АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ БИС

КНИГА 2

Функционально-логическое проектирование БИС

Под редакцией
лауреата
Государственной премии СССР,
доктора технических наук,
профессора Г. Г. Казеннова

МОСКВА «ВЫСШАЯ ШКОЛА» 1990

ББК 32.844.1
А 22
УДК 621.38

Рекомендовано Государственным комитетом СССР по народному образованию для использования в учебном процессе студентами высших технических учебных заведений

Рецензенты: чл.-кор. АН СССР Б. В. Баталов (Научно-исследовательский институт систем автоматизированного проектирования РЭА и СБИС АН СССР); кафедра микроэлектроники Московского инженерно-физического института (зав. кафедрой — д-р техн. наук, проф. А. В. Шальнов)

А $\frac{2004060000-086}{001(01)-90}$ 145—89

ISBN 5-06-000093-1

СПИСОК СОКРАЩЕНИЙ

БД — банк данных
БИС — большая интегральная схема
БМК — базовый матричный кристалл
ДНФ — дизъюнктивная нормальная форма
ИС — интегральная схема
КФУ — комбинационный функциональный узел
КА — конечный автомат
КНФ — конъюнктивная нормальная форма
ЛЭ — логический элемент
МДНФ — минимальная дизъюнктивная нормальная форма
МКНФ — минимальная конъюнктивная нормальная форма
ПФУ — последовательностный функциональный узел
САПР — система автоматизированного проектирования
СДНФ — совершенная дизъюнктивная нормальная форма
СКНФ — совершенная конъюнктивная нормальная форма
СУБД — система управления базой данных
ТПР — таблица проверки работоспособности
ТФУ — типовой функциональный узел
ФЛМ — функционально-логическое моделирование
ФЛП — функционально-логическое проектирование
ФУ — функциональный узел
Э1 — укрупненная схема
Э2 — функциональная схема
Э3 — логическая схема

ВВЕДЕНИЕ

При разработке современных цифровых устройств широкое применение находят большие интегральные схемы (БИС). Для сокращения сроков и повышения качества их проектирования созданы различные типы промышленных систем автоматизированного проектирования (САПР), обеспечивающих сквозной цикл разработки БИС, к числу которых следует отнести САПР ТОПАЗ-3000, ПРОЛОГ, КОМПАС-82, АСП-6М, SL-2000 и др. Если первые сквозные САПР БИС разрабатывались на базе ранее созданных независимых подсистем, реализующих отдельные этапы проектирования БИС, то современные САПР разрабатываются с общих методологических позиций, обеспечивающих в первую очередь единство лингвистического и информационного обеспечения САПР. В настоящее время САПР являются эффективным средством разработки БИС различных типов.

В книге рассмотрены основные проблемы автоматизации одного из основных этапов проектирова-

ния БИС — функционально-логического проектирования (ФЛП). Данный этап является достаточно сложным и трудоемким процессом, остающимся еще во многом эвристическим по своему характеру, так как большая доля работ по синтезу схем выполняется не ЭВМ, а человеком. Важность этапа ФЛП определяется его местом в общем процессе разработки БИС. Являясь начальным этапом, функционально-логическое проектирование во многом определяет качество БИС. В целом функционально-логическое проектирование БИС — это комплексный процесс принятия множества решений, требующий от разработчиков разносторонних знаний в области вычислительной техники, микросхемотехники, методологии построения и применения САПР, проектирования и производства БИС. Поэтому подготовка высококвалифицированных инженеров — разработчиков БИС возможна только в процессе реального проектирования на промышленной САПР.

Структура книги, состав и методика изложения материала ориентированы на индивидуальную самостоятельную работу с целью формирования у обучаемых знаний о технологии создания средств автоматизации ФЛП, а также практических умений решения различных задач ФЛП БИС на основе САПР. Алгоритмы и методы автоматизации, используемые в системах функционально-логического проектирования, достаточно подробно описаны в ряде монографий и учебных пособий. Поэтому в книге основное внимание уделено не получившему достаточного освещения в литературе одному из важнейших вопросов автоматизации ФЛП БИС — единству и неразрывности лингвистического и информационного обеспечения современных САПР. Показаны место и роль банка данных в структуре сквозной САПР как основного средства интеграции различных этапов проектирования БИС. При этом лингвистическое обеспечение служит единым средством описания БИС на различных уровнях ФЛП, управления этим процессом, накопления в банке данных системы типовых решений и адаптации САПР к различным объектам проектирования. Программное обеспечение функционально-логического моделирования выступает как средство отладки логического проекта БИС и аттестации его качества.

Главы 1 и 2 являются введением в функционально-логическое проектирование БИС. В них изложены математические основы логического проектирования, дана характеристика БИС как объекта ФЛП, рассматривается иерархия моделей и уровней проектирования, отмечаются особенности этого процесса и возможность реализации двух стратегий: восходящего и нисходящего ФЛП. Основное внимание в гл. 3 и 4 уделено созданию средств автоматизации ФЛП, в том числе разработке специализированного банка данных, входного языка проектирования, настройке системы на заданный объект проектирования и др., а также методике и технологии разработки логического проекта БИС с помощью САПР ФЛП-3000. Содержание гл. 5 и 6 направлено на формирование у обучаемых умений и навыков проектирования различных функциональных узлов (фрагментов БИС) на основе методики формализованного синтеза и системы ФЛП-3000.

Глава 1

ОСОБЕННОСТИ АВТОМАТИЗИРОВАННОГО ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ БИС

Большая интегральная схема содержит на одном кристалле десятки тысяч компонентов, что позволяет реализовать достаточно сложные функции (обработка и хранение информации, управление внешними устройствами и др.). По функциональной сложности БИС приближаются к цифровым устройствам на дискретных компонентах, ЭВМ или их отдельным частям. Методология проектирования БИС и методология проектирования цифровых устройств во многом совпадают, особенно на ранних этапах — при разработке архитектуры и логики функционирования. Так, при ФЛП БИС и ФЛП цифровых устройств используется один и тот же математический аппарат — булева алгебра, теория автоматов, теория графов и др.

В области автоматизации ФЛП цифровых устройств достигнуты значительные успехи. Созданы и успешно эксплуатируются САПР цифровых устройств, такие, как РАПИРА-2, ПРАМ-2, ПУЛЬС, КОНДИЦИЯ. Тем не менее при ФЛП БИС встают новые задачи, решение которых связано с известными трудностями и требует новых подходов, прежде всего необходимо использовать более совершенные методы и средства автоматизации проектирования, учитывающие особенности объекта проектирования — БИС.

§ 1.1. Характеристика БИС как объекта функционально-логического проектирования

Известно, что БИС в отличие от цифровых устройств — неремонтопригодные изделия, поэтому для них становятся неприемлемыми традиционные способы разработки, такие, как макетирование и постепенная доводка опытного образца путем физической замены электрорадиоэлементов или корректировки схемы соединений между ними. Основной особенностью автоматизированного проектирования БИС в отличие от проектирования устройств на дискретных электрорадиоэлементах является требование бездефектности выполнения проектных работ. Обнаружение и исправление ошибок, допущенных при ФЛП, требует значительных усилий разработчиков, непроизводительных расходов и затрат времени на последующих этапах проектирования и при изготовлении БИС. Ввиду этого необходимость выявления и исправления

ошибок при ФЛП БИС становится очевидной. Данная задача может быть решена на основе высокоэффективных методов и средств автоматизации ФЛП. Однако в настоящее время этот процесс реализуется в автоматизированном режиме, когда синтез логических схем выполняется вручную эвристическим способом или на основе формализованных методов, а задачи анализа работоспособности БИС решаются на соответствующих уровнях ФЛП с помощью программ моделирования. В настоящее время существуют достаточно эффективные алгоритмы и программные средства логического синтеза БИС, построенных на программируемых логических матрицах либо на элементах И-НЕ ИЛИ-НЕ. Однако автоматический синтез логических схем в произвольно заданном базисе логических элементов еще находится в стадии исследовательских работ.

БИС имеют ограниченное число внешних выводов, в то время как реализуемые ими функции весьма сложны. С увеличением степени интеграции БИС отношение числа выводов к числу компонентов на кристалле уменьшается и задача тестирования усложняется, так как для подачи и наблюдения сигналов в различных точках схемы не хватает внешних выводов. Число точек наблюдения ограничивается только входными и выходными выводами БИС. Поэтому в настоящее время значительное внимание уделяется поиску и разработке новых методов оценки качества тестов, генерации тестов, активно развиваются подходы по разработке легкотестируемых схем.

Некоторые особенности БИС как объекта ФЛП проявляются на этапах топологического проектирования, изготовления и эксплуатации в составе аппаратуры. Например, наличие паразитных емкостей соединительных проводников в топологии БИС приводит к временной задержке сигналов, следствием чего являются такие эффекты, как состязания сигналов, риски сбоя, искажения формы и запаздывание синхронизирующих импульсов. В результате может нарушиться работоспособность БИС. К числу других дестабилизирующих факторов, влияющих на нормальную работу БИС, можно отнести изменение температуры окружающей среды, колебания напряжения источников питания, внешние нагрузки и др. Поэтому по завершении процесса проектирования БИС необходимо проверить его результаты с учетом действия внешних факторов с помощью специальных программ аттестации, входящих в САПР.

Таким образом, особенности БИС как объекта функционально-логического проектирования (неремонтопригодность, сложность тестирования и аттестации) вынуждают разработчиков этих изделий и специалистов в области САПР активно вести поиск новых методов и средств проектирования, постоянно совершенствовать технологию разработки БИС, чтобы обеспечить высокую производительность труда инженеров и безошибочность проектирования. Многие трудности преодолены только в сквозных интегрированных САПР, включающих элементы экспертных оценок.

§ 1.2. Иерархия моделей БИС на этапе функционально-логического проектирования

Согласно блочно-иерархическому подходу к проектированию любой достаточно сложный объект может иметь иерархию уровней сложности. Так, радиоэлектронные средства по функциональной сложности подразделяются на комплексы (радиоизмерительные и радиоуправляющие системы и др.), функциональные устройства (передатчики, приемники, микропроцессоры и др.), функциональные узлы (регистры, счетчики и др.), функциональные элементы (усилители, логические элементы, триггеры и др.), электрорадиоэлементы (транзисторы, резисторы, конденсаторы, полупроводниковые структуры и др.). В свою очередь, элементами конструктивной иерархии ЭВМ могут быть: стойки, рамы, панели, типовые элементы замены, БИС.

В общем случае при определении уровней сложности какого-либо объекта производят его деление на составные части. Так, если на некотором уровне i_1 имеем объект S, то на соседнем, более низком уровне i_2 происходит разделение S на части: $s_1, ..., s_j$, и каждая часть рассматривается с большей степенью детализации, чем на уровне i_1.

Для БИС условно можно выделить шесть иерархических уровней: 1) БИС; 2) функциональный блок (ФБ); 3) функциональный узел (ФУ); 4) функциональная ячейка (ФЯ); 5) логический элемент (ЛЭ); 6) компонент. На первом уровне БИС в целом представляется как «черный ящик» с соответствующими входами и выходами. На втором БИС задается архитектурой в виде укрупненной схемы (Э1), элементами которой являются функциональные блоки. На третьем БИС можно рассматри-

вать в виде функциональной схемы (Э2), элементы которой — функциональные узлы (регистры, счетчики, сумматоры, дешифраторы и др.). На четвертом уровне БИС представляется в виде совокупности функциональных ячеек. Под функциональной ячейкой понимается топологически законченный аналоговый или цифровой элемент библиотеки. На пятом уровне БИС задается в виде логической (принципиальной) электрической схемы (Э3), элементами которой могут быть функциональные ячейки и/или логические элементы. Под логическим элементом понимается простейшая функционально законченная часть ячейки, например элементы типа И, ИЛИ, И-НЕ, ИЛИ-НЕ. Необходимость представления БИС в виде схемы Э3 (как совокупности элементов) возникает при логическом моделировании, а в виде схемы Э3 (как совокупности ячеек) — при решении задач топологического проектирования БИС. На шестом уровне БИС можно представить в виде электрической схемы, как совокупности компонентов (транзисторов, резисторов, конденсаторов). Однако практически необходимость такого представления БИС возникает крайне редко.

Обычно при функционально-логическом проектировании БИС на основе базового матричного кристалла (БМК) разработчик в основном использует четыре уровня иерархического представления:

1) большая интегральная схема,
2) функциональный блок,
3) функциональный узел,
4) ячейка.

Последний уровень является основным уровнем представления матричных БИС, а возможные более низкие уровни разработчика обычно не интересуют. Однако в САПР при логическом моделировании и схематическом анализе для более точного отображения процессов, протекающих в БИС, имеется возможность автоматического перехода на уровень логических элементов или компонентов.

Таким образом, цифровая БИС в процессе функционально-логического проектирования представляется тремя видами схем: укрупненной схемой (Э1), функциональной схемой (Э2) и логической схемой (Э3).

Рассмотрим иерархию моделей БИС при функционально-логическом проектировании (табл. 1.1), в качестве которых выступают три вида схем (Э1—Э3), являющихся различными формами представления БИС.

Таблица 1.1

Иерархия моделей БИС при функционально-логическом проектировании

Уровни проектирования	Алгоритмические модели	Структурные модели
Архитектурный	БИС	
	Блок	Э1
Функциональный	Узел	Э2
Логический	Ячейка	Э3

Так как ФЛП БИС — сложный и трудоемкий процесс, то он разбивается на отдельные этапы — уровни проектирования: *архитектурный, функциональный, логический.* На каждом уровне ФЛП решаются определенные задачи, расширяющие и углубляющие знания о проектируемой БИС.

Различают алгоритмические и структурные модели БИС.

Алгоритмическая модель представляется в виде «черного ящика», и с ее помощью устанавливаются логические связи между входными и выходными переменными в соответствии с алгоритмом функционирования БИС. Алгоритмическая модель для каждого уровня ФЛП характеризует работу соответственно БИС, функционального блока и функционального узла.

Структурная модель БИС на каждом уровне проектирования определяется совокупностью соответствующих элементов (блоков, узлов или ячеек) и связей между ними. Каждый элемент структурной модели описывается алгоритмической моделью. Например, на архитектурном уровне проектирования элементами структурной модели БИС являются алгоритмические модели функциональных блоков. Обычно структурная модель обладает более высокой точностью отображаемых процессов, чем алгоритмическая модель, но уступает ей по быстродействию. Поэтому на каждом уровне проектирования сначала работают по алгоритмической модели, оперативно получая соответствующую информацию о функционировании проектируемой БИС, а затем — по структурной модели, уточняя и корректируя полученную ранее информацию.

§ 1.3. Анализ процесса функционально-логического проектирования

При функционально-логическом проектировании БИС разработчик выполняет комплекс работ по синтезу и анализу укрупненной функциональной и логической схем, функциональных тестов и таблиц логической проверки работоспособности. Задачи синтеза схем и функциональных тестов на всех уровнях ФЛП очень трудно поддаются автоматизации, поэтому обычно они выполняются вручную на основе знаний, опыта, интуиции и творческой изобретательности разработчика. Задачи верификации (проверки) принятых решений на архитектурном, функциональном и логическом уровнях выполняются с помощью различных описаний исследуемого объекта и функционально-логического моделирования.

В общем случае при ФЛП БИС могут быть реализованы *две стратегии проектирования*: *восходящее* (*снизу вверх*) *и нисходящее* (*сверху вниз*). Выбор той или иной стратегии в основном определяется постановкой задачи на проектирование, объемом исходной информации и техническими требованиями к БИС.

Восходящее проектирование. Пусть требуется разработать БИС на БМК, которая должна быть аналогом некоторого цифрового устройства, ранее разработанного на ИС малой и средней степени интеграции. Иными словами, задана некоторая логическая схема ЭЗ, элементами которой являются корпусные ИС, и на ее основе необходимо разработать логический проект БИС. Задан также БМК конкретного типа. При такой постановке задачи имеем большой объем исходной информации, так как известен в деталях не только алгоритм функционирования проектируемой БИС, но и ее отработанная исходная логическая схема ЭЗ, построенная, правда, на другом элементном базисе. Поэтому разработку логического проекта БИС целесообразно проводить согласно стратегии «снизу вверх», которая достаточно проста в «реализации». Прежде всего предполагается создать на ячейках БМК библиотеку функциональных узлов — аналогов корпусных ИС, а затем разработать и отладить логическую схему БИС на основе этой библиотеки. Здесь максимально используется накопленный ранее опыт (схемные и функциональные решения), сокращаются сроки и стоимость создания проекта БИС на БМК.

Нисходящее проектирование. Стратегия создания логического проекта «сверху вниз» более сложна и трудоемка в реализации, чем разработка логического проекта по стратегии «снизу вверх», в силу большой неопределенности исходной информации. Применение нисходящего проектирования целесообразно в случае, когда известен только алгоритм функционирования БИС и ее желаемые технические характеристики, а информация о структуре логической схемы БИС отсутствует. Поэтому для разработки логической схемы необходимо предварительно провести работы на этапах архитектурного и функционального проектирования, т. е. разработать сначала схемы Э1 и Э2. Таким образом, нисходящее проектирование БИС сводится к такой последовательности действий: 1) разрабатывается архитектура БИС (схема Э1) как совокупность функциональных блоков; 2) каждый блок описывается алгоритмической моделью и проводится моделирование схемы Э1 на регистровом уровне. По результатам моделирования уточняются и конкретизируются технические требования на БИС, формируются требования к блокам, уточняется алгоритм функционирования БИС в целом; 3) на основе библиотеки ячеек БМК проводится разработка функциональной схемы (Э2), а затем логической схемы БИС (Э3). Стратегия нисходящего проектирования предполагает иерархию не только алгоритмических и структурных моделей БИС, используемых на уровнях архитектурного, функционального и логического проектирования БИС, но и иерархию тестов для проверки работоспособности ячеек БМК, функциональных узлов, блоков и БИС в целом.

Обе стратегии разработки логического проекта БИС, как и любого сложного объекта, удачно согласуются с блочно-иерархическим подходом, согласно которому структура проектируемой БИС должна разбиваться по функциональному признаку на отдельные фрагменты (функциональные блоки и узлы). Проектирование различных по сложности фрагментов можно проводить независимо друг от друга. Таким образом, при ФЛП преодолевается проблема функциональной сложности БИС в целом.

При разработке логического проекта БИС большой объем работ падает на его верификацию, которая выполняется с помощью программ функционально-логического моделирования на регистровом и логическом уровнях. Например, с помощью программ функционально-

логического моделирования матричных БИС решаются следующие задачи:

1) верификация ячеек БМК, типовых функциональных узлов;

2) уточнение алгоритма функционирования БИС, ее структуры и спецификаций элементов структуры;

3) отладка логической схемы отдельных элементов структуры и БИС в целом;

4) отладка функциональных тестов и таблиц логической проверки работоспособности (сокращенно таблиц проверки работоспособности — ТПР);

5) анализ влияния топологии на правильность работы логической схемы;

6) оценка устойчивости проекта БИС к воздействию внешних факторов, таких, как изменение температуры, напряжения источников питания, разброс технологических параметров.

Как уже отмечалось, проектирование БИС — это процесс последовательного накопления и уточнения знаний о будущей БИС (техническое задание, уточненный алгоритм функционирования, структурная схема, логическая схема, топология, ТПР и т. д.). Чем более детально представляется работа БИС, тем сильнее возрастают требования к адекватности моделирования — степени соответствия результатов моделирования истинному поведению исследуемой БИС. Например, при моделировании алгоритма функционирования БИС достаточно отразить реакцию БИС на внешние воздействия и проверить ее на соответствие техническому заданию вне зависимости от конкретной структурной реализации БИС. На завершающей стадии работы над проектом БИС (например, при имитации температурных испытаний или моделирования с учетом топологии) адекватность моделирования в значительной степени определяет качество проекта БИС в целом. Однако следует учитывать, что повышение адекватности моделирования, как правило, ведет к увеличению затрат машинного времени и требуемого объема оперативной памяти. Поэтому необходимый и достаточный уровень адекватности моделирования зависит от класса исследуемых БИС, вида решаемых задач, имеющихся в распоряжении вычислительных ресурсов, возможностей САПР и т. д.

Одним из основных направлений эффективного решения проблемы повышения точности и быстродействия функционально-логического моделирования БИС явля-

ется разработка программ с регулируемой адекватностью моделирования. В общем случае адекватность и точность моделирования зависят от выбранного метода и алгоритма моделирования, используемых моделей сигналов, состава параметров моделей логических элементов, способов учета временных соотношений между сигналами.

Известны различные методы и алгоритмы логического моделирования. Основные из них: синхронные и асинхронные, сквозные и событийные. По алгоритмам синхронного моделирования осуществляется моделирование логических схем с нулевыми или единичными задержками логических элементов (ЛЭ). Эти алгоритмы обладают высоким быстродействием, но не позволяют получать временных диаграмм работы логических элементов схемы, решать другие задачи, связанные с учетом временных соотношений между сигналами. Алгоритмы асинхронного моделирования реализуют моделирование логических схем с учетом задержек ЛЭ. В основе алгоритмов событийного моделирования, в отличие от алгоритмов сквозного моделирования, обработке подлежат только ЛЭ с изменившимися входными значениями сигналов. Анализ работы цифровых схем показывает, что одновременно в активном состоянии находятся лишь 1...10 % всех логических элементов схемы. Поэтому событийное моделирование обладает более высоким быстродействием по сравнению со сквозным. Наибольшее распространение получил алгоритм асинхронного событийного моделирования, соединивший положительные свойства асинхронного и событийного моделирования.

