vee p_1) = p_2 A_2$;

для конечного оператора A_3

$p_1 \bar{p}_2 A_3 \vee \bar{p}_2 A_3 \vee \bar{p}_2 A_3 = \bar{p}_2 A_3 (p_1 \vee 1 \vee 1) = \bar{p}_2 A_3$.

Теперь проверим выполнение условия п. 46: для начального оператора *Ai* в *Мо^* имеем

$\bar{p}_1 \cdot \bar{p}_1 = \bar{p}_1$;

для начального оператора Лз 'в *Мл,* имеем

$p_2 \cdot p_2 = p_2$;

для начального оператора *Аз* в *Мл^* имеем

$\bar{p}_2 \cdot \bar{p}_2 = \bar{p}_2$.

Таким образом, четвертое условие также выполняется.

Схема реализованного автомата в базисе УЯ изображена на рис. Ь.2. так видно из рисунка, выход уя1 соединен со входами УЯ2 и УЯз, так как после *Ма* возможен переход к *Ма2* и *Ма3.* При этом входами *Xi,* .... Xn уя1 являются pi и *p2,* так как переход МА, осуществляется от МА, при pi=0 и p2=1. В случае перехода МА во внутреннее состояние, соответствующее М^. он остается в нем до тех пор, пока значение pi=0 не сменится на pi=l. Аналогично во внутреннее состояние, соответствующее Мд,. микропрограммный автомат перейдет после *МА* (вход от уя1) и *Ма,* (вход от УЯз) при pi=l и p2=1. При этом МА останется во внутреннем состоянии, соответствующем *МА2* , до тех пор, пока значение *p2=1* не сменится на p2=0 (см. *Ма2).* Заметим, что на незадействованные входы *Xi, ..., Xn* должен постоянно подаваться единичный сигнал.

Таблица 6.1

	00	10	01	11	Выход	УЯ
1	(1)	3	(1)	2	z_{A_1}	$УЯ_1$
2	3	3	(2)	(2)	z_{A_2}	$УЯ_2$
3	(3)	(3)	1	2	z_{A_3}	$УЯ_3$

Для сравнения языка ЛСА с языком таблиц .переходов приведем запись условий рассматриваемого автомата на языке таблиц переходов (табл. 6.1, в которой указан номер 1-й ячейки, соответствующей i-му внутреннему состоянию автомата).

Таблица 6.2

p_1p_2	00	10	01	11	Выход	УЯ
1	3	(1)	3	2	z_{A_1}	$УЯ_1$
2	1	1	(2)	(2)	z_{A_2}	$УЯ_2$
3	(3)	1	(3)	1	z_{A_3}	$УЯ_3$

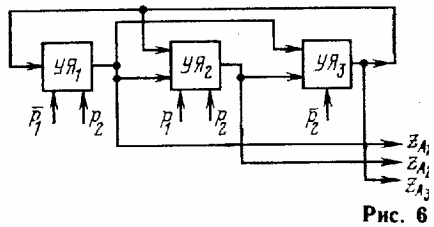


Рис. 6

Пример 6.2. Пусть заданы логическая схема алгоритма

$\mathfrak{U} = \downarrow^2 A_1 p_1 \uparrow^1 p_2 \uparrow^2 \downarrow^3 A_2 p_2 \uparrow^2 \omega \uparrow^3 \downarrow^1 A_3 p_1 \uparrow^1 \omega \uparrow^2$

и система микрокоманд:

$$M_1 = \left\{ \begin{array}{c} \overset{+}{\nearrow}[A_2] \\ p_2 \\ \overset{+}{\nearrow} \quad \overset{-}{\searrow} 1 \\ \downarrow^1 A_1 \rightarrow p_1 \\ \underset{-}{\searrow}[A_3] \end{array} \right\};$$

$$M_2 = \left\{ \begin{array}{c} \overset{+}{\nearrow} 1 \\ \downarrow^1 A_2 \rightarrow p_2 \\ \underset{-}{\searrow}[A_1] \end{array} \right\}; \quad M_3 = \left\{ \begin{array}{c} \overset{+}{\nearrow}[A_1] \\ \downarrow^1 A_3 \rightarrow p_1 \\ \underset{-}{\searrow} 1 \end{array} \right\}.$$

Из рассмотрения этой системы микрокоманд видно, что в ней выполняются три первых условия. Как и ранее, убедимся в выполнении четвертого условия:

для оператора Ai *(piVp1)p1p2=p1p2:*

для оператора *A2 p1p2-p2=p1p2;*

для оператора *A3 p1p1=p1.* Схема МА изображена на рис. 6.3, а таблица переходов — в табл. 6.2.

Следует заметить, что указанные выше первые два условия н сам метод синтеза обобщаются на случай наличия в микрокомандах в качестве начального и конечного членов не только операторов, но и логических условий.

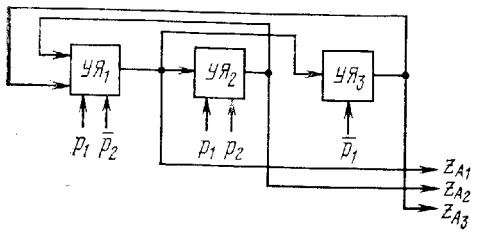


Рис. 6.3

Рассмотренная УЯ может применяться при синтезе ДУ лишь тогда, когда в каждом внутреннем состоянии автомата имеется хотя бы одно устойчивое состояние (т. с. в каждой микрокоманде должна быть ветвь, заканчивающаяся начальным членом). Если во внутреннем состоянии нет устойчивого состояния, что обычно бывает в микропрограммном автомате, то необходимо использовать второй тип УЯ, которую обозначим УЯ'. Эта ячейка (рис. 6.4) отличается от УЯ тем, что в цепи обратной связи (рис. 6.4,а)

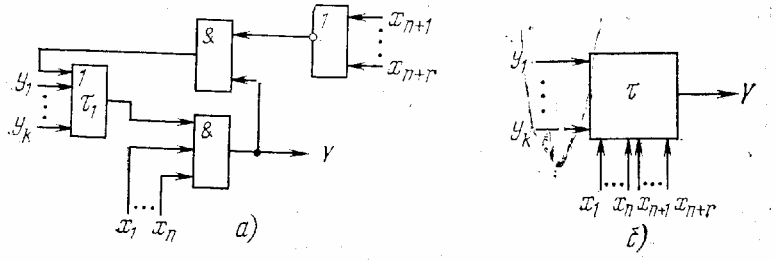


Рис. 6.4

введены элементы И с двумя входами и ИЛИ—НЕ с входами *Xn+i, ..., Xn+r.* Поэтому после включения ячейки (т. е. возникновения сигнала в цепи обратной связи) она будет находиться в этом состоянии до тех пор. пока на один пз входов *Xn+\, ..., Xn+г* не придет единичный сигнал. При появлении единичного сигнала сигналы на выходах элементов ИЛИ-НЕ и И станут нулевыми и, таким образом, обратная связь нарушится, ячейка перейдет в исходное (невключенное) состояние. Условное обозначение УЯ' приведено на рис. 6.4,6.

Пример 6.3. Пусть заданы ЛСА (3.9) и система микрокоманд, полученная по этой ЛСА в примере 4.7. Из системы микрокоманд видно, что в каждом внутреннем состоянии автомата нет устойчивых состояний, в микрокомандах *Мр* и *Мр* начальными членами являются не операторы, а логические условия. Конечными членами в микрокомандах *Mrj, Мр* и *Мр,* являются логические условия. Использование в качестве начальных и конечных членов микрокоманд логических условий — характерное свойство МА. При нахождении микропрограммного автомата в определенных внутренних состояниях на его вход могут поступать не все сигналы (значения ЛУ), а лншь некоторые из них. Например, из пяти логических устройств при выполеные микрокоманды *Мр* запрашиваются лишь значения *pi* и *рз.*

В рассматриваемой системе микрокоманд для *Мр* не выполняется второе условие, так как конечный оператор /-', входит к микрокоманду дважды. Поэтому данная система микрокоманд реализуется в базисе не УЯ, а УЯ'. Кроме того, эту систему необходимо преобразовать, оборвав в *Мр* ветвь на логическом условии *pt* н введя новую микрокоманду *Мр1.* В результате вместо одной микрокоманды *Мр* получим две следующие:

$$M'_{p_4} = \left\{ \begin{array}{l} \overset{+}{\nearrow}[p_2] \\ p_5 \\ \overset{+}{\nearrow}\ \overset{-}{\searrow}[F_j] \\ p_4 \\ \overset{-}{\searrow} A_3 \rightarrow [F_i] \end{array} \right\}; \quad M_{p_2} = \left\{ \begin{array}{l} \overset{+}{\nearrow}[A_1] \\ p_2 \\ \overset{-}{\searrow}[F_j] \end{array} \right\}$$

для которых второе условие выполняется.

Для микрокоманд *Мр , Мр* и *Мл*, не выполняется также третье условие, поэтому преобразуем их следующим образом:

$$M'_{p_1} = \left\{ \begin{array}{l} \overset{+}{\nearrow}[F_j] \\ p_3 \\ \overset{+}{\nearrow}\ \overset{-}{\searrow}[F_i] \\ \overset{-}{\searrow}[A_k] \end{array} \right\}; \quad M''_{p_4} = \left\{ \begin{array}{l} \overset{+}{\nearrow}[p_2] \\ p_5 \\ \overset{+}{\nearrow}\ \overset{-}{\searrow}[F_j] \\ p_4 \\ \overset{-}{\searrow}[A_3] \end{array} \right\};$$

$$M_{A_3} = \{A_3 \rightarrow [F_i]\}; \quad M'_{A_1} = \{A_1 \rightarrow [A_2]\}; \quad M_{A_2} = \{A_2 \rightarrow [F_i]\}$$

При этом учитываем, что микрокоманда *MFJ* уже имеется.

Теперь проверим выполнение четвертого условия:

для оператора *A0(Ak) pi;*

для логического условия *p4* 1 *[p4]* VI [P4] =$P^{4;}$
для оператора Fj *p1p3FjVp4p5Fj;*
для оператора A1 *p4p5p2A1;*
для оператора A2 1A2;
для оператора Аз *p4A3;*
для оператора Fj p1p3FiV1Fi;
для логического условия p2 p4p5[p2].

Как видно, для операторов *Fi* и *Fj* четвертое условие не выполняется. Поэтому заданную систему микрокоманд нельзя реализовать только в оазисе *уя.* Однако в тех случаях, когда добавляются логические элементы ИЛИ либо ФБ. реализующие операторы Fi и *Fj* имеют несколько входов, система микрокоманд может быть реализована вводом для каждого из операторов *Fi* и Fj нескольких УЯ' (рис. 6.5). Каждая из этих УЯ может быть выполнена в виде БИС или в базисе ИМС с малой интеграцией, например в базисе элементов ИМС серии 155.

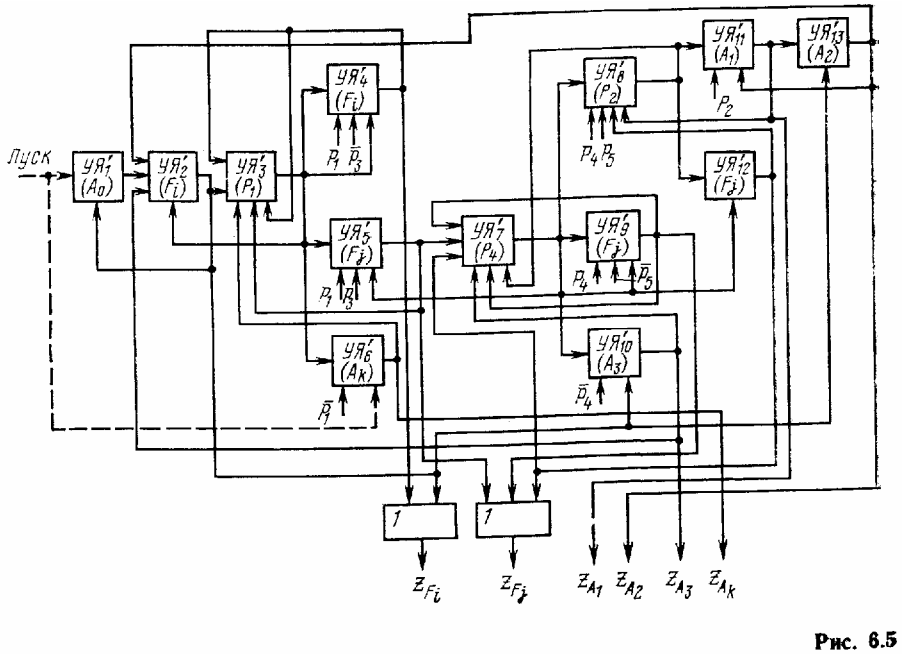


Рис. 6.5

Сложность (т. е. число корпусов) ИМС серии 155 как при стандартной реализации микропрограммного автомата в виде схемы Уилкса '(см. гл. 4), так при использовании УЯ' существенно зависит от алгоритма функционирования МА. В общем случае при синтезе МА наиболее предпочтительно использовать его стандартную реализацию. Однако в том случае, когда (см. примеры 6.*1* и 6.2) все или хотя бы большинство (более 75%) микрокоманд имеют ветви, заканчивающиеся начальными членами этих микрокоманд, может оказаться целесообразным использование базиса УЯ, особенно если выполняется четвертое условие.

При выборе элементного базиса необходимо иметь в виду, что применение БИС вместо малых ИМС сокращает число межкорпусных соединений (паек), а следовательно, повышает надежность проектируемого ДУ. С этой точки зрения при аппаратной реализации ДУ в качестве элементного базиса наиболее предпочтительно использовать выполненные в виде БИС *программируемые логические матрицы* (ПЛМ), которые освоены промышленностью, и перспективные БИС на основе *матричных однородных сред* (МОС).

6.3. Синтез дискретного устройства на программируемых логических матрицах

Программируемые логические матрицы [5, 21] можно рассматривать как один из наиболее простых типов однородных сред. При этом ПЛМ, представляющая собой БИС, создана на базе полупроводниковой технологии запоминающих устройств. Структурная схема простейшей ПЛМ (рис. 6.6,а) состоит из двух матриц М1 и М2. Условное обозначение ПЛМ приведено на рис.

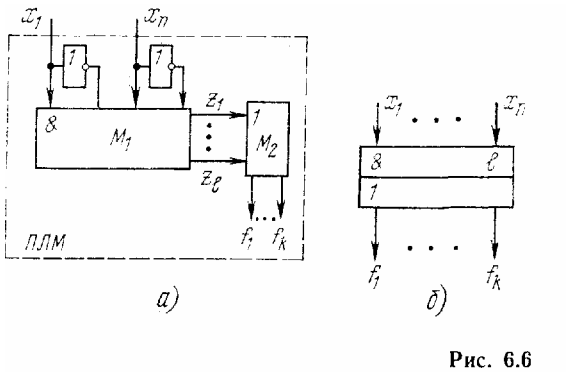


Рис. 6.6

6.6.б. Число входов X1, ..., *Xn* и выходов *f1, ..., fk* в ПЛМ ограничено и зависит от степени интеграции (т. е. от числа компонент) БИС. Инверсные входы *x1, ..., Xn* образуются непосредственно в буфере ПЛМ, включающем инверторы НЕ. В матрице *Mi* формируются / конъюнкций *Zi=f(x1, ..., Xn),* i=1, ..., l, от прямых и инверсных входных переменных, сопоставленных с прямыми и инверсными входами ПЛМ. Выходы z1, ..., *zl* матрицы *M1,* которым соответствуют образованные в *M1* конъюнкции, являются входами матрицы *M2,* в которой конъюнкции Z1, ... *Zl* логически складываются, образуя изъюнктивные нормальные формы (ДНФ) функций *f1, ..., fk,* описывающих сигналы на одноименных выходах матрицы *M2.* Иными словами, функционально матрица Mi представляет собой l многовходовых (2n-входовых) логических элементов И, а матрица *M2. — k* многовходовых (l-входовых) логических элементов ИЛИ (рис. 6.7). Однако в ПЛМ допускается настройка (программирование) и *M1,* и *M2* путем изменения состава входов в логические элементы как И, так и ИЛИ. Поэтому на ПЛМ может быть реализована любая система *k* булевых (переключательных) функций от *n* переменных, представленных в ДНФ. Учитывая вместе с тем, что реально в ПЛМ число выходов в *Mi* берется значительно меньше, чем возможное число конъюнкций от *n* переменных, т. е. /<2", на ПЛМ могут быть реализованы только такие ДНФ, в которых число конъюнкций не превышает /. Вместо логических элементов И в М] или ИЛИ в Afa могут применяться логические элементы ИЛИ—НЕ.

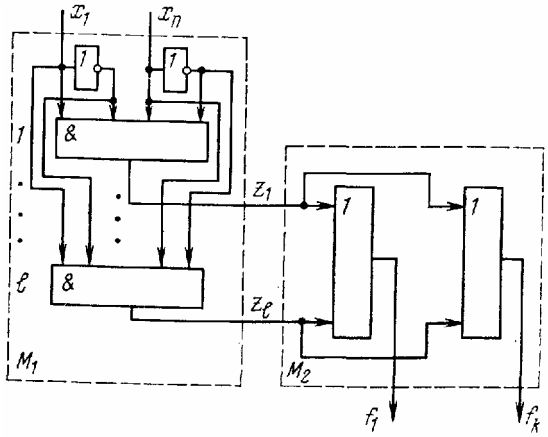


Рис. 6.7

Приведенную на рис. 6.6 ПЛМ можно представить в виде системы вертикальных и горизонтальных шин, в каждой точке пересечения которых стоит элемент (диод, транзистор и т. п.), соединяющих соответствующие горизонтальную и вертикальную шины (рис. 6.8). В зависимости от типа применяемых элементов различают биполярные ПЛМ и МОП (металл, окисел, полупроводник) ПЛМ. На рис. 6.9,а представлены возможные элементы в *M* и *M-г* биполярной ПЛМ, а на рис. 6.9,а — МОП ПЛМ.

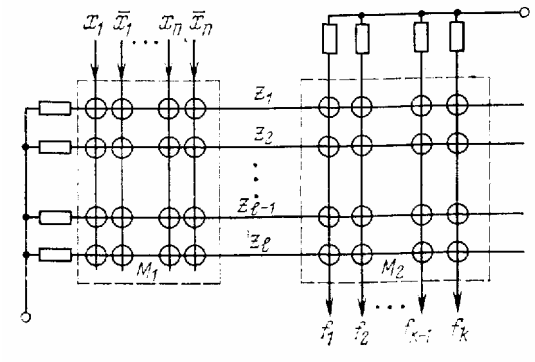
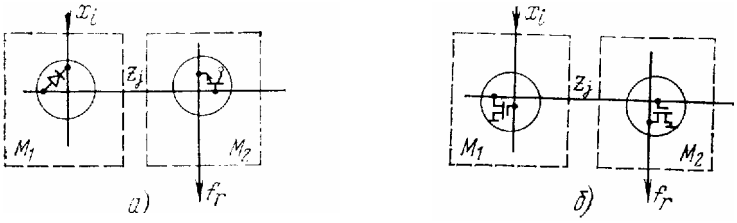


Рис. 6.8

Очевидно, прежде чем реализовать систему функций при использовании МОП ПЛМ, каждую из этих функции необходимо перевести в двухъярусную форму, аналогичную ДНФ, но использующую операцию ИЛИ—НЕ (функция Вебба) [I]. В зависимости от способа настройки (программирования) элементов матриц. *A'li* и *Ms* различают два основных типа ПЛМ: программируемые в процессе изготовления в программируемые пользователем.



В программируемых логических матрицах первого типа информация в матрицы (т. е. соединения вертикальных и горизонтальных шин) заносится в процессе их изготовления с помощью маски и в дальнейшем не может изменяться. В биполярных ПЛМ маска используется для подключения элементов к шинам матриц путем металлизации нужных участков схемы. В МОП ПЛМ ненужные (замаскированные) элементы не образуются.

Программируемые логические матрицы второго типа поставляются незапрограммироваиными. Информация в матрицы *Mi* и ЛЬ заносится пользователем с помощью специального оборудования. Таким образом, потребитель может сам программировать ПЛМ при создании на их базе тех или иных ДУ. При этом следует отметить, что среди ПЛМ второго типа имеются как однократно, так и многократно программируемые (т. с. допускающие перепрограммирование) .

Пример 6.4. Пусть задана следующая система представленных в совершенной дизъюнктивной нормальной форме (СДНФ) [1, 16] булевых функций. описывающая комбинационный автомат:

$$\begin{cases} f_1 = x_1 x_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3; \\ f_2 = x_1 \bar{x}_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_3 x_4 \vee x_1 x_2 x_3 x_4; \\ f_3 = x_1 x_2 \vee \bar{x}_1 \bar{x}_2. \end{cases}$$

Необходимо реализовать данную систему в базисе ПЛМ биполярнэгс типа.

Решение. Вначале необходимо минимизировать функции [1, 16], причем так, чтобы было минимальным число различных конъюнкции, так как число / выходов ПЛМ ограничено. Получение же минимального числа вхождений переменных в функции какого-либо значения не имеет, поскольку в *M,* каждой прямой и инверсной переменной все равно отведена одна вертикальная шина. Та<им образом, при минимизации системы функций необходимо стремиться получить не минимальные, а кратчайшие ДНФ, причем такие, которые имели *Сы* наибольшее число одинаковых конъюнкций. В результате минимизации задан.ной системы функций получаем следующие кратчайшие ДНФ:

f *f1* = *X1* X3 **V *X2 X3* ;**

{* f2 = *X1 X2 X4 V X1 **X3 X4 ;**

[f3 = X1 X2 V X1 X2

Прежде чем реализовать полученную систему ДНФ на ПЛМ, удобно составить сокращенную таблицу соответствия [1, 16] (табл. 6.3).

Т а б л и ц а 6.3

X1 X2 X3 X4	f1	f2	f3
1 X 1 X	1	0	0
X 0 0 X	1	0	0
1 0 X 1	0	1	0
1 X 1 1	0	1	0
1 1 X X	0	0	1
0 0 X X	0	0	1

3 табл. 6.3 вхождению прямой (без отрицания) переменной в конъюнкцию соответствует 1, а инверсной—0. Знак «X* обозначает, что соответствующая переменная в конъюнкцию не входит. От такой таблицы легко перейти к программируемой логической матрице (рис. 6.10), где точками указаны те элеменгь- матриц *M* и *Mt,* которые должны быть образованы в ПЛМ.

При использовании МОП ПЛМ от ДНФ необходимо перейти к функциям в базисе ИЛИ—ПЕ, а затем уже реализовать их на ПЛМ. Методы перехода от ДНФ к формам в базисе ИЛИ—НЕ изложены в [I], а более подробно — в i[19].

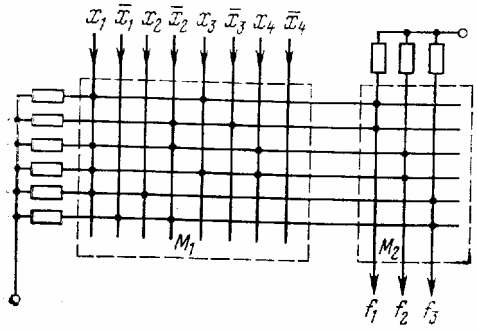


Рис. 6.10

Рассмотрим теперь реализацию в базисе ПЛМ автомата с памятью. В данном случае на ПЛМ реализуется логический преобразователь ЛП автомата (рис. 6.11), а для блока элементов памяти могут быть использованы любые типы триггеров, например *RS*-триггеры, имеющие два входа и два выхода (инверсный и прямой). При этом в систему функций, реализуемых на ПЛМ (матрицы *M1* и *M2),* войдут как функции, описывающие сигналы на выходах *Zi, ..., Zk* автомата, так и фукции выключения и вклгачения триггеров Yri. и Ysi, i=1, ..., m, от входных X1, ... *Xn* и внутренних y1, ... *yn* переменных.

В том случае, когда весь логический преобразователь автомата реализован на одной стандартной ПЛМ с ограниченными размерами, необходимо осуществить декомпозицию системы функций.

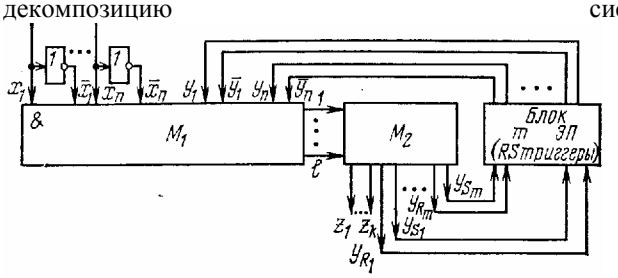


Рис. 6.11

В настоящее время существует ряд достаточно эффективных методов декомпозиции системы функций при их реализации на ПЛМ (см., например, [5]).

6.4. Синтез дискретного устройства в базисе МОС

Матричная, как и любая другая однородная среда [14, 20] характеризуется тем, что ее функциональные элементы (ФЭ) и связи между ними являются однотипными. Матричная однородная среда, относящаяся к классу сред с индивидуальной настройкой ФЭ, представляет собой прямоугольную итеративную решетку из *M* горизонтальных и *N* вертикальных шин (рис. 6.12,а). Функциональные элементы МОС, которые обычно называются *ячейками среды,* расположены на пересечении горизонтальных и вертикальных шин. Каждая из ячеек МОС подключена к одной горизонтальной и одной вертикальной шине, причем связь между шинами МОС возможна только через ячейки среды.

Каждая ячейка МОС может выполнять одну из следующих четырех одноместных операций (рис. 6.12,б):

О1 — передачу инвертированного сигнала с горизонтальной шины на вертикальную;

О2 — передачу инвертированного сигнала с вертикальной шины на горизонтальную;

О3 — передачу сигнала с горизонтальной шины на вертикальную;

04 — развязку между шинами.

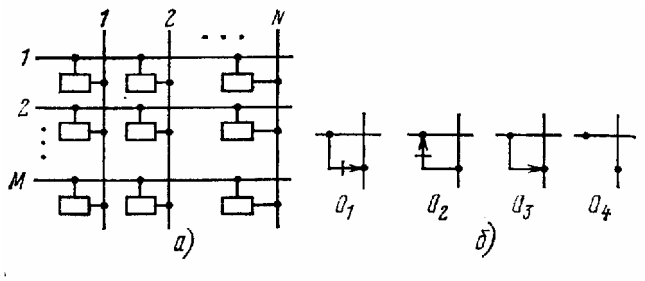


Рис. 6.12

При этом ячейка МОС вносит задержку т при передаче сигнала с одной шины на другую. Параллельное подключение выходов ячеек к горизонтальной (при операции *О2)* или вертикальной (при операциях О, и Оз) шине соответствует дизъюнкции переменных.

сопоставленных с этими выходами. С помощью внешних сигналов ячейка МОС настраивается (программируется) на реализацию одной из четырех операций О1, ..., О4. Настройка осуществляется

координатным методом одновременно всех ячеек. Показано, что в такой среде могут быть реализованы любая система булевых функцнй и любой автомат с памятью.

Пример 6.5. Пусть задана **система** из трех булевых функций, рассмотренная в примере 6.4. Необходимо ее реализовать на матричной однородной среде.

Решение. При синтезе системы булевых функций на МОС, так же как и при синтезе на ПЛМ, должны быть найдены кратчайшие ДНФ и выбраны такие кратчайшие формы, которые обеспечили бы наличие максимального числа одинаковых конъюнкций в различных ДНФ. Пусть после минимизации построена табл. 6.3. От табл. 6.3 легко перейти к настройке МОС (рис. 6.13). При этом входы и выходы автомата подключаются к горизонтальным шинам. На вертикальной шине образуется дизъюнкция переменных, соответствующих сигналам, снимаемым с выходов подключенных к данной шине ячеек. Вертикальная шина, на которой образована дизъюнкция А, переменных xji, ... *xjl . l<n* где n — общее число переменных во всех функциях, подключается к горизонтальной шине через инвертор. Поэтому если в конъюнкции ДНФ входит неременная без отрицания, то она подается на вертикальную шину через ячейку, настроенную на реализацию операции О1 (т. е. Эта переменная инвертируется). Если она входит в ДНФ без отрицания, то подается через ячейку, настроенную на реализацию операции Оз. Таким образом, в нашем случае получим для функции fI две дизъюнкции, образованные на первых двух вертикальных шинах: k1=x1Vx3 и *k2=x2Vx3.* После их инвертирования ячейками, настроенными на реализацию операции *О2* и их объединения на горизонтальной шине получаем *f1=k1\/k2=x1Vx3VX2VXз=x1X3Vx2xз.* Аналогично получена настройка МОС на реализацию функций f2 и f3: f2=k3Vk4=x1Vx2Vx4Vx1Vx3Vx4=***X1X2X4VX1X3X4; f3=k5Vk6=x1VX2VX1VX2=X1X2VX1X2.***

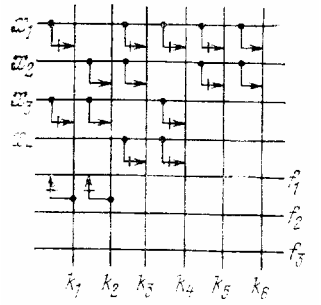


Рис. 6.13

Рассмотрим реализацию в МОС автоматов с памятью. Как и при использовании ПЛМ, для реализации автоматов с памятью в МОС логический преобразователь автомата можно построить в МОС, а в качестве ЭП использовать триггеры. Однако учитывая, что ячейка МОС имеет задержку т, ее можно представить в виде композиции элемента, реализующего логическую функцию f, и элемента задержки т (см. рис. 2.4,6). При этом автомат может рассматриваться как автомат с рапределенной задержкой (см. рис. 2.5).

Пример 6.6. Задана таблица включений, полученная в примере 4.1 (см. ркс. 4.1.в). Требуется реализовать автомат в базисе МОС с выделенным блоком памяти (БП), построенным на малогабаритных электромагнитных реле.

Решение. Припишем входным и внутренним переменным следующие веса: переменной x1 — вес 2^0=1; *x2* — вес 2^1=2 и у—вес 2^2=4 (на рис. 6.14 эти веса указаны в скобках). Подсчитаем номера конституентов и укажем их в соответствующих тактах таблицы включений. Теперь легко выписать в цифровом виде булевы функции, описывающие сигнал на выходе z и сигнал включения ЭП (реле) Y: Fz=5(2,6); *FY=3, 7,* 5 (2, 6). В скобках указаны номера 2 и 6 (в таблице включений не встречаются, поэтому их можно отнести к условным).

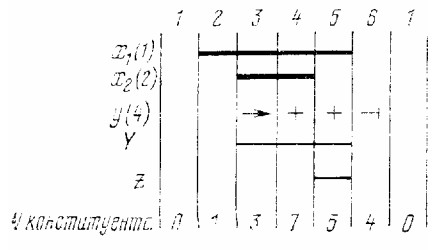


Рис. 6.14

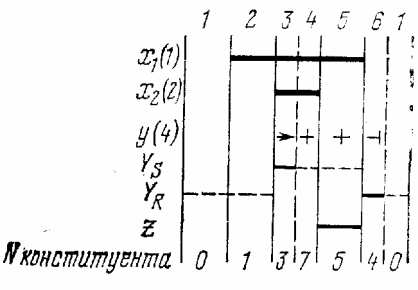


Рис. 6.17

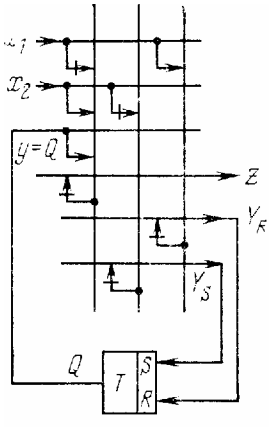


Рис. 6.18

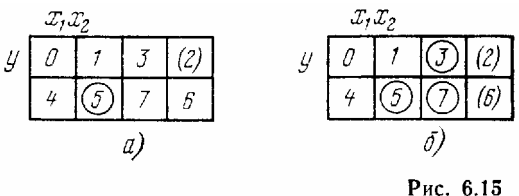


Рис. 6.15

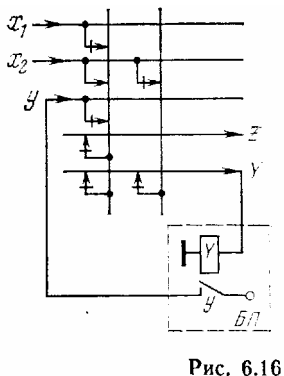


Рис. 6.16

Осуществим минимизацию этих функций, для чего можно использовать любой из известных методов (см. гл. 5 настоящего пособия, а также [1, 7, 17, 19]). Воспользуемся, например, методом карт Карно. Карта Карно для Fi представлена на рис. 6.15,а, а для *Fr*—на рис. 6.15,6. Исходя из карт Карно получаем, что Fx=*x1x2y; Fy=X2Vx1(x2/1)y.* По этим функциям настроим ячейки МОС и подключим реле к МОС (рис. 6.16). В функция *FY* целесообразно не исключать условную переменную *хг,* так как конъюнкция *x1x2y* имеется и в функции *FZ,* следовательно, на две одинаковые конъюнкции потребуется только одна вертикальная шина МОС.

Пример 6.7. Задана та же таблица включений, что и в примере 6.6. Требуется синтезировать автомат с использованием в блоке памяти *RS-ТРИГГЕРА*.

Решение. При использовании /?5-триггера необходимо иметь цепь включения триггера *Y—Fys и* цепь выключения триггера *У—FYR*- Поэтому вместо таблицы включений, изображенной на рис. 6.14, составим таблицу включений, представленную на рис. 6.17. Получим следующие функции включения: ^=5(2,6); *FYS=3(2,* 6, 7, 5); Fyr=4 (2, 6, 0, 1). После их минимизации *Fz=x1x2y; Fys=X2; Fyr=X1.*

Настройка МОС с БП приведена на рис. 6.18.

При реализации в МОС дискретного устройства в виде автомата с распределенной задержкой вместо ЭПг (триггера) вводится дополнительная вертикальная шина, к которой через ячейку, настроенную на О2, подключается шина Yi.

Пример 6.8. Автомат, условия работы которого сформулированы в примере 6.6, необходимо реализовать только на МОС в виде автомата с распределенной задержкой.

Решение. Используем кратчайшие ДНФ, полученные в примере 6.6: Fz=*X1X2y; Fy=x2Vx1X2Y.* По ним настроим ячейки МОС, введя дополнительную вертикальную шину *D.* На рис. 6.19 кружочками обведены ячейки *] и 2.* При этом ячейка *1,* настроенная на реализацию операции Оа, передает инвертированный сигнал с шины *Y* на шину *D,* а ячейка 2 — инвертированный сигнал с шины D на шину *y.* Таким образом, эти две ячейки, обеспечивая задержку в передаче сигнала с шины *У* на шину *y,* реализуют ЭП и образуют цепь обратной связи с выхода *Y* на вход *y* с задержкой 2т в логическом преобразователе автомата с распределенной задержкой.

Пример 6.9. Автомат задан таблицей перехода, полученной в примере 4.6 (см. табл. 4.11). Требуется синтезировать автомат в базисе МОС с выделен гым БП, в котором используются малогабаритные электромагнитные реле.

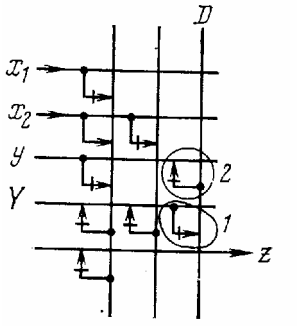


Рис. 6.19

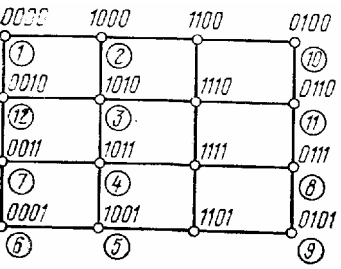


Рис. 6.21

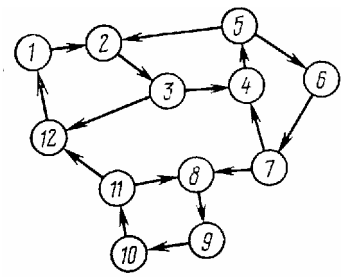


Рис. 6.20

-

Решение. Сначала необходимо закодировать внутренние состояния таким образом, чтобы избежать критических состязании ЭП. Наиболее простой способ устранения состязании ЭП состоит в выборе так называемого соседнего кодирования, при котором любой переход автомата из одного внутреннего состояния в другое сопровождается изменением только одного ЭП. Число s переменных y1, ... *y,* в

нашем случае не должно быть выше четырех (2s>12), поэтому для осуществления соседне
го кодирования удобно воспользоваться методом «натягивания» графа переходов па развертку четырехмерного куба.

По табл. 4.11 получаем граф переходов (рис. 6.20), который «натягиваем» на развертку четырехмерного куба (рис. 6.21). На рис. 6.21 в кружочках указаны номера внутренних состояний автомата. После этого по табл. 4.11 \'- рис. 6.21 строим кодированную таблицу переходов (табл. 6.4).

Таблица 6.4

x1 x2	10	11	01	ZiZ^gZiZs
У1У2У3У4 00				
0000 (0000)	1000	(0000)	(0000)	00000
1000 1010	(1000)	—	(1000)	10000
1010 (1010)	1011	(1010)	0010	10000
1011 1001	(1011)	—	(1011)	01000
1001 (1001)	0001	(1001)	1000	01000
0001 0011	(0001)	—	(0001)	00100
0011 (0011)	0111	;0011)	1011	00100
0111 0101	(0111)	—	(0111)	00010
0101 (0101)	0100	0101	0001	00010
0100 0110	(0100)	—	—	00001
0110 (0110)	0010	(0110)	Dili	00001
0010 0000	(0010)	—	(0010)	00000

По кодированной таблице можно составить булевы функции, описывающие как выходные сигналы (si, ..., 25), так и сигналы включения ЭП (y,, ... ^a). Приписав переменным веса: *Xi*—вес 1, *ху.*—вес 2, *у*\—ес 4, *у-г*—вес 8, ;,',. — вес 16 и г/4—вес 32,—запишем булевы функции в цифровом виде:

Fz1 = 4, 5, 6, 20, 21, 22, 23 (7);
Fz2 = 36, 37, 38, 39, 52, 53, 54 (55);
Fz3 = 32, 33, 35, 48, 49, 50, 51 (34);
Fz4 = 40, 41, 42, 43, 56, 57, 59 (58);
Fz5 = 8, 9, 24, 25, 26, 27 (10, 11);
Fy1 = 1, 4, 5, 6, 20, 21, 23, 36, 38, 39, 50, 52, 53, 54 (7, 55, 35, 59, 11, 10, 19);
Fy2 = 8, 9, 24, 26, 27, 40, 41, 43, 49, 56, 57, 58 (7, 10, 11, 19, 35, 55, 59);
FY3 = 4, 8, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 32, 48, 49, 50, 51, 53, 54, 57, 58
(7, 10, 11, 19, 35, 55, 59);
Fy4 = 21, 26, 32, 33, 34, 36, 37, 39, 40, 42, 43, 48, 49, 50, 51, 52, 53, 54, 56,
57, 58 (7, 10, 11, 19, 35, 55, 59).

Ко всем функциям можно добавить неиспользуемые наборы, которые соответствуют неиспользуемым наборам значении переменных *y1, ..., y4,* т. е. в качестве условных можно взять наборы 12, 13, 14, 15, 28, 29, 30, 31, 44, 45, 46^ 47, 60, 61, 62, 63. В связи с тем, что данные функции зависят от шести переменных, карты Карно применять неудобно. Поэтому необходимо использовать. другой метод, например метод Битсона [161. После минимизации получим следующие ДНФ, представленные в виде общего решения [1, 16]:

$$F_{z_1} = y_1 \bar{y}_2 \bar{y}_4;$$

$$F_{z_2} = y_1 \bar{y}_2 y_4;$$

$$F_{z_3} = \bar{y}_1 \bar{y}_2 y_4;$$

$$F_{z_4} = \bar{y}_1 y_2 y_4;$$

$$F_{z_5} = \bar{y}_1 y_2 \bar{y}_4;$$

$$F_{y_1} = x_1 \bar{x}_2 \bar{y}_2 \bar{y}_3 \bar{y}_4 \vee \bar{x}_1 y_1 \frac{\bar{y}_2}{1} \bar{y}_3 \vee \bar{x}_2 y_1 \bar{y}_2 y_3 \vee x_1 \frac{x_2}{1} y_1 \frac{\bar{y}_2}{1} \frac{y_3}{1} \bar{y}_4 \vee$$

$$\vee \frac{x_1}{1} x_2 y_1 \frac{\bar{y}_2}{1} \bar{y}_3 \frac{y_4}{1} \vee \bar{x}_1 x_2 \bar{y}_2 y_3 y_4;$$

$$F_{y_2} = \bar{x}_2 \frac{\bar{y}_1}{1} y_2 \bar{y}_3 \vee \bar{x}_1 \frac{\bar{y}_1}{1} y_2 y_3 \vee \frac{x_1}{1} x_2 \frac{\bar{y}_1}{1} y_2 \frac{y_3}{1} \bar{y}_4 \vee x_1 \bar{y}_1 y_2 y_4 \vee$$

$$\vee x_1 \bar{x}_2 \bar{y}_1 \frac{\bar{y}_2}{1} y_3 y_4;$$

$$F_{y_3} = x_1 \bar{x}_2 \bar{y}_1 \bar{y}_2 \frac{\bar{y}_3}{1} \bar{y}_4 \vee \bar{x}_1 \frac{\bar{y}_1}{1} y_2 \bar{y}_4 \vee y_1 \frac{\bar{y}_2}{1} y_3 \bar{y}_4 \vee \bar{y}_1 y_3 \vee$$

$$\vee \bar{x}_1 \bar{x}_2 \bar{y}_1 \bar{y}_2 \bar{y}_4 \vee \bar{y}_1 \bar{y}_2 y_3 y_4;$$

$$F_{y_4} = x_1 \bar{x}_2 \frac{y_1}{1} \bar{y}_2 y_3 \vee \bar{x}_1 x_2 \frac{\bar{y}_1}{1} y_2 y_3 \vee \bar{x}_2 \bar{y}_2 y_3 \vee x_1 \bar{y}_1 y_4 \vee x_1 x_2 y_4.$$

Схема автомата представлена на рис. 6.22.