К наиболее часто применяемым моделям сигналов относят: 0 — состояние логического (лог.) нуля; 1 — состояние логической (лог.) единицы; $\times$ — логически неопределенное состояние (значение сигнала неизвестно); Z — высокоимпедансное (отключенное) состояние; / — переключение из 0 в 1; \ — переключение из 1 в 0.

Алфавит моделирования включает определенную совокупность символов и позволяет учесть соответствующие особенности сигналов в схеме. Наибольшее распространение получили следующие алфавиты моделирования: {0, 1} — двоичный; {0, 1, $\times$} — троичный; {0, 1, $\times$, Z} — четырехзначный; {0, 1, $\times$, /, \} — пятизначный; {0, 1, $\times$, /, \, Z} — шестизначный. Отметим, что существуют и другие варианты алфавитов моделирования, однако с ростом значимости алфавита быстро увеличиваются

время моделирования и требуемый объем оперативной памяти.

Модели логических элементов могут обладать совокупностью параметров, которые существенно влияют на адекватность моделирования. В простейшем случае модель ЛЭ может иметь один параметр — время задержки распространения сигнала. В более сложных моделях могут быть учтены формирующие свойства элементов, нагрузка на выходе, пороги срабатывания по входу, выходные сопротивления источников сигнала, длительности фронтов, температурные зависимости и другие параметры.

В настоящее время широкое развитие получают сквозные интегрированные САПР, охватывающие все этапы проектирования и подготовки производства БИС. Разработка подобной САПР требует принятия и обоснования множества решений по реализации математического, лингвистического, информационного, программного и технического обеспечения. Принятие и обоснование каждого решения производят на основании трех важнейших принципов построения САПР:

1) целостность системы — современные САПР содержат отдельные подсистемы, с помощью которых реализуется единый технологический цикл сквозного проектирования, обеспечивающий высокую производительность труда разработчика и бездефектность проектирования БИС. Связи между отдельными подсистемами не должны нарушать целостность всей системы;

2) гибкость системы — САПР должна обладать средствами оперативной настройки на БИС различных типов, технологию изготовления, топологию и т. п.;

3) открытость системы — современная САПР должна обладать максимальной открытостью для модификации и развития. Это требование определяется средой функционирования САПР, которая постоянно изменяется: усложняются объекты проектирования, меняются средства технологической подготовки производства, нормативно-технические документы, развивается и сама методология проектирования БИС. Все это нередко приводит к неэффективному применению ранее разработанных средств САПР, к необходимости их доработки или замены. САПР, не обладающая свойством открытости, не может развиваться и приспосабливаться к изменяющейся среде функционирования.

Реализация принципов построения САПР во многом

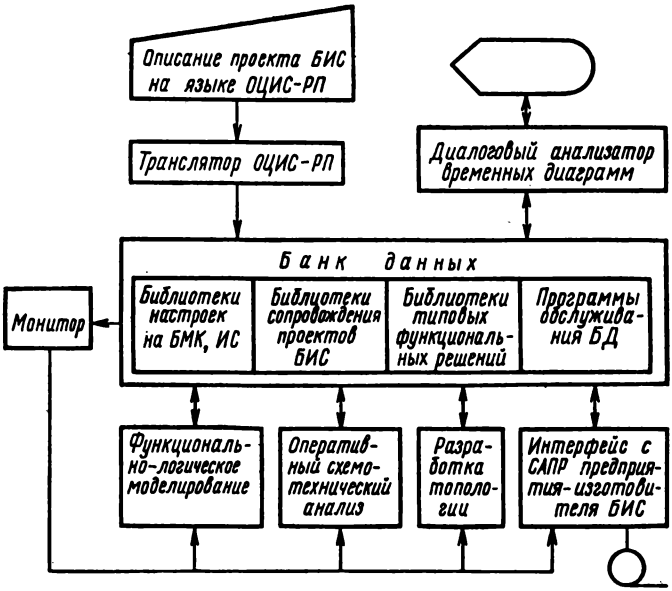


Рис. 1.1. Структура и состав САПР ФЛП-3000

зависит от того, насколько удачно, гибко и надежно построена информационная среда функционирования САПР: банк данных и лингвистическое обеспечение (интерфейс между разработчиком БИС и системой).

В качестве примера сквозной интегрированной САПР, позволяющей эффективно решать задачи функционально-логического проектирования БИС, рассмотрим систему ФЛП-3000. Система предназначена для проектирования матричных БИС и цифровых устройств на их основе и содержит следующие подсистемы (рис. 1.1): транслятор с языка ОЦИС-РП — языка иерархического описания цифровых схем на уровне регистровых передач и функциональных тестов; специализированный банк данных; монитор запуска подсистем проектирования; подсистему функционально-логического моделирования (ФЛМ); диалоговый анализатор временных диаграмм; подсистему оперативного схемотехнического анализа электрических схем; подсистему разработки топологии БИС; интерфейс с САПР предприятий — изготовителей БИС.

Язык ОЦИС-РП относится к классу многоуровневых структурно-функциональных языков регистровых передач. Язык ориентирован на инженера-разработчика БИС и позволяет описывать алгоритм функционирования и структуру цифрового устройства, БИС и ее фрагментов. С его помощью осуществляется также настройка САПР ФЛП-3000 на конкретный БМК и управление запуском соответствующих подсистем проектирования.

Система имеет единый специализированный банк данных (БД), обеспечивающий информационный интерфейс между подсистемами и

содержащий информацию о настройке системы на различные типы БМК, а также промежуточные и окончательные результаты работы подсистем.

Монитор запуска подсистем проектирования осуществляет вызов обрабатывающих программ соответствующих подсистем в последовательности, указанной пользователем средствами языка ОЦИС-РП при запуске системы.

Подсистема ФЛМ содержит ряд обрабатывающих программ и является одной из основных подсистем САПР ФЛП-3000. Она обеспечивает асинхронное событийное моделирование на различных уровнях верификации проекта БИС или цифрового устройства, позволяет проводить совместное моделирование БИС или ее фрагментов, одна часть которых представлена алгоритмическими, а другая — структурными моделями. Использование разных по точности и сложности моделей диктуется необходимостью выбора разумного соотношения точность — время и память — время. Повышенная адекватность моделирования БИС достигается в подсистеме ФЛМ учетом различных параметров: узловых емкостей и выходных сопротивлений источников сигналов, порогов срабатывания, задержек и формирующих свойств логических элементов или функциональных узлов, задержек в межсоединениях топологии БИС, эквивалентных сопротивлений и емкостей контрольно-измерительного оборудования, напряжения источников питания, температуры окружающей среды и др. В подсистеме ФЛМ проводится раскрытие описания схемы до заданного уровня детализации, логико-временное моделирование, выявляются риски сбоя и критические состязания сигналов в проектируемых схемах, осуществляется контроль нагрузочной способности логических элементов, оценка качества тестов.

Результатом работы подсистемы ФЛМ являются отлаженная логическая схема БИС, на основе которой синтезируются топология и таблица логической проверки работоспособности.

Подсистема оперативного схемотехнического анализа служит для одновариантного схемотехнического анализа электрических схем и используется на этапе настройки САПР ФЛП-3000 на библиотеку элементов с целью определения динамических параметров базовых логических элементов ячеек БМК. Рассчитанные параметры входят в алгоритмическую модель базовых логических элементов, используемых при ФЛМ БИС и ее фрагментов.

Подсистема разработки топологии содержит прикладные программы для решения различных задач: 1) автоматического синтеза топологии БИС; 2) восстановления логической схемы БИС из топологии; 3) контроля соответствия восстановленной и исходной логической схемы БИС; 4) расчета задержек в межсоединениях ячеек для последующей аттестации проекта БИС.

Диалоговый анализатор временных диаграмм предназначен для вызова из банка данных и оперативного отображения на экране дисплея результатов моделирования.

Интерфейс с САПР предприятия-изготовителя содержит программы, которые автоматически транслируют отлаженную логическую модель БИС с языка ОЦИС-РП на язык САПР предприятия-изготовителя и записывают на магнитную ленту информацию о проекте БИС с учетом требований карты заказа на изготовление. Обязательными компонентами карты заказа являются описание топологии БИС для изготовления фотошаблонов и таблица логической проверки работоспособности для автоматизированного контрольно-измерительного оборудования.

Глава 2

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ БИС

Для выполнения работы, связанной с функционально-логическим проектированием БИС и других цифровых устройств, разработчику необходимы знания математических основ проектирования — алгебры логики и теории цифровых автоматов. Ясное понимание принципов и методов, лежащих в основе формирования моделей БИС и их фрагментов, исключительно важно для каждого, кто предполагает заниматься ФЛП БИС. В алгебре логики (в отличие от обычной алгебры) переменные принимают только два значения 0 или 1. Представление чисел символами 0 и 1 полностью соответствует характеру работы цифровых БИС: 0 — цепь разомкнута и 1 — цепь замкнута; 0 и 1 — напряжение низкого и высокого уровня соответственно. Разработка основ алгебры логики связана с именем ирландского математика Дж. Буля (1815—1864), поэтому существуют и другие названия: алгебра Буля или булева алгебра.

§ 2.1. Основные понятия и законы алгебры логики

Логическая (двоичная) переменная характеризует простое высказывание, которое содержит одну законченную мысль. Она обозначается буквой x и может принимать значения 0 или 1.

Логическая функция — это сложное высказывание, состоящее из нескольких простых, связанных между собой соединительными союзами. Она записывается аналитически в виде $Y = f(x_1, x_2, ..., x_n)$, где x_i — двоичная переменная, $x_i \in \{0, 1\}$; $Y \in \{0, 1\}$.

Входной набор — это определенная комбинация значений двоичных переменных в логической функции. Максимальное число входных наборов определяется выражением $m = 2^n$, где n — число переменных. Максимальное число логических функций n переменных определяется выражением $N = 2^{2^n}$.

Таблица истинности — это представление логической функции в виде таблицы, в левой части которой записываются входные наборы, а в правой — соответствующие им значения функции. *Полностью определенная функ-*

ция — это логическая функция, имеющая определенные значения 0 или 1 на всех входных наборах. *Частично определенная функция* — это логическая функция, значения которой определены не на всех входных наборах. *Рабочие наборы* — это входные наборы, для которых логическая функция полностью определена. *Безразличные наборы* — это входные наборы, для которых логическая функция не определена. Частично определенную функцию можно сделать полностью определенной (доопределить), приписав безразличным наборам какие-либо значения функции.

Функции одной переменной. Так как $n = 1$ и $m = 2$, то максимальное число функций одной переменной $N = 4$, а именно: $f_0 = 0$ — нулевая функция; $f_1 = 1$ — единичная функция; $f_2 = x_1$ — функция повторения x_1; $f_3 = \bar{x}_1$ — функция отрицания, или инверсия x_1 (читается «f_3 есть не x_1»).

Функции двух переменных являются основными функциями алгебры логики. Полный состав, название и алгебраические выражения функций двух переменных представлены в табл. 2.1. Из алгебраических выражений этих функций видно, что для записи любой функции достаточно иметь три: отрицание (f_{10}, f_{12}), конъюнкцию

Таблица 2.1

Функции двух переменных

$x_1=0$ $x_2=0$	0 1	1 0	1 1	Название	Алгебраическое выражение
0	0	0	0	Нулевая	$f_0=0$
0	0	0	1	Конъюнкция	$f_1=x_1x_2$
0	0	1	0	Запрет x_2	$f_2=x_1 \leftarrow x_2=x_1\bar{x}_2$
0	0	1	1	Повторение x_1	$f_3=x_1$
0	1	0	0	Запрет x_1	$f_4=x_2 \leftarrow x_1=\bar{x}_1x_2$
0	1	0	1	Повторение x_2	$f_5=x_2$
0	1	1	0	Исключающее ИЛИ	$f_6=x_1 \oplus x_2=x_1\bar{x}_2 \vee \bar{x}_1x_2$
0	1	1	1	Дизъюнкция	$f_7 = x_1 \vee x_2$
1	0	0	0	Стрелка Пирса	$f_8 = x_1 \downarrow x_2 = \overline{x_1 \vee x_2}$
1	0	0	1	Равнозначность	$f_9 = x_1 \sim x_2 = x_1x_2 \vee \bar{x}_1\bar{x}_2$
1	0	1	0	Отрицание x_2	$f_{10}=\bar{x}_2$
1	0	1	1	Импликация от x_2 к x_1	$f_{11}=x_2 \rightarrow x_1=x_1 \vee \bar{x}_2$
1	1	0	0	Отрицание x_1	$f_{12}=\bar{x}_1$
1	1	0	1	Импликация от x_1 к x_2	$f_{13}=x_1 \rightarrow x_2=\bar{x}_1 \vee x_2$
1	1	1	0	Штрих Шеффера	$f_{14} = x_1/x_2 = \overline{x_1x_2}$
1	1	1	1	Единичная	$f_{15} = 1$

(f_1) и дизъюнкцию (f_7), которые образуют функционально полную систему функций. Минимальные функционально полные системы содержат одну функцию (И-НЕ или ИЛИ-НЕ) или две [(НЕ, И) или (НЕ, ИЛИ)].

Аксиомы алгебры логики:

$$0 \vee x_1 = x_1, \qquad x_1 x_1 = x_1,$$
$$0 \cdot x_n = 0, \qquad x_1 \vee x_1 = x_1,$$
$$0 \cdot x_1 x_2 \ldots x_n = 0, \qquad x_1 \vee x_1 = 1,$$
$$1 \vee x_1 = 1, \qquad x_1 \bar{x}_1 = 0,$$
$$1 \cdot x_1 = x_1, \qquad \bar{\bar{x}}_1 = x_1,$$
$$1 \vee x_1 \vee x_2 \vee \ldots \vee x_n = 1, \qquad \bar{\bar{x}}_1 = \bar{x}_1.$$

Законы алгебры логики:

1) переместительные (коммутативные):
$x_1 \vee x_2 = x_2 \vee x_1$, $x_1 x_2 = x_2 x_1$;
2) сочетательные (ассоциативные):
$x_1 \vee x_2 \vee x_3 = x_1 \vee (x_2 \vee x_3) = x_2 \vee (x_1 \vee x_3) =$
$= x_3 \vee (x_1 \vee x_2)$,
$x_1 x_2 x_3 = x_1(x_2 x_3) = x_2(x_1 x_3) = x_3(x_1 x_2)$;
3) распределительные (дистрибутивные):
$x_1(x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3$, $x_1 \vee x_2 x_3 = (x_1 \vee x_2)(x_1 \vee x_3)$;
4) законы де-Моргана:
$\overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n$,
$\overline{x_1 x_2 \ldots x_n} = \bar{x}_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_n$.

В алгебре логики существуют понятия: элементарная конъюнкция/дизъюнкция, конституента единицы/нуля, ранг элементарной конъюнкции/дизъюнкции, соседние элементарные конъюнкции/дизъюнкции.

Элементарная конъюнкция/дизъюнкция — это конъюнкция/дизъюнкция входных переменных и/или их отрицания. Любой символ переменной в элементарной конъюнкции/дизъюнкции может встречаться один раз. Например, $f_1(x_1, x_2, x_3, x_4) = x_1 x_2 x_3 x_4$ есть элементарная конъюнкция, а $f_2(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 x_4 x_1$ и $f_3(\bar{x}_1, x_2, \bar{x}_3, x_4) = x_1 x_2 \overline{x_3 x_4}$ не являются элементарной конъюнкцией.

Конституента единицы/нуля — элементарная конъюнкция/дизъюнкция, в которую входят все n переменных в прямом или инверсном виде. Конституенту единицы/нуля часто называют минтермом/макстермом. *Ранг элементарной конъюнкции/дизъюнкции* — число входных переменных в элементарной конъюнкции/дизъюнкции. Так, $f(x_1) = \bar{x}_1$ — это элементарная конъюнкция первого

ранга; $f(x_1, x_2, x_3) = x_1\bar{x}_2x_3$ — элементарная конъюнкция третьего ранга.

Соседние элементарные конъюнкции/дизъюнкции — две элементарные конъюнкции/дизъюнкции одного и того же ранга, если они являются функциями одних и тех же переменных и отличаются только знаком отрицания (инверсии) одной из переменных. Например, конъюнкции $\bar{x}_1x_2x_3x_4$ и $x_1x_2x_3x_4$ являются соседними, так как они — функции одних и тех же переменных и отличаются знаком инверсии только одной переменной x_1.

На основе законов алгебры логики могут быть сформулированы правила преобразования логических выражений.

Правило склеивания для соседних элементарных конъюнкций: логическую сумму двух соседних элементарных конъюнкций некоторого ранга r можно заменить одной элементарной конъюнкцией ранга $r-1$, являющейся общей частью исходных слагаемых. Это правило является следствием распределительного закона первого рода. Например,

$$\bar{x}_1x_2\bar{x}_3x_4 \vee x_1x_2\bar{x}_3x_4 = x_2\bar{x}_3x_4, \quad x_1x_2 \vee x_1\bar{x}_2 = x_1.$$

Правило склеивания для соседних элементарных дизъюнкций: логическое произведение двух соседних элементарных дизъюнкций ранга r можно заменить одной элементарной дизъюнкцией ранга $r-1$, являющейся общей частью исходных конъюнкций. Это правило является следствием распределительного закона второго рода и применяется для упрощения логических выражений. Например:

$$(\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4)(x_1 \vee x_2 \vee \bar{x}_3 \vee x_4) = x_2 \vee \bar{x}_3 \vee x_4,$$
$$(x_1 \vee x_2)(x_1 \vee \bar{x}_2) = x_1.$$

Правило поглощения для элементарных конъюнкций: дизъюнкцию двух элементарных конъюнкций разных рангов, из которых одна является собственной частью другой, можно заменить конъюнкцией, имеющей меньший ранг. Это правило является следствием распределительного закона первого ранга. Например,

$$\bar{x}_1x_2\bar{x}_3x_4 \vee \bar{x}_1x_2 = \bar{x}_1x_2, \quad x_1 \vee x_1x_2 = x_1.$$

Кроме того, справедливы выражения:

$$x_1 \vee \bar{x}_1x_2 = x_1 \vee x_2, \quad \bar{x}_1 \vee x_1x_2 = \bar{x}_1 \vee x_2.$$

Правило поглощения для элементарных дизъюнкций:

логическое произведение двух элементарных дизъюнкций разных рангов, из которых одна является общей частью другой, можно заменить дизъюнкцией, имеющей меньший ранг. Это правило является следствием распределительного закона второго рода и находит широкое применение для упрощения логических функций. Например,

$$(\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4)(\bar{x}_1 \vee x_2) = \bar{x}_1 \vee x_2, \quad x_1(x_1 \vee x_2) = x_1.$$

В алгебре логики существуют две теоремы разложения:

$$f(x_1, x_2, \ldots, x_n) = x_1 f(1, x_2, \ldots, x_n) \vee \bar{x}_1 f(0, x_2, \ldots, x_n), \tag{2.1}$$

$$f(x_1, x_2, \ldots, x_n) = [x_1 \vee f(0, x_2, \ldots, x_n)][\bar{x}_1 \vee f(1, x_2, \ldots, x_n)] \tag{2.2}$$

и четыре соотношения, позволяющие упрощать логические функции:

$$x_1 f(x_1, \bar{x}_1, x_2, x_3, \ldots, x_n) = x_1 f(1, 0, x_2, x_3, \ldots, x_n), \tag{2.3}$$

$$\bar{x}_1 f(x_1, \bar{x}_1, x_2, x_3, \ldots, x_n) = \bar{x}_1 f(0, 1, x_2, x_3, \ldots, x_n), \tag{2.4}$$

$$x_1 \vee f(x_1, \bar{x}_1, x_2, x_3, \ldots, x_n) = x_1 \vee f(0, 1, x_2, x_3, \ldots, x_n), \tag{2.5}$$

$$\bar{x}_1 \vee f(x_1, \bar{x}_1, x_2, x_3, \ldots, x_n) = \bar{x}_1 \vee f(1, 0, x_2, x_3, \ldots, x_n). \tag{2.6}$$

Из соотношений (2.3) и (2.6) следует, что в функции f все переменные x_1 следует заменить на единицу, а $\bar{x}_1$ — на нуль. В соотношениях (2.4) и (2.5), наоборот, все переменные x_1 нужно заменить на нуль, а переменные $\bar{x}_1$ — на единицу. Рассмотрим примеры использования некоторых соотношений.

Пример 2.1. Упростить логическое выражение $Y = x_1(x_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3)$.