Пример 6.10. По условиям примера 6.9 построить схему автомата на МОС без использования выделенного БП (модель автомата с оаспределеннои задержкой).

Решение. Булевы функции в этом случае будут такими же, как и в примере 6.9. Отличие в решении будет состоять только в том, что в МОС должны быть введены четыре вертикальные шины для реализации памяти автомата (ри=. 6.23).

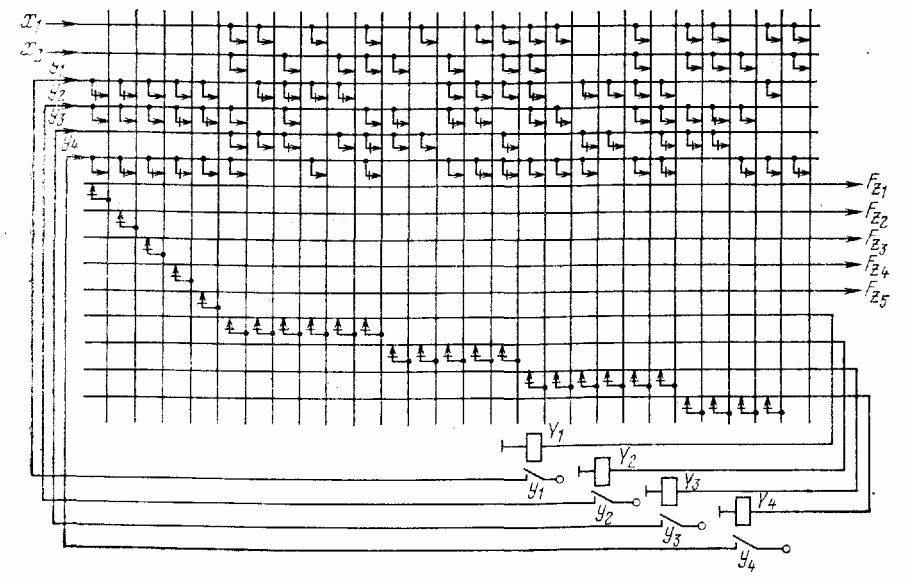


Рис. 6.22

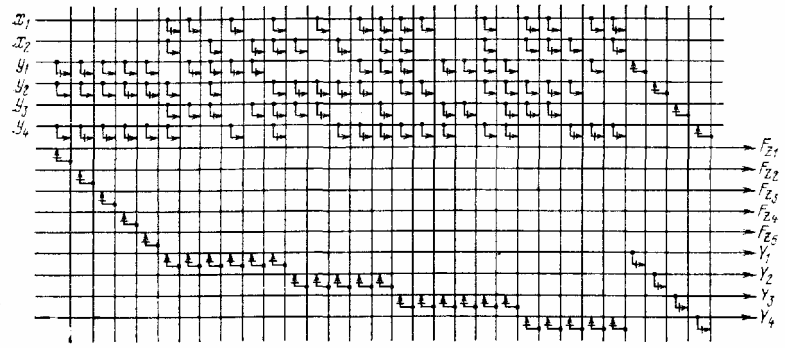


Рис. 6.23

в.5. Реализация микропрограммного автомата на МОС

Микропрограммный автомат, как и любой другой, может быть построен в матричной однородной среде. Однако учет особенностей алгоритма его функционирования и структуры [2, 3] позволяет упростить процесс построения МА и разработать канонический метод синтеза его структуры по логической схеме алгоритма.

Представим структуру МА в виде k функциональных схем ФС1, ..., ФСi, ..., ФСk, каждая из которых реализует одну микрокоманду, где *k* — число микрокоманд. Отведем каждой функциональной схеме lj вертикальных шин в МОС, где *lj*—число ветвей в j-й микрокоманде.

Каждой логической переменной *ра* , а=1, 2, ..., n, где n — число логических условий в ЛСЛ, отведем свою горизонтальную шину а; каждой выходной переменной (za)p , p=1, 2, ..., *F,* где *F*— число операторов в ЛСА,—свою горизонтальную шину (n+p), каждой внутренней переменной *y^,y--=\, 2,* ..., 5, *S=]\ogsh[* — две горизонтальные шины: *(n+F+y)* и *(n+F+S+y).* На одной из них, а именно на *(n+F+y)* записано значение внутренней переменной Yy в i-такте; на другой—значение уу в (i+1)-м такте. Тогда для реализации МЛ используется участок среды размерностью *(lk+k) (n+F+2S+l),* где /== 2 /=i

Так как каждая микрокоманда в схеме Уилкса реализуется за два такта (см. гл. 5), то используемый участок МОС можно условно разбить на два. Один из них предусматривается для реализации ФС,, /=1, 2, ... *k,* другой—для передачи значения кода микрокоманды, выполняющейся в (t+1)-м такте, от горизонтальных шин *(S+n+F+y) к.* шинам *(n+F+y),* где **y=1, 2, ... S.**

Пример 6.11. Пусть алгоритм функционирования микропрограммного автомата задан в виде ЛСА

$$A_0\, p_1 \uparrow^1 p_2 \uparrow^2 A_1\, p_3 \uparrow^2 \downarrow^1 p_4 \uparrow^2 A_2 \uparrow^2 A_3\, A_h.$$

При этом получены следующие микрокоманды:

$$M_1 = \left\{ A_0 \to p_1 \begin{cases} + \nearrow [M_2] \\ - \searrow [M_3] \end{cases} \right\};$$

$$M_2 = \left\{ A_1 \to p_2 \begin{cases} + \nearrow [M_3] \\ - \searrow [M_5] \end{cases} \right\};$$

$$M_3 = \left\{ p_4 \begin{cases} + \nearrow [M_4] \\ - \searrow [M_5] \end{cases} \right\}$$

$$M_4 = \{A_2 \to [M_5]\};$$

$$M_5 = \{A_3 \to [M_6]\};$$

$$M_6 = \{A_h\}.$$

Закодируем микрокоманды (табл. 6.5). Число кодирующих переменных (т. е. ЭП), которое потребуется для этого, .S='|log2 6[-3.

Для реализации функциональных связей потребуется (10+6) =16 вертикальных и (9+3+3) =15 горизонтальных шин. На вертикальных шинах реализуется последовательно: ФС1 (первая—третья шины), ФС2 (четвертая, пятая шины), ФСз (шестая, седьмая шины), ФС4 (восьмая шина), ФС4 (девятая шина) и ФС6 (десятая шина). Вертикальные шины с 11-и по 16-ю входят в часть II и служат для перезаписи кода следующей микрокоманды. На горизонтальные шины последовательно подаются переменные

$$p_1, p_2, p_3, p_4, A_0, A_1, A_2, A_3, y_1, y_2, y_3, y_1^1, y_2^1, y_3^1, q.$$

Разберем подробно реализацию первой ветви первой микрокоманды, в результате которой автомат переходит к выполнению микрокоманды *Ms.* Исходя из заданной ЛСА запишем ветвь Z,i:

$$L_1 = A_0\, p_1\, p_2 \to [M_2].$$

Так как логические переменные *pi* и *рг* входят в ветвь L] без инверсии, то соответственно ячейки (оо1,1 и оо2.1 настроены на передачу инверсного сигнала с горизонтальной шины на вертикальную. Ячейки оо3,1 и оо4,1 настроены на развязку, так как переменные *р3* и р4 не входят в *L1.* Ячейка *(оо5,* i настроена на инверсную передачу с вертикальной шины на горизонтальную, поскольку

Таблица 6.5

Микрокоманда	y_1	y_2	y_3
M_1	0	0	0
M_2	0	0	1
M_3	0	1	0
M_4	0	1	1
M_5	1	0	0
M_6	1	0	1

А0 входит в Li, а ячейки (оо6, 1, ... оо8, 1 настроены на развязку, поскольку *Ai, ... A3* не входят в L1. Ячейки (оо9, 1, оо10, 1, оо11. 1 настроены на передачу прямого сигнала с горизонтальной шины на вертикальную, так как у1, у2, у3, *которыми* закодирована микрокоманда *M1* с входящей в нее *L1* равны 0 (см. табл. 6.1). После *L1* по условиям алгоритма функционирования должна выполняться микрокоманда *M2,* следовательно, из ячеек оо12, 1, оо13, 1, оо14, 1 только последняя настроена на инверсную передачу с вертикальной шины на горизонтальную, так как микрокоманда *M2* закодирована следующим образом: *y1=0,* у2= 0, у3=1.

Полная схема реализации микропрограммного автомата в МОС приведена на рис. 6.24. На вход *q* поступает сигнал от тактового генератора ТГ (см *гл.* 5).

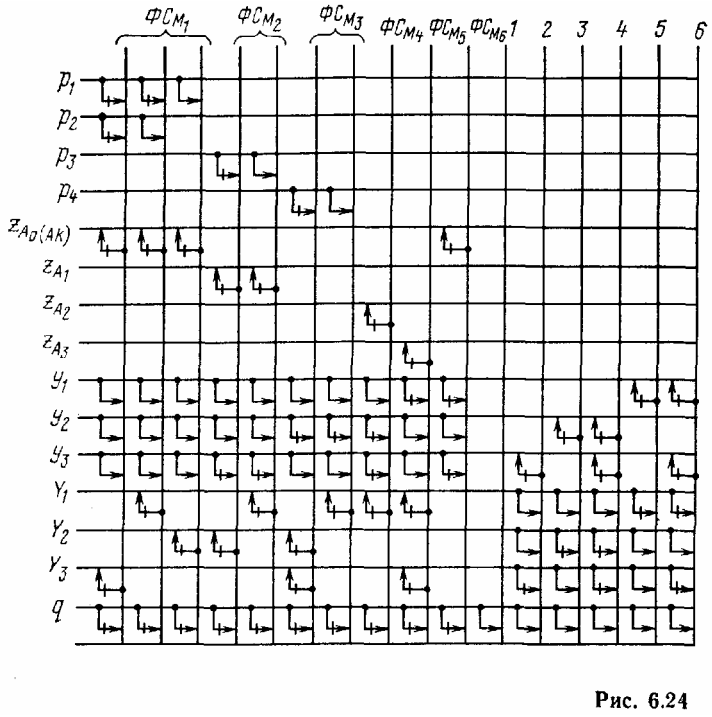


Рис. 6.24

Более подробно метод синтеза микропрограммного **автомата в** базисе МОС изложен в [22].

Контрольные вопросы

1. В чем состоит особенность проектирования дискретных устройств в базисе БИС?
2. Что гакое ПЛМ и каким обсазом можно реализовать дискретное устройство в базисе ПЛМ?
3. В чем состоит принципиальное отличие МОС от ПЛМ?

Глава 7.

СИНТЕЗ УПРАВЛЯЮЩИХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ МИКРОПРОЦЕССОРОВ

7.1. Принципы построения микропроцессоров и микропроцессорных систем

Микропроцессоры и микропроцессорные системы являются наиболее массовыми средствами вычислительной техники. Вместе с тем микропроцессоры из-за дешевизны и малых габаритных размеров их стали широко использовать в различного рода системах управления, в том числе в управляющих устройствах узлов коммутации.

Общепринятого строгого определения понятия «микропроцессор» пока нет, что объясняется быстрым развитием интегральной технологии и схемотехники. Обычно под *микропроцессором* (МП) понимают функционально законченное устройство, которое выполняет функции программной

обработки информации, аналогичные функциям процессора обычной ЭВМ, и реализуется на одной БИС или в виде модуля, содержащего несколько БИС [21, 23, 24].

На основе набора БИС, включающего БИС МП и ограниченное число других типов БИС, строятся более .полные системы программной обработки информации — микропроцессорные системы (МПС) и микро-ЭВМ.

Микропроцессорной системой часто называют любую вычислительную или управляющую систему, в которой в качестве процессорного блока используется микропроцессор [21]. В общем случае МПС может содержать несколько микропроцессоров.

В дальнейшем под микропроцессорной системой будем понимать систему, содержащую один микропроцессор. Системы, имеющие несколько МП, будем называть *многомикропроцессорными* (ММПС). В настоящее время нет четкого разграничения между понятиями МПС и микро-ЭВМ. Однако под микро-ЭВМ обычно понимают конструктивно завершенную МПС, оформленную в виде автономного вычислительного устройства с собственным источником питания, интерфейсом ввода — ывода и комплексом программного обеспечения [21, 23, 24].

В состав МПС, как правило, входят следующие основные функциональные блоки: микропроцессор МП, постоянное или полупостоянное запоминающее устройство (ПЗУ или ППЗУ), оперативное ЗУ (ОЗУ), генератор тактовых сигналов (ТГ), блок приоритетного прерывания (БПП), схемы внешнего интерфейса, обеспечивающие сопряжение МИС с периферийными устройствами (интерфейс ввода— вывода ИВВ) и схемы внутреннего интерфейса (ВИ), обеспечивающие передачу данных, адресов и команд между функциональными блоками МПС.

При построении МПС обычно применяется 'магистральный принцип соединения их функциональных блоков. Согласно этому принципу блоки, реализованные в виде функционально законченных модулей, соединяются между собой одной или несколькими шинами (рис. 7.1). В последнем случае это, как правило, шины: данных (ШД), адресов (ША) и управляющих сигналов (ШУ).

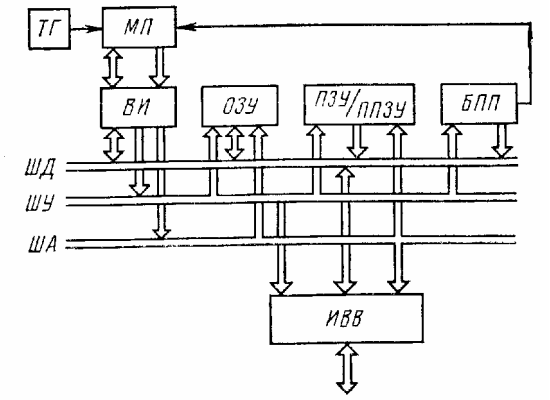


Рис. 7.1

В состав МП кроме основного блока, выполняющего функции собственно процессора и называемого *центральным процессором* (ЦП), часто входят устройства синхронизации и управления системой, а также некоторые вспомогательные схемы. Как ЦП, так и каждое из устройств, входящих в состав МП, выполняются в виде отдельных БИС или одной БИС, образуя IB последнем случае БИС микропроцессора.

Центральный процессор (рис. 7.2) содержит следующие основные функциональные блоки: арифметическо-логическое устройство (АЛУ), устройство управления (УУ), схему распределения тактовых сигналов (устройство синхронизации УС), блок внутренних регистров, который состоит из специальных регистров и регистров общего назначения (РОИ). К специальным регистрам относятся:
регистр команды (РК), аккумулятор (А), счетчик команд (программный счетчик ПС), индексные регистры (ИР), регистр состояния (регистр признаков результата РП), регистровое стэковое ЗУ (стэк) или указатель стэка (УкСт), буферные регистры данных и адресов (БРД и БРА).

Построение структуры центрального процессора также основано на .магистральном принципе соединения его функциональных блоков. Этот принцип, являющийся характерной особенностью архитектуры как ЦП, так и МПС *в* целом, реализуется в ЦП путем использования общей внутренней шины данных (см. рис. 7.2), разрядность которой, как правило, совпадает с разрядностью слов, обрабатываемых в АЛУ. Функциональные блоки микропроцессорной системы могут быть сгруппированы и реализованы с помощью БИС различными способами, что влияет на основные свойства этой системы. Набор совместимых БИС, требуемых для построения МПС и разрабатываемых специально для этой цели, составляет комплект или семейство микропроцессорных БИС (микропроцессорный комплект).

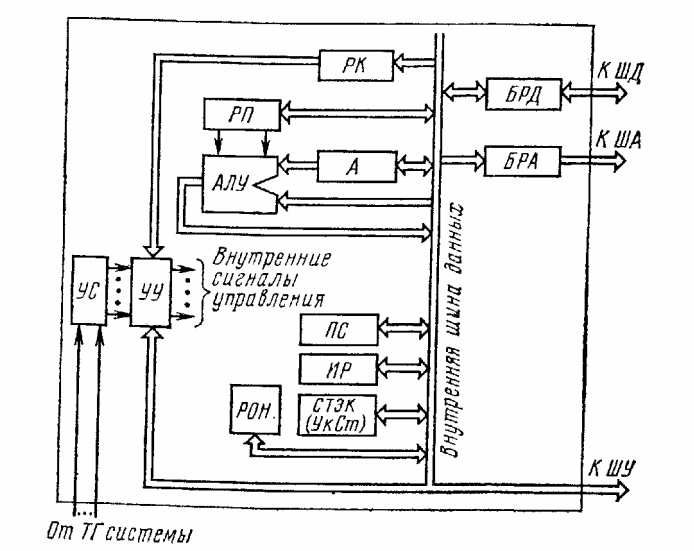


Рис. 7.2

Первый микропроцессор появился в 1971 г. (Gntel4040). В настоящее время известно уже около 100 различных типов МП, точнее микропроцесорных комплексов (наборов) МПК (МПН) БИС {21J, так как фирмы-изготовители .выпускают обычно наборы из нескольких БИС: МП, ОЗУ, ПЗУ, ППЗУ, ИВВ (порты [24]) и др. Гакое многообразие МП определяется различным сочетанием их характеристик.

Общими для всех типов МП являются следующие их основные характеристики и свойства [24]:

малая разрядность слова: 2, 4, 8, 12 или 16 бит;

ограниченная мощность набора команд (обычно требуются двепять команд для выполнения операции, эквивалентной одной команде мини-ЭВМ);

аппаратно подкрепленная организация связи подпрограмм (с помощью стека);

программно-управляемый ввод—вывод;

низкая стоимость микро-ЭВМ (обычно 1—5% стоимости мини-ЭВМ

7.2. Методы программной реализации дискретных устройств

В последнее время все большее распространение получает реализация ДУ на основе МП, МПС и ММПС. Методы программной реализации булевых функций известны довольно давно, по крайней мере, с тех пор, как появились языки программирования, позволяющие производить операции над булевыми переменными. Однако только с появлением микропроцессоров — этого широкодоступного и дешевого элементного базиса — началось широкое их практическое использование. При этом программно реализуя систему булевых функций, тем самым реализуем комбинационный автомат с памятью.

Большое внимание уделяется разработке простых и эффективных методов программной реализации, позволяющих экономично использовать ограниченные ресурсы микропроцессора (особенно объем памяти) и вычислять значения булевых функций за небольшое число шагов. В отличие от понятия сложности схем при схемной реализации управляющих устройств, которое, как правило, сводится к числу элементов в схеме, можно говорить, по крайней мере, о трех понятиях сложности для программ: числе команд в тексте вычислений;

объеме памяти для хранения промежуточных результатов вычислений;

быстродействии программы, т. е. числе команд, выполняемых в процессе вычисления.

Рассмотрим кратко четыре подхода к программной реализации булевых функций, получивших наибольшее распространение.

Непосредственное вычисление булевых функций.

Рассмотрим для примера булеву функцию

$$f = x(\bar{y}z \vee y\bar{z}) \vee \bar{x}yz. \qquad (7.1)$$

Существует несколько способов непосредственного вычисления этой функции. Так, можно предложить следующую процедуру. Вычислить:

1) логическое произведение *у-z* и занести результат в ячейку **Л;**

2) логическое произведение *у-г* и занести результат в ячейку *В;*

3) логическую сумму содержимого ячеек Л и В, занести результат в ячейку Л;

4) логическое произведение *x*- (содержимое ячейки Л) и занести результат в ячейку Л;

5) логическое произведение *x*-*y* и занести результат в ячейку ***B;***

6) логическое произведение *z*- (содержимое ячейки *B)* и занести результат в ячейку *B;*

7) логическую сумму содержимого ячеек Л и В и занести результат (значение искомой функции) .в ячейку *B.*

При реализации вычисления этой булевой функции на основе МПС необходимо иметь программу в памяти системы, реализующую данную вычислительную процедуру. Эта программа должна быть составлена в системе команд используемого МП. Система команд наиболее распространенного отечественного однокристального 8-разрядного микропроцессора К-580 ИК-80, который входит в МПК серии К-580, приведена в [21].

Как нетрудно понять, сложность программы и время вычисления функции существенно зависят от формы ее представления. При этом следует заметить, что, хотя скобочная форма булевой функции, 'например вида (7.1), позволяет упростить вычислительную процедуру, в практике программной реализации она не получила достаточно широкого распространения. Более часто используются дизъюнктивные нормальные формы. Кроме того, необходимо иметь программу ввода исходных данных через входные порты в виде набора значений переменных *x, y, г* и программу вывода через выходные порты результатов в виде значения функции *f.* Входные и выходные порты в МПС реализуются в виде отдельных интегральных микросхем с определенным числом входов и выходов. Поэтому если число входных переменных в булевой функции превышает число входов во втором порту, то при проектировании МПС 'необходимо использовать несколько портов или организовать поочередную подачу на порт групп переменных, входящих в булеву функцию. Аналогично если число булевых функций в системе, соответствующей различным выходам комбинационного автомата или логического преобразователя автомата с памятью, превышает число выходов в выходном порту, то также необходимо использовать несколько портов или поочередно выводить значения групп функций.

Преимущества рассмотренного подхода к вычислению булевых функций в основном сводится к его простоте и наглядности. Но, как видно, с ростом числа входных переменных существенно величиваются как длина программы, так и время ее выполнения. Кроме того, при переходе к вычислению новой или частично измененной функции требуются составление и отладка новой программы, что приводит к дополнительным затратам. Наконец, при применении многоразрядных микропроцессоров (8- или 16-разрядных) нерационально используется их память, так как под одну булеву переменную приходится отводить полностью одно машинное слово.

Исходя из этого данный подход к программной реализации булевых функций применяется для вычисления относительно простых и зависящих от небольшого числа переменных булевых функций.

Метод отображения входного набора. Рассмотрим функцию

$$f=wxy\backslash jz \qquad (7.2)$$

Положим, что требуется вычислить значение этой функции навходном наборе переменных *w, y, x, z,* равном 1000. Будем считать, что такой входной набор определяет входное слово **1000,:**

Функция (7.2) включает в себя две конъюнкции: *wyx* и *z.* Если прямому вхождению переменной в конъюнкцию поставить в соответствие 1, а инверсному вхождению—0, то конъюнкции *wyx* будет соответствовать набор ОЮх, а конъюнкции *z*—набор ххх1. Символом «х» обозначены переменные, не входящие в рассматриваемую конъюнкцию. Построенные таким образом наборы далее будем называть словами-конъюнкциями. Очевидно, что если входное слово совпадает по всем разрядам с одним из слов конъюнкций рассматриваемой функции, то значение функции на данном входном наборе будет равно 1. В противном случае функция равна 0. Если конъюнкция образована не всеми переменными, то сравниваются только те разряды слова-конъюнкции, которые не содержат символа «х».

Такой подход положен в основу программного вычисления булевых функций по методу отображения входного набора. Поскольку в этом случае время вычисления булевых функций, а также объем занимаемой памяти будут пропорциональны числу конъюнкций іВ ДНФ реализуемой функции, то предварительно необходимо минимизировать искомую функцию, т. е. построить ее кратчайшую ДНФ (с минимальным числом конъюнкций).

Обратимся к реализации такого подхода. Как и ранее, в качестве искомой функции возьмем функцию (7.2). Будем считать, что входные переменные уже введены в память микропроцессора и упорядочены в одно входное слово, каждый разряд которого соответствует одной входной переменной. С каждым из слов-конъюнкций искомой функции сопоставляется маскирующее слово. В разрядах маскирующего слова записаны 1, если соответствующие разряды слова-'конъюнкции являются определенными, т. е. содержат 1 или 0. Если некоторый разряд слова-конъюнкции является неопределенным (т. е. содержит символ «х»), то в соответствующий разряд маскирующего слова записывается 0. Хранение маскирующих слов и слов-конъюнкций осуществляется в стеке. Пусть стек организован в оперативном ЗУ, а обращение к нему осуществляется через регистр указателя стека, в котором постоянно хранится адрес первого слова стека. По мере продвижения данных из стека в рабочие регистры микропроцессора содержимое регистра указателя стека автоматически увеличивается на единицу. Последним словом стека всегда является нулевое слово, т. е. слово, все разряды которого равны нулю.

Схема организации стека для данного примера приведена в табл. 7.1. Будем считать, что

микропроцессор оперирует 8-разрядными слова.ми, а число переменных, от которых зависит функция, равно четырем. Поэтому первые четыре разряда стека не используются, т. е. первые четыре разряда маскирующих слов будут нулевыми. Рабочими разрядами как маскирующих слов, так и словонъюнкций являются последние четыре разряда. Первым словом стека будет маскирующее слово для конъюнкции *wyx,* затем следует слово-конъюнкция и т. д.

Блок-схема программы вычисления функции (7.2) по методу

Таблица 7.1

Неиспользуемые разряды	Переменные				Применение
	w	*y*	*x*	*z*	
0 0 0 0	1	1	1	0	Первое слово стека
x x x x	0	1	0	x	
0 0 0 0	0	0	0	1	
x x x x	x	x	x	1	
0 0 0 0	0	0	0	0	Конец стека

отображения входного набора представлена на рис. 7.3. Первоначально осуществляется поразрядное сложение по модулю 2 разрядов входного слова и текущего слова-конъюнкции. Если соответствующие разряды этих двух слов совпадают, то разряд резуль

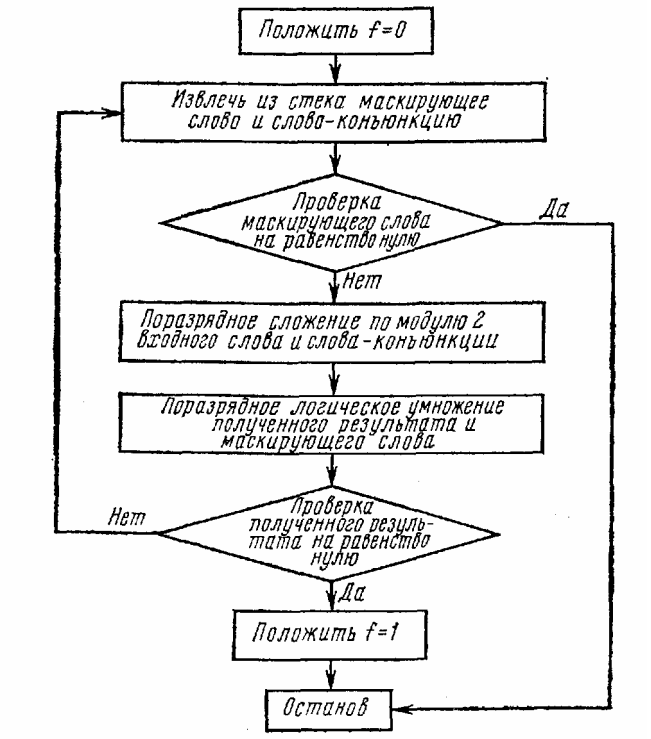


Рис. 7.3

тирующего слова будет нулевым. В случае несовпадения этих разрядов соответствующий разряд результирующего слова будет единичным. Для исключения влияния неопределенных разрядов в слове-конъюнкции производится поразрядное логическое умножение результирующего слова и соответствующего маскирующего слова. Если все разряды полученного при этом слова будут нулевыми, то функции присваивается единичное значение. При считывании из стека нулевого слова вычисления прекращаются.

В соответствии с блок-схемой программы, представленной на рис. 7.3, осуществляется программирование в системе команд используемого МП.

При использовании этого метода искомая функция будет равна единице только в том случае, когда наблюдается полное совпадение входного слова и одного из слов-конъюнкций в тех разрядах, для которых в маскирующем слове записаны единицы.

По сравнению с методом непосредственного вычисления булевых функций рассмотренный подход обладает рядом преимуществ. Во-первых, программа вычислений в значительной степени является универсальной и единственное изменение, которое требуется при переходе к реализации новой функции, сводится либо к изменению содержимого стека, либо к организации дополнительного стека и занесению адреса первого слова нового стека в регистр указателя стека. Во-вторых, данный метод является более быстродействующим (особенно при большом числе входных переменных), так как для вычисления функции требуется сравнение входного слова не со всеми словами-конъюнкциями, а до обнаружения первого совпадения. В-третьих, более рационально используется память микропроцессора за счет

размещения в одной ячейке нескольких (в данном случае восьми) входных переменных.

Недостатки данного подхода заключаются в возможной значительной длине стека при большом числе конъюнкций в ДНФ реализуемой функции, а также в необходимости сохранения содержимого регистра указателя стека при обращении к этому регистру системных программ (например, при обработке прерываний) или других программ пользователей.

Метод адресных переходов. В соответствии с этим методом осуществляется предварительное вычисление значений реализуемой функции на каждом из возможных входных наборов. Затем полученные значения заносятся в таблицу значений искомой функции, которая хранится в памяти микропроцессора [21]. Размещение таблицы значений в памяти осуществляется таким образом, что с каждым входным набором однозначно сопоставляется одна ячейка памяти, в которой хранится соответствующее значение функции. Так, если реализуется функция, зависящая от восьми переменных, то для хранения всех возможных значений функции потребуется 256 одноразрядных ячеек памяти или 32 восьмиразрядных слова. При этом адресация может быть организована таким образом, что первые пять разрядов входного набора будут определять адрес ячейки памяти, а оставшиеся три разряда — местоположение 'необходимого блока *в* выбранном слове значений функции.

В табл. 7.2 иллюстрируется возможный подход к вычислению функции (7.2) ло методу адресных переходов. Здесь с каждым входным набором сопоставляется адресное слово. Поскольку

Таблица 7.2

Адресное слово	Выходное слово	Адресное слово	Выходное слово
00000000	00000000	00001000	00000000
00000001	00000001	00001001	00000001
00000010	00000000	00001010	00000000
00000011	00000001	00001011	00000001
00000100	00000001	00001100	00000000
00000101	00000001	00001101	00000001
00000110	00000000	00001110	00000000
00000111	00000001	00001111	00000001

функция (7.2) зависит от четырех переменных, то первые четыре разряда адресных слов не используются и 'всегда равны нулю. Переменным *w, y, x* и *г* соответствуют пятый, шестой, седьмой и восьмой разряды адресного слова соответственно. В правом столбце таблицы приведены выходные слова, .последний разряд которых определяет значение функции.

На рис. 7.4 приведена блок-схема универсальной программы вычисления произвольной булевой функции от восьми входных переменных по методу адресных переходов. Таблица значений функции состоит из 32 восьмиразрядных слов; пять младших раз

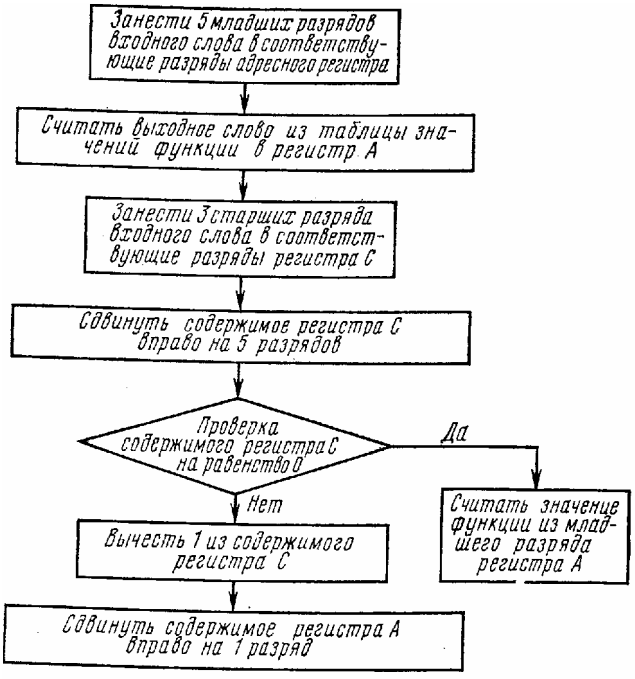


Рис. 7.

рядов адресного слова определяют местоположение строки таблицы значений, а три старших разряда — местоположение конкретного бита выходного слова, т. е. значение функции.

Важным преимуществом метода адресных переходов является его быстродействие, особенно при

вычислении систем булевых функций. Так, при использовании 8-разрядного микропроцессора по заданному входному набору можно одновременно определить значения до восьми .булевых функций, считая, что каждый разряд выходного слова соответствует значению одной булевой функции. К недостаткам метода можно отнести несколько больший, чем в случае метода отображения входного набора, объем памяти. В общем случае он будет пропорционален числу входных переменных и числу реализуемых функций. Как и предыдущий метод, метод адресных переходов отличается своей универсальностью, т. е. одна и та же программа может быть использована для вычисления различных функций. Изменения вносятся только в таблицу значений функции.

Сравнительные характеристики рассмотренных выше подходов к программной реализации булевых функций при использовании МПК К-580 приведены в табл. 7.3. Каждый из методов сравни-

Т а б л и ц а 7.3

Метод вычислений	Пример № 1		Приме р № 2		Приме р ,\» з	
	Объем Байт	*Т, мкс*	Объем байт	*Т,* мкс	Объем байт	*Т,* мкс
Непосредственное вычисление	20	40	—	____	____	____
Отображение вход-нзго набора	20	70	40	320	190	2600
	60	90	60	90	260	10

вался по двум параметрам—объему необходимой памяти и времени вычислений. Анализ производился для трех примеров. Первый пример относится к реализации функции (7.2). Второй пример соответствует реализации произвольной булевой функции от восьми входных переменных, ДНФ которой состоит из десяти конъюнкций. Третий пример относится к реализации произвольной системы из восьми булевых функций. Каждая из этих функций зависит от восьми входных переменных, а ее ДНФ содержит по десять конъюнкций.

Для метода непосредственного вычисления во втором и третьем примерах оценки объема необходимой памяти и времени вычислений резко возросли и их точное значение не определялось. Поскольку в третьем примере при применении метода адресных переходов операции по сдвигу выходного слова таблицы значений для каждой из булевых функций .не использовались, то быстродействие метода оказалось выше, чем при рассмотрении второго примера. Однако затраты памяти возросли более чем в 4 раза.

Из рассмотрения таблицы следует, что метод непосредственного вычисления почти не имеет никаких преимуществ даже при реализации относительно простых функций. Метод отображения входного набора характеризуется несколько меньшими затратами памяти, чем метод адресных переходов, однако имеет более низкое быстродействие, особенно при реализации систем 'булевых функций.

Таким образом, при выборе того или иного из рассмотренных выше трех методов программной реализации булевых функций не- обходимо исходить от предъявляемых к разрабатываемой системе требований по быстродействию, объему допустимых аппаратных затрат и соотношению между стоимостью разработки аппаратного и программного обеспечении системы управления.

Метод бинарных программ. Еще одним подходом к программной реализации булевых функций, получившим распространение в последние годы, является метод бинарных программ. Первоначально эти программы использовались для синтеза релейноконтактных схем, однако с широким распространением микропроцессоров наблюдается повторный интерес к применению бинарных программ для реализации булевых функций.

В общем случае бинарная программа представляет собой последовательность команд вида

i: ЕСЛИ *А* ТО / ИНАЧЕ *k,*

где i—порядковый номер команды; *А*—булева переменная, значение которой проверяется данной командой. При *А* = 1 осуществляется переход к выполнению команды с порядковым номером/, при Л=0—переход к выполнению команды с порядковым номером *k.*

Рассмотрим для примера реализацию булевой функции (7.1) с использованием бинарной программы:

1 ЕСЛИ *x* **ТО** 4 ИНАЧЕ 2
2 ЕСЛИ *y* **ТО 3** ИНАЧЕ 6
3 ЕСЛИ *z* ТО 7 ИНАЧЕ 6
4 ЕСЛИ *y* ТО 5 ИНАЧЕ 3
5 ЕСЛИ z ТО 6 ИНАЧЕ 7
6 /=0
7 *f*=\

Команды с порядковыми номерами от первого до пятого есть команды проверки значений булевых

переменных, а последние две команды—команды присвоения значения функции. Практическое использование метода бинарных программ в настоящее время ограничивается отсутствием эффективных методов по составлению и упрощению бинарных программ с целью минимизации объема занимаемой памяти микропроцессора.

7.3. Построение многомикропропессорных управляющих систем

Многомикропроцессорную систему можно рассматривать как распределенное дискретное устройство. Среди этих систем важное место занимают управляющие системы (ММУС), в которых микропроцессор или микропроцессорные модули (МПМ) непосредственно находятся на объекте управления (см. гл. 1).

Одним из наиболее ответственных этапов построения многомикропроцессорной управляющей системы (ММУС) является вы| бор ее структуры. Структура определяет производительность системы и пропускную способность каналов связи между отдельными ее блоками, а также между системой и блоками объекта управления (БОУ).

В настоящее время известен ряд типов .структур ММУС. Рассмотрим два наиболее характерных типа — структуру, построенную по децентрализованному принципу (см. рис. 1.1,я), и структуру распределенного типа (см. 1.1,6). Первый тип структуры эффективно используется 'в сосредоточенных ММУС, а второй — в распределенных, когда БОУ 'разнесены на достаточно большие расстояния.

Четкого определения распределенной ММУС в данное время нет. Однако во всех работах, где пытаются выявить отличия распределенной ММУС от сосредоточенной, основными параметрами, различающими эти системы, являются расстояния между МПМ и требуемый между ними обмен информацией. Например, считается, что ММУС относится к сосредоточенной, если расстояние между МПМ не превышает 0,1 км, а потребная скорость обмена данными должна быть более 10^6 бит/с. (В этих случаях говорят, что существуют сильные связи между МПМ.) Если же расстояние между МПМ превышает 0,1 км, а потребная скорость обмена между МПМ может быть менее 10^6 бит/с, то такие ММУС относят к распределенным. (Говорят, что существуют слабые связи между МПМ.) Указанное разделение, конечно, очень условно, но тем не менее оно позволяет как-то качественно оценить принадлежность ММУС к одному из двух типов и соответственно выбирать тип структуры системы.

Среди структур распределенных ММУС наибольший интерес представляют сети МП, в которых (рис. 7.5) в явном виде выделены функциональные микропроцессорные модули (ФМПМ), выполняющие частные алгоритмы функционирования системы по управлению процессами в объекте управления, и сеть связи ФМПМ, включающая каналы связи и коммутационные МПМ микропроцессорные модули (КМПМ). Сеть связи ФМПМ обеспечивает взаимосвязь ФМПМ в процессе выполнения ММУС своего алгоритма функционирования.

Структура сосредоточенных ММУС может быть централизованного и децентрализованного типов, построенных на основе ЦУУ.

Структура децентрализованной многомикропроцессорной управляющей системы.

Как отмечалось в гл. 2, алгоритм функционирования управляющего устройства УУ удобно представить в виде композиции частных алгоритмов функционирования отдельных этапов или фаз процесса управления (см., например, для узла коммутации [2]).

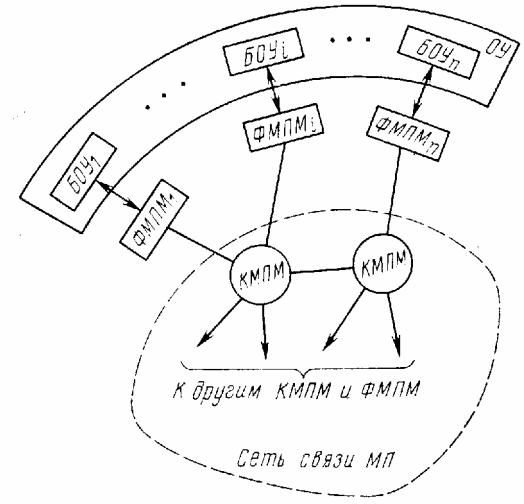


Рис. 7.5

При проектировании УУ в зависимости от технико-экономических показателей (капитальных и эксплуатационных затрат, простоты и удобства эксплуатации, надежности, быстродействия и т. д.) может быть принято одно из двух альтернативных решений о реализации частного алгоритма функционирования: в ЦУУ или в МПМ, находящихся непосредственно у БОУ. В дальнейшем программу МПМ и аппаратные средства, реализующие частный алгоритм ^*i,* будем называть *автономным функциональным блоком* (АФБ).

Когда частные алгоритмы функционирования реализуются в АФБ, яа ЦУУ возлагается задача по выполнению алгоритма над частными алгоритмами, т. е. координация работы АФБ. Однако в ЦУУ могут выполняться и некоторые частные алгоритмы функционирования, если по каким-либо причинам их реализация в АФБ оказывается менее предпочтительной.

После выполнения алгоритма функционирования 21{ вырабатывается сигнал прерывания для реализации частного алгоритма 2l,+i очередной фазы. При этом если *Щ* реализуется в АФБ, то в ЦУУ сигнал прерывания вырабатывает АФБ<. По этому сигналу, являющемуся по существу сигналом заявки на выполнение алгоритма 2t»-i-i, ЦУУ принимает решение о постановке указанного алгоритма в очередь на ожидание или о его реализации, возможно с прерыванием выполняемого в данный момент другого частного алгоритма. Если ^-*г*+\ должен выполняться в АФБ, то ЦУУ вырабатывает сигнал занятия и активизирует соответствующий АФБ. Если же SIj+i должен выполняться в ЦУУ, то активизируется соответствующая программа непосредственно в ЦУУ.