Решение. Воспользовавшись соотношением (2.3), получим

$$Y = x_1(x_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3) = x_1(1 \cdot x_2 \vee 0 \cdot x_3 \vee x_2x_3) = x_1(x_2 \vee x_2x_3) = x_1x_2.$$

Пример 2.2. Упростить логическое выражение

$$Y = \bar{x}_1(x_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3). \tag{2.7}$$

Решение. Воспользовавшись соотношением (2.4), запишем

$$Y = \bar{x}_1(x_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3) = \bar{x}_1(0 \cdot x_2 \vee 1 \cdot x_3 \vee x_2x_3) = \bar{x}_1(x_3 \vee x_2x_3) = \bar{x}_1x_3.$$

§ 2.2. Формы представления логических функций

На отдельных этапах проектирования комбинационных схем используют различные формы представления логических функций: словесную, табличную, аналитическую, геометрическую и кубическую.

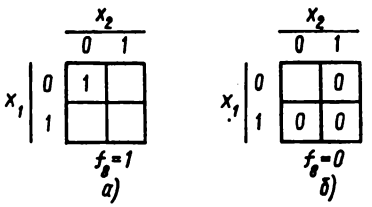


Рис. 2.1. Изображение функции f_8 на карте Карно

Таблица 2.2

Таблица истинности функции f_8

x_1	x_2	f_8
0	0	1
0	1	0
1	0	0
1	1	0

Любая логическая функция может быть представлена в виде *словесного описания*. Например, функция $f_8 = x_1 \downarrow x_2$ словесно может быть описана так: $f_8 = 1$ тогда и только тогда, когда обе переменные x_1 и x_2 равны нулю. Эту же функцию можно представить в *табличной форме* (*таблицей истинности или картой Карно*). Таблица истинности, как и карта Карно (табл. 2.2 и рис. 2.1), содержит все 2^n возможных входных наборов и значения функции, соответствующие каждому из наборов.

Для двух переменных карта Карно представляет собой квадрат, разделенный на четыре ячейки, по одной на каждый входной набор. Строки карты связаны с переменной x_1, столбцы — с переменной x_2. Следовательно, расположенная слева вверху ячейка соответствует входному набору (00) или минтерму ($\bar{x}_1\bar{x}_2$), а расположенная справа внизу ячейка — входному набору (1 1) или макстерму ($x_1 \vee x_2$). Такого рода карта называется картой Карно на две переменные (рис. 2.1). Представление логической функции на карте Карно производится в соответствии с таблицей истинности. Если функция $f_8 = \bar{x}_1\bar{x}_2 = 1$ на входном наборе (00), то этот факт отражается на карте Карно записью в левую верхнюю ячейку единицы (рис. 2.1, *а*). Остальные ячейки остаются незаполненными. Карта Карно может заполняться нулями в те ячейки, на входных наборах которых функция равна нулю. На рис. 2.1, *б* приведен пример заполнения карты Карно для функции $f_8 = 0$.

Пример 2.3. Представить функцию $f(x_1, x_2) = x_1\bar{x}_2 \vee x_1x_2$ с помощью карты Карно.

Решение. Имеем функцию двух переменных, поэтому она может быть представлена на карте Карно для двух переменных единицами в двух нижних ячейках с координатами (1, 0) и (1, 1).

В случае трех переменных карта Карно (рис. 2.2) содержит восемь ячеек, по одной для каждого входного

x_1 \ x_2x_3	00	01	11	10
0	000	001	011	010
1	100	101	111	110

Рис. 2.2. Карта Карно для функции трех переменных

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Рис. 2.3. Карта Карно для функции четырех переменных

набора, указанного внутри ячейки. Переменная x_1 связана с двумя строками карты, а переменные x_2 и x_3 — с четырьмя столбцами. Таким образом, любые две рядом расположенные ячейки являются соседними и их координаты отличаются только одной переменной. Кроме того, соседними являются ячейки, стоящие в первом и последнем столбцах карты.

Поскольку для четырех переменных существует 16 входных наборов, карта Карно разделена на 16 ячеек (рис. 2.3). Каждая ячейка пронумерована в соответствии с порядковым номером входного набора.

В случае пяти переменных целесообразно использовать две 16-ячеечные карты (рис. 2.4), а не одну 32-ячеечную (рис. 2.5). Каждая из указанных на рис. 2.4 карт связана с одним из значений переменной x_5.

В случае шести переменных потребуется уже четыре 16-ячеечные карты. Каждая карта должна быть связана с одной из четырех возможных комбинаций переменных x_5 и x_6. Для логических функций с числом переменных $n>6$ карты Карно становятся громоздкими и неудобными для практического применения.

x_1x_2 \ x_3x_4	00	01	11	10
00	0	2	6	4
01	8	10	14	12
11	24	26	30	28
10	16	18	22	20

а)

x_1x_2 \ x_3x_4	00	01	11	10
00	1	3	7	5
01	9	11	15	13
11	25	27	31	29
10	17	19	23	21

б)

Рис. 2.4. Представление функции пяти переменных на двух 16-ячеечных картах Карно:
а — при $\bar{x}_5$; *б* — при x_5

x_1x_2 \ $x_3x_4x_5$	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Рис. 2.5. Карта Карно для функции пяти переменных

В алгебре логики существуют две основные аналитические формы представления функций: совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная форма (СКНФ). Каждая логическая функция имеет одну СДНФ и одну СКНФ.

СДНФ логической функции — это дизъюнкция конституент единицы (минтермов), соответствующих наборам входных переменных, для которых функция равна единице. В общем случае СДНФ можно представить в форме

$$f(x_1, x_2, ..., x_n) = \bigvee_1 x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}, \qquad (2.8)$$

где $\alpha_1, \alpha_2, ..., \alpha_n$ — двоичный набор;

$$x_i^{\alpha_i} = \begin{cases} x_i, \text{ если } \alpha_i = 1, \\ \bar{x}_i, \text{ если } \alpha_i = 0. \end{cases} \qquad (2.9)$$

СКНФ логической функции — это конъюнкция конституент нуля (макстермов), соответствующих входным наборам, для которых функция равна нулю. В общем случае СКНФ можно представить в форме

$$f(x, x_2, .., x_n) = \mathop{\&}_0 \overline{x_1^{\alpha_1}} \vee \overline{x_2^{\alpha_2}} \vee ... \vee \overline{x_n^{\alpha_n}}. \qquad (2.10)$$

Рассмотрим алгоритмы перехода от табличного описания логической функции к ее аналитическому описанию (СДНФ, СКНФ).

Алгоритм перехода от таблицы истинности логической функции к ее записи в виде СДНФ:

1) выбрать в таблице такие входные наборы, на которых функция обращается в единицу;

2) записать конституенты единицы (минтермы) для выбранных входных наборов;

3) полученные минтермы соединить между собой знаками дизъюнкции.

Пример 2.4. Получить СДНФ логической функции f_9, заданной таблицей истинности (см. табл. 2.1).

Решение. Функция $f_9 = 1$ на нулевом и третьем входных наборах (00) и (11). Для выбранных наборов записываем минтермы в соответствии с условием (2.9).

Соединив знаком дизъюнкции записанные минтермы, получим СДНФ для f_9: $f_9 = \bar{x}_1\bar{x}_2 \vee x_1x_2$.

Алгоритм перехода от табличного задания логической функции к ее записи в виде СКНФ:

1) выбрать в таблице истинности такие входные наборы, на которых функция имеет нулевые значения;

2) записать конституенты нуля (макстермы) для выбранных входных наборов;

3) полученные макстермы соединить между собой знаками конъюнкции.

Пример 2.5. Получить СКНФ логической функции f_9 (см. табл. 2.1).

Решение. Функция $f_9 = 0$ на первом (01) и втором (10) наборах. Для выбранных наборов записываем макстермы: $\bar{x}_1^0 \vee \bar{x}_2^1 = x_1 \vee \bar{x}_2$, $\bar{x}_1^1 \vee \bar{x}_2^0 = \bar{x}_1 \vee x_2$. Соединив знаком конъюнкции записанные макстермы, получим СКНФ для f_9:

$$f_9 = (x_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2). \qquad (2.11)$$

Правильность полученного аналитического выражения (2.11) можно проверить, подставив в него значения переменных x_1 и x_2, соответствующие входным наборам, на которых функция f_9 обращается в нуль. На наборе 01 функция $f_9 = (0 \vee 0)(1 \vee 1) = 0$. На наборе 10 $f_9 = (1 \vee 1)(0 \vee 0) = 0$. Так как на данных наборах $f_9 = 0$, следовательно, СКНФ (2.11) записана верно.

Помимо двух основных (СДНФ и СКНФ) аналитических форм представления функций в алгебре логики существуют дизъюнктивные и конъюнктивные нормальные формы (ДНФ и КНФ), а также минимальные дизъюнктивные и конъюнктивные нормальные формы (МДНФ и МКНФ).

ДНФ/КНФ — это дизъюнкция/конъюнкция элементарных конъюнкций/дизъюнкций. ДНФ/КНФ называется минимальной, если она содержит наименьшее число букв по сравнению со всеми ДНФ/КНФ, существующими для данной функции. При помощи правил развертывания элементарных конъюнкций и дизъюнкций ДНФ и КНФ могут быть преобразованы в СДНФ и СКНФ.

В общем виде СДНФ и СКНФ записываются выражениями (2.8) и (2.10), которые могут быть слишком громоздкими. Для их упрощения может быть использована более компактная *числовая форма* записи, предусматривающая только десятичные эквиваленты двоичным наборам, на которых функция принимает значение единицы (для СДНФ) и нуля (для СКНФ), соединив их знаками дизъюнкции (для СДНФ) и конъюнкции (для СКНФ).

Пример 2.6. Записать в числовой форме СДНФ функции f_9, полученную в примере 2.4.

Решение. Поскольку из аналитической формы (СДНФ) и таб-

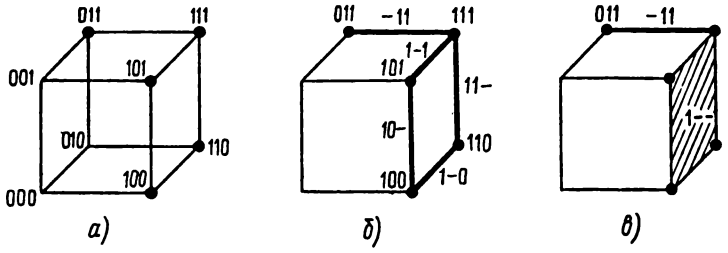


Рис. 2.6. Геометрическое и кубическое представление функции $Y = \vee(3, 4, 5, 6, 7)$:
а — 0-кубы; *б* — 1-кубы; *в* — 2-куб

лицы истинности (см. табл. 2.1) видно, что $f_9 = 1$ на нулевом и третьем наборах, то числовая форма СДНФ имеет вид

$$f_9 = 0 \vee 3 = \vee(0, 3).$$

Пример 2.7. Записать в числовой форме СКНФ функции f_9, полученную в примере 2.5.

Решение. Поскольку $f_9 = 0$ на первом и втором наборах, т. е. СКНФ в числовом виде: $f_9 = 1 \cdot 2 = \&(1, 2)$.

Геометрическое и кубическое представления логических функций. В геометрическом смысле каждый двоичный набор $(\alpha_1, \alpha_2, ..., \alpha_n)$ можно рассматривать как n-мерный вектор, определяющий точку n-мерного пространства. Исходя из этого, все множество наборов, на которых определена функция n переменных, представляется в виде вершин n-мерного куба. Отмечая точками вершины, в которых функция имеет единичные значения, получают ее геометрическое представление, например функция $Y = \vee(3, 4, 5, 6, 7)$ геометрически представляется, как показано на рис. 2.6, *а*.

На основе геометрических представлений строятся кубические представления логических функций, в которых используются элементы n-мерных кубов. Геометрически каждому набору $(\alpha_1, \alpha_2, ..., \alpha_n)$ соответствует вершина n-мерного куба с данными координатами. Каждую вершину, на которой функция принимает единичное значение, принято называть *0-кубом* (нулевым кубом). Множество 0-кубов образует *нулевой кубический комплекс* K_0. Например, для функции $Y = \vee(3, 4, 5, 6, 7)$ нулевой кубический комплекс имеет вид $K_0 = \{011, 100, 101, 110, 111\}$ и в данном случае состоит из пяти 0-кубов, каждый из которых соответствует определенной вершине трехмерного куба (рис. 2.6, *а*).

Если два 0-куба из комплекса K_0 различаются только

по одной координате, то они образуют один 1-куб (единичный куб). Он представляется общими элементами 0-кубов, а на месте координаты, по которой различаются 0-кубы, указывается символ «-», обозначающий независимую координату. Например, два 0-куба: 100, 101 различаются только по третьей координате и, следовательно, образуют единичный куб 10-, которому соответствует ребро трехмерного куба. Все множество единичных кубов функции образует *единичный кубический комплекс* K_1. Для функции $Y = \bigvee(3, 4, 5, 6, 7)$ он состоит из пяти 1-кубов: $K_1 = \{$-11, 10-, 1-0, 1-1, 11-$\}$ и определяет все множество ребер, на которых функция Y принимает единичные значения (рис. 2.6, *б*).

Если два единичных куба из комплекса K_1 имеют общую независимую координату и различаются только по одной координате, то они образуют один *2-куб* (*двоичный куб*). Его запись состоит из общих компонент 1-кубов, а координата, принимающая различные значения в 1-кубах, обозначается в 2-кубе как независимая координата «-» (пробел). Например, два единичных куба 1-0 и 1-1 образуют один двоичный куб 1--. Для рассматриваемой функции Y комплекс имеет вид $K_2 = \{$1--$\}$ и состоит из одного двоичного куба, соответствующего грани трехмерного куба (рис. 2.6, *в*). Размерность куба определяется числом независимых координат (символов «-»).

Объединение кубических комплексов K_0, K_1, ..., K_n функции $f(x_1, x_2, ..., x_n)$ образует *кубический комплекс функции* $K(f)$. Для рассматриваемого примера комплекс $K(f)$ соответствует объединению комплексов K_0, K_1 и K_2.

В отличие от аналитических форм записи логических функций, использующих большой набор индексированных букв и математических знаков, кубические представления позволяют задавать логические функции в виде множества кубов, компонентами которых являются только три символа: 0, 1, -. Ограниченное число символов в записи функций позволяет использовать кубические представления в ЭВМ при решении задач логического проектирования БИС.

§ 2.3. Методы минимизации логических функций

Наиболее эффективно задача минимизации логических функций выполняется с использованием кубических представлений этих функций. Каждая логическая функ-

ция $f(x_1, x_2, ..., x_n)$ рассматривается как множество $П$-кубов, принадлежащих комплексу $K(f)$, таких, что каждая вершина комплекса K_0 заключена по крайней мере в один куб множества $П$. Полученное таким образом множество $П$ называется покрытием комплекса $K(f)$ или *покрытием логической функции* $П(f)$. Каждому выбранному покрытию $П(f)$ соответствует своя ДНФ функции f. Например, функции $f = \bigvee(3, 4, 5, 6, 7)$ соответствует комплекс $K(f) = \{011, 100, 101, 110, 111, -11, 10-, 1-0, 1-1, 11-, 1--\}$. В покрытие данной функции могут быть включены различные совокупности кубов, принадлежащих $K(f)$ и охватывающих все вершины n-мерного куба, в которых $f = 1$:

$$\left\{\begin{matrix}011\\1--\end{matrix}\right\}, \quad \left\{\begin{matrix}-11\\1--\end{matrix}\right\}, \quad \left\{\begin{matrix}-11\\1-1\\1-0\end{matrix}\right\}, \quad \left\{\begin{matrix}-11\\10-\\11-\end{matrix}\right\}$$

и др. От выбранного покрытия функции легко перейти к ее ДНФ. Например, для второго покрытия можно записать $f = x_2x_3 \bigvee x_1$.

Для оценки сложности логических функций вводится понятие *цены покрытия* ($Ц_П$), равной сумме цен всех кубов, составляющих покрытие $П(f)$. Цена r-куба, соответствующего конституенте функции n переменных, определяется как разность $n-r$ и соответствует числу букв в конституенте. Например, цена второго покрытия $Ц_П = 3$, а цена первого покрытия равна четырем, третьего — шести, четвертого — шести. Покрытие, имеющее минимальную цену, назовем покрытием комплекса (функции) в смысле Квайна, а ДНФ, определенная по минимальному покрытию, — МДНФ. Так как цена второго покрытия имеет минимальное значение из всех возможных, то соответствующая ему ДНФ $f = x_2x_3 \bigvee x_1$ является МДНФ.

Из большого числа различных приемов и методов минимизации логических функций рассмотрим два наиболее типичных: табличный метод (с помощью карт Карно) и метод Квайна и Мак-Класки.

Минимизация при помощи карты Карно. Карта Карно представляет собой таблицу для задания логических функций в форме СДНФ и СКНФ. Расположение клеток в таблице позволяет легко определить склеивающиеся между собой члены. В других расчетных методах минимизации процесс отыскания склеивающихся членов является наиболее трудоемким, так как при выполнении опе-

рации склеивания необходимо сравнивать все возможные пары членов исходной функции. В табличном методе поиск склеивающихся членов максимально упрощен. Каждой клетке карты Карно соответствует вершина куба. Склеивающиеся между собой конституенты располагаются в соседних ячейках и выделяются в покрытие функции. Минимальное покрытие выбирается интуитивно на основе анализа различных вариантов покрытий минимизируемой функции.

Рассмотрим *правила минимизации при помощи карт Карно*:

1) 2^k соседних клеток, содержащих единицы/нули и расположенных по вертикали либо по горизонтали, а также в виде прямоугольника либо квадрата, соответствуют одной элементарной конъюнкции/дизъюнкции, ранг которой r меньше ранга конституенты n на k единиц. Чем больше клеток в покрытии, тем проще выражаемый им член логической функции — импликанта;

2) в любой карте Карно соседними клетками, к которым можно применить правило склеивания, являются не только рядом стоящие ячейки, но и клетки, находящиеся на противоположных концах любой строки и любого столбца (эти клетки оказываются смежными, если карту свернуть по горизонтали или вертикали в цилиндр).

Таким образом, процесс минимизации сводится к нахождению наиболее крупных покрытий из 2^k соседних (заполненных) ячеек. Следует стремиться к тому, чтобы каждая заполненная ячейка входила в какое-либо покрытие. Импликанта, соответствующая некоторому покрытию заполненных ячеек, содержит символы тех переменных, значения которых совпадают у всех объединенных ячеек.

Пример 2.8. Минимизировать с помощью карты Карно функцию вида $Y = \bigvee(1, 2, 3, 5, 7)$.

Решение. Поскольку исходная функция задана для трех переменных в числовой форме, для минимизации выбираем карту Карно на три переменные и на указанных в исходной функции наборах записываем в соответствующие ячейки единицы (рис. 2.7). Так как исходных наборов пять, то и карта Карно должна содержать пять единиц. После заполнения карты Карно проводим склеивание конституент соседних ячеек, для чего выбираем два соседних покрытия. В первое покрытие входят четыре ячейки, находящиеся в средней части карты. Эти ячейки покрываются переменной x_3 (т. е. склеиваются по переменным x_1 и x_2) и образуют двоичный куб --1. Во второе покрытие входят две ячейки (011, 010), которые склеиваются по переменной x_3 и образуют единичный куб (01-). В результате получаем минимальное покрытие, состоящее из двух кубов:

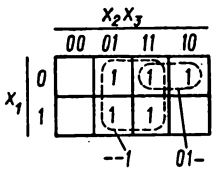


Рис. 2.7. Минимизация на карте Карно функции $Y = \vee(1, 2, 3, 5, 7)$

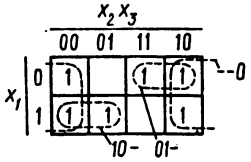


Рис. 2.8. Минимизация на карте Карно функции $Y = \vee(0, 2, 3, 4, 5, 6)$

$$П(Y_{МДНФ}) = \left\{ \begin{matrix} --1 \\ 01 \end{matrix} \right\}$$

Тогда МДНФ записываем в виде $Y_{МДНФ} = x_3 \vee \bar{x}_1 x_2$.

Цена полученного покрытия $Ц_{МДНФ} = 3$. Для сравнения укажем, что цена покрытия исходной функции $Ц_{СДНФ} = 15$.

Пример 2.9. Минимизировать с помощью карты Карно функцию $Y = \vee(0, 2, 3, 4, 5, 6)$.

Решение: 1) выбираем карту Карно на три переменные; 2) представляем на карте исходную функцию Y, т. е. заполняем карту единицами на указанных входных наборах; 3) выбираем покрытия соседних ячеек. Получаем минимальное покрытие функции, состоящее из трех кубов (рис. 2.8):

$$П(Y_{МДНФ}) = \left\{ \begin{matrix} --0 \\ 01- \\ 10- \end{matrix} \right\};$$

4) записываем МДНФ минимизируемой функции $Y_{МДНФ} = \bar{x}_3 \vee \bar{x}_1 x_2 \vee \vee x_1 \bar{x}_2$.

Метод Квайна и Мак-Класки. Достоинства метода: использование числового представления логических функций и реализация алгоритма минимизации на ЭВМ; практически отсутствие ограничений на число переменных.

Основные этапы минимизации логических функций методом Квайна и Мак-Класки.

I. *Нахождение простых импликант.* Сначала все 0-кубы разделяются на группы с одинаковым числом единиц. Затем попарно сравниваются между собой все 0-кубы соседних групп. Если два 0-куба различаются только по одной координате, то они образуют 1-куб (0-кубы, образующие 1-куб, отмечаются). После построения всех 1-кубов, образующих комплекс K_1, производится построение 2-кубов и т. д. При построении r-кубов все $(r-1)$-кубы, образующие r-кубы, также отмечаются. Этап заканчивается, когда ни один $(r+1)$-куб не может быть построен. Все не отмеченные кубы комплекса $K(f)$ являются простыми импликантами и образуют покрытие $П(f)$ функции f.

II. *Построение таблицы покрытий матрицы Квайна.* Строки таблицы покрытий соответствуют простым импликантам, а столбцы — 0-кубам (конституентам единицы) минимизируемой функции. На пересечении строки i и столбца j ставится метка единица, если импликанта i покрывает конституенту j. Импликанта покрывает конституенту, если она отличается от нее только независимыми координатами.

III. *Отыскание минимального покрытия функции.* Для нахождения минимального покрытия необходимо удалить из таблицы покрытия некоторые лишние простые импликанты. С этой целью в методе Квайна и Мак-Класки реализуется следующий алгоритм решения задачи покрытия.

1. Выделение ядра Квайна. Если в каком-либо столбце таблицы покрытий имеется только одна единица, то импликанта, стоящая в строке, является существенной (обязательной) и должна входить в ядро Квайна, а следовательно, и в минимальное покрытие, поскольку не используя ее, невозможно покрыть все конституенты.

2. Вычерчивание строк и столбцов, покрываемых импликантами, входящими в ядро Квайна. Если в полученной таблице покрытий имеются столбцы, содержащие по одной единице, то операцию, описываемую в п. 1, следует повторить.

3. Поглощение лишних столбцов (сжатие по столбцам). Из матрицы вычерчивается тот столбец, в который полностью входит любой другой столбец. Иначе, если в таблице покрытий имеется такая пара столбцов d_i и d_j, что $d_i \subseteq d_j$, то столбец d_j вычерчивается, поскольку покрытие вычеркнутого столбца может производиться путем покрытия оставшегося столбца.

4. Поглощение лишних строк (сжатие по строкам). Если в таблице покрытий имеется такая пара строк q_i и q_j, что $q_i \supseteq q_j$, то строка q_j вычерчивается (строка q_i поглощает строку q_j).

5. Последовательное применение двух преобразований (сжатие по столбцам и строкам). В результате исходная таблица покрытий приводится к так называемой «циклической», импликанты которой должны входить в минимальное покрытие функции.

IV. *Получение минимальной формы логической функции.* Простые импликанты, соответствующие строкам, которые входят в минимальное покрытие, соединяются знаками дизъюнкции и образуют МДНФ булевой функции.

	0 группа	1 группа	2 группа	3 группа	4 группа
0-кубы	0000(0)	0001(1) 0010(2) 0100(4) 1000(8)	0101(5) 1010(10) 1100(12)	0111(7) 1110(14)	1111(15)
1-кубы	000-(0,1) 00-0(0,2) 0-00(0,4) -000(0,8)	0-01(1,5) -010(2,10) 010-(4,5) -100(4,12) 10-0(8,10) 1-00(8,12)	01-1(5,7) 1-10(10,14) 11-0(12,14)	-111(7,15) 111-(14,15)	
2-кубы	0-0-(0,1,4,5) -0-0(0,2,8,10) 0-0-(0,4,1,5) --00(0,4,8,12) -0-0(0,8,2,10) --00(0,8,4,12)	1--0(8,10,12,14) 1--0(8,12,10,14)			

Рис. 2.9. Нахождение простых импликант функции $Y = \vee(0, 1, 2, 4, 5, 7, 8, 10, 12, 14, 15)$

Пример 2.10. Минимизировать логическую функцию четырех переменных $Y = \vee(0, 1, 2, 4, 5, 7, 8, 10, 12, 14, 15)$ методом Квайна и Мак-Класки.

Решение. 1. Находим простые импликанты в соответствии с первым этапом алгоритма минимизации функции (рис. 2.9). Все отмеченные кубы комплекса $K(Y)$ являются импликантами и образуют покрытие

$$\Pi(Y) = \begin{Bmatrix} 01\text{-}1 \\ \text{-}111 \\ 111\text{-} \\ 0\text{-}0\text{-} \\ \text{-}0\text{-}0 \\ \text{--}00 \\ 1\text{--}0 \end{Bmatrix}$$

2. Строим таблицу покрытий функции Y (рис. 2.10, *а*).

3. Определяем оптимальное покрытие. С помощью рис. 2.9, *а* устанавливаем, что импликанта 0-0- является существенной для конституент 0000, 0001, 0100, 0101, и импликанта -0-0 является существенной для столбцов 0000, 0010, 1000, 1010. Исходя из этого, на данном этапе в минимальное покрытие функции должны быть включены импликанты 0-0- и -0-0. В результате таблица упрощается (рис. 2.10, *б*).

Затем проводим сжатие таблицы (рис. 2.10, *б*). Сжатие по столбцам невозможно. Строка *Б* поглощает строку *А*, а строка *Д* — строку *Г*. В результате получаем таблицу, приведенную на рис. 2.10, *в*. После сжатия по столбцам получаем таблицу, приведенную на рис. 2.10, *г*, в которой импликанта 111- является лишней, так как не содержит единиц ни в одном столбце. После ее исключения из таблицы (рис. 2.10, *г*) получаем циклическую матрицу (рис. 2.10, *д*), импликанты -111 и 1--0 которой должны входить в минимальное покрытие функции.

Импликанта	Конституента единицы										
	0000	0001	0010	0100	0101	0111	1000	1010	1100	1110	1111
0 1 – 1					1	1					
– 1 1 1						1					1
1 1 1 –										1	1
0 – 0 –	1	1		1	1						
– 0 – 0	1		1				1	1			
– – 0 0	1			1			1		1		
1 – – 0							1	1	1	1	

а)

	Импликанта	0111	1100	1110	1111
А	0 1 – 1	1			
Б	– 1 1 1	1			1
В	1 1 1 –			1	1
Г	– – 0 0		1		
Д	1 – – 0		1	1	

б)

Импликанта	0111	1100	1110	1111
– 1 1 1	1			1
1 1 1 –			1	1
1 – – 0		1	1	

в)

Импликанта	0111	1100
– 1 1 1	1	
1 – – 0		1

д)

Импликанта	0111	1100
– 1 1 1	1	
1 1 1 –		
1 – – 0		1

г)

Рис. 2.10. Решение задачи покрытия методом Квайна и Мак-Класки:
а — исходная матрица Квайна; *б—д* — скорректированные матрицы

Таким образом, минимальное покрытие функции имеет вид

$$\Pi(Y) = \begin{Bmatrix} 0\text{-}0\text{-} \\ \text{-}0\text{-}0 \\ \text{-}111 \\ 1\text{--}0 \end{Bmatrix} \tag{2.12}$$

4. На основании (2.12) записываем МДНФ

$$Y = \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4 \vee x_2x_3x_4 \vee x_1\bar{x}_4.$$

В настоящее время нет достаточно эффективных методов нахождения целочисленного решения задачи покрытия

большой размерности, поэтому часто используют *эвристические алгоритмы покрытия*. Рассмотрим один из них, разработанный авторами.

1. Находим ядро Квайна так же, как и в классическом алгоритме Квайна и Мак-Класки.

2. Вычерчиваем все столбцы, которые покрываются строками, взятыми в оптимальное покрытие.

3. Вычисляем веса столбцов скорректированной таблицы покрытий. Вес столбца равен числу единиц в нем. Выбираем столбец или столбцы, содержащие минимальное число единиц.

4. По формуле (2.13) вычисляем веса строк, покрывающих выбранные столбцы:

$$S_k = \sum_i w_i = m\sum_i \frac{1}{\alpha_i}, \quad i = 1, 2, ..., t, \qquad (2.13)$$

где S_k — вес k-й строки матрицы; w_i — вес i-го столбца, равный числу единиц в этом столбце; m — число строк в матрице; α_i — число единиц в столбце, на пересечении с которым k-я строка содержит единицу; t — число единиц в k-й строке.

5. Выбираем строку с минимальным весом и исключаем ее из таблицы.

6. Для скорректированной таблицы вычисляем веса столбцов.

7. Если вес одного или нескольких столбцов равен единице, то строки, покрывающие эти столбцы, должны быть отнесены в оптимальное покрытие и вместе с данными столбцами исключены из таблицы.

8. Если покрыты еще не все столбцы, то необходимо перейти к п. 3. В противном случае поиск прекращается.

Данный эвристический алгоритм обладает высокой точностью благодаря постоянному анализу возможности выделения ядра Квайна и исключения из дальнейшего рассмотрения строки с минимальным весом из некоторого числа строк, покрывающих столбцы с одинаковым минимальным весом.

Пример 2.11. Определить по эвристическому алгоритму минимальное покрытие некоторой функции, заданной матрицей Квайна (рис. 2.11, *а*).

Решение. 1. Вычисляем веса столбцов таблицы (см. рис. 2.11, *а*) и записываем их в низу каждого столбца. Устанавливаем, что столбцы 2, 6, 9 и 10 имеют вес, равный двум.

2. Вычисляем веса строк (см. рис. 2.11, *а*), покрывающих столбцы 2, 6, 9, 10. Строка 6 имеет минимальный вес и поэтому подлежит-

	1	2	3	4	5	6	7	8	9	10	Вес строки
1	1			1				1			
2		1	1		1			1			11,2
3		1				1	1				11,2
4			1	1				1		1	12
5				1			1		1		9,7
6	1		1			1					9 +
7					1		1				
8	1				1			1	1		11,2
9	1		1		1		1			1	13,5
Вес столбца	4	2	4	3	4	2	4	4	2	2	

а)

	1	2	3	4	5	6	7	8	9	10
1	1				1			1		
2		1	1		1			1		
3		1					1	1		
4			1	1				1		1
5					1		1		1	
7						1		1		
8	1					1			1	1
9	1		1			1		1		1
Вес столбца	3	2	3	3	4	1	4	4	2	2

б)

	1	3	4	5	8	9	10	Вес строки
1	1		1		1			
2		1			1			
4		1	1		1		1	10
5			1			1		6 +
7				1				
8	1			1	1	1		9,3
9	1	1		1			1	10
Вес столбца	3	3	3	3	4	2	2	

в)

	1	3	4	5	8	9	10
1	1		1		1		
2		1			1		
4		1	1		1		1
7				1			
8	1			1	1	1	
9	1	1		1			1
Вес столбца	3	3	2	3	4	1	2

г)

	3	4	10	Вес строки
1		1		2,5 +
2	1			
4	1	1	1	6,6
7				
9	1		1	4,1
Вес столбца	3	2	2	

д)

	3	4	10
2	1		
4	1	1	1
9	1		1
Вес столбца	3	1	2

е)

Рис. 2.11. Нахождение минимального покрытия функции эвристическим методом:
а — исходная матрица Квайна; *б* — *е* — скорректированные матрицы

исключению из таблицы (см. рис. 2.11, *а*). После исключения строки 6 получаем таблицу (см. рис. 2.11, *б*).

3. Вычисляем веса столбцов таблицы (см. рис. 2.11, *б*). Столбец 6 имеет вес, равный единице, поэтому строку 3, покрывающую столбец 6, выбираем в покрытие. Исключаем ее из дальнейшего рассмотрения вместе со столбцами 6 и 7. Тогда получаем таблицу (рис. 2.11, *в*).

4. Вычисляем веса столбцов таблицы (см. рис. 2.11, *в*). Столбцы 9 и 10 имеют минимальный вес, равный двум.

5. Вычисляем веса строк, покрывающие столбцы 9 и 10. Исходя из минимального веса строку 5 исключаем из таблицы (см. рис. 2.11, *в*). В результате получаем таблицу (см. рис. 2.11, *г*).

6. Вычисляем веса столбцов таблицы (см. рис. 2.11, *г*). Столбец 9 имеет вес, равный единице. Тогда строку 8, покрывающую столбец 9, выбираем в покрытие, исключаем ее из дальнейшего рассмотрения вместе со столбцами 1, 5, 8 и 9. Получаем таблицу (см. рис. 2.11, *д*).

7. Вычисляем веса столбцов таблицы (см. рис. 2.11, *д*). Столбцы 4 и 10 имеют минимальный вес, равный двум.

8. Вычисляем веса строк (1, 4, 9), покрывающих столбцы 4 и 10. Исключаем из таблицы (см. рис. 2.11, *д*) строку 1. Получаем новую таблицу (рис. 2.11, *е*).

9. Вычисляем веса столбцов таблицы (см. рис. 2.11, *е*). Столбец 4 имеет вес, равный единице. Тогда строку 4 записываем в покрытие. Так как она покрывает все столбцы таблицы (см. рис. 2.10, *е*), процесс минимизации закончен.

Таким образом, в оптимальное покрытие вошли строки 3, 4 и 8.

Кроме рассмотренных существует большое число методов минимизации булевых функций, среди которых следует выделить метод Рота, допускающий задание минимизируемой функции в произвольной ДНФ.

§ 2.4. Формы автоматного описания БИС

Математический аппарат булевой алгебры применим только к комбинационным схемам, в которых отсутствуют элементы памяти. Поэтому для анализа и синтеза цифровых БИС, содержащих логические элементы и элементы памяти (триггеры), инженеру недостаточно знать только основы алгебры логики, ему необходимы сведения из теории цифровых автоматов.

Цифровой автомат — это устройство, служащее для преобразования дискретной информации, перехода из одного состояния в другое под воздействием входных сигналов и сохранения состояния в отсутствие последних. Цифровыми автоматами являются: ЭВМ, БИС, функциональные блоки, узлы, элементы библиотеки БМК. В общем случае математической моделью цифрового автомата служит конечный автомат (КА):

$$\text{КА} = \{M_X, M_Y, M_S, \varphi, \Psi\}, \qquad (2.14)$$

где M_X, M_Y, M_S — множество входных, выходных и внутренних состояний; φ — функция переходов; Ψ — функция выходов. При изучении КА используют абстрактную и структурную теорию автоматов.

Абстрактный автомат — это конечный автомат, эле-

менты которого X, Y, S являются символами входных, выходных и внутренних переменных определенной размерности. В абстрактной теории автоматов изучаются наиболее общие законы их поведения без учета конечной структуры автомата и физической природы информации. Для представления информации принят алфавитный способ задания:

$$X = \{x_1, x_2, ..., x_m\} \text{ — входной алфавит,}$$
$$Y = \{y_1, y_2, ..., y_n\} \text{ — выходной алфавит,}$$
$$S = \{s_0, s_1, ..., s_q\} \text{ — алфавит внутренних состояний.}$$

Функция переходов φ реализует отображение $S \times X \to S$, функция выходов Ψ — отображение $S \times X \to Y$ (для автомата Милли) и $S \to Y$ (для автомата Мура).

Автомат Милли — это конечный автомат, работа которого описывается уравнениями:

$$\begin{aligned} S(t+1) &= \varphi\{S(t),\ X(t)\}, \\ Y(t) &= \Psi\{S(t),\ X(t)\}, \end{aligned} \tag{2.15}$$

где t — время.

Автомат Мура — это конечный автомат, работа которого описывается уравнениями:

$$\begin{aligned} S(t+1) &= \varphi\{S(t),\ X(t)\}, \\ Y(t) &= \Psi\{S(t)\}. \end{aligned} \tag{2.16}$$

Комбинационный автомат — это конечный автомат вида КА $= \{X, Y, \Psi\}$, в котором выходной вектор Y зависит только от входного вектора X. Такие автоматы называют еще автоматами без памяти или без обратных связей, и к ним относят различные комбинационные функциональные узлы: сумматоры, полусумматоры, шифраторы и дешифраторы, мультиплексоры и демультиплексоры, преобразователи кодов и др. Основные формы представления логических функций, описывающих работу комбинационного автомата, изложены в § 2.2.

Последовательностный автомат — это конечный автомат, алгоритм функционирования которого описывается уравнениями вида (2.15) или (2.16). Такие автоматы называют также автоматами с памятью. К их числу относят различные последовательностные функциональные узлы: счетчики, регистры, генераторы чисел и др. Основными формами описания работы автоматов являются: 1) аналитическая (2.15), (2.16); 2) табличная — таблицы переходов и выходов; 3) граф автомата; 4) микро-

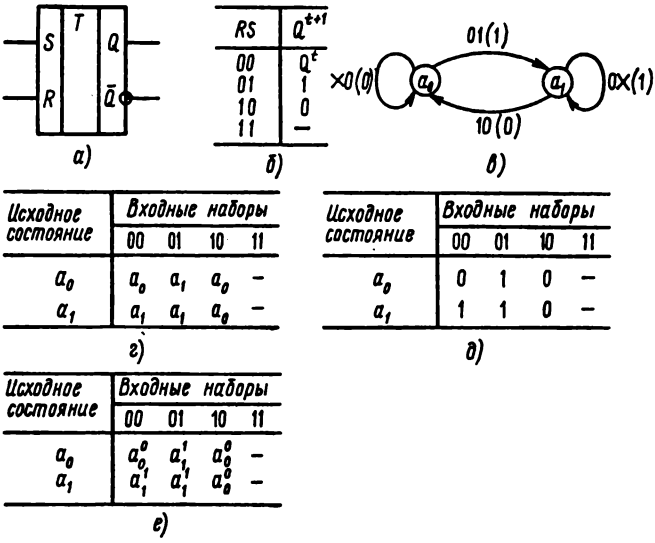


Рис. 2.12. Формы автоматного представления RS-триггера:
а — условное графическое обозначение; *б* — таблица истинности; *в* — автоматный граф; *г* — таблица переходов; *д* — таблица выходов; *е* — совмещенная таблица переходов и выходов

программа. Примеры автоматных описаний RS-триггера представлены на рис. 2.12.

Граф автомата — это ориентированный граф, в котором множеству вершин соответствует множество внутренних состояний, а множеству дуг — множество возможных переходов из одного состояния в другое. Примером изображения такого графа служит рис. 2.12, *в*. Запись на дугах определяет условие перехода (входной набор), а цифра в скобках (0/1) указывает на значение выходного сигнала в новом состоянии автомата.

Таблица переходов (*смены состояний*) — это матрица, каждому внутреннему состоянию которой отводится одна строка, а каждому входному состоянию (набору) — один столбец. Элемент s_{ij} матрицы есть внутреннее состояние, в которое переходит автомат из состояния s_i при входном векторе X_j. Примером таблицы переходов служит рис. 2.12, *г*.

Таблица выходов — это матрица, аналогичная таблице переходов, элемент y_{ij} которой есть выходное состояние (0/1) при переходе автомата из внутреннего со-

стояния s_i под воздействием входного вектора X_j. Примером таблицы выходов служит рис. 2.12, *д*.

Совмещенная таблица переходов и выходов — это матрица, также аналогичная таблице переходов. Элемент s_{ij} матрицы есть внутреннее состояние, в которое переходит автомат из состояния s_i под действием входного вектора X_j с указанием состояния выхода ($k = 0$ либо $k = 1$).

Глава 3

ПРОГРАММНЫЕ СРЕДСТВА АВТОМАТИЗАЦИИ ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ БИС

Объем программных средств автоматизации проектирования БИС составляет сотни тысяч операторов и разрабатывается 2...5 лет. Эффективность применения САПР БИС во многом определяется организацией интерфейса между разработчиком БИС и САПР, способом организации обмена данными между подсистемами проектирования, возможностью настройки (адаптации) САПР на объект проектирования, оценки качества проектных решений, накопления и выбора типовых функциональных решений. Основное внимание в главе уделено трем основным проблемам, определяющим ядро интегрированной САПР и ее эффективность: 1) специализированному банку данных (БД) как совокупности базы данных и аппарата ее формирования и обслуживания; именно БД должен обеспечить информационное единство всех данных о проектируемой БИС; 2) входному языку проектирования, обеспечивающему алгоритмическое и структурное описание БИС и ее фрагментов; 3) средствам настройки на заданный объект проектирования.

§ 3.1. Специализированный банк данных САПР БИС

Различают два подхода к разработке БД: 1) на основе системы управления базами данных (СУБД) общего назначения; 2) путем создания специализированного БД. СУБД общего назначения представляет собой автономную программную систему, ориентированную на работу с детально структурированными данными, и позволяет обеспечить независимость прикладных программ от конкретного способа организации и структуры данных и тем самым сделать базу данных инвариантной к пользователям или прикладным программам. Свойство инвариантности позволяет использовать ее в САПР БИС. Однако чрезмерная универсальность СУБД снижает ее эффективность.