В качестве ЦУУ децентрализованного ММУС до последнего времени использовались мини-ЭВМ или машины более высокой производительности. Однако с появлением высокопроизводительных микропроцессоров и микро-ЭВМ вполне возможно использовать в качестве ЦУУ микро-ЭВМ и МПМ с высокопроизводительными микропроцессорами.

При управлении ЦУУ множеством параллельно выполняемых в ОУ технологических процессов в режиме разделения времени может оказаться, что из-за ограниченного быстродействия ЦУУ время осуществления технологического процесса окажется выше допустимого. Тогда можно или использовать более быстродействующее ЦУУ, или реализовать часть частных алгоритмов функционирования в АФБ.

Как первый, так .и второй пути решения давной задачи связаны с дополнительными затратами. Однако второй путь, как правило, оказывается более предпочтительным, так как переход к более быстродействующей модели ЦУУ, если это вообще возможно, обычно приводит к значительному удорожанию ММУС. Кроме того, перевод *Щ* в АФБ, сокращает объем управляющей информации, передаваемой между ЦУУ и АФБ, а следовательно, дополнительно уменьшает число соединений (проводность) ММУС.

Ниже описывается метод, позволяющий оптимально выбрать те частные алгоритмы функционирования, которые можно выполнять в АФБ, сохранив необходимую производительность ММУС без увеличения быстродействия ЦУУ при минимальных дополнительных затратах на введение АФБ.

При наиболее простой циклической дисциплине обслуживания поступивших заявок на выполнение частных алгоритмов реализация заявок осуществляется по принципу «первым пришел — первым обслужен». Если в момент начала обслуживания заявки на ^*i* не окажется свободного АФБ требуемого типа *i,* т. е. такого, где может быть выполнен 21{, заявка ставится в очередь на ожидание. Освободившиеся АФБ, занимаются для выполнения *Щ* в порядке поступления заявок в очередь.

Требуемая производительность ММУС задается величиной среднего допустимого времени выполнения технологического процесса Гдоп. Задача состоит в выборе такой структуры ММУС, которой соответствует при заданном быстродействии ЦУУ наименьшая суммарная стоимость всех АФБ, входящих в состав этой системы,

по сравнению с другими структурами, обеспечивающими заданную производительность. При этом предполагается, что задано максимальное число *N* одновременно выполняемых технологических процессов, которыми должна управлять проектируемая ММУС; допускается управление различными технологическими процессами на основе различных алгоритмов функционирования и известны вероятности *pj* выполнения каждого алгоритма функционирования ММУС 2Р, *j*=\, ..., *N,* р1+ ... +p^N=\, где *N*— число различных алгоритмов функционирования.

Для 'каждого алгоритма St^7 задана соответствующая матрица вероятностей перехода от одного частного алгоритма к другому. Известны средние длительности выполнения каждого частного алгоритма в ЦУУ—/, и в АФБ—г'дфБ^ , а также стоимость АФБ,—Сг,,
1=1, ..., от, где *m*—число различных частных алгоритмов во всех заданных алгоритмах функционирования.

Метод выбора структуры ММУС состоит из двух этапов. На первом этапе выбираются типы АФБ, создание которых может обеспечить требуемую производительность ММУС, а на втором этапе определяется число АФБ каждого типа.

Представим алгоритм функционирования 21.' в виде логической схемы алгоритма ЛСА, где в качестве i-го члена ЛСА 21' используется частный алгоритм 2t'^. Тогда все *N* алгоритмов 21', ..., а" представим в виде объединенной ЛСА 21 [3], а зная матрицы переходных вероятностей для каждого из объединяемых алгоритмов функционирования и вероятности их выполнения, можно получить матрицу переходных вероятностей *L* для объединенного алгоритма.

Решая матричное уравнение **aL=a,** определяем неподвижный вероятностный вектор а. Затем определяется *&',* элементы которого a'i= (1/ад)а, нормированы в предположении, что а\==1, где ад и *&'k*—элементы соответствующих векторов а и а', сопоставленные с оператором конца *Аъ* объединенной ЛСА 2t. Тогда среднее время однократного выполнения объединенного алгоритма 2t

$$T_{\text{ср}} = \sum_{i=1}^{m} \mathbf{a}'_i t_i, \qquad (7.3)$$

где **a',** есть среднее число выполнения частного алгоритма 21, при однократной реализации объединенного алгоритма 21. При этом заметим, что из условия объединения *N* алгоритмов функционирования [3] следует, что однократному выполнению объединенного алгоритма 21 соответствует однократное выполнение одного из объединяемых алгоритмов 21', /=1, ..., *N,* в зависимости от необходимости управления тем или иным технологическим процессом. В связи с этим в дальнейшем будем различать однократное выполнение объединениого алгоритма 21 и однократное выполнение объединенного алгоритма $2I^7$.

Если разница между средним и допустимым временем выполнения алгоритма функционирования, меньше или равна нулю, т. е,

$$B = T_{\text{ср}} - T_{\text{доп}} \leqslant 0,$$

то заданное быстродействие ЦУУ обеспечивает требуемую производительность ММУС и, следовательно, необходимости в создании АФБ нет. В этом случае микропроцессорные модули в ММУС либо выполняют функции по согласованию работы ЦУУ с БОУ, либо вообще не создаются, т. е. используется централизованный принцип построения ММУС.

Если Гср>7'доп, то быстродействие ЦУУ недостаточно и необходимо ввести АФБ, которые за счет параллельной работы с ЦУУ по выполнению алгоритмов функционирования обеспечили бы необходимую производительность ММУС.

Обозначим среднюю величину сокращения затрат времени ЦУУ на обслуживание алгоритма 21 при выполнении частного алгоритма фазы 2t; в АФБ, через *bi,* т. е.

$$b_i = a'_i (t_i - t'_i), \qquad (7.4)$$

где *t'i* — время занятости ЦУУ выполнением 21; при наличии АФБг (т. е. время, затрачиваемое ЦУУ на активизацию АФБ и восприятие сигнала прерывания от *АФБг* после окончания выполнения в нем частного алгоритма 21;). Тогда среднее значение сокращения времени занятости ЦУУ выполнением алгоритма 21 составит ^ *Б^,*

1^m.

Первый этап построения структуры ММУС состоит в выборе такого подмножества частных алгоритмов фаз, реализация которых в АФБ обеспечивает необходимую производительность ММУС, т. е. выполнение следующего неравенства:

$$\sum_{i=1}^{l} b_i \geqslant B. \qquad (7.5)$$

Назовем подмножество частных алгоритмов /^{^ц ..., ^i;}, *1'Sstn, допустимым подмножеством,* если удовлетворяется условие (7.5). *Минимально допустимым подмножеством* будет такое допустимое подмножество, исключение из которого какого-либо алгоритма приводит к нарушению неравенства (7.5).

Два подмножества *R{* и *Rj* назовем *сравнимыми,* если они содержат одинаковое число частных алгоритмов и каждому частному алгоритму 2t; s^, можно взаимно-однозначно поставить в соответствие частный алгоритм 2Г; »е/?„ так что при *y1^-^y^* выполняются условия

$$b_{i_\xi} \geqslant b_{j_\xi}, \quad c_{i_\xi} \leqslant c_{j_\xi}, \tag{7.6}$$

где *Cit(Cjt)*—стоимость одного АФБ, (АФБ,) и г/<Лг/„)—удельная нагрузка на одну группу АФБ^(АФБ^), т. е. нагрузка на АФБ.(„(АФБ„) при однократном выполнении 21.

При невыполнении условия (7.6) подмножества являются несравнимыми. В случае выполнения условия (7.6) для двух подмножеств *Ri* и *Rj* подмножеству *Ri* соответствует равноценная или более экономичная структура ММУС, чем подмножеству *Rj.* Поэтому будем говорить, что из двух сравнимых подмножеств подмножество *Rj* покрывается подмножеством *Ri.*

Пример 7.1. Пусть для частных алгоритмов 2ti, ..., 21т, входящих в ЛСА

Я = i^2 2Ii T^1 ив ^ T^2 2Гз 1^s 21, ЭДв T^{3}1^{1}2I2 *wi* ^,

заданы следующие времена кх выполнения *ti,* н стоимости с, .в условных единицах:

$\mathfrak{A}_1$: $t_1 = 20$, $c_1 = 3$;
$\mathfrak{A}_2$: $t_2 = 40$ $c_2 = 4$;
$\mathfrak{A}_3$: $t_3 = 40$, $c_3 = 5$;
$\mathfrak{A}_4$: $t_4 = 30$, $c_4 = 10$;
$\mathfrak{A}_5$: $t_5 = 50$, $c_5 = 10$;
$\mathfrak{A}_6$: $t_6 = 40$, $c_6 = 15$;
$\mathfrak{A}_7$: $t_7 = 100$, $c_7 = 20$.

Кроме того, известна следующая матрица переходных вероятностей:

$$L = \begin{array}{c|ccccccc} & \mathfrak{A}_1 & \mathfrak{A}_2 & \mathfrak{A}_3 & \mathfrak{A}_4 & \mathfrak{A}_5 & \mathfrak{A}_6 & \mathfrak{A}_7 \\ \mathfrak{A}_1 & & 1/3 & & & 2/3 & & \\ \mathfrak{A}_2 & 1 & & & & & & \\ \mathfrak{A}_3 & & & & & & & 1 \\ \mathfrak{A}_4 & 3/5 & & 2/5 & & & & \\ \mathfrak{A}_5 & & & & 1 & & & \\ \mathfrak{A}_6 & & 1/4 & & & & & 3/4 \\ \mathfrak{A}_7 & & & & & 1 & & \end{array}.$$

Для построения вектора а выпишем следующую систему уравнений:

$$\begin{cases} a_2 + (3/5)\,a_4 = a_1; \\ (1/3)\,a_1 + (1/4)\,a_6 = a_2; \\ (2/5)\,a_4 = a_3; \\ a_5 = a_4; \\ (2/3)\,a_1 = a_5; \\ a_7 = a_6; \\ a_3 + (3/4)\,a_6 = a_7; \\ a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 = 1. \end{cases}$$

После решения этой системы получим ai=3/16; й2=9/80; йз=1/20; а*=1/8; as=°-1/8; аб=1/5; в7=1/5. Приравняем Д2=1. Тогда после пересчета по формуле a',=ai/a2 получим:
a\=W; a'-,=\; a'з=4/9; a\=10/9: a'5=10/9; *a's=l6i9:* д'7=16/9.

В соответствии с (7.3) получим $T_{ср} = \sum_{i=1}^{7} a'_i t_i = 449$ с. Пусть $T_{доп} = 300$ с. Тогда $B = T_{ср} - T_{доп} = 149$ с. Следовательно, в соответствии с (7.5) необходимо иметь $\sum_{i=1}^{7} b_i x_i \geqslant 149$ с, где $x_i = \{0; 1\}$.

Найдем допустимые подмножества частных алгоритмов, считая, что <<==•=2 с и конечный частный алгоритм 21» должен выполняться в ЦУУ. Кроме того, примем /.A„ ==<i.

Рассмотрим вначале подмножества, включающие по одному частному алгоритму (табл. 7.4) *, при этом значение Ь< вычисляем по формуле (7.4).

- Здесь и далее в таблицах вместо **2t<** указан **номер t.**

Из табл. 7.4 видно, что выполнение только одного частного алгоритма в АФБ позволит создать ММУС с аданной производительностью. Поэтому принимаем R1= {St7}. Очевидно, R1 является минимально допустимым подмножеством.

Таблица 7.4

$\mathfrak{A}_i$	1	3	4	5	6	7
Σb_i	30	3,5	31,1	66,7	65,3	174,2
Σc_i	3	5	10	10	15	20

Найдем другие допустимые подмножества. Однако прежде чем начать их поиск, найдем покрываемые подмножества. Для этого вычислим вначале удельные нагрузки на АФБ{, t=l, 3, 4, 5, 6, 7, по формуле

$$y_i = (t_{a_i} + t_{\text{АФБ}i})\, a'_i.$$

Примем $ta_i = t'_1$. Тогда:

$y_2 = (2+40)\ 4/9 = 18{,}7$; $y_1 = (2+20)\ 5/3 = 36{,}7$;

$y_5 = (2+50)\ 10/9 = 57{,}8$ $y_4 = (2+30)\ 10/9 = 35{,}6$;

$y_7 = (2+100)\ 16/9 = 181{,}3$ $y_6 = (2+40)\ 16/9 = 74{,}7$.

Легк0 видеть, что условие (7.6) выполняется только для двух подмножеств *[St5]* и {St6}, причем подмножество {Sts} покрывает подмножество {2tt}. Следовательно, подмножество {St6} из дальнейшего рассмотрения исключается.

Рассмотрим теперь подмножества из двух частных алгоритмов, за исключением тех, в которые входит St6 (в табл. 7.5).

Таблица 7.5

$\{\mathfrak{A}_i\}$	{1,3}	{1,4}	{1,5}	{1,7}	{3,4}	{3,5}	{3,7}	{4,5}	{4,7}	{5,7}
Σb_i	33,5	61,1	96,7	204,2	34,6	70,2	177,7	97,8	205,3	240,9
Σc_i	8	13	13	23	15	15	25	20	30	30

Из табл. 7.5 с учетом удельных нагрузок видно, что отсутствуют покрываемые подмножества. При этом выявились следующие допустимые подмножества: ^2={2ll, 217}; ^3={Яз, Я7}; /?4={2t4, 2t7}; ^5={3l5, 2Iz}.

Рассмотрим теперь подмножества из трех частных алгоритмов (табл. **7.6).** Из табл. 7.6 видно, что имеются следующие допустимые подмножества: R6={St1, St44, St7}; R7={St1, St5, St7}; R8={St1, St4, St7}; R9={St3, St4, St7}; R10= {St4, *St5,* St7}. При этом нет покрываемых подмножеств.

Таблица 7.6

$\{\mathfrak{A}_i\}$	{1,3,4}	{1,3,5}	{1,3,7}	{1,4,7}	{1,5,7}	{3,4,5}	{3,4,7}	{3,5,7}	{1,4,5}	{1,3, 4,6}	{4,5,7}
Σb_i	64,6	100,2	207,7	235,3	270,9	101,3	209,8	244,4	127,8	129,9	272,0
Σc_i	18	18	28	33	33	25	35	35	23	33	40

Таблица 7.7

$\{\mathfrak{A}_i\}$	{1,3,4,5}	{1,3,4,7}	{1,3,5,7}	{1,4,5,7}	{3,4,5,7}	{1,3,4,5,7}
Σb_i	131,3	238,4	274,4	302,0	275,5	305,5
Σc_i	28	38	38	43	45	48

Аналогично получаем и остальные допустимые подмножества (табл. 7.7). Из табл. 7.7 находим следующие допустимые подмножества:

$R_{12} = \{\mathfrak{A}_1, \mathfrak{A}_3, \mathfrak{A}_4, \mathfrak{A}_7\}$; $R_{13} = \{\mathfrak{A}_1, \mathfrak{A}_3, \mathfrak{A}_5, \mathfrak{A}_7\}$;

$R_{14} = \{\mathfrak{A}_1, \mathfrak{A}_4, \mathfrak{A}_5, \mathfrak{A}_7\}$;

$R_{15} = \{\mathfrak{A}_3, \mathfrak{A}_4, \mathfrak{A}_5, \mathfrak{A}_7\}$ и $R_{16} = \{\mathfrak{A}_1, \mathfrak{A}_3, \mathfrak{A}_4, \mathfrak{A}_5, \mathfrak{A}_7\}$.

Таким образом, на первом этапе получено множество всевозможных несравнимых допустимых подмножеств, удовлетворяющих у ел с/вию (/,о),

дгь^6^ BbInoJIHeHM ^i в АФБ» состоит из времени активизации АФЬ—?^. и времени выполнения 2t,—^^ . Однако при наличию конечного числа АФБ, для заданной нагрузки *у,* может оказаться, что в момент, когда необходимо включить *АФБ,,* не найдется ни одного свободного АФБ,. Тогда ЦУУ поставит заявку в очередь на включение АФБ, после его освобождения. В связи с этим фактическое выполнение 21, в АФБ, с учетом времени ожидания в очереди ^, составит /„„ *+1* ^ *+t*^,.

Легко видеть, что удельная .нагрузка на группу АФБ, при однократном выполнении объединенного алгоритма функционироваяия *yi*— (^a^+^АфБ,)a',, а общая нагрузка при *N* одновременно. выполняемых технологических процессов У,=г/,^.

На втором этапе проектирования структуры ММУС для каждого допустимого подмножества частных алгоритмов из полученного на первом этапе подмножества несравнимых допустимых подмножеств частных алгоритмов определяется число АФБ в каждой группе.

Примем, что включение однотипных АФБ в каждой группе полнодоступное, а поступающий поток заявок на включение АФБ оудем считать простейщим. Тогда необходимое число АФБ, в, группе для удовлетворения необходимого качества обслуживания заявок, выражающееся в выполнении условия

*ож1 ^ 7ож.доиЧ (7.7)

где *toxf* — среднее время ожидания в очереди к АФБ,, может быть получено методами теории массового

обслуживания [25]. В ^^ ^У^е общее время выполнения Я. при его реализации вАФЬс состоит из времени, затрачиваемого ЦУУ на включение АФЬ, и принятие от него сигнала прерывания, *t'i^ta,* времени ожидания в очереди на занятие АФБ, /ож, и времени выполнения' St. в АФБ, ^ , т. е.
^АФБ, = ^ + ^ож; + ^АФБ,-

Следовательно, с учетом того, что при организации АФБ из *m* частных алгоритмов функционирования / частных алгоритмов ЭДц, ..., 21^ выполняется в АФБ, а остальные 2^, ..., ^-^^— непосредственно в ЦУУ, среднее время однократного выполнения алгоритма 21 вместо (7.3) определяется выражением

$$T^{p} = {}^{\wedge} a^{\wedge} {}^{\wedge}oB,g + {}''^{\wedge} a \setminus t,y \qquad \textbf{(7.8).}$$

Получение наиболее экономичного варианта структуры ММУС, соответствующего каждому допустимому подмножеству частных алгоритмов *Ri,* выполняется независимо от других допустимых подмножеств. Затем результаты, соответствующие всем допустимым подмножествам, сравниваются друг с другом. Принят будет тот вариант структуры, которому соответствует наименьшая суммарная стоимость АФБ.

Получение наиболее экономичного варианта структуры ММУС, •соответствующего одному ^is{J?g}, сводится к решению следующей задачи.

Требуется определить целочисленные неотрицательные значения переменных x,,, ..., *x^,* которые удовлетворяют ограничению

S "'§ ^ож tg (-^ig) ^ ^ож.доп- (7.9)

и обращают в минимум линейную функцию

<c=^^^'

S=i

где *tox.ic(xit)* —время ожидания в очереди к АФБ^ при их числе, равном x,. ; ^ож.доп—допустимое время ожидания в очередях ко всем АФБ 'в процессе выполнения объединенного алгоритма 21, а значит, и любого из 2t's{2t[1], ..., St^}.

Время /ож.доп должно быть, очевидно, таким, чтобы

^p^Tдo,,. . (7.10)

Решение этой задачи, полученное для подмножества *Ri* алгоритма 21, назовем оптимальным решением для ^?i. Среди оптимальных решений для всех подмножеств *Rii={R^}* найдем такое, для которого при удовлетворении условия (7.10) требуются наименьшие затраты на организацию АФБ. Это решение будем называть *оптимальным* для алгоритма 21. Структуру ММУС, соответствующую оптимальному решению для алгоритма 2t, будем называть *оптимальной.*

Пример 7.2. Для рассматриваемого примера на первом этапе (см. пример 7.1) получено 16 допустимых несравнимых решений. Найдем вначале оптимальное решение для *R;.* При этом допустимое время ожидания

^ж.доп<2б,—5,,. (7.11)

Таким образом, для ^ **время** /ож.доп 1=174,2—149=25,2 с. По номограммам для систем массового обслуживания с ожиданием и экспоненциальным распределением времени обслуживания найдем время ожидания при заданной нагрузке на группы АФБ (см. приложение 3) и различном числе **АФБ в** группе.

При Л/=10 получим

$$Y_7 = N y_7 = \frac{181,3}{3600} \cdot 10 = 0,5 \text{ Эрл.}$$

Тогда при числе АФБт 1/7=4 задержка *tow* 7 (4) =21.. с. В этом случае стоимость группы АФБт для /?i

1

равна Сд =20-4=80.

Рассмотрим теперь допустимое подмножество *Rs=[Sii,* Sir}. Для *Rs* **имеем** *foHs.soa* 2=204,2—149=55,2 с, которое должно быть больше или равно времени ожидания в очереди как к АФБ!, так и к АФБ/.