В автоматизированных системах управления и информационно-поисковых системах, в которых широко применяются СУБД общего назначения, база данных есть совокупность записей, содержащих сами данные и

ключи, определяющие смысловое содержание записи и позволяющие выделить ее среди других. Каждая запись информационно слабо связана с другими записями и может существовать и использоваться независимо от них. СУБД позволяет выполнять ввод, накопление, хранение, корректировку и обработку данных. Суть обработки заключается в выделении смыслового подмножения информации с использованием алгебры отношений между данными (эквивалентно, равно, больше и т. п.). Например, если база данных содержит независимые смысловые части А1—А6, то пользователю могут потребоваться различные подмножества и конфигурации данных: цепочка А1, А3, А6, цепочка А4, А2, А6 и т. п. Пользователь СУБД формирует необходимые ему запросы и выступает в роли потребителя данных. Круг и разнообразие возможных запросов (требуемых конфигураций данных) могут сильно различаться и часто заранее не определены. Поэтому основное функциональное назначение СУБД — обеспечить с приемлемыми затратами машинного времени гарантированное получение данных по произвольному запросу из уже существующей базы данных. Причем при выполнении запросов изменение самих данных не производится. В процессе эксплуатации база данных может быть значительно расширена без изменения структуры хранимых в ней данных.

В САПР БИС база данных полностью определяется объектом проектирования, характеризует его с различных сторон и с различной степенью детализации. Круг возможных запросов достаточно четко ограничен и определяется потребностями различных подсистем САПР, которые на основе информации, уже имеющейся в базе данных, в процессе работы порождают новые данные о проекте БИС. Следовательно, процесс эволюции базы данных не является процессом механического накопления данных, а есть процесс их анализа и синтеза. Разработчик БИС как пользователь СУБД и базы данных выступает в роли не столько потребителя данных, сколько в роли их создателя.

В этом смысле можно провести определенные параллели между трудом разработчика БИС и программиста. Программист реализует разработанный им алгоритм на удобном и понятном ему языке программирования. Далее транслятор преобразует текст программы в форму объектного модуля, удобную для работы программы-редактора. И в завершении программа-редактор формирует из

совокупности объектных модулей программу в виде загрузочного модуля для выполнения на ЭВМ. Аналогично, СУБД и база данных для САПР выполняют роль информационного связующего звена между разработчиком БИС, с одной стороны, и подсистемами проектирования, с другой. Поэтому при разработке перспективных сквозных САПР БИС для повышения их эффективности целесообразно развивать концепции создания специализированных БД.

В основу построения САПР ФЛП-3000 была положена идея приблизить технологию разработки проекта БИС к технологии разработки программного обеспечения. Эта идея определила *принципы организации специализированного БД* системы.

База данных БД представляет собой совокупность специализированных библиотек, построенных на основе файлов прямого доступа. Типовым элементом данных в библиотеке является раздел. Роль СУБД выполняет пакет подпрограмм обслуживания библиотек базы данных, реализующий следующие функции: начальную организацию и разметку; организацию раздела; запись информации в разделы с заданного байта; чтение информации из раздела с заданного байта; удаление разделов; печать оглавления; копирование библиотеки.

Каждая библиотека базы данных организационно состоит из нулевого блока, блоков оглавления и блоков хранения информации. Принадлежность блоков соответствующим разделам или свободной части библиотеки и порядок следования определяются списковой структурой, которая хранится в нулевом блоке. С ее помощью реализован процесс автоматической «сборки мусора», состоящий в том, что после удаления какого-либо раздела список принадлежащих ему блоков автоматически включается в список свободных блоков библиотеки, которые становятся доступными для повторного использования при организации новых разделов или расширении существующих.

Раздел идентифицируется в оглавлении библиотеки двойным именем и типом. Имена 1 и 2 формируемого раздела указываются пользователем системы в процессе описания БИС или заданий на запуск подсистем проектирования. Тип раздела, структура и состав хранимой в нем информации определяются соответствующей подсистемой проектирования в момент его организации.

Информация в разделах библиотеки представляется

в виде совокупности структур и массивов данных. Запись данных в разделе осуществляется непосредственно из оперативной памяти ЭВМ без каких-либо преобразований. В начале раздела находится информация о размерах структур данных, адресах начала их расположения в разделе, длине (в байтах) отдельных массивов и подструктур. Все манипуляции с библиотеками и разделами система осуществляет автоматически, сопровождая их печатью соответствующих информационных сообщений. Пользователь косвенно управляет процессом создания и обновления разделов в библиотеках, используя средства языка ОЦИС-РП.

Все библиотеки базы данных разделяются на три группы: системные, сопровождения проектов БИС и рабочие. Системные библиотеки являются библиотеками общего пользования и предназначены для хранения информации о настройке системы ФЛП-3000 на различные типы БМК, серии ИС, а также для хранения и постепенного накопления инженерного опыта в виде типовых фрагментов схем (типовых функциональных решений). В библиотеках сопровождения проектов хранится информация о конкретных разрабатываемых БИС (результаты трансляции фрагментов схемы, раскрытия схемы до заданного уровня детализации, результаты моделирования и синтеза топологии и т. п.). Рабочая библиотека создается каждый раз в момент запуска системы и служит для временного хранения информации. По завершении работы системы она уничтожается.

База данных САПР ФЛП-3000 содержит разделы хранения информации о разрабатываемой БИС девяти основных типов: ПОДСИСТЕМА, ЗАДАНИЕ, ФРАГМЕНТ, СХЕМА, ДИНАМИКА, СТАТИКА, ТПР, РАЗМЕЩЕНИЕ, ТОПОЛОГИЯ. Взаимодействие разделов с соответствующими подсистемами САПР ФЛП-3000 представлено на рис. 3.1.

Раздел ПОДСИСТЕМА формируется транслятором ОЦИС-РП в одноименной директиве, содержит сведения о параметрах запуска подсистем проектирования и используется для мнемонической настройки директивы ЗАДАНИЕ транслятора ОЦИС-РП. Раздел ЗАДАНИЕ формируется транслятором ОЦИС-РП в одноименной директиве. Он располагается в рабочей библиотеке системы и содержит конкретные значения параметров запуска подсистем проектирования. Информация раздела ЗАДАНИЕ используется монитором системы и соответ-

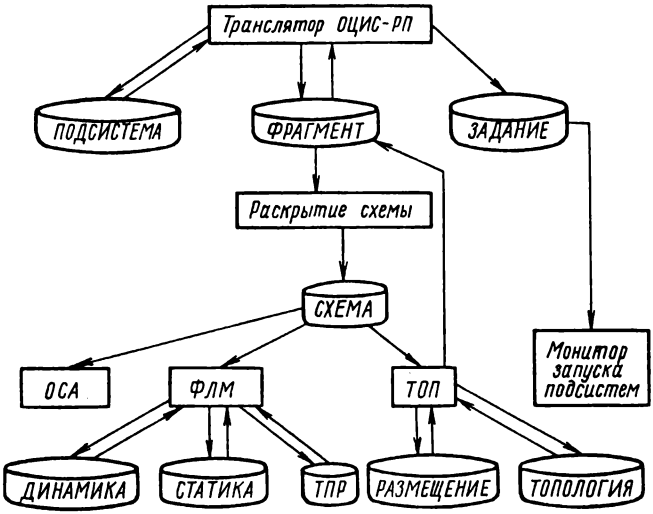


Рис. 3.1. Взаимодействие разделов базы данных с подсистемами САПР ФЛП-3000

ствующей подсистемой проектирования для определения режима ее работы.

Раздел ФРАГМЕНТ формируется транслятором ОЦИС-РП при трансляции описания фрагмента схемы и подсистемой ТОП при восстановлении логической схемы из топологии. Раздел содержит информацию о структуре фрагмента схемы, алгоритме его функционирования, внешних выводах и параметрах и используется при раскрытии схемы до элементов заданного типа. Раздел СХЕМА формируется программой раскрытия схемы, вызываемой из подсистем. Раздел содержит описание структуры раскрытой схемы, идентификаторы элементов, параметры элементов для соответствующей подсистемы.

Разделы ДИНАМИКА и СТАТИКА формируются подсистемой ФЛМ. Оба эти раздела хранят результаты логического моделирования и используются при выводе диаграмм динамического и статического режимов работы схемы на экран дисплея (с помощью диалогового анализатора) и на АЦПУ. Раздел ТПР формируется подсистемой ФЛМ, хранит таблицы проверок работоспособности и используется при оценке полноты контролирующих тестов, аттестации проекта БИС и подготовке магнитной ленты для автоматизированного контрольно-измеритель-

ного оборудования. Раздел РАЗМЕЩЕНИЕ формируется подсистемой синтеза топологии БИС, содержит описание привязки ячеек схемы на поле БМК и используется при трассировке межсоединений. Раздел ТОПОЛОГИЯ формируется подсистемой синтеза топологии БИС или на рабочих станциях при ручной разводке или коррекции топологии БИС. Раздел содержит описание переменных слоев топологии БИС в формате SOURCE. Он используется при контроле соответствия топологии БИС исходной логической схеме и подготовке магнитной ленты для фотонаборной установки.

Для надежного функционирования САПР важное значение имеет контроль информационного единства базы данных. В САПР ФЛП-3000 эта задача решается путем проверки соблюдения причинно-следственных связей между разделами. Правильность связей проверяется по датам организации разделов и контрольным суммам.

§ 3.2. Средства алгоритмического и структурного описания объектов функционально-логического проектирования

В сквозной интегрированной САПР БИС желательно иметь один входной язык проектирования, удовлетворяющий ряду требований.

1. Входной язык должен быть ориентирован на работу с банком данных системы и быть у пользователя основным средством управления процессом занесения информации в БД и процессом ее корректировки. В связи с этим возможности входного языка существенно влияют на эффективность применения БД.

2. Входной язык должен обладать возможностью настройки подсистем САПР на объект проектирования. Широкий спектр задач, решаемых в процессе проектирования БИС, требует вводить во все подсистемы проектирования большое количество различных по характеру и форме представления данных об объекте проектирования. Например, для настройки САПР ФЛП-3000 на конкретный БМК требуется средствами входного языка описать имена, внешние выводы, алгоритм функционирования, динамические свойства ячеек БМК; топологию кристалла; множество технологических и топологических ограничений и т. п.

3. Входной язык должен на различных уровнях представления описывать объект проектирования с различной

степенью абстракции. Разработка БИС — итерационный процесс постепенного перехода от словесного описания алгоритма функционирования БИС к детально отработанной структуре логической схемы и топологии. Поэтому входной язык должен обеспечивать возможность описания БИС с различной степенью точности от ее алгоритмической модели до структурных моделей Э1—Э3.

4. Входной язык должен описывать среду функционирования объекта проектирования путем задания различных входных воздействий на БИС, влияние на функционирование БИС автоматизированного контрольно-измерительного оборудования, изменения напряжения источника питания, температуры окружающей среды и др.

5. Входной язык должен управлять порядком запуска и режимом работы подсистем САПР для реализации сквозного проектирования БИС. Каждая вновь запускаемая подсистема основывается на информации в БД, подготовленной предыдущей подсистемой. Поэтому средства входного языка обязательно должны иметь механизм управления запуском подсистем. Гибкость такого механизма и его адаптация к вновь создаваемым подсистемам существенно влияют на эффективность отдельных подсистем и САПР в целом.

Для САПР ФЛП-3000 разработан специальный многоуровневый структурно-функциональный входной язык ОЦИС-РП, удовлетворяющий всем перечисленным требованиям.

Синтаксис языка ОЦИС-РП. При разработке синтаксиса языка используются синтаксические диаграммы как средство наиболее наглядного и точного его описания. Элементами изображения синтаксических диаграмм являются стрелки, прямоугольники, кружки и овалы. Стрелки устанавливают порядок разбора синтаксической диаграммы, прямоугольник означает ссылку на другую синтаксическую диаграмму, в кружках и овалах записывают отдельные символы или совокупности символов, являющиеся конечными элементами синтаксиса.

Основными элементами языка ОЦИС-РП являются: директивы, разделители, идентификаторы, числа, пробелы и комментарии. *Директивы* — это специальные ключевые слова (ПОДСИСТЕМА, ЗАДАНИЕ и др.), предшествующие синтаксической диаграмме и однозначно ее идентифицирующие. *Разделитель* — это символ, необходимый для отделения элементов языка: ,, ., :,; ,',(,),=, /, ¤ , &, [,], ∗, >, <, !, ¬ , », #, _, +, —, %, @ . *Идентифи-*

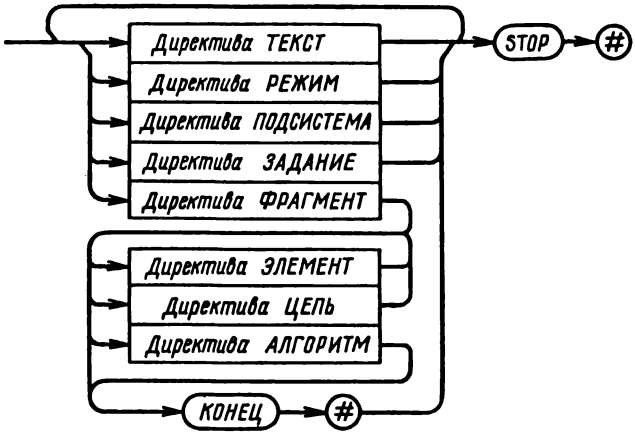


Рис. 3.2. Порядок следования директив

каторы — совокупности букв и цифр, используемые для обозначения описываемых объектов. Числа могут быть целыми и вещественными. Комментарии начинаются с комбинации разделителей /*, а завершаются разделителями *-/.

Директивы языка ОЦИС-РП делятся на три группы: 1) *сервисные директивы* (ТЕКСТ, РЕЖИМ); 2) *директивы алгоритмического и структурного описания фрагментов схемы* (ФРАГМЕНТ, ЭЛЕМЕНТ, ЦЕПЬ, АЛГОРИТМ); 3) *директивы формирования заданий на запуск подсистем проектирования* (ПОДСИСТЕМА, ЗАДАНИЕ). Порядок следования директив регламентируется синтаксической диаграммой (рис. 3.2). Все директивы начинаются с ключевого слова и заканчиваются разделителем (#). Ключевое слово STOP указывает на окончание исходного описания и завершение работы транслятора языка ОЦИС-РП.

Сервисные директивы (ТЕКСТ, РЕЖИМ). Для повышения гибкости использования и сокращения повторяющихся частей исходного описания схемы в языке ОЦИС-РП предусмотрен аппарат текстовых переменных. Текстовая переменная — это идентификатор, которому соответствует некоторый текст. При необходимости этот текст может быть включен в любое место описания схемы путем ссылки на соответствующую текстовую переменную. Допускается взаимная вложенность текстовых переменных, а также присвоение уже существующей пере-

менной нового значения. Задание значения текстовой переменной выполняется с помощью директивы ТЕКСТ. Признаком включения текстовой переменной является разделитель %.

Пример 3.1. Присвоить некоторым параметрам ТЗ и ТФ с помощью текстовой переменной следующие значения: ТЗ = 35 нс, ТФ = 10 нс.

Решение. На основе синтаксической диаграммы директивы ТЕКСТ записываем

```
ТЕКСТ: ПАРАМ(ТЗ,ТФ)='ТЗ=%ТЗ,ТФ=%ТФ'#
```

Тогда в необходимых местах описания можно записать %ПАРАМ (35, 10), что означает присвоение: ТЗ = 35 нс и ТФ = 10 нс.

Директива РЕЖИМ предназначена для оперативного изменения режимов работы транслятора языка ОЦИС-РП. Она позволяет изменять размеры рабочих структур транслятора, управлять режимом печати исходного описания и т. п. Конкретный состав параметров, возможный диапазон их изменения и значения, устанавливаемые по умолчанию, определяются соответствующей версией транслятора языка.

Если, например, требуется изменить старое значение параметра К8 (максимальное число элементов в транслируемом фрагменте БИС) на новое, равное 500, то для этого необходимо с помощью директивы РЕЖИМ указать новое значение параметра К8 в виде РЕЖИМ:К8 = = 500#.

Структурное описание фрагмента схемы реализуется в двух вариантах: 1) с помощью директив ФРАГМЕНТ и ЭЛЕМЕНТ; 2) на основе директив ФРАГМЕНТ, ЭЛЕМЕНТ и ЦЕПЬ. В первом варианте структура схемы описывается по элементам, во втором — по цепям. В обоих вариантах описание структуры фрагмента схемы завершается по ключевому слову КОНЕЦ и разделителем # (см. рис. 3.2).

Директива ФРАГМЕНТ (рис. 3.3) позволяет задать имя и тип описываемого фрагмента схемы, его библиотеку в БД, состав и порядок следования внешних контактов, а также совокупность параметров фрагмента, необходимых для работы соответствующих подсистем проектирования. Директива ФРАГМЕНТ не имеет однозначного ключевого слова. Тип фрагмента — это идентификатор, который обозначает принадлежность фрагмента к некоторому классу схем. Например, БИС, БЛОК, ФУ (функциональный узел), ТРИГГЕР. По завершении трансляции каждого фрагмента схемы формируется раз-

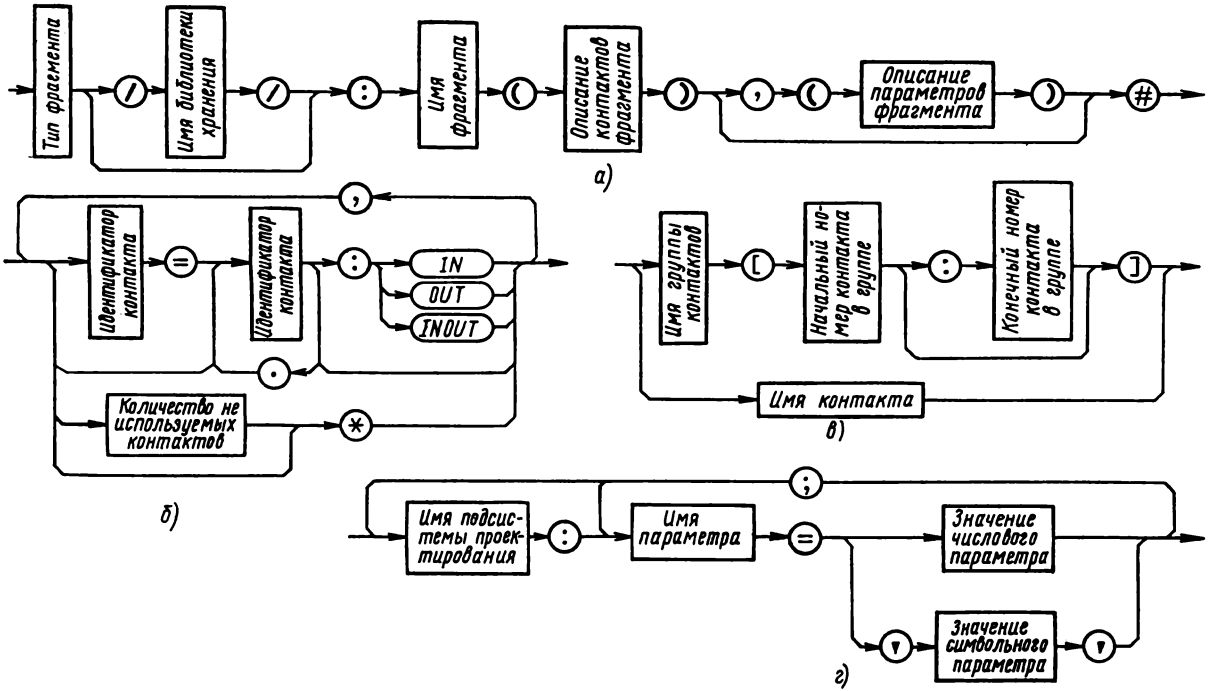


Рис. 3.3. Синтаксическая диаграмма директивы ФРАГМЕНТ:

а — основная диаграмма; *б* — описание контактов фрагмента; *в* — описание идентификатора контакта; *г* — описание параметров фрагмента схемы для подсистем проектирования

дел с описанием соответствующего фрагмента, либо в библиотеке БД, указанной в директиве, либо в рабочей библиотеке при отсутствии ссылки в директиве на какую-либо библиотеку.

Контакты фрагмента схемы могут быть описаны одиночно и группами. Одиночные контакты идентифицируются по имени, а группы — по имени группы и номеру в группе. Если при описании контактов фрагмента схемы важна их последовательность, то допускается запись неиспользованных контактов с помощью разделителя *. Кроме того, с помощью разделителя = могут быть заданы имена — синонимы контактов. Допускается их сцепление через разделитель «.» (точка).

В директиве ФРАГМЕНТ могут быть заданы параметры фрагмента, необходимые для работы подсистем проектирования. Конкретный состав параметров зависит от подсистемы. Параметры могут быть числовыми и символьными. Значение символьного параметра может представлять сложную синтаксическую конструкцию, синтаксический анализ которой осуществляется подсистемой проектирования, для которой описывается такой параметр.

Рассмотрим некоторые примеры записи директивы ФРАГМЕНТ.

```
ЯЧЕЙКА /BANK.BMK/:
  W14(K2=A,K4=C,K5=S,K6=D,K8=B),
     (FLM:
       ДОП_НАГР_ВЫХ='0,0,3,0,0';
       ЭКВ_НАГР_ВХ='1,1,0,1,1')#
```

В данном примере описаны внешние контакты ячейки W14 (см. приложение) и приведены предельные характеристики их нагрузок. Предполагается, что результаты трансляции данного фрагмента должны быть помещены в библиотеку BANK. BMK:

```
БЛОК:
  RG(CI[1:3]=УСТ.СБРОС.СИНХР,ВХ[0:7],2*,ВЫХ[15:0],ЗЕМЛЯ)#
```

Этот пример иллюстрирует применение операции сцепления при задании синонимов группы контактов CI [1:3]. Синонимом контакту с идентификаторов CI [1] задан идентификатор УСТ, контактам CI [2] и CI [3] соответственно заданы СБРОС и СИНХР. Кроме того, указано, что контакты 12 и 13 фрагмента не используются, а контакт 30 подключен к земле.

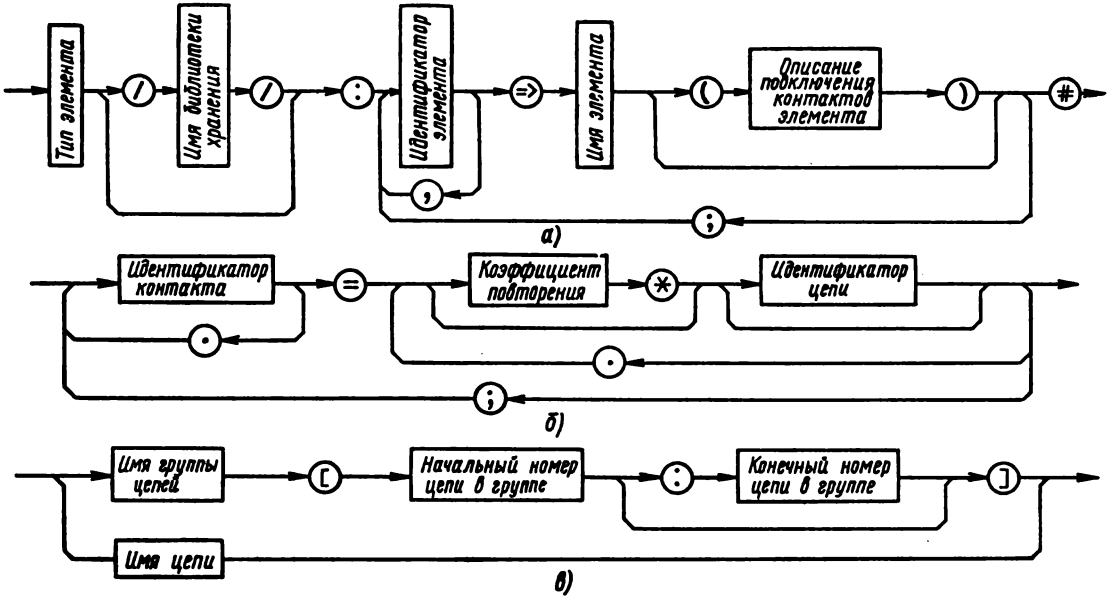


Рис. 3.4. Синтаксическая диаграмма директивы ЭЛЕМЕНТ:
а — основная диаграмма; *б* — описание подключения контактов элемента; *в* — идентификатор цепи

В *директиве ЭЛЕМЕНТ* (рис. 3.4) описываются имя и тип элементов фрагмента схемы, их библиотека в БД системы, идентификаторы элементов в соответствии с обозначениями их на схеме, а также порядок подключения контактов элементов к внешним и внутренним цепям фрагмента. Ссылка на библиотеку БД может отсутствовать. В этом случае предполагается, что информация об элементе хранится в рабочей библиотеке системы. Если через запятую перечислено несколько идентификаторов элементов, то это означает, что они имеют одинаковое имя и контакты описываются как параллельно подключенные. Подобный принцип описания целесообразно использовать для регулярных схем. Описание подключения контактов элементов может отсутствовать полностью или в нем может быть описано подключение только части контактов. В этом случае подключение оставшихся контактов должно быть выполнено в директиве ЦЕПЬ.

Идентификаторы цепи представляют любую комбинацию символов, не являющихся разделителями. Они выбираются разработчиком БИС исходя из назначения цепи. Внутренние цепи, как и внешние контакты фрагмента, могут быть одиночными или объединены в группы цепей. Одиночная цепь идентифицируется именем, цепь в группе — именем группы цепей и номером цепи в группе.

Для облегчения описания структуры схемы допускается сцепление контактов и соответствующих им цепей в группы с помощью разделителя . (точка). При этом порядок подключения контакта к цепи определяется порядковым номером контакта и цепи в полученных группах контактов и цепей. Неиспользуемые контакты должны помечаться разделителем *. Контакты, подключаемые к источникам логического нуля и единицы, подсоединяются к цепям ЗЕМЛЯ и ПИТАНИЕ, которые являются глобальными и всегда присутствуют в схеме независимо от того, указаны они в списке внешних контактов фрагмента или нет.

Приведем в качестве примера описание по элементам на языке ОЦИС-РП схемы полусумматора, собранного на трех ячейках БМК:

```
ФУ:
  PSM(A,B,S,P)#
  ЯЧЕЙКА /BANK.BMK/:
    D1=>W7(K6.K4=2*A;K8.K2=B.B;K7=D1_7;K1=D1_1);
    D2=>V1(K2=D1_1;K1=P;K4=ЗЕМЛЯ;K5=*);
    D3=>V3(K4=P;K2=D_7;K3=S)#
КОНЕЦ#
```

В приведенном структурном описании полусумматора обозначения внутренних цепей схемы записаны в виде составных имен источников сигнала в этой цепи. Например, обозначение D1_7 составлено из идентификатора ячейки D1 и номера контакта ее выхода 7. Такой метод обозначения внутренних цепей прост в понимании и, главное, не требует введения дополнительных обозначений в схеме.

Директива ЦЕПЬ позволяет выполнить описание схемы по цепям (рис. 3.5). В этом случае формируется список контактов элементов, подключенных к соответствующим цепям. Цепи могут иметь идентификатор или быть безымянными. Неиспользуемые контакты элементов собираются в отдельную цепь, помеченную разделителем ✳. После идентификатора цепи могут быть указаны ее параметры. В существующей версии ОЦИС-РП используется один параметр С (узловая емкость). Одна цепь отделяется от другой разделителем ; (точка с запятой). Контакты элементов в цепи записываются в последовательности: идентификатор элемента, разделитель (/), идентификатор контакта. Допускается указать несколько идентификаторов элементов и имен контактов в круглых скобах. При этом считается, что контакты данных элементов подключены к описываемой цепи. Например, описание цепи (D1, D7)/(K6, K3); полностью идентично описанию D1/K6, D1/K3, D7/K6, D7/K3;

В качестве иллюстрации применения директивы ЦЕПЬ приведено структурное описание полусумматора по цепям на языке ОЦИС-РП:

```
ФУ:
  PSM(A,B,S,P)#
  ЯЧЕЙКА /BANK.BMK/:
    D1=>W7;D2=>V1;D3=>V3#
  ЦЕПЬ:
    A => D1/(K6,K4);
    B => D1/(K8,K2);
    S => D3/K3;
    P => D2/K1,D3/K4;
    D1/K7,D3/K2;
    D1/K1,D2/K2;
    ЗЕМЛЯ => D2/K4;
    *     => D2/K5#
КОНЕЦ#
```

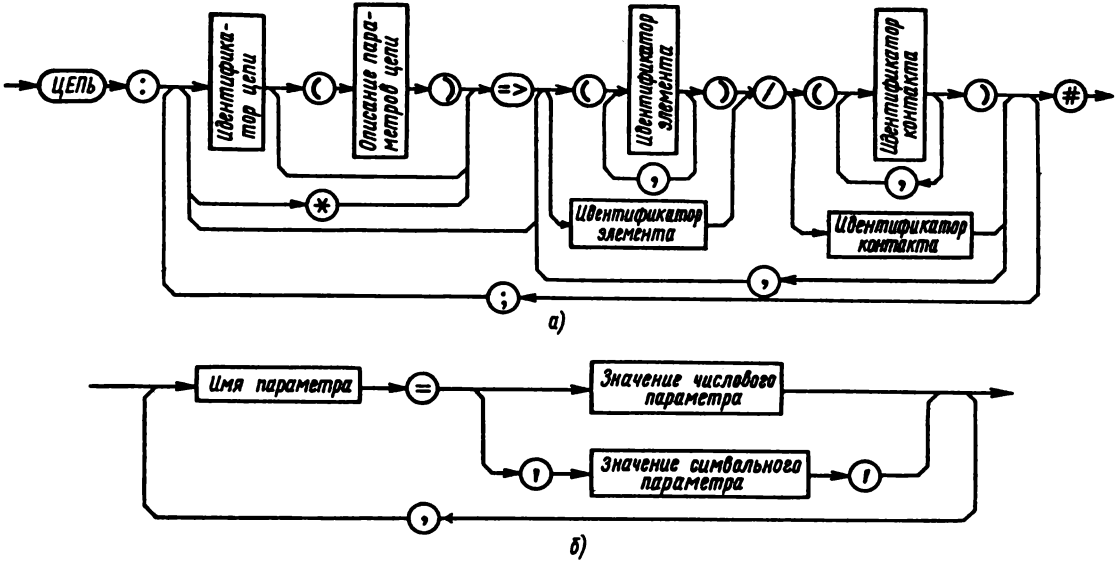


Рис. 3.5. Синтаксическая диаграмма директивы ЦЕПЬ:
а — основная диаграмма; *б* — описание параметров цепи

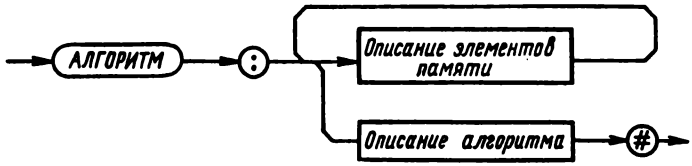


Рис. 3.6. Синтаксическая диаграмма директивы АЛГОРИТМ

Алгоритмическое описание фрагмента схемы на языке ОЦИС-РП осуществляется с помощью директивы АЛГОРИТМ (рис. 3.6) и семи операторов: 1) описание элементов памяти (ПАМЯТЬ); 2) описание процесса (ПРОЦЕСС); 3) описание условия (ЕСЛИ); 4) вычисление логического состояния; 5) описание цикла (ЦИКЛ-ПОКА); 6) вызов процесса; 7) завершение работы алгоритма (ЗАВЕРШИТЬ).

Оператор описания элементов памяти фрагмента (рис. 3.7) позволяет ввести мнемонические обозначения для ОЗУ, ПЗУ, регистров памяти, отдельных триггеров, используемых в схеме, при необходимости задать начальные их значения. Каждый элемент памяти может принимать значения: 0, 1, × — логически неопределенное состояние. При отсутствии начальной установки элементов памяти считается, что они находятся в неопределенном состоянии. Начальная установка массивов памяти осуществляется поразрядно слева направо, сверху вниз. При этом восьмеричные, десятичные и шестнадцатеричные цифры заменяются комбинацией нулей и единиц. Одна восьмеричная цифра соответствует трем двоичным разрядам, одна десятичная и шестнадцатеричная — четырем двоичным разрядам. Например, записи цифры 16 в восьмеричной системе 16″8 в двоичной системе соответствует 001110″2; 69″10—01101001″2; OF3″16—000011110011″2.

В качестве иллюстрации применения директивы ПАМЯТЬ опишем ПЗУ, содержащее четыре 8-разрядных числа:

```
ПАМЯТЬ ПЗУ1[1:4,0:7]<FO"16,2*11"16,11001110>;
```

Такая запись означает, что элементам матрицы ПЗУ1 присваиваются следующие значения:

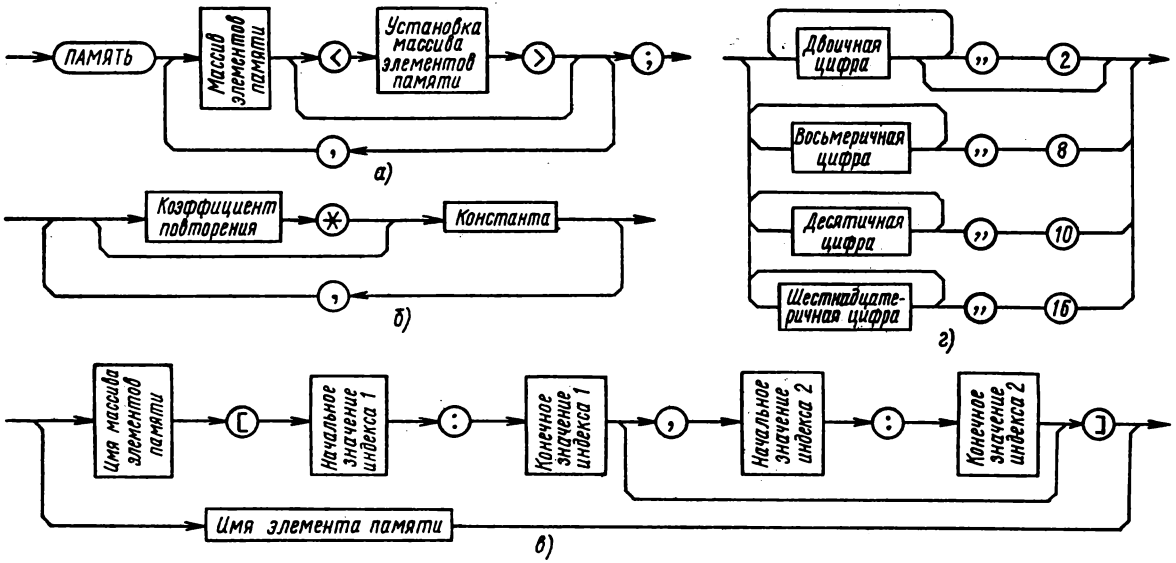


Рис. 3.7. Синтаксическая диаграмма описания элементов памяти:
а — основная диаграмма; *б* — установка элементов памяти; *в* — массив элементов памяти; *г* — константа

Число	Разряды							
	0	1	2	3	4	5	6	7
1	1	1	1	1	0	0	0	0
2	0	0	0	1	0	0	0	1
3	0	0	0	1	0	0	0	1
4	1	1	0	0	1	1	1	0

Описание алгоритма (рис. 3.8) позволяет выделить некоторую совокупность операторов алгоритма (по аналогии с подпрограммами в языках программирования) и, ссылаясь на имя данного процесса, выполнять их в различных местах алгоритма.

Оператор вычисления логического состояния (рис. 3.9) является основным, наиболее сложным и функционально насыщенным оператором описания алгоритма. Он позволяет выполнять над отдельными переменными и векторами следующие операции: ¤—поразрядное суммирование по модулю два; ! — поразрядная дизъюнкция; & — поразрядная конъюнкция; * — арифметическое умножение; / — арифметическое деление; + — арифметическое сложение; — — арифметическое вычитание; > — больше; < — меньше; >= — больше или равно; <= — меньше или равно; ¬ = — не равно; = — равно; SV1 — свертка по единице; SV0 — свертка по нулю; SL — сдвиг влево; SP — сдвиг вправо; CSL — циклический сдвиг влево; CSP — циклический сдвиг вправо.

В логическом выражении допускаются операции сцеп-

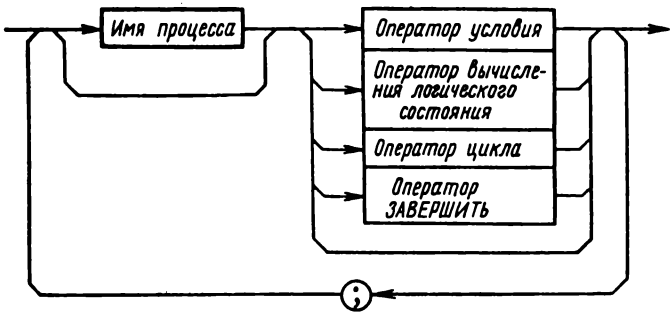
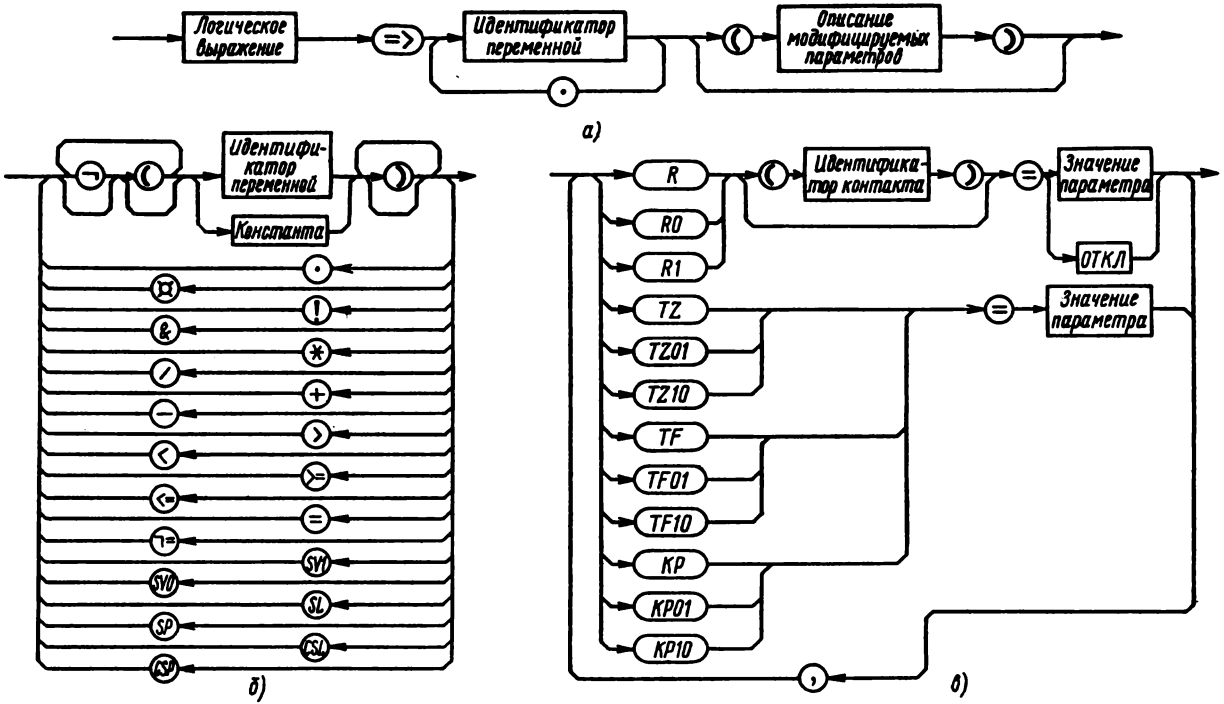


Рис. 3.8. Синтаксическая диаграмма описания алгоритма функционирования


Логическое выражение
=>
Идентификатор переменной
Описание модифицируемых параметров
а)
Идентифи-катор переменной
Константа
SV1
SV0
SL
SP
CSL
CSV
б)
R
R0
R1
TZ
TZ01
TZ10
TF
TF01
TF10
KP
KP01
KP10
Идентификатор контакта
Значение параметра
ОТКЛ
Значение параметра
в)

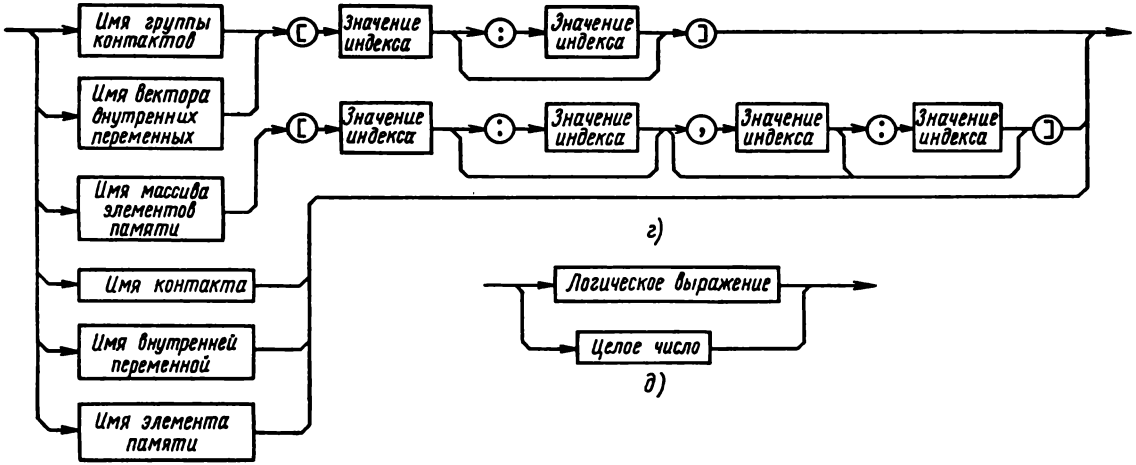


Рис. 3.9. Синтаксическая диаграмма оператора вычисления логического состояния:
а — основная диаграмма; *б* — логическое выражение; *в* — описание модифицируемых параметров; *г* — идентификатор переменной; *д* — значение индекса

ки (обозначаются разделителем .) отдельных переменных и векторов.