Поскольку в подмножестве *Rs* содержатся два частных алгоритма, для получения оптимального решения воспользуемся алгоритмом дискретного программирования. Для этого составим табл. 7.8, в которой п<—число блоков АФБ<, которые добавляются к основному для уменьшения времени ожидания, а в клетках указаны величины, характеризующие уменьшения времени ожидания при введении очередного *(n+1}-го* блока для *Yi=0,t2* Э.рл и Ут=0,5 Эрл. В данном случае для одного АФБ) <ож.|(1)=11 с и для АФБг <ож.7(1) =93 с, т. е. общее время ожидания *fox* i, 7(1,1) =104 с, тогда как допустимое время ожидания /ож.доп 2=55,2 с. Таким' образом, недостает <,,ж i, 7(1,1)—<ож.доп2=104—

55,2=48,8 с.

Таблица 7.8

$\mathfrak{A}_i$	c_i	Величина сокращения времени ожидания, с, при числе n_i блоков АФБ$_i$					
		1	2	3	4	5	6
$\mathfrak{A}_1$	3	5,4	1,8	1,0	0,6	0,4	0,3
$\mathfrak{A}_7$	20	55	12	5	4,5	2	1,8

Вначале выбираем при га<=1 алгоритм St?, так как он обеспечивает максимальное сокращение времени ожидания.

Если взять АФБ7, то стоимость его создания равна 20. При стоимости в 20 единиц можно создать шесть АФБ]. Однако экономия во времени ожидания при этом составит только на 9,5 с. Следовательно, на первом шаге выбирается АФБ? (в табл. 7.9 число 55 с указано в скобках с индексом 1). При этом получаем экономию во времени ожидания, равную 55 с, тогда как требовалось 48,8 с. Таким образом, оптимальным решением для подмножества .Ri» является .следующее: один АФБ, и два АФБ/. При этом *to* ж i, 7(1, 2) =104— 55==49 с будет меньше, чем <o1я.допа=55,2 с.

Стоимость .группы АФБ, и АФБ? для подмножества *Rs* составит Сд = =3+2.20=43.
г

Теперь получим решение для подмножества Дз={Яз, SI?}. Для *Рз* нагрузки /а=0,05 Эрл и ^7=0,5 Эрл. Следовательно, *to»* 3,7(1,1) =<ож s(l) +<ож т(1) = =5+93=98 с; *torn* 3=177,7—149=28,7 с. Поэтому *1ож* з, 7(1, 1)—/ож 3=98— 28,7= = 69,3 с.

Затем строим табл. 7.9.

Из табл. 7.9 видно, что окончательное решение на первом шаге — выбор АФБ?. При этом превышение временем ожидания допустимого значения составит 69,3—55=14,3 с. Поэтому приступаем к выполнению второго шага алгоритма. Из оставшихся значений времени ожидания выбираем наибольшее значение. В нашем случае это 12 с (если взять второй блок АФБз). При стоимости в>20 единиц можно взять четыре АФБз. Однако в данном случае время ожидания сократится только на 3,4 с. Следовательно, оптимальным решением на вто

Таблица 7.9

$\mathfrak{A}_i$	c_i	Величина сокращения времени ожидания, с, при числе n_i блоков АФБ$_i$				
		1	2	3	4	5
$\mathfrak{A}_3$	5	[1,8	0,8	0,5]$_3$	0,3	0,2
$\mathfrak{A}_7$	20	[55]$_1$	[12]$_2$	5	4,5	2,0

ром шаге является выбор второго дополнительного блока АФБ? (в табл_7а указано в скобках с индексом 2). После выполнения второго шага превышение временем ожидания допустимого значения составит 14,3—12=2,3 с. Поэтому приступаем к выполнению третьего шага алгоритма^

Очевидно, на третьем шаге оптимальным решением будет выбор трех блоков АФБз (в табл. 7.9 указано в скобках с индексом 3), так как наибольшее оставшееся 'значение, равное 5 с, соответствует третьему блоку АФБ;, однако его стоимость превышает стоимость трех блоков АФЬэ.

Таким образом, оптимальным решением для подмножества является выбор четырех блоков АФБз и трех блоков АФБ;. При этом ^,, ,(4, 3)-98-55-12—3,1=27,9 с будет меньше, чем ^о-к.доп з=28,/ с, а стоимость Свд=4.5+

i o ^ пл_ on

Аналогично получаем решения и для остальных 13 допустимых подмножеств(та^ ^. ^^^ ^ оптимальным решением для заданного алгорит-

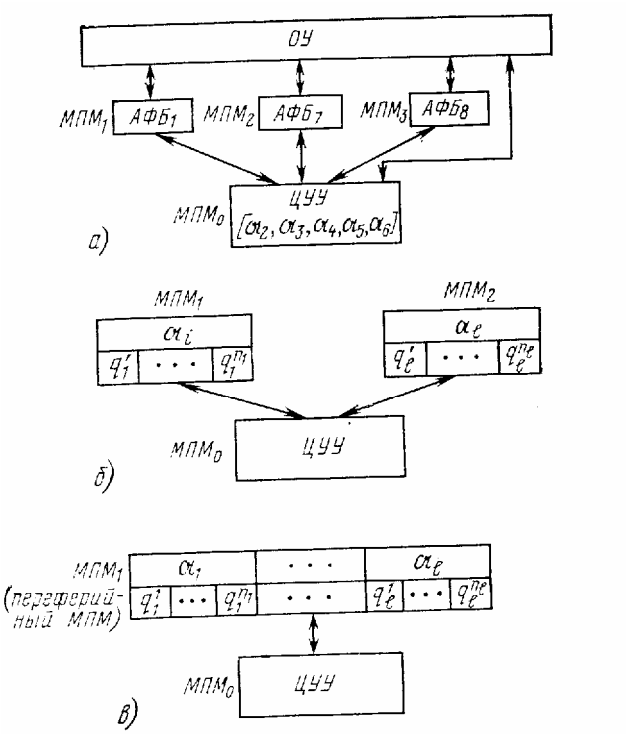

ОУ
МПМ1
АФБ1
МПМ2
АФБ7
МПМ3
АФБ8
МПМ0
ЦУУ
[α2, α3, α4, α5, α6]
а)
МПМ1
αi
МПМ2
αℓ
МПМ0
ЦУУ
б)
МПМ1
(периферийный МПМ)
α1
αℓ
МПМ0
ЦУУ
в)

Рис. 7.6

ма функционирования будет решение, соответствующее оптимальному решению для допустимого подмножества *Rs,* при котором создаются один блок АФБ, и два блока АФБ;.

Таким образом, оптимальной децентрализованной структурой ММУС для рассматриваемого примера будет структура, изображенная на рис. 7.6,а. При реализации такой децентрализованной ММУС на основе микропроцессорных модулей один модуль необходим для создания ЦУУ, в котором должны храниться программы, реализующие частные алгоритмы Sta, Яз. Ш:4, Ms.

Таблица 7.10

л,	Состав блока	Число блоков	Стоимость	Л,	Состав йлока	Число блоков	Стоимость
^i	АФБт	4	80	*Рч*	АФБ4 АФБэ АФБт	1 1 2	60
R2	АФБ, АФБ2	1 2	*л* о	*Ri2*	АФБ! АФБ2 АФБт	1 1 2	58
Да	АФБз АФБ2	4 3	80				
/?t	АФБ» АФБ7	1 2	*v*	*RLS*	АФБ! АФБз АФБ АФБт	2 1 1	51
R'.	АФБэ АФБт	1 2	50				
Re	АФБ! АФБ4 АФБ7	1 1	53	*Ru*	АФБ1 АФБ, АФБз	2 1	46
R-.	АФБ! АФБ. АФБт	2 2 1	46				
Rs	АФБ! АФБ4 АФБт	1 1 2	53	*R^*	АФБз АФБ4 АФБз АФБт	1 2 2 1	53
R»	АФБз АФБ4 АФБ7	) 1 2	53	*Rie*	АФБ! АФБз АФБ4 АФБз АФБт	2 1 1 1 1	51
Rn>	АФБ., АФБ:. АФБт	1 1 2	53				

При реализации АФБ могут быть приняты различные решения в зависимости от используемых для их построения микропроцессорных модулей, наличия расположенных на расстоянии блоков в ооъекте управления ОУ, сложности реализуемых в АФБ частных алгоритмов и т. п. Наиболее характерными являются следующие три типа ММУС:

1. Реализация каждого АФБ в отдельном микропроцессорном модуле (рис. 7.6,а). Такой принцип реализации целесообразен в тех случаях, когда используются малопроизводительные МПМ или когда частные алгоритмы являются достаточно сложными.

2. Реализация в одном микропроцессорном модуле всех АФБ одного типа (рис. 7.6,6). В этом случае каждый МПМ является специализированным для решения одной задачи (выполнения одного частного алгоритма) и функционирует в режиме разделения времени с числом мест пользования его ресурсами, равным числу АФБ одного соответствующего типа. Данный принцип реализации часто оказывается достаточно эффективным даже при низком использовании микропроцессорного модуля, так как в этом случае в модуле хранится только одна программа, а устройства обмена между МПМ и ЦУУ оказываются наиболее простыми. Наиболее эффективен этот принцип реализации в тех случаях, когда в ОУ выполняется большое число процессов каждого из типов.

3. Реализация в одном микропроцессорном модуле всех АФБ (рис. 7.6,в). В этом случае ММУС состоит из двух МПМ, один из которых (ЦУУ) является главным, а второй — вспомогательным, часто называемым периферийным, так как он приближен к ОУ. Этот принцип целесообразен, когда ОУ является сосредоточенным, а производительность МПМ обеспечивает реализацию всех АФБ. Другой, более простой метод построения децентрализованной ММУС описан в [2].

7.4. Метод построения структуры управляющей сети микропроцессоров

Как отмечалось ранее, перспективным принципом построения распределенной ММУС является управляющая сеть микропроцессоров, в которой алгоритм функционирования системы управления, представленный в виде композиции частных алгоритмов и реализуемый в виде программ, распределен по отдельным ФМПМ сети микропроцессоров без выделения ЦУУ для координации работы ФМПМ.

В ФМПМ сети МП программы, как правило, хранятся в постоянных или полупостоянных запоминающих устройствах, тогда как данные, характеризующие состояние одного или нескольких БОУ, управляющих воздействий, а также различного рода промежуточных результатов хранятся в оперативных запоминающих устройствах ФМПМ. В процессе функционирования распределенной ММУС, представленной в .виде управляющей сети МП, между ФМПМ происходит обмен этими данными. Кроме того, вообще говоря, не исключается и возможность обмена программами между ФМПМ.

Таким образом, возникают потоки данных между ФМПМ управляющей сети МП, передаваемые по каналам сети связи МП.

Будем говорить, что ФМПМ^ связан с ФМПМ^-, если между .ними в процессе функционирования ММУС существует лоток сообщений ,с интенсивностью /1„>0.

В распределенных ММУС на сеть связи МП падает значительная доля стоимости всей ММУС. В связи с этим одной из первых задач, возникающих при проектировании такой ММУС, является задача минимизации обмена сообщениями между ФМПМ. При этом минимизируется нагрузка, поступающая в сеть связи МП, а следовательно, последняя становится менее сложной, улучшаются ее стоимостные показатели.

В настоящее время имеется ряд методов решения задачи минимизации нагрузки на сети МП, один из которых изложен в [26]. Если программы распределены по ФМПМ, можно определить тяготения между каждой парой ФМПМ. Эти тяготения являются исходными данными для построения сети связи МП. Построение сети связи МП удобно представить в виде трех этапов.

На первом этапе выбирается необходимое число коммутационных микропроцессорных модулей КМПМ, к которым подключаются ФМПМ. При выборе следует стремиться к тому, чтобы оомен информацией между каждой парой ФМПМ осуществлялся как можно через меньшее число КМПМ. В случае минимального числа транзитных КМПМ задержка передаваемых между ФМПМ сообщении (т. е. один транзитный КМПМ) будет минимальной а общая производительность ММУС,—очевидно, максимальной при этом нагрузка на сеть связи КМПМ будет также минимальной. легко понять, что в частном случае, когда все связные ФМПМ подключены к одному и тому же коммутационному микропроцессорному модулю, нагрузка между КМПМ будет вообще отсутствовать, а следовательно, не будет и сети связи между ними Таким образом, задача выбора необходимого числа КМПМ ^уф^пм^{3}^^{46} $Р^{33161"1"1"1}$ множества ФМПМ на группы связия", ^{фмпм}- Рассмотрим метод ее решения вначале для данного частного случая, т. е. когда отсутствует сеть связи между КМПМ назовем группу ФМПМ *связной,* если каждая пара ФМПМ *является* связной. Группа связных ФМПМ называется максимальной если в нее вошли все *m* ФМПМ, между которыми имеется плипт,'" $^{\wedge fiwn}{\sim}^{m}$ оставшихся не найдется ни одного, который одновременно был бы связан со всеми *m* модулями данной груп^vnn$^{к0\ \wedge Tloня\wedge ч\ г}r^{о}$ '"^Утенные таким образом *г* максимальных S $^{язных\ фмпм}$. вообще говоря, могут быть пересекающи/.„^^мт-т^ $^{с\ одной}$ "аксимальной группой связных ФМПМ один КМПМ, получим *г* КМПМ, между которыми будет отсутстnv ^{ать} ^^-•я"^{60}, нагрузка, так как все потоки сообщений между связными ФМПМ будут проходить только через один КМПМ для нахождения минимального числа *г^n* КМПМ сети связи МП при отсутствии потоков сообщений между КМПМ необходимо найти минимальное число максимальных групп связных ФМПМ.

для решения этой задачи может быть применен один из методов минимизации числа внутренних состояний автомата, изложенных в гл. 3, причем аналогом максимальной группы совместимых внутренних состояний автомата в данном случае является максимальная группа связных ФМПМ, минимальное число которых должно покрыть все возможные пары связных ФМПМ.

Как и при минимизации числа внутренних состояний автомата, составляем треугольную таблицу связных пар ФМПМ. Далее по таблице покрытий выбираем минимальное число максимальных групп связных ФМПМ.

Пример 7.2. Пусть в ММУС имеется восемь ФМПМ, нагрузка между каждой парой которых указана в треугольной табл. 7.11. Требуется построить управляющую сеть МП с минимальным числом КМПМ и отсутствием между ними нагрузки.

Та блица 7.11

ФМПМг	5		
ФМПМз	5	5	
ФМПМ»	—	1	—

ФМПМа	—	—	2	5			
ФМПМг	2	—	5	1	5		
ФМПМт	1	—	—	2	—	1	
ФМПМв	—	2	1	—	2	5	5

ФМПМ, ФМПМг ФМПМэ ФМПМ, ФМПМэ ФМПМе ФМПМ?

Данная таблица может рассматриваться в качестве треугольной таблицы связности для определения максимальных связных групп ФМПМ. При этом если *Yi,>0, то* ФМПМ(и ФМПМ^ являются связными. В таблице черточка указана в тех клетках, которые соответствуют парам несвязных ФМПМ.

Найдем по таблице следующие максимальные группы связных ФМПМ (для простоты будем указывать только номера ФМПМ): а=1, 2, 3; 6=1, 6, 7; с= *=2^;* д=2, 3, 8; е=3, 5, 6, *S; f=4^5^Г, g=4, 6, 7;* /i=6, 7, 8; тI=I, 3, 6.

Строим таблицу покрытий (табл. 7.12).

Строки этой таблицы сопоставлены максимальным группам, а столбцы — парам связных ФМПМ. Покрытие пары связных ФМПМ максимальной группы обозначим знаком V. Если в каком-либо столбце имеется только один знак V. то он помещается ,в круглые скобки. Максимальные группы, соответствующие
строкам, в которых имеется хотя бы один знак (V), образуют ядро. Значит, минимальное число максимальных групп связных ФМПМ равно восьми: *a, Ь, c,* гi> *e, f, g, h.* Группа т) является избыточной.

Таким образом, в сети связи МП должно быть не менее восьми КМПМ для того, чтобы между ними не было никакой связи (рис. 7.7).

Если максимальные группы ядра не образуют покрытия, то из оставшихся максимальных групп к ним должно быть добавлено минимально необходимое число групп, которые можно получить, пользуясь функцией Петрика [3].

Таблица 7.12

Максимальная группа	Номера пар ФМПМ																			
	1,2	1,3	1,6	1,7	2,3	2,4	2,8	3,5	3,6	3,8	4,5	4,6	4,7	5,6	5,8	6,7	6,8	7,8
V*a*	(V)	V			V													
—η		V	V						V									
V*b*			V	(V)												V		
V*c*						(V)												
V*d*					V		(V)			V								
V*e*								(V)	V	V				V	V		V	
V*f*											(V)	V		V				
V*g*												V	(v)			V		
V*h*																V	V	(v)

Из рис. 7.7, а также из рассмотрения соответствующих максимальных групп видно, что каждый ФМПМ включен в несколько КМПМ.

Таким образом, исключение связей между КМПМ усложняет связи ФМПМ с КМПМ. Поэтому данное решение не всегда может

оказаться целесообразным. В тех случаях, когда можно допустить некоторую потерю производительности ММУС, целесообразно отойти от выбора мини мального числа КМПМ без их связи друг с другом. Можно, например, включать в один и тот же КМПМ не все связные пары ФМПМ, а только те, которые находятся на небольшом расстоянии друг от друга. Тогда вместо выбора максимальных групп связных ФМПМ будут выбраны квазимаксимальные, удовлетворяющие данному критерию по расстоянию между ФМПМ. В качестве квазимаксимальных групп могут рассматриваться непересекающиеся максимальные группы связных ФМПМ или с каким-либо

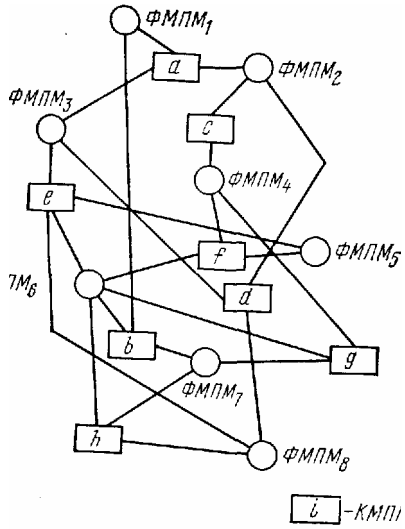


Рис. 7.7

допустимым пересечением. Если взяты непересекающиеся группы, то каждый ФМПМ будет подключен только к одному КМПМ. Может быть поставлено условие о допустимости таких непересекающихся групп, для которых один ФМПМ входит только в две, три и т. п. квазимаксимальные группы связных ФМПМ. В этих случаях, очевидно, каждый ФМПМ будет одключен соответственно к двум, трем и т. д. КМПМ.

При использовании квазимаксимальных групп, в частности непересекающихся групп связных ФМПМ, передача сообщений между ФМПМ,'вошедшими в разные квазимаксимальные группы, будет осуществляться уже не через один, а через два или даже более КМПМ, т. е. через сеть связи МП. При этом с увеличением числа транзитных КМПМ общая производительность ММУС снижается. Поэтому стоит задача исключить ФМПМ из таких максимальных групп, в которых он создает меньшую нагрузку к другим ФМПМ данной группы по сравнению с нагрузкой этого же ФМПМ к ФМПМ во всех других максимальных группах. В этом случае ФМПМ с сильной связью по нагрузке окажутся подключенными к одному и тому же КМПМ, а со слабой связью — к разным КМПМ. При этом будут минимальными общая нагрузка на сеть связи МП, что приведет к построению наиболее дешевой сети связи МП и снижению производительности ММУС.

Устранение пересечений максимальных групп связных ФМПМ является задачей комбинаторного типа, размерность которой зависит от характера пересечений и числа ФМПМ.

Введем понятие е-максимальной группы связных ФМПМ, на основе которого квазима'ксимальные группы могут быть получены также и в том случае, когда слабые связи между ФМПМ; и ФМПМ„ т. е. при У„<е, не принимаются во внимание. Пусть в =2, тогда при построении треугольной таблицы связных ФМПМ на основе табл. 7.11 в клетках ФМПМ4—ФМПМа, ФМПМе— ФМПМ4, ФМПМт^ФМПМь ФМПМ7—ФМПМб .и ФМПМа— ФМПМз должны быть черточки, т. е. соответствующие пары ФМПМ считаются несвязными.

Метод устранения пересечений максимальных групп рассмотрим яа следующем примере.

Пример 7.4. Пусть имеются максимальные группы связных ФМПМ, полученные на примере 7.2 и образующие покрытие. Требуется найти минимальное число непересекающихся групп связных ФМПМ.

Решение. Исключим вначале пересечения групп по ФМПМ|. Легко видеть, что ФМПМ(входит в две максимальные группы а и *Ь.* Рассчитаем два варианта нагрузки Ус , которая возникает в сети связи КМПМ, если исключить ФМПМ] из группы *а* или из группы 6, т. е.

$$b = \widetilde{1,\ 6,\ 7}.$$

Поскольку У^^Ус,, целесообразно оставить ФМПМ1 в группе *а,* из группы 6 — исключить. Исключение ФМПМ, из группы Ъ запишем так:

Аналогично рассмотрим и семь остальных максимальных групп:

$$\text{ФМПМ}_1 = \begin{cases} \text{из } a: y^a_{c_1} = y_{1,2} + y_{1,3} = 5 + 5 = 10; \\ \text{из } b: y^b_{c_1} = y_{1,6} + y_{1,7} = 3. \end{cases}$$

$$\text{ФМПМ}_2 = \begin{cases} \text{из } a: y^a_{c_2} = y^a_{1,2} + y^a_{2,3} = 10; \\ \text{из } c: y^c_{c_2} = y_{2,4} = 1; \\ \text{из } d: y^d_{c_2} = y_{2,3} + y_{2,8} = 7. \end{cases}$$

Следовательно, $c = \widehat{2}, \overline{4}$ и $d = \widehat{2}, \overline{3, 8}$.

$$\text{ФМПМ}_3 = \begin{cases} \text{из } a: y^a_{c_3} = y_{1,3} + y_{2,3} = 10; \\ \text{из } d: y^d_{c_3} = y_{2,3} + y_{3,8} = 6; \\ \text{из } e: y^e_{c_3} = y_{3,5} + y_{3,6} + y_{3,8} = 8. \end{cases}$$

Следовательно, $d = \widehat{3}, \overline{2, 8}$ и $e = \widehat{3}, \overline{5, 6, 8}$.

$$\text{ФМПМ}_4 = \begin{cases} \text{из } c: y^c_{c_4} = y_{2,4} = 1; \\ \text{из } f: y^f_{c_4} = y_{4,5} + y_{4,6} = 6; \\ \text{из } g: y^g_{c_4} = y_{4,6} + y_{4,7} = 3. \end{cases}$$

Следовательно, $c = \overline{2}, \widehat{4}$, $g = \widehat{4}, \overline{6, 7}$.

$$\text{ФМПМ}_5 = \begin{cases} \text{из } e: y^e_{c_5} = y_{3,5} + y_{5,6} + y_{5,8} = 9; \\ \text{из } f: y^f_{c_5} = y_{4,5} + y_{5,6} = 10. \end{cases}$$

Следовательно, $e = \widehat{5}, \overline{3, 6, 8}$.

$$\text{ФМПМ}_6 = \begin{cases} \text{из } b: y^b_{c_6} = y_{1,6} + y_{6,7} = 3; \\ \text{из } e: y^e_{c_6} = y_{3,6} + y_{5,6} + y_{6,8} = 15; \\ \text{из } f: y^f_{c_6} = y_{4,6} + y_{5,6} = 6; \\ \text{из } h: y^h_{c_6} = y_{6,7} + y_{6,8} = 6. \end{cases}$$

Следовательно, $b = \overline{1, 7}, \widehat{6}$, $f = \overline{4, 5}, \widehat{6}$ и $h = \widehat{6}, \overline{7, 8}$.

$$\text{ФМПМ}_7 = \begin{cases} \text{из } b: y^b_{c_7} = y_{1,7} + y_{6,7} = 2; \\ \text{из } g: y^g_{c_7} = y_{4,7} + y_{6,7} = 3; \\ \text{из } h: y^h_{c_7} = y_{6,7} + y_{7,8} = 6. \end{cases}$$

Следовательно, $b = \overline{1, 6}, \widehat{7}$ и $g = \overline{4, 6}, \widehat{7}$.

$$\text{ФМПМ}_8 = \begin{cases} \text{из } d: y^d_{c_8} = y_{2,8} + y_{3,8} = 3; \\ \text{из } e: y^e_{c_8} = y_{3,8} + y_{5,6} + y_{6,8} = 8; \\ \text{из } h: y^h_{c_8} = y_{6,8} + y_{7,8} = 10. \end{cases}$$

Следовательно, $d = \overline{2, 3}, \widehat{8}$ и $e = \overline{3, 5, 6}, \widehat{8}$.

Таблица 7.13

	ФМПМ$_1$	ФМПМ$_2$	ФМПМ$_3$	ФМПМ$_4$	ФМПМ$_5$	ФМПМ$_6$	ФМПМ$_7$	ФМПМ$_8$
a	(V)	(V)	(V)					
b	∽						∽	∽
c		∽		∽	∽			
d		∽	∽					
e			∽		∽	V		∽
f				(V)	(V)	∽		
g				∽		∽		
h						∽	(V)	(V)

Построим таблицу покрытий (табл. 7.13), в которой строкам сопоставлены максимальные группы, а

столбцам — отдельные ФМПМ.

Знак V s таблице указывает на то, что в группу, сопоставленную с этой строкой, входит соответствующий ФМПМ, а знак ~ —что из заданной группы может быть исключен соответствующий ФМПМ.

Знак (У)указывает на то, что выполняются два условия: 1) в столбце имеется только один знак V и 2) в строке имеется еще хотя бы один знак V

Выполнение этих условий означает, что в данную группу обязательно должны войти хотя бы два ФМПМ. Заметим, что при наличии в группе только одного ФМПМ установка КМПМ не обязательна, так как в него будет включен только один ФМПМ.

Группы, которым соответствуют строки хотя бы с одним знаком (V)i входят в ядро покрытия. В нашем случае это группы a, *f* и *h.* В строках, соответствующих группам *Ь, c, d, g,* имеются только знаки .~. Это означает, что. в данные группы могут не входить ФМПМ. Значит, для минимизации числа КМПМ их необходимо исключить. В группу *e* должен быть включен только один ФМПМ. Следовательно, КМПМ в этом случае может не создаваться. Вместе с тем ФМПМе может быть включен в двух группах *(f* и *h),* вошедших в ядро. Следовательно, для сокращения числа КМПМ и нагрузки на сеть связи целесообразно ФМПМ включить в группу *f* или *h.* Тогда получаем два решения

1_ Выбираем три непересекающиеся группы: a=1, 2, 3, f'=4, 5, 6 и *h'*= =7, 8

2. Выбираем три непересекающиеся группы: a=1, 2, 3, /"==4, 5 и *h"*= -6, 7, 8.

Связи с указанием нагрузок между КМПМ для этих двух решений приведены соответственно на рис. 7.8,д и *б.*

Заметим, что при подключении ФМПМ не более чем к двум КМПМ всего потребуется четыре КМПМ при общей нагрузке на сеть связи Уобщ=8 Эрл.

На этом первый этап построения сети связи МП заканчивается.

На втором этапе определяется способ физической реализации КМПМ-. Во-первых, подбираются ФМПМ и КМПМ необходимой производительности. Во-вторых, исходя из загрузки ФМПМ решается вопрос о передаче функции того или иного КМПМ на один из ФМПМ, вошедших в данную группу.

На третьем этапе проектируется сеть связи МП. Этот этап аналогичен этапу проектирования 'сети связи, т. е. необходимо получить топологию сети связи, выбрать метод коммутации и рассчитать пропускные способности каналов и КМПМ.

Вопросы проектирования сети связи здесь не рассматриваются, так как они изучаются в курсе «Теория сетей связи».

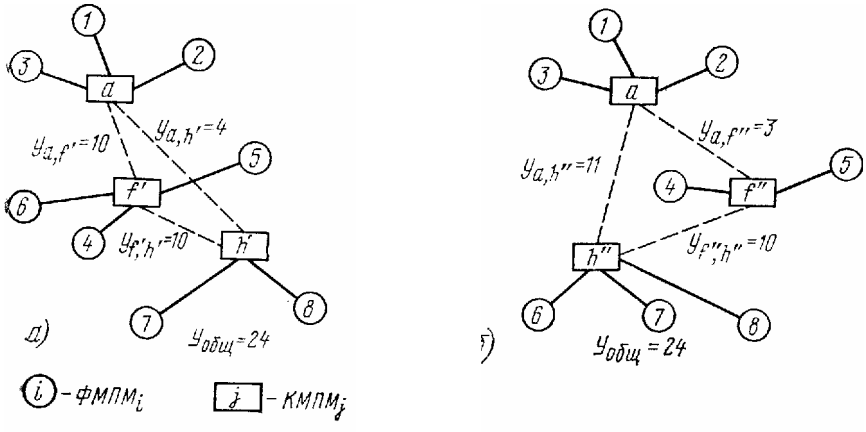


Рис. 7.8

Контрольные вопросы

1. Назовите состав основных функциональных блоков микропроцессорной системы.
2. Какие существуют методы программной реализации дискретного устройства в базисе микропроцессорных систем?
3. Какими средствами можно повысить производительность многомикропроцессорного дискретного устройства?
4. **В чем** суть метода построения распределенного многомикропроцессорного дискретного устройства?

ПРИМЕР ТИПОВОГО ЗАДАНИЯ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ
РАЗРАБОТКА СИСТЕМЫ ДИСКРЕТНОГО УПРАВЛЕНИЯ

1. Назначение системы

Система предназначена для управления объектом, представляющим собой совокупность блоков аппаратуры связи, расположенных в нескольких зданиях, удаленных друг от друга на расстояние до 5 км.

2. Исходные данные

1.1. Здания (Зд.), в которых расположена аппаратура связи, находятся на территории общей площадью 25 км^2; расстояния между зданиями в километрах приведены в табл. П.1.

Таблица П.1

	$Зд_1$	$Зд_2$	$Зд_3$
$Зд_2$	4		
$Зд_3$	4,9	1,9	
$Зд_4$	2,2	3,7	3,5

2.2. Размещение блоков аппаратуры связи (блоков объекта управления — БОУ) по зданиям приведено в табл. П.2.

2.2.1. Расстояние в метрах между БОУг и BOVj, t, /=1, ..., 7, *i^i,* в здании Зд1 приведены в табл. П.3.

2.2.1.1. Алгоритм функционирования подсистемы управления объектами, расположенными в Зд1, состоят из (восьми частных алгоритмов, взаимодействие которых представлено в виде ЛСА.

ЭД' = i*9Ii ^эд, ^1 ^ ^ **214** I[1] **Ms 21в** T[3] **йт %** со t[1], (П.1)

в которой 212, 21з, 21в—логические условия, а остальные—операторы. Матрица переходных вероятностей, определяющая вероятность перехода от Я; к 2tj, имеет вид

	$\mathfrak{A}_1$	$\mathfrak{A}_2$	$\mathfrak{A}_3$	$\mathfrak{A}_4$	$\mathfrak{A}_5$	$\mathfrak{A}_6$	$\mathfrak{A}_7$	$\mathfrak{A}_8$
$\mathfrak{A}_1$		1						
$\mathfrak{A}_2$			0,8		0,2			
$\mathfrak{A}_3$		0,1		0,9				
$\mathfrak{A}_4$					1			
$\mathfrak{A}_5$						1		
$\mathfrak{A}_6$			0,4				0,6	
$\mathfrak{A}_7$								1
$\mathfrak{A}_8$	1							

. (П.2)

2.2.1.2. Управление БОУ осуществляется согласно частному алгоритму $\mathfrak{A}_i$ (табл. П.4). Знак «х» в таблице указывает на то, что БОУ$_j$ находится под управлением $\mathfrak{A}_i$.

Таблица П.2

БОУ	Здания 1	2	3	4
$БОУ_1$	×			
$БОУ_2$	×			
$БОУ_3$	×			
$БОУ_4$	×			
$БОУ_5$	×			
$БОУ_6$	×			
$БОУ_7$	×			
$БОУ_8$		×		
$БОУ_9$			×	
$БОУ_{10}$				×
$БОУ_{11}$				×

Таблица П.3

	$БОУ_1$	$БОУ_2$	$БОУ_3$	$БОУ_4$	$БОУ_5$	$БОУ_6$
$БОУ_2$	15					
$БОУ_3$	7,5	7,5				
$БОУ_4$	12,5	25	17			
$БОУ_5$	10	18	11	5		
$БОУ_6$	15	10	4	20,5	12,5	
$БОУ_7$	25	12,5	17	30	25	12,5

2.2.1.3. Среднее время *i,* выполнения частных алгоритмов и стоимость с« в условных единицах АФБ<, предназначенного для реализации 21,, приведены в табл. П.5. Время выполнения частных алгоритмов имеет экспоненциальное распределение.

Примечание. В случае реализации 2Ii в АФБ, на его активизацию ЦУУ затрачивает *tt*= 0,002 с.

Таблица П.4

	БОУ$_1$	БОУ$_2$	БОУ$_3$	БОУ$_4$	БОУ$_5$	БОУ$_6$	БОУ$_7$
$\mathfrak{A}_1$	×	×	×				
$\mathfrak{A}_2$				×			
$\mathfrak{A}_3$				×	×		
$\mathfrak{A}_4$				×	×	×	
$\mathfrak{A}_5$							×
$\mathfrak{A}_6$						×	×
$\mathfrak{A}_7$	×	×					
$\mathfrak{A}_8$	×		×				

Таблица П.5

Частные алгоритмы	1	2	3	4	5	6	7	8
Длительность выполнения $\mathfrak{A}_i$, с	12	0,05	0,25	0,08	0,2	0,01	3	1
Стоимость АФБ$_i$, у. е.	40	10	5	12	7	2	6	6

Таблица П.6

Частные алгоритмы	$\mathfrak{A}_1$	$\mathfrak{A}_2$	$\mathfrak{A}_3$	$\mathfrak{A}_4$	$\mathfrak{A}_5$	$\mathfrak{A}_6$	$\mathfrak{A}_7$	$\mathfrak{A}_8$
$t_{ож.доп}$, с	3	0,03	0,1	0,01	0,05	0,01	1	1

2.2.1.4. Допустимые времена ожидания Гож.доп.* начала обслуживания 21< приведены в табл. П.6.

2.2.1.5. Число одновременно выполняемых технологических процессов в части объекта управления, расположенной в Здь A/i=25.

2.2.1.6. Допустимое время однократного выполнения подсистемой управления алгоритма функционирования в ЗД[7'доп.1=0,5 с.

2.2.1.7. В соответствии с частным алгоритмом функционирования 2Ii реализуются операции по подсчету числа импульсов от одного до десяти, поступающих в систему управления от БОУ.

2.2.1.8. В соответствии с частным алгоритмом функционирования 21г осуществляется кодирование двоичным кодом чисел от одного до десяти; полученная кодовая комбинация обрабатывается 21з-

2.2.1.9. Частный алгоритм ^ выполняет операции по вычислению значения булевой функции *f⁴=XlX2Xзx^xs\/XlX-гXsX4Xs'^* л-^в-Сз-ОДУ *XsXsXsX^xsV x^xyXзX^*

2.2.1.10. В соответствии с частными алгоритмами SIs—Sts реализуются операции по решению систем функций.

2.2.2. В здании Здэ расположен один БОУб, управление которым производится с помощью частного алгоритма функционирования Яд.

2.2.2.1. Частный алгоритм 24 позволяет вычислить значение булевой функции ^=0, 1, 3, 5, 6, 8, 9, 10, 11, 17, 25, 26, 30, 31 (2, 4, 16, 16, 18, 24, 27, 28).

2.2.3. В здании Здз расположен БОУ9 управление которым производится с помощью алгоритма 21ю.

2.2.3.1. Частный алгоритм Stio позволяет вычислить значения булевой функции /'о=1, 5, 6, 7, 10, 45, 46, 47, 54, 55. 61 [0, 16, 24, 32, 63].

2.2.4. В здании Зд4 находятся два БОУ: БОУю и БОУц, расстояние между которыми составляет около 5 м.

2.2.4.1. Частный алгоритм функционирования подсистемы управления БОУ^ расположенным в Зд4, в' свою очередь, состоит из двух следующих частных алгоритмов, взаимодействие которых представлено в виде ЛСА

(П.3>

$$\mathfrak{A}^{IV} = r_1 \uparrow^1 \mathfrak{A}_1^{IV}\ \omega \uparrow^2 \downarrow^1 \mathfrak{A}_2^{IV} \downarrow^2.$$

2.2.4.1.1. Частные алгоритмы функционирования St^i, St^'s имеют следующий вид:

$$\mathfrak{A}_1^{IV} = A_0\ A_1\ p_1 \uparrow^1 A_2\ A_3 \downarrow^1 p_2 \uparrow^2 p_3 \uparrow^2 A_4 \downarrow^2; \quad (П.4)$$

$$\mathfrak{A}_2^{IV} = A_0\ A_1 \downarrow^1 A_2\ A_3\ p_1 \uparrow^1 p_2 \uparrow^2 p_3 \uparrow^2 A_4 \downarrow^2. \quad (П.5)$$

2.2.4.1.2. Совместимость операторов показана в табл. П.7.

Таблица П.7

	A_0	A_1	A_2	A_3	A_4
A_1	×				
A_2	×	V			
A_3	×	V	V		
A_4	×	×	V	V	
A_k	V	×	×	×	×

Таблица П.8

	A_0	A_1	A_2	A_3	A_4	A_5
A_1	×					
A_2	×	V				
A_3	×	V	V			
A_4	×	×	×	V		
A_5	×	V	V	V	V	
A_k	V	×	×	×	×	×

2.2.4.1.3. Распределения сдвигов для частных алгоритмов St^i и ЗГ^э имеют вид:
АО, *Ak—{pi,p2.Ps}*
*Ai,As—{pi} ; As—{Pi}; **Ai—{—}.***

2.2.4.2. Частный алгоритм функционирования подсистемы управления БОУю расположенным в Зд4, имеет вид

$91^{\wedge}$= Ло Pi $T^{1\wedge}$ T^1 *I^ AI Pi* T^2 *A, p,* f I^4 *Aa* I^l *A^ ^ AsP»* T^4. (П.6)

2.2.4.2.1. Совместимость операторов показана в табл. П.8.

2.2.4.2.2. Распределение сдвигов для частного алгоритма $Я^{1\prime\wedge}$ имеет вид:
Ац, Ah, A\—[pi, ра, рз]; As, Аз—{—}; Л4, *As—[рз}.*

2.2.4.3. В связи с необходимостью обеспечения высокого быстродействия в ограниченного **времени ожидания** начала обслуживания БОУэ и БОУю я ФМПМ вынесены функции по выполнению частных алгоритмов SI^i и St^s в АФБд и Я^з в АФБэ на БОУд и БОУю и и АФБю расположены соответственно должны быть реализованы аппаратно.

Таблица П.9

	ФМПМ$_1$	ФМПМ$_2$	ФМПМ$_3$
ФМПМ$_2$	10		
ФМПМ$_3$	5	1	
ФМПМ$_4$	2	—	8

2.3. Для управления -БОУ, расположенными в одном и том же здании ЗА! установлен ФМПМ,.

2.3.1. Нагрузка (тяготения) между каждой парой ФМПМ в условных единицах (у. е.) приведена в табл. П.9.

2.4. Для построения системы управления можно использовать микропроцессорные наборы серии К-580, интегральные микросхемы серии 155 («Логика-2»), ПЛМ и МОС.

3. Задание

Необходимо разработать систему дискретного управления (СДУ), удовлетворяющую исходным данным п. 2.

3.1. Разработать структуру взаимосвязи ФМПМ с использованием КМПМ.

3.1.1. Выявить необходимое число КМПМ при отсутствии ограничения на число КМПМ, к которым может подключиться ФМПМ.

3.1.2. Выявить необходимое число КМПМ при подключении каждого ФМПМ не более чем к одному КМПМ.

3.1.3. Сравнить варианты, полученные в п. 3.1.1 и п. 3.1.2, и выбрать из них тот, который имеет наименьшее число КМПМ.

3.2. Разработать подсистему управления, расположенную в здании Зд1.

3.2.2. Если производительность ЦУУ будет недостаточной, то выявить минимально необходимое число частных алгоритмов, выполнение которых в АФБ обеспечит заданные в п. 2.2.1 временные характеристики подсистемы управления при общей минимальной стоимости всех АФБ.

3.2.3. Выбрать элементный базис для АФБ и составить функциональную **схему** подсистемы управления.

3.2.4. Разработать схему АФБ в одном из базисов интегральных микросхем серии 155 («Логика-2»), ПЛМ или МОС.

3.3 Разработать вычислительную процедуру для булевой функции f^9, выполняемую в ФМПМа, реализованном на основе микропроцессорного набора серии К-580.

3.3.1. Выбрать метод вычисления булевой функции *Р.*

3.3.2. Составить блок-схему программы вычисления булевой функции $/^9$.

3.3.3. Составить программу вычисления булевой функции в кодах микропроцессорного набора серии К-580.

3.4. Разработать вычислительную процедуру для 'булевой функции $/^{10}$, выполняемой в ФМПМз,

реализованном на основе микропроцессорного набора серии К-580.

3.4.1. Выбрать метод вычисления булевой функции $/^{10}$.

3.4.2. Составить блок-схему программы вычисления булевой функции f^{10}.

3.4.3. Составить программу вычисления булевой функции f^{10} в кодах микропроцессорного набора серии 'К-580.

3.5. Разработать подсистему управления, расположенную в здании Зд4.

3.5.1. Составить функциональную схему подсистемы управления.

3.5.2. Разработать блок микропрограммного управления, реализующий частные алгоритмы St"! и 21^2.

3.5.2.1. Объединить ЛСА SI^i и SI^s.