Например, запись A.B.C [0:4] означает, что переменные *A* и *B* сцеплены с вектором C [0:4] и образуют новый вектор, состоящий из семи разрядов, два старших разряда которого определяются переменными *A* и *B*, а пять младших разрядов — вектором C [0:4].

При работе с векторами внешних контактов, внутренних переменных и массивами элементов памяти допускается опосредованное указание значения индексов через логические выражения, значения которых вычисляются в процессе работы алгоритма.

Пусть, например, имеется вектор элементов памяти A [0:7]. Необходимо выбрать одно значение (один элемент памяти) вектора. Решение этой задачи можно записать в виде

$$A\,[\,\neg B.C.(L\&M[6])\,], \qquad (3.1)$$

где B, C, L, M[6] — некоторые логические переменные, определяющие выбор одного из восьми значений. В результате вычисления логического выражения, заключенного в квадратные скобки, получается адрес элемента памяти. В зависимости от значений переменных B, C, L, M[6] адрес может принимать значения от 000 до 111, т. е. от 0 до 7. Схемотехническое решение выбора одного из восьми элементов памяти согласно выражению (3.1) приведено на рис. 3.10.

В операторе вычисления логического состояния результат логического выражения может присваиваться отдельным переменным, векторным переменным или векторам, полученным сцеплением переменных. Старшие разряды результата располагаются слева.

Если, например, требуется сложить два 16-разрядных числа A и B, то эту операцию можно записать в виде A[1:16] + B[1:16] = P·S[1:16], где S[1:16] — сумма 16-разрядных чисел A и B; P — 17-й разряд переполнения.

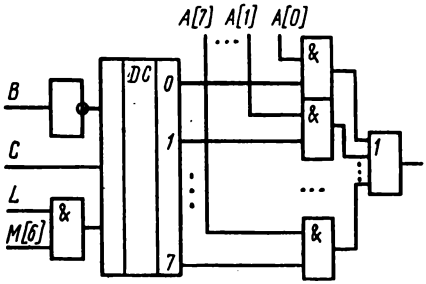


Рис. 3.10. Схемотехнический аналог алгоритмического выражения A[¬B.C.(L&M[6])]

Для внешних контактов фрагмента схемы, которые являются выходами или входами/выходами, могут быть заданы совокупности параметров, определяющие динамические свойства работы фрагмента схемы по соответствующему выходу:

1) RO, RI, R —

сопротивления (кОм) источника сигнала на выходе (R0 — при формировании лог. 0, R1 — при формировании лог. 1, R означает, что R0 = R1 = R). Если задан параметр ОТКЛ, то при формировании соответствующего логического состояния выход находится в высокоимпедансном состоянии;

2) TZ01, TZ10, TZ — задержка (нс) распространения сигнала от момента активизации модели фрагмента до появления реакции на выходе, т. е. до начала переключения (TZ01 — время задержки переключения из 0 в 1; TZ10 — время задержки переключения из 1 в 0; TZ означает, что TZ01 = TZ10 = TZ);

3) TF01, TF10, TF — длительность фронта переключения (нс) сигнала на выходе при идеальном сигнале на входе. Данные параметры характеризуют влияние на выход внутренних емкостных цепей фрагмента (TF01 — длительность фронта переключения из 0 в 1; TF10 — длительность фронта переключения из 1 в 0; TF означает, что TF01 = TF10 = TF);

4) KP01, KP10, KP — коэффициент передачи фронта с входа на выход, определяющий, какая часть длительности фронта на входе, активизирующем модель, учитывается при вычислении длительности сигнала на выходе.

По умолчанию, принимается R = 0, TZ = 1, TF = = KP = 0. Параметры TZ01, TZ10, TZ могут быть указаны для элементов памяти, т. е. может быть задано время распространения сигнала от регистра к регистру внутри модели фрагмента.

Оператор условия (рис. 3.11) позволяет реализовать выбор из нескольких возможных альтернатив. Если вы-

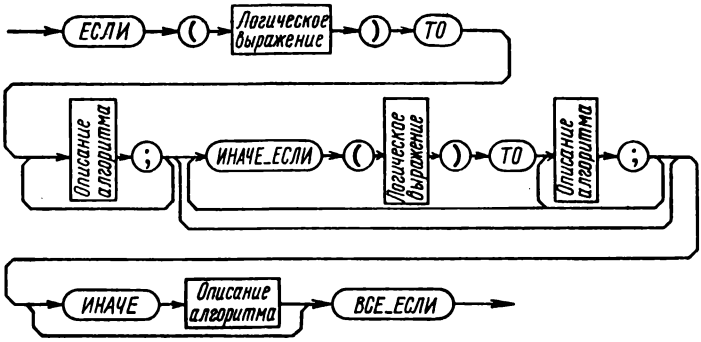


Рис. 3.11. Синтаксическая диаграмма оператора условия

численное значение логического выражения истинно, то выполняется совокупность операторов после ключевого слова ТО, в противном случае — совокупность операторов после ключевого слова ИНАЧЕ. Ключевое слово ВСЕ_ЕСЛИ служит признаком завершения оператора условия и является обязательным.

В качестве примера приведем запись:

```
ЕСЛИ (A&B) ТО ВХ1=>П1;ВХ2=>П2;

ИНАЧЕ_ЕСЛИ (-D) ТО П1=>D;

ИНАЧЕ -Q=>Q(TZ=10);

ВСЕ_ЕСЛИ;
```

Первая строка этой записи означает, что если $A\&B = 1$, то значения сигналов с входов 1 и 2 следует заслать соответственно в П1 и П2. Вторая строка означает, что если $A\&B = 0$ и $\bar{D} = 1$, то значение П1 переслать в D, третья означает, что если $A\&B = 0$ и $\bar{D} = 0$, то значение $\bar{Q}$ заслать на QI, с задержкой в 10 нс.

Оператор цикла (рис. 3.12) позволяет циклически повторять выполнение совокупности операторов до тех пор, пока логическое выражение, указанное после ключевого слова ЦИКЛ-ПОКА, является истинным. Признаком завершения оператора цикла служит ключевое слово ВСЕ_ЦИКЛ. Вызов процесса осуществляется путем указания в нужном месте алгоритма имени процесса. При каждой встрече в потоке операторов имени процесса выполняются операторы, принадлежащие данному процессу.

Оператор ЗАВЕРШИТЬ (рис. 3.13) позволяет прервать выполнение алгоритма модели фрагмента, передать в подсистему ФЛМ код прерывания и текст с комментариями о прерывании для вывода на АЦПУ.

Пример 3.2. Записать на языке ОЦИС-РП алгоритмическую модель стробирующего дешифратора типа «3—8» и на основе его словесного описания. При сигнале разрешения $C = 1$ каждой комбинации на входах A[0:2] должно соответствовать напряжение высокого уровня на одном из восьми выходов Y[0:7], а при $C = 0$ на всех выходах должен быть лог. 0. Лог. 1 на одном из выходов должна установиться через 25 нс после подачи сигнала $C = 1$, выходное сопротивление должно быть равно 2 кОм.

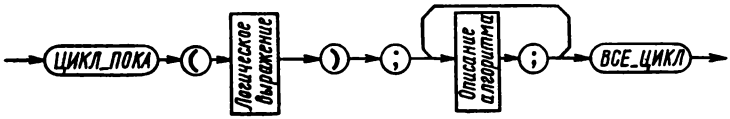


Рис. 3.12. Синтаксическая диаграмма оператора цикла

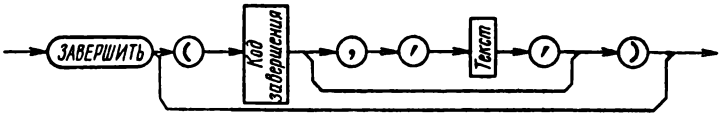


Рис. 3.13. Синтаксическая диаграмма оператора ЗАВЕРШИТЬ

Решение. Для составления алгоритмической модели дешифратора «3—8» используем директивы ФРАГМЕНТ и АЛГОРИТМ:

```
ФУ:
  DC_3_8(C,A[0:2],Y[0:7])#        /*-ОПИСАНИЕ ВНЕШНИХ ВЫВОДОВ;
  АЛГОРИТМ:
    0=>П[0:7];                    /*-УСТАНОВКА В НУЛЬ РАБОЧИХ ПЕРЕМЕННЫХ;
    ЕСЛИ (C) ТО
      1=>П[A[0:2]];               /*-УСТАНОВКА В '1'
    ВСЕ_ЕСЛИ;                     /* ПЕРЕМЕННОЙ ПО АДРЕСУ;
    П[0:7]=>Y[0:7](TZ=25,R=2)#    /*-ФОРМИРОВАНИЕ РЕАКЦИИ ВЫХОДОВ Y;
КОНЕЦ#
```

Оформление заданий на запуск подсистем проектирования. В языке ОЦИС-ПР предусмотрен единый аппарат подготовки заданий на запуск подсистем проектирования. При этом предполагается, что любая подсистема проектирования может иметь свой уникальный состав параметров запуска.

Каждый параметр различается по имени и типу (целочисленный, вещественный и символьный). Целочисленные и вещественные параметры задаются в виде чисел. Особую и важную роль играют символьные параметры, с помощью которых можно передать в подсистему различную текстовую информацию в виде любой синтаксической конструкции. Фактически это позволяет, оставаясь в рамках общих правил оформления заданий на запуск подсистем проектирования, создавать иерархию описания их параметров.

Для оформления заданий на запуск подсистем проектирования используются две директивы: ПОДСИСТЕМА и ЗАДАНИЕ. *Директива ПОДСИСТЕМА* (рис. 3.14) служит для определения состава параметров запуска той или иной подсистемы. При этом задаются имя, тип (I — целочисленный, R — вещественный, C — символьный), диапазон изменения и номинальное значение целочисленных и вещественных параметров, а для символь-

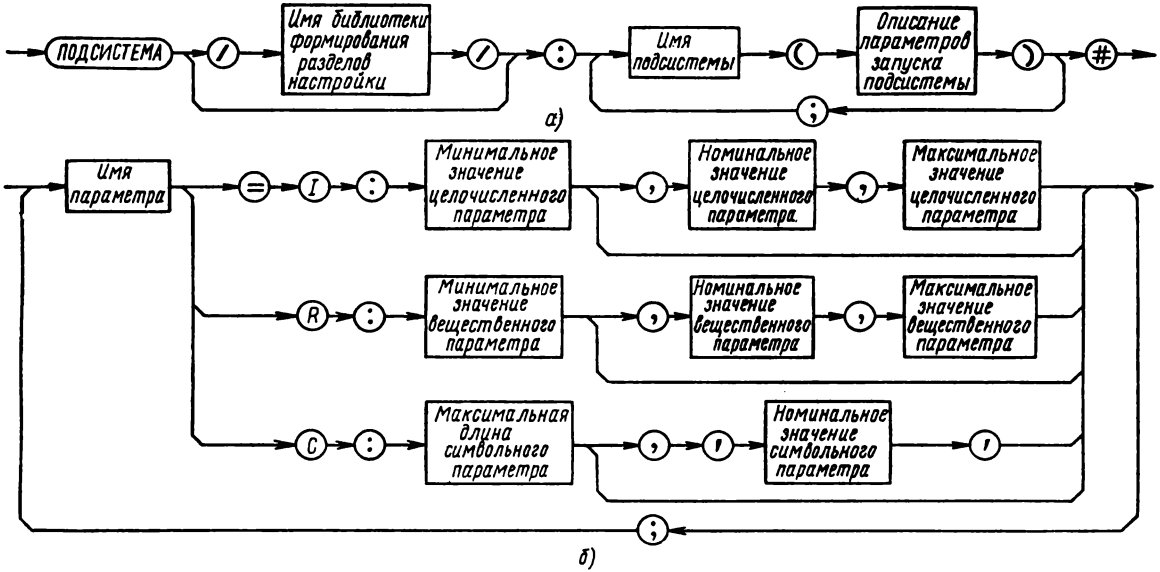


Рис. 3.14. Синтаксическая диаграмма директивы ПОДСИСТЕМА:
а — основная диаграмма; *б* — описание параметров запуска подсистемы

ных параметров также максимальная длина и значение, принимаемое по умолчанию. Состав, мнемонические имена, синтаксис символьных параметров определяются в момент разработки соответствующей подсистемы проектирования. Конкретные значения параметров задаются разработчиками САПР при настройке системы на объект проектирования и могут изменяться для различных типов БМК и серий ИС. В результате трансляций данной директивы в указанной библиотеке БД формируется раздел с описанием параметров запуска подсистемы проектирования.

Этот раздел является исходным для работы директивы ЗАДАНИЕ.

Рассмотрим в качестве примера вариант настройки на параметры программы RASKR, выполняющей раскрытие схемы до элементов заданного типа:

```
ПОДСИСТЕМА /BANK.BMK/:
  RASKR(ФРАГМЕНТ    =C:60;
        РАСКРЫТЬ_ДО=C:100,'ЭЛЕМЕНТ';
        ПОДСИСТЕМА =C:16, 'FLM';
        СХЕМА      =C:100;
        РАЗМЕР1    =I:1000,1500,3000)#
```

Как видим, данная подсистема имеет четыре символьных и один целочисленный параметр. Параметр ФРАГМЕНТ позволяет задать имя, тип и библиотеку хранения головного фрагмента схемы. Параметр РАСКРЫТЬ_ДО содержит список типов элементов нижнего уровня (по умолчанию раскрытие производится до уровня элементов). Параметр ПОДСИСТЕМА указывает, для какой подсистемы будет производиться раскрытие схемы. Параметр СХЕМА служит для указания имени и библиотеки хранения в БД раскрытой схемы. Параметр РАЗМЕР1 указывает размер рабочей структуры программы.

При трансляции *директивы ЗАДАНИЕ* (рис. 3.15) вначале считывается раздел из БД, подготовленный директивой ПОДСИСТЕМА, затем выполняется контроль параметров на состав, тип, допустимость значений. В одной директиве может быть описано несколько заданий на запуск различных подсистем проектирования. Все задания после проверки корректности их оформления накапливаются в рабочей библиотеке системы.

По завершении трансляции описания схемы и заданий на ее обработку автоматически запускается монитор системы, который распознает в рабочей библиотеке разделы

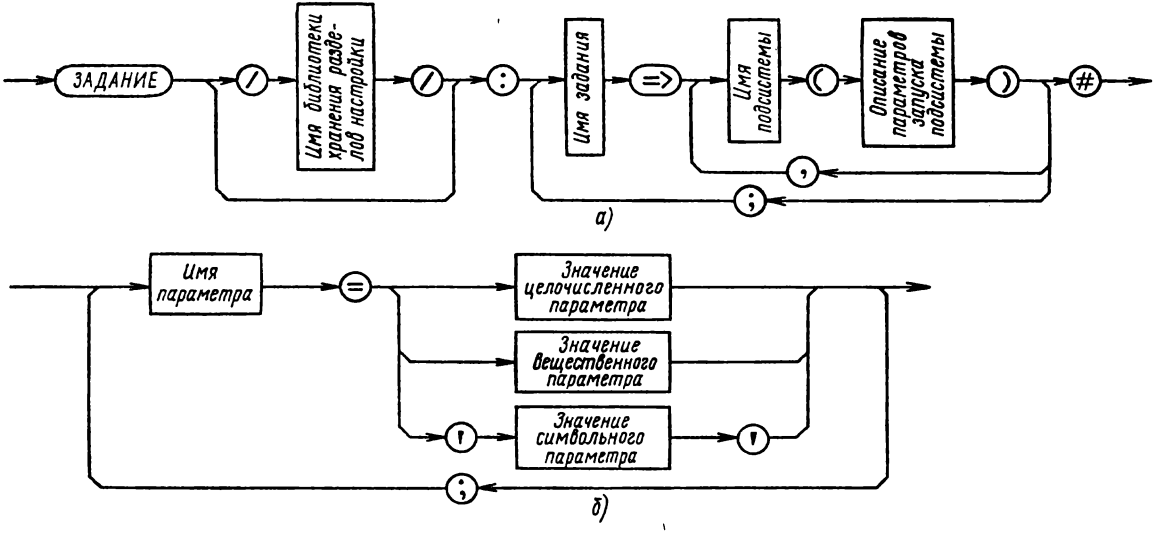


Рис. 3.15. Синтаксическая диаграмма директивы ЗАДАНИЕ:
а — основная диаграмма; *б* — описание параметров запуска подсистемы

хранения заданий и в порядке их следования вызывает соответствующие подсистемы проектирования и передает им управление.

Рассмотрим пример оформления задания на запуск программы RASKR:

```
ЗАДАНИЕ:

  ОТЛАДКА =>

    RASKR(ФРАГМЕНТ='AK1,БЛОК/BANK.BIS.AK/';

          РАСКРЫТЬ_ДО='ЯЧЕЙКА';

          СХЕМА        ='AK1/BANK.BIS.AK/')#
```

Описанное задание имеет имя ОТЛАДКА и содержит вызов программы RASKR, которая должна выполнить раскрытие схемы для уровня ячеек, БМК для подсистемы FLM (имя подсистемы FLM берется из значения, принятого, по умолчанию, для параметра ПОДСИСТЕМА). Головной фрагмент раскрываемой схемы хранится в библиотеке сопровождения проекта БИС BANK.BIS.AK, имеет имя AK1 и тип БЛОК. В результате работы программы RASKR сформированная схема помещается в ту же библиотеку BANK.BIS.AK (имя 1 раздела AK1, имя 2 — FLM, тип раздела СХЕМА).

§ 3.3. Средства функционально-логического моделирования БИС

Основу подсистемы функционально логического моделирования САПР ФЛП-3000 составляет программа асинхронного событийного моделирования БИС с управляемой адекватностью. Суть управления адекватностью заключается в том, что на разных стадиях отладки проекта БИС разработчик может по своему усмотрению указывать те или иные параметры логических моделей элементов и топологии БИС, параметры внешней среды ее функционирования, имитировать различные режимы работы АКИО и др.

В начале отладки проекта БИС используются высокоскоростные и менее точные модели элементов, а на заключительной стадии — высокоточные модели элементов, учитывающие электрические параметры, традиционно считающиеся параметрами схемотехнического анализа. Такой подход к процессу ФЛМ позволяет выбрать рациональное соотношение между быстродействием и точностью моделирования, обеспечивает высокую скорость отладки и бездефектность проектирования БИС.

Учет причинно-следственных связей между сигналами в схеме осуществляется с помощью двух списков данных — очереди будущих событий (ОБС) и очереди активизации элементов (ОАЭ). Именно с помощью этих

структур реализуется алгоритм асинхронного событийного моделирования, реализованный в подсистеме ФЛМ САПР ФЛП-3000. Приведем данный алгоритм на псевдокоде:

```
ЦИКЛ ПОКА ПРИЗНАК МОДЕЛИРОВАНИЯ РАВЕН "ДА";
  ПОМЕЩАЕМ ВХОДНЫЕ ВОЗДЕЙСТВИЯ В ОБС;
  НАХОДИМ tмод, ДЛЯ КОТОРОГО ИМЕЕТ МЕСТО ОБС ИЛИ ОАЭ;
  ЕСЛИ ОБС И ОАЭ ПУСТЫЕ ТО
    ПРИЗНАК МОДЕЛИРОВАНИЯ РАВЕН "НЕТ";
  ВСЕ ЕСЛИ;
  ЕСЛИ ДЛЯ tмод ИМЕЮТСЯ ЧЛЕНЫ ОБС ТО
    ЦИКЛ ПО ВСЕМ ЧЛЕНАМ ОБС ДЛЯ tмод;
      ВЫБИРАЕМ ОЧЕРЕДНОЕ СОБЫТИЕ ИЗ ОБС И МОДИФИЦИРУЕМ
      СОСТОЯНИЕ СООТВЕТСТВУЮЩЕГО ИСТОЧНИКА СИГНАЛА;
    ВСЕ ЦИКЛ;
    ИСКЛЮЧАЕМ ОТРАБОТАННЫЕ СОБЫТИЯ ИЗ ОБС;
    ЦИКЛ ПО ВСЕМ МОДИФИЦИРОВАННЫМ ИСТОЧНИКАМ СИГНАЛА;
      ВЫЧИСЛЯЕМ НОВОЕ СОСТОЯНИЕ УЗЛА СХЕМЫ К КОТОРОМУ ПОДКЛЮЧЕН
      МОДИФИЦИРОВАННЫЙ ИСТОЧНИК СИГНАЛА;
      ЕСЛИ СОСТОЯНИЕ УЗЛА СХЕМЫ ИЗМЕНИЛОСЬ ТО
        ОПРЕДЕЛЯЕМ ПАРАМЕТРЫ АКТИВИЗАЦИИ ЭЛЕМЕНТОВ СХЕМЫ,
        ПОДКЛЮЧЕННЫХ ВХОДАМИ К ДАННОМУ УЗЛУ, И ЗАНОСИМ ИХ В ОАЭ;
        НАКАПЛИВАЕМ ИНФОРМАЦИЮ В БД О НОВОМ СОСТОЯНИИ УЗЛА СХЕМЫ;
      ВСЕ ЕСЛИ;
    ВСЕ ЦИКЛ;
  ВСЕ ЕСЛИ;
  ЕСЛИ ДЛЯ tмод ИМЕЮТСЯ ЧЛЕНЫ ОАЭ ТО
    ЦИКЛ ПО ВСЕМ ЧЛЕНАМ ОАЭ ДЛЯ tмод;
      ВЫЗЫВАЕМ МОДЕЛЬ АКТИВИЗИРУЕМОГО ЭЛЕМЕНТА СХЕМЫ;
      ЗАНОСИМ СИГНАЛЫ С ВЫХОДА ДАННОГО ЭЛЕМЕНТА В ОБС;
    ВСЕ ЦИКЛ;
    ИСКЛЮЧАЕМ ИЗ ОАЭ ОТРАБОТАННЫЕ ЭЛЕМЕНТЫ;
  ВСЕ ЕСЛИ;
ВСЕ ЦИКЛ;
```

Очередь будущих событий необходима для временного хранения сигналов (событий) с генераторов входных воздействий и с выходов логических элементов, появляющихся с задержкой относительно времени их вычисления. Сигналы из ОБС считываются, если текущее время моделирования $t_{мод}$ совпадает с временем начала данного сигнала. В этот момент сигнал может изменить состояние узла схемы, на который он поступает. При изменении состояния узла вычисляются и помещаются в ОАЭ времена активизации моделей элементов, подключенных входами или входами/выходами к данному узлу.