3.5.2.2. Сформировать микрокоманды по объединенной ЛСА St^i.a.

3.5.2.3. Выбрать разрядность РМК, закодировать микрокоманды и построить функциональную схему блока микропрограммного управления.

3.5.2.4. Разработать **схему** блока микропрограммного управления в одном из базисов интегральных микросхем серии 155, ПЛМ или МОС.

3.5.3. Разработать-блок микропрограммного управления, реализующий частный алгоритм Я^з.

3.5.3.1. Сформировать микрокоманды по ЛСА й^з.

3.5.3.2. Выбрать разрядность РМК, закодировать микрокоманды и построить функциональную схему блока микропрограммного управления.

3.5.3.3. Разработать схему блока микропрограммного управления в одном из базисов интегральных микросхем серии 155, ПЛМ или МОС.

3.6. Составить спецификацию используемых в СДУ элементов.

ПРИЛОЖЕНИЕ 2

РЕКОМЕНДАЦИИ О ПОРЯДКЕ ВЫПОЛНЕНИЯ ПРОЕКТА

1. ОБЩИЕ ЗАМЕЧАНИЯ

Изложенный в приложении 1 пример типового задания на курсовое проектирование позволяет в процессе проектирования освоить все основные вопросы по построению в современных элементных базисах сложных дискретных систем управления (СДУ) распределенными объектами. При этом задание составлено так, чтобы студент мог пользоваться в основном данным учебным пособием. Однако в зависимости от подготовленности студента и его наклонностей задание может быть соответствующим образом скорректировано. В задание может быть введена экономическая часть, в которой целесообразно задать технико-экономическое сравнение вариантов проектируемой системы или ее отдельных элементов, реализованных, например, в различных базисах.

Весь процесс проектирования СДУ в соответствии с приведенным заданием можно условно разделить на следующие этапы:

1) построение структуры управляющей сети микропроцессоров;

2) разработка децентрализованной подсистемы управления, расположенной в здании Зд1;

3) разработка схем функциональных блоков подсистемы децентрализованного управления, расположенных в здании Зд[;

4) разработка блоков микропрограммного управления, расположенных в здании Зд4;

5) разработка программ для ФМПМ, расположенных в зданиях Зда в Здз.

Эти этапы целесообразно выполнять последовательно один за другим. Однако перед их выполнением целесообразно подробно ознакомиться с основами проектирования дискретных устройств по материалу гл. 1 учебного пособия, обратив особое внимание на принципы построения дискретных устройств и систем **(см.** разд. 1.1). В качестве дополнительного материала можно рекомендовать [2,3, 6, 211."

2. РЕКОМЕНДАЦИИ О ПОРЯДКЕ ВЫПОЛНЕНИЯ ЭТАПОВ ПРОЕКТИРОВАНИЯ СДУ

2.1. Построение структуры управляющей сети микропроцессоров

При выполнении этого этапа необходимо использовать материалы разд. 7.3', (до п. А) и 7.4. Вспомогательный материал дан в [4, 26].

Исходные данные для выполнения первого этапа приведены в пп. 2.1, 2.3, 2.4 приложения 1. При этом расстояния между ФМПМ необходимо принимать такими же, как и между зданиями, в которых они располагаются (см. табл. П.1.1). Размещение БОУ по зданиям приведено в табл. П.1.2.

Задание на разработку структуры управляющей сети микропроцессоров приведено в п. 3.1, включающем пп. 3.1.1—3.1.3 (см. приложение 1). Примеры решения аналогичной задачи по

выполнению этих пунктов задания приведены в раэд. 7.4 (примеры 7.2 и 7.3). При выполнении задания на этом этапе при выявлении е-маисимальных групп связных ФМПМ значение е может задаваться произвольно. Однако с учетом табл. П.1.9 целесообразно брать следующие значения показателя связности: е=0, е=1 и е=2.

2.2. Разработка децентрализованной подсистемы управления

Материал, необходимый для выполнения этого этапа, приведен в разд. 7.3. В качестве вспомогательной литературы можно использовать [2, З].

Исходные данные для выполнения второго этапа проектирования приведены в п. 2.2.1 (пп. 2.2.1.1—2.2.1.6) приложения 1. Задание на разработку децентрализованной подсистемы дискретного управления, расположенной в здании ЗД[, Приведено в п. 3.2 (пп. 3.2.1—3.2.3) приложения 1.

При выполнении данного пункта задания целесообразно руководствоваться примером 7.1. При этом время обслуживания для заданной нагрузки на группу АФБ **может** быть найдено по номограммам или по формуле, приведенным в приложении 3. Для построения структуры децентрализованной системы управления может быть использован и более простой метод, изложенный в [2], разд. 2.3, с. 48—52.

Следует заметить, что, как и в разд. 7.3, каждый из АФБ может быть реализован в базисе микропроцессорных наборов. Однако для разнообразия в задании, приведенном в приложении 1, предусматривается аппаратная реализация АФБ в базисах интегральных микросхем серии 155, ПЛМ и МОС. Поскольку специальный модуль универсальной ячейки может быть реализован на интегральных микросхемах серии 155, он не указан в перечне элементов, приведенном в п. 2.4 задания **(см.** приложение 1). Однако универсальную ячейку также можно использовать при построении АФБ, реализовав ее затем в базисе интегральных микросхем серия 155.

Элементный базис для построения АФБ может быть уточнен в задании или выбран на основе технико-экономического сравнения вариантов реализации АФБ в процессе проектирования СДУ. При выборе элементного базиса необходимо использовать материал **гл. 2 и 6 пособия.**

2.3. Разработка схем **функциональных блоков** подсистемы децентрализованного **управления**

Материал, необходимый для выполнения данного этапа проектирования, изложен в гл. 3—6 пособия. Исходные данные для синтеза АФБ, расположенных в здании Зд1, приведены в пп. 2.2.1.7—2.2.1.10, а задание на разработку — в п. 3.2.4 (см. приложение 1).

Из всех АФБ, которые должны быть созданы в соответствии с результатами выполнения второго этапа проектирования, необходимо разработать схемы, реализующие частные алгоритмы Я, (п. 2.2.1.7), йг (п. 2.2.1.8) и ^ (п. 2.2.1.9). Число и состав подлежащих к реализации в проекте АФБ могут быть изменены в зависимости от результатов второго этапа проектирования и от необходимости усложнения или упрощения типового задания.

2.3.1. Разработка схемы A<E>Bi, реализующего алгоритм SIi

Схема АФБ[(т. е. АФБ, реализующего 2Ii) может быть разработана в любом из указанных в п. 2.4 (см. приложение 1) задания элементном базисе. Однако вначале необходимо выполнить абстрактный синтез АФБ, т. е. задать условия его работы на одном из автоматных языков, когда элементный базис учитывается в очень слабой степени или вообще не учитывается. Поскольку в данном случае **имеется** одна последовательность «вход—выход», в качестве начального языка удобно использовать первоначальную таблицу включений. Однако можно составить и первоначальную таблицу переходов. Материал с соответствующими примерами для выполнения этого этапа абстрактного синтеза АФБ] изложен в разд. 3.1 и 3.2. После данного этапа осуществляется минимизация числа внутренних состояний автомата, представляющего абстрактную модель АФБ[при использовании языка таблиц переходов или выбор минимального числа ЭП при переводе первоначальной таблицы включений в реализующую таблицу включений. Этот материал также с соответствующими примерами изложен в разд. 4.1.

Если условия работы АФБ задаются на языке таблиц переходов, то после минимизации числа внутренних состояний необходимо осуществить их кодирование (см. разд. 4.3).

В качестве дополнительного материала для выполнения этапа абстрактного синтеза АФБ[рекомендуются [1, 3, 7], а при использовании таблиц включений, кроме того, и [16].

После этапа абстрактного синтеза АФБ начинается этап структурного синтеза. При этом в зависимости от выбранного элементного базиса вначале выписываются булевые функции, описывающие выходные сигналы АФБ и сигналы включения ЭП. Этот вопрос изложен в гл. 4. Методы структурного синтеза (включающего минимизацию булевых функций) в базисе интегральных микросхем серии 155 рассмотрены в гл. 4. Методы структурного синтеза в базисе электромагнитных

реле и интегральных микросхем малой степени интеграции, к которым относятся микросхемы серии 155, описываются в [1] и более подообно в [3, 11, 19]. Если в качестве элементного базиса выбран один из базисов ПЛМ, УЛЯ или МОС, то при структурном синтезе необходимо пользоваться материалом гл. 6. В качестве дополнительной литературы в данном случае можно использовать: [17, 18] при построении АФБ в базисе универсальной ячейки; [5, 15, 21, 32] в базисе ПЛМ и [14, 20, 31] в базисе однородных сред.

Методы минимизации булевых функций достаточно подробно изложены в [7. 16, 19].

2.3.2. Разработка схемы АФБг, **реализующего алгоритм 2tx**

Устройство, реализующее 21г, может быть представлено в виде комбинационного автомата с десятью входами и четырьмя выходами. При этом появление сигнала на і-м входе (число f) вызывает появление комбинации выходных сигналов. Поэтому для построения АФБ; вначале необходимо сопоставить с каждым і-м входом одну определенную комбинацию выходных сигналов. **Таким** образом, абстрактный синтез в этом случае имеет вырожденный характер. Структурный же синтез АФБг выполняется так же, как л АФБь

2.3.3. Разработка схемы АФБ4, реализующего алгоритм Sti

Этап абстрактного синтеза отсутствует, так как булева функция уже задана. Поэтому сразу начинается этап структурного синтеза, который выполняется так же, как для АФБ[и АФБа.

Необходимо отметить, что процесс разработки АФБ следует начинать АФБ< как наиболее простого; затем разрабатывается более сложный АФБа и только после этого АФБ!.

В дополнительном задании можно предусмотреть составление монтажных схем и разработку конструкций АФБ. При этом рекомендуется воспользоваться материалом гл. 1 и дополнительной литературой [5, б].

2.4. Разработка блоков микропрограммного управления

При выполнении четвертого этапа проектирования необходимо использовать материал раэд. 3.3—3.5, 4.2, 5.5 и 6.5. В качестве дополнительной литературы можно использовать [2, З].

Исходные данные для разработки блоков микропрограммного управления, расположенных в здании Зд4, приведены в п. 2.2.4 (пп. 2.2.4.1—2.2.4.3), а задания на их проектирование — в п. 3.5 (пп. 3.5.1—3.5.3) приложения 1.

Процесс проектироваяия целесообразно начинать с блока микропрограммного управления (БМПУ), реализующего Я^з как наиболее простого. Формальной моделью БМПУ служит микропрограммный автомат, определение которого приведено в разд. 3.1, а языки задания условий его работы—в разд. 3.3. В разд. 3.4 описан процесс составления Л СА. После того как составлена ЛСА (см. ириложение 1, п 2.2.4.2), необходимо сформировать микрокоманды методами, описанными в разд. 4.2). При этом сведения о совместимости операторов приведены в **в. 2.2.4.2.1,** а о распределении сдвигов—в п. 2.2.4.2.2 приложения 1. На основе сформированных микрокоманд выбирается разрядность РМК и строятся вначале структурная, а **затем** и функциональная **схемы** БМПУ в выбранном элементном базисе Метод синтеза структурной схемы БМПУ рассмотрен в разд. 5.5, а метод реализации БМПУ в базисе МОС -- в разд. 6.5. Заметим, что БМПУ может быть эффективно реализован также и в базисе ПЛМ.

Синтез БМПУ, реализующего St^i я З^а, аналогичен рассмотренному. Однако отличие состоит в том, что перед формированием микрокоманд необходимо осуществить объединение ЛСА St^i и З![1]^ методом, изложенным в разд 3.5. Исходные данные для разработки этого БМПУ приведены в п. 2.2.4.1 (пл. 2.2.4.1.1—2.2.4.1.3), а задание — в п. 3.5.2 (пп. 352Л—35.2.4) приложения 1. После того как будут построены схемы этих двух ЬМ11У, в соответствии с заданием (см. п. 3.5.1 приложения 1) составляется функциональная схема подсистемы управления, расположенной в здании Зд4.

2.5. Разработка программ для ФМПМ

На данном этапе осваивается процесс программирования задач для ФМПМ. При этом программируются два алгоритма функционирования: для ФМПМз и ФМПМ4. Материал, необходимый для выполнения этого этапа проектирования, изложен в разд. 7.2. В качестве дополнительной литературы в первую очередь рекомендуются [21, 27—30], где описывается система команд для микропроцессорного набора К-580, даются рекомендации с разбором примеров по составлению программ для микропроцессоров. Кроме того, можно воспользоваться также [23, 24], где приводятся полезные сведения о микропроцессорах и микропроцессорных системах. Исходные данные для составления программ приведены в п. 2.2.2 (для ФМПМа) и п. 2.2.3 (для ФМПМз), а задания — соответственно в п. 3.3 (пп. 3.3.1—3.3.3) и 3.4 (пп. 3.4.1—3.4.3) приложения 1. Рекомендации о выборе метода вычисления булевых функций приведены в разд. 7.2.

После того как будут разработаны отдельные подсистемы, необходимо привести общую функциональную схему СДУ со спецификацией используемых элементов в СДУ (см. п. 3.6 приложения 1). Кроме того, может быть введен раздел технико-экономического обоснования выбранного решения или дан стоимостный расчет разработанной СДУ.

ПРИЛОЖЕНИЕ 3

НОМОГРАММЫ ВЫЧИСЛЕНИЯ ^ож,(п) ПРИ ЭКСПОНЕНЦИАЛЬНОМ РАСПРЕДЕЛЕНИИ *ti*

Номограммы* для вычисления *tow.i[n)* при экспоненциальном распределении *ti* приведены на рис. П.1.

По номограм1.мам легко определить среднее время ожидания при *n* АФБ,— *tox.i(n),* если задана .нагрузка У в эрлангах (ч-зан/ч) и среднее время выполнения алгоритма в АФБ;—^. Для этого необходимо отложить на оси абсцисс заданное значение нагрузки и из этой точки провести вертикальную линию до пересечения с кривой, соответствующей заданному числу АФБ{, т. е. *n.* Проводя далее от точки пересечения горизонтальную линию, на оси ординат определяем значение *to».i(ri')/ti.* Чтобы вычислить tow.i(n), необходимо полученное число разделить на *ti* — среднее значение времени выполнения АФЬ,-

Среднее время ожидания начала обслуживания можно также определить по формуле

<ож ,(")==<,/(»— *Y).*

* Номограммы взяты из книги: Telephone Traffic Theory, Tables and Charts. P. 1.—Siemens Axiengesellschaft, Berlin. — Munchen, 1970, p. 420.

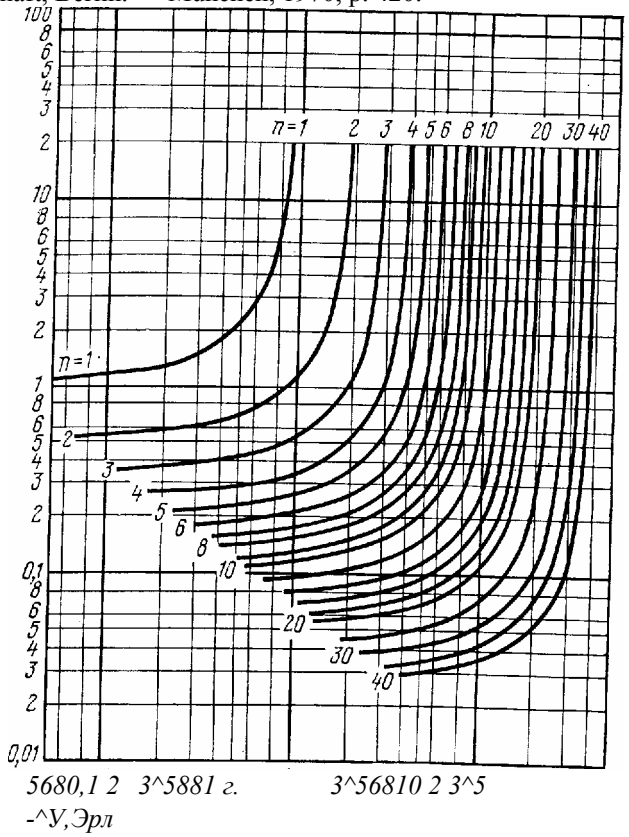


Рис. П.1

1. **Ершова Э. Б., Рогинский В. Н., Маркин Н. П.** Основы дискретной автоматики в электросвязи.— **М.:** Связь, 1980.—232 с.
2. **Лазарев В. Г., Пийль Е. И., Турута Е. Н.** Программное управление на узлах коммутации.—М.: Связь, 1978.—264 с.

3.Лазарев **В. Г., Пийль Е. И.** Синтез управляющих автоматов.—М.: Энергия, 1978.—408 с.

4. Лазарев В. Г., Пийль Е. И., Турута Е. Н. Построение программируемых управляющих устройств.— М.: Энергоатомяздат, 1984.—192с.
5. Автоматизация проектирования цифровых устройств,—Л.: Судостроение, 1979.—264 с.
6. Преснухин Л. Н., Шахнов В. А., Кустов В. А. Основы конструирования микроэлектронных вычислительных машин. — М.: Высшая школа, 1976 — 408 с-
7. Закревский А. Д. Алгоритмы синтеза дискретных автоматов.—М.: Наука, 1971.-512 с.
8. Глушков В. И., Капитонова Ю. В., Летичевский А. А. Автоматизация проектирования вычислительных машин. — Киев: Наукова Думка, 1975. —

Применение вычислительных **машин** для проектирования цифровых устройств/Под ред. Н. Я. Матюхина.—М.: Сов. радио, **1968.**
Гаврилов М. А., Девятков В. В. Пупырев Е. И. Логическое проектирование дискретных автоматов.—М.: Наука, 1977.—332 с.
Автоматизация проектирования сложных логических структур/Под ред. В. А. Горбатова.—М.: Энергия, 1978.—362 с.
Проектирование цифровых вычислительных машин.—М.: Высшая школа, 1972.—344с.
Юрин О. Н. Единая система автоматизации проектирования ЭВМ.—М,: Сов. радио, 1976.—176 с.
Евреинов Э. В., Прангишвили И. В. Цифровые автоматы с настраиваемой структурой.—М.: Энергия, 1976.—240 с.
Логические матрицы, программируемые в условиях эксплуатации. — Электроника, 1975, № 6, с. 89—90.
Иванова О. Н., Лазарев В. Г., Пийль Е. И. Синтез электронных схем дискретного действия.—М.: Связь, 1964.—176 с.
Якубайтис Э. А. Логические автоматы и микромодули. — Рига: Зинатне, 1975.—259 с.
Лазарев В. Г. Алгоритмический синтез асинхронных автоматов в базисе универсальной ячейки,—В кн.: Построение устройств управления сетями связи.—М.: Наука, 1977, с. 60—64.
Поспелов Д. А. Логические методы анализа и синтеза схем.—М.: Энергия, 1974.—368 с.
Однородные структуры: Анализ. Синтез. Поведение/В. И. Варшавский, В. Б. Мараховский, В. А. Песканский, Л. Я. Розенблюм.—М.: Энергия, 1973.—1152 с.
Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы.—М.: Радио и связь, 1981.—328 с.
Зорева Л. Н., Лазарев В. Г. Метод построения микропрограммного автомата в матричной однородной среде.—В кн.: Сети связи и дискретные устройства управления.—М.: Наука, 1976, с. 91—100.
Прангишвили И. В. Микропроцессоры и микро-ЭВМ. — М.: Энергия, 1979.— 232 **с.**
Каган Б. М., Стешин В. В. Микропроцессоры в цифровых системах.—М.: Энергия, 1Q79.— 192 с.
Лившиц Б. С., Пшеничников А. П., Харкевич А. Д. Теория телетрафика.— М.: Связь, 1979.—224 с.
Донианц В. Н. Оптимизация статического распределения программ в сети микро-ЭВМ.—В кн.: Синтез управляющих устройств на основе микропроцессоров и однородных сред.—М.: Наука, 1980, с. 20—23.
Зеленков Г., Панов В., Попов С. Первый шаг.—Радио, 1982, № 9, с. 33—36.
Зеленков Г., Панов В., Попов С. Система команд микропроцессора КР 580ИКЖ— Радио, 1982, № 10, с. 24—28.
Зеленков Г., Панов В., Попов С. Знакомство с программированием. — Радио, 1982, № 11, с. 38—11.
Зеленков Г., Панов С. Знакомство продолжается.—Радио, 1982, *№* 12, с. 31^-34.
Евреинов Э. В. Однородные вычислительные системы, структуры и среды. — М.: Радио и связь, 1981.—208 с.
Баранов С. И., Синев В. Н. Автоматы и программируемые матрицы. — Минск: Вышэйшая школа, 1980.—136 с.
Лутов **М. Ф.,** Жарков М. А., Юнаков П. А. Квазиэлектронные и электронные АТС.—М.: Радио и связь, 1982.—264 с. Ферриды. Библиотека по автоматике.—М.: Энергоатомиздат, 1981.—88 с.

Оглавление