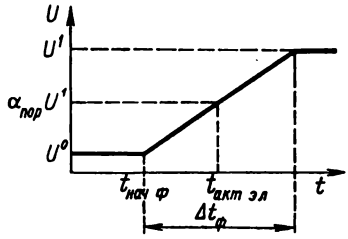


Рис. 3.16. Определение параметров сигнала

Необходимость ОАЭ заключается в том, что при моделировании с учетом длительности фронта переключения и порога срабатывания ЛЭ $\alpha_{пор}$ (рис. 3.16) моменты начала фронта переключения $t_{нач\,ф}$ и активизации элемента ($t_{акт\,эл}$) не совпадают из-за наличия длительности фронта ($\Delta T_ф$). Время активизации ЛЭ, как следует из рис. 3.16, определяется по формуле

$$t_{акт\,эл} = t_{нач\,ф} + \alpha_{пор}\Delta T_ф, \qquad (3.2)$$

где $\alpha_{пор}$ — порог срабатывания элемента $0 \leqslant \alpha_{пор} \leqslant 1$. При совпадении времени активизации элемента с текущим временем моделирования из ОАЭ извлекается номер активного элемента, вызывается его логическая модель и вычисляются новые состояния его выходов, внутренних переменных и элементов памяти, значение которых с учетом указанных в алгоритме задержек помещается в ОБС. Затем цикл работы алгоритма повторяется.

В процессе моделирования логическая схема БИС представляется в виде совокупности моделей элементов и моделей межэлементных связей (цепей). Каждый элемент представляется в виде «черного ящика» с заданным алгоритмом функционирования, т. е. логической связью между сигналами на его входах и выходах, и совокупностью его динамических параметров

Моделирование схемы обычно начинается из логически неопределенного состояния, так как в начальный момент неизвестно, в каком состоянии находятся внутренние переменные элементов, их входы и выходы. Ис-

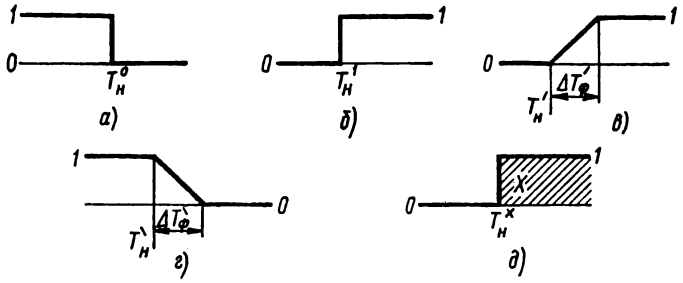


Рис. 3.17. Логические сигналы пятизначного алфавита моделирования:

а — лог. 0; *б* — лог. 1; *в* — переключение из 0 в 1 (/); *г* — переключение из 1 в 0 (\); *д* — логически неопределенное состояние

ключение составляют модели элементов памяти, в которых начальные значения переменных указаны явно.

Для отображения статического режима работы элементов в подсистеме ФЛМ использован троичный алфавит моделирования {0, 1, ×}. Для более адекватного моделирования переходных процессов в цепях схемы использован пятизначный алфавит {0, 1, ×, /, \}. Причем сигналы пятизначного алфавита имеют параметр — сопротивление источника, формирующего сигнал, т. е. выходное сопротивление логического элемента или генератора входных воздействий. Дополнительно они характеризуются девятью дополнительными параметрами (рис. 3.17): $(T_н^0/T_н^1)$ — время установления в узле состояния 0/1; (R_0/R_1) — сопротивления источника сигнала логического нуля/единицы; $T_н'/T_н'$ — время начала переключения из 0 в 1/из 1 в 0; $\Delta T_ф'/\Delta T_ф'$ — длительность фронта переключения; $T^{\times}{}_н$ — время установления в узле состояния ×. Заметим, что параметры сигнала R_0 и R_1 могут быть использованы для отображения высокоимпедансного состояния Z в цепях схемы, для этого достаточно задать $R_0 = R_1 = \infty$. Сигналы трехзначного алфавита моделирования имеет только параметр — время установления в узле соответствующего состояния: $T_н^0$, $T_н^1$, $T_н^{\times}$.

Переход от пятизначного алфавита моделирования к трехзначному, и наоборот, осуществляется на границе цепь — элемент. На входах элементов — от пяти- к трехзначному алфавиту, на выходах — от трехзначного к пятизначному. Как уже указывалось, при расчете времени активизации элемента используется параметр $\alpha_{пор}$ — по-

рог срабатывания элемента в момент $t_{\text{акт эл}}$ (вызова модели элемента). Фронт переключения аппроксимируется скачком из 0 в 1 и из 1 в 0 и определяется новое состояние входного контакта, соответственно 1 и 0. Переход к пятизначному алфавиту моделирования осуществляется следующим образом. В начале в соответствии с алгоритмом функционирования элемента вычисляется логическое состояние его выхода в троичном алфавите. Далее выполняется расчет времени начала сигнала ($T_{\text{н}}$) и длительности фронта переключения $\Delta T_{\text{ф}}$ и при $\Delta T_{\text{ф}} \neq 0$ изменяется состояние выхода. Параметр $T_{\text{н}}$ рассчитывается по формуле $T_{\text{н}} = t_{\text{акт эл}} + t_{\text{зд эл}}$, где $t_{\text{зд эл}}$ — задержка распространения сигнала в элементе.

Длительность фронта переключения определяется выражением

$$\Delta T_{\text{ф}} = \Delta t_{\text{ф эл}} + R_{\text{вых}} C_{\text{вых}} + K_{\text{п}} \Delta T_{\text{ф вх}} + \Delta t_{\text{ф ц}}, \qquad (3.3)$$

где $\Delta t_{\text{ф эл}}$ — собственная длительность фронта элемента, т. е. минимально возможная длительность фронта на его выходе, определяемая при идеальном сигнале на входе и отсутствии нагрузки на выходе, характеризующие внутренние RC-цепи элемента; $R_{\text{вых}}$ — сопротивление источника сигнала на выходе ЛЭ (R_0 или R_1); $C_{\text{вых}}$ — собственная выходная емкость элемента; $K_{\text{п}}$ — коэффициент формирования, определяющий, какая часть длительности фронта на входе ($\Delta T_{\text{ф вх}}$), вызвавшем реакцию на выходе элемента, передается на выход; этот параметр характеризует усилительные свойства элемента; $\Delta t_{\text{ф ц}}$ — длительность фронта цепи, к которой подключен данный выход элемента. В общем случае эта величина может быть различной для каждого входного контакта элемента, подключенного к данной цепи.

Алгоритм функционирования и параметры элементов задаются при описании моделей элементов средствами языка ОЦИС-РП и поступают в подсистему ФЛМ из банка данных.

Отметим, что двухступенчатый алфавит моделирования и большое число параметров сигналов и моделей элементов позволяют путем автоматической коррекции параметров существенно расширить возможности моделирования различных процессов в схеме, оценить ее устойчивость к разбросу технологических параметров, температуры, напряжения источников питания, учесть задержки в топологии БИС и другие факторы.

Важное значение для программы логического моде-

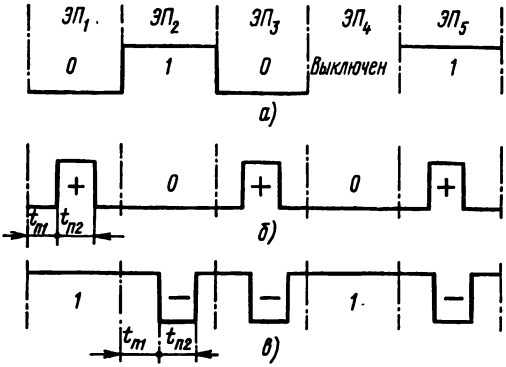


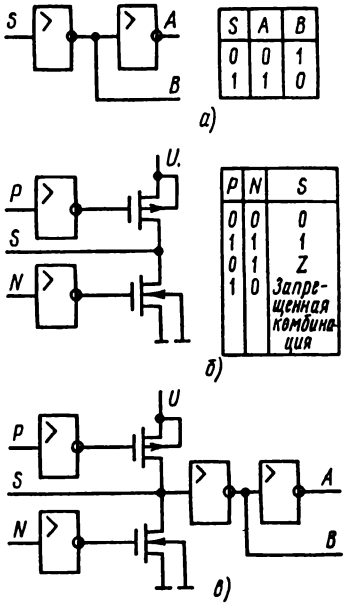
Рис. 3.18. Временные диаграммы: статического генератора входных воздействий *(а)*; импульсного генератора типа 0—1—0 *(б)* и типа 1—0—1 *(в)*

лирования имеет *описание работы генераторов входных воздействий*. Возможности таких генераторов полностью определяются возможностями современного автоматизированного контрольно-измерительного оборудования. Временная диаграмма работы генератора разделена на равные промежутки времени, называемые элементарными проверками. Так как генерация внутри БИС запрещена, то переключения внутри схемы могут быть вызваны только изменениями логических состояний выводов БИС, т. е. входными воздействиями.

Генераторы могут быть статическими и импульсными. Статические генераторы могут изменять свое состояние только в начале элементарных проверок (ЭП), т. е. состояние статического генератора фиксируется на время всей элементарной проверки и может принимать только одно значение из трех: 0, 1 или выключен (рис. 3.18, *а*). Импульсные генераторы в отличие от статических могут менять свое состояние дважды на протяжении одной элементарной проверки. Различают импульсные генераторы типа 0—1—0 и 1—0—1 (рис. 3.18, *б, в*). Импульсные генераторы имеют всего два параметра $t_{п1}$ и $t_{п2}$, которые определяют моменты появления импульсов в пределах элементарной проверки.

Все выводы БИС разделяются на входные, выходные и типа вход/выход, которые могут выступать в качестве как входов, так и выходов. В соответствии с этим имеется

три типа периферийных ячеек: входные, выходные и совмещенные (входные/выходные). На рис. 3.19 представлены логические схемы периферийных ячеек БМК различных типов, обеспечивающие связь БИС с внешней средой. Входная ячейка (рис. 3.19, *а*) обеспечивает усиление и инвертирование входного сигнала, поступающего на внешний контакт S. Выходная ячейка (рис. 3.19, *б*) формирует на выходном контакте S сигналы 0, 1 или Z в зависимости от управляющих сигналов P, N. Совмещенная ячейка (рис. 3.19, *в*) объединяет обе предыдущие схемы и может обеспечить как ввод, так и вывод информации.



S	A	B
0	0	1
1	1	0

P	N	S
0	0	0
1	1	1
0	1	Z
1	0	Запрещенная комбинация

Рис. 3.19. Логические схемы и таблицы работоспособности периферийных ячеек БМК:
а — входная ячейка; *б* — выходная; *в* — совмещенная

При описании временных диаграмм в подсистеме ФЛМ используют следующие символы:

1) для статических генераторов 0 — состояние лог. 0; 1 — состояние лог. 1; D — контролируемый выход или вход/выход; X — неконтролируемый выход или вход/выход;

2) для импульсных генераторов типа 0—1—0 символ 0 — это состояние лог. 0 (импульс отсутствует); + — генератор вырабатывает импульс;

3) для импульсных генераторов типа 1—0—1: 1 — состояние лог. 1 (импульс отсутствует); — — генератор вырабатывает импульс.

Аналогично другим подсистемам САПР ФЛП-3000 запуск программы моделирования осуществляется с помощью директивы ЗАДАНИЕ. Программа моделирования имеет большое число параметров запуска, которые позволяют активно управлять процессом моделирования. Условно параметры запуска можно разделить на четыре группы: 1) управления процессом моделирования; 2) управления адекватностью моделирования; 3) имитации

работы автоматического контрольно-измерительного оборудования; 4) управления накопления результатов моделирования в БД.

Параметры управления процессом моделирования указывают: какую схему необходимо отмоделировать; в какой библиотеке БД она хранится; какова длительность элементарной проверки; сколько элементарных проверок необходимо выполнить; в каких узлах необходимо контролировать возникновение гонок фронтов и др.

К параметрам управления адекватностью моделирования можно отнести: шаг моделирования, который определяет минимальный дискрет, до которого округляются все временные параметры схемы (параметры логических моделей, длительности фронтов и т. д.); максимально допустимая длительность короткого замыкания в узле сигналов противоположного типа (/, \); допустимые уровни помехи, используемые при анализе гонок фронтов (если амплитуда сигнала сбоя не превышает уровня помехи, то он считается не опасным; в противном случае фиксируется сигнал сбоя); признак, указывающий на необходимость учета задержки в топологических цепях, и др.

Параметрами имитации работы измерительного оборудования являются: сопротивление генераторов внешних воздействий, температура окружающей среды, напряжение источников питания, емкость выводов, параметры имитации дребезга при включении и отключении генераторов внешних воздействий; временная диаграмма работы и др.

Параметры управления накоплением результатов моделирования в БД позволяют указать, в каких узлах схемы в процессе моделирования необходимо накапливать (хранить) временные диаграммы, состояния узлов схемы в конце элементарных проверок по завершении всех переходных процессов, состояния выводов БИС для формирования управляющей ленты для контроля.

§ 3.4. Средства настройки САПР на библиотечные элементы

Настройка подсистемы ФЛМ проводится в три этапа. На первом этапе определяется состав базовых ЛЭ библиотеки ячеек БМК, рассчитываются их динамические параметры и разрабатываются алгоритмические модели,

на втором разрабатываются структурная и алгоритмическая модели ячеек БМК, на третьем — из ячеек БМК создается библиотека типовых функциональных решений, которая обычно содержит несколько десятков наиболее часто применяемых функциональных узлов и блоков.

Определение динамических параметров базовых ЛЭ осуществляется путем постановки ряда экспериментов по программе схемотехнического анализа ОСА-ЕС. Расчет параметров сигналов на выходе ЛЭ осуществляется в программе логического моделирования по следующим формулам:

$$T_{н} = T_{акт\ эл} + \Delta t_{зд\ эл}, \qquad (3.4)$$

$$\Delta T_{ф\ вых} = K_{п} \Delta T_{ф\ вх} + C_{нагр} R_{вых} + \Delta t_{ф\ эл}, \qquad (3.5)$$

где $T_{н}$ — время начала фронта сигнала на выходе элемента; $T_{акт\ эл}$ — время активизации элемента; $\Delta t_{зд\ эл}$ — собственная задержка элемента; $\Delta T_{ф\ вых}$ — длительность фронта сигнала на выходе элемента; $K_{п}$ — коэффициент передачи длительности фронта сигнала с входа элемента на его выход; $T_{ф\ вх}$ — длительность фронта сигнала на активном входе элемента; $C_{нагр}$ — емкостная нагрузка на выходе элемента; $R_{вых}$ — сопротивление источника сигнала; $\Delta t_{ф\ эл}$ — собственная длительность фронта сигнала на выходе элемента.

Параметры $\Delta t_{зд\ эл}$ и $\Delta t_{ф\ эл}$ определяются из графиков переходных процессов в схеме базового ЛЭ при подаче на его входы трапецеидальных импульсов с длительностью фронта, близкой нулю, и при отсутствии нагрузки на выходе. При этом $\Delta t_{зд\ эл}$ есть промежуток между моментом подачи входного сигнала и моментом начала переключения на соответствующем выходе. Параметр $\Delta t_{ф\ эл}$ вычисляется путем аппроксимации реального фронта выходного сигнала прямой линией (рис. 3.20).

Как видно из формулы (3.5), сопротивление источника сигнала $R_{вых}$ можно рассчитать при $\Delta T_{ф\ вх} = 0$ по формуле

$$R_{вых} = (\Delta T_{ф\ вых}|_{при\ \Delta T_{ф\ вх} = 0} + \Delta t_{ф\ эл}) / C_{нагр}, \qquad (3.6)$$

а коэффициент передачи при $C_{нагр} = 0$ по формуле

$$K_{п} = (\Delta T_{ф\ вых}|_{при\ C_{нагр} = 0} + \Delta t_{ф\ эл}) / \Delta T_{ф\ вх}. \qquad (3.7)$$

Из (3.6) следует, что для расчета параметра $R_{вых}$ соответствующего выхода модели ЛЭ необходимо по программе схемотехнического анализа выполнить моделирование схемы ЛЭ при идеальном сигнале на входе для

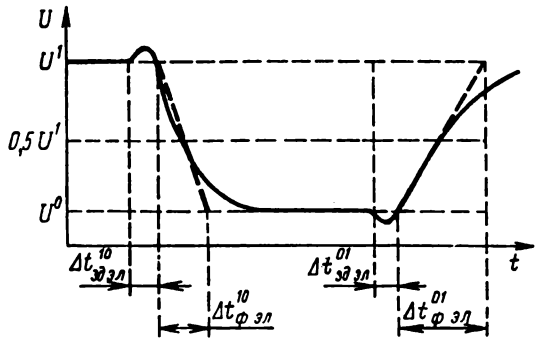


Рис. 3.20. Параметры базового логического элемента

различных емкостных нагрузок, вычислить из графиков значения $\Delta T_{ф\,вых}$, рассчитать и усреднить полученные значения параметра $R_{вых}$. Соответственно из (3.7) следует, что для расчета параметра $K_п$ проводится моделирование схемы элемента при отсутствии емкостной нагрузки на выходе для входных сигналов с различной длительностью фронта и последующее усреднение полученных значений $K_п$.

Отметим, что в общем случае расчет параметров модели необходимо производить для фронтов переключения: 01 и 10, т. е. максимально в модели может быть учтено восемь параметров: $R^1_{вых}$, $R^0_{вых}$, $\Delta t^{01}_{зд\,эл}$, $\Delta t^{10}_{зд\,эл}$, $\Delta t^{01}_{ф\,эл}$, $\Delta t^{10}_{ф\,эл}$, $K^{01}_п$, $K^{10}_п$. Полученные параметры вводятся в САПР при описании алгоритмов функционирования моделей базовых ЛЭ в директиве АЛГОРИТМ языка ОЦИС-РП.

В качестве иллюстрации процесса настройки САПР ФЛП-3000 на библиотеку БМК рассмотрим примеры.

Пример 3.3. Настроить подсистему ФЛМ на ячейку W16 БМК (см. приложение), представляющую RS-триггер, собранный на базовых ЛЭ 2ИЛИ-НЕ.

Решение поставленной задачи сводится к разработке алгоритмической модели элемента 2ИЛИ-НЕ и двум моделям (алгоритмической и структурной) ячейки W16.

1. Для разработки алгоритмической модели элемента 2ИЛИ-НЕ рассчитаем по программе схемотехнического анализа ОСА динамические параметры. Пусть в результате проведенных расчетов получены следующие значения параметров: $\Delta t^{01}_{зд\,эл} = 2$ нс; $\Delta t^{10}_{зд\,эл} = 0{,}5$ нс; $\Delta t^{01}_{ф\,эл} = 2$ нс; $\Delta t_{ф\,эл} = 1{,}5$ нс; $R^0_{вых} = 3{,}4$ кОм; $R^1_{вых} = 13$ кОм; $K^{01}_п = 0{,}09$.

2. С учетом данных параметров запишем алгоритмическую модель элемента 2ИЛИ-НЕ на языке ОЦИС-РП:

```
ЭЛЕМЕНТ /BANK.BMK/:
  2ИЛИ-НЕ(ВХ1,ВХ2,ВЫХ),
        (FLM:
           ДОП_НАГР_ВЫХ='0,0,3';
           ЭКВ_НАГР_ВХ='1,1,0')#
  ЦЕПЬ: ВХ1<C=0.12>,ВХ2<C=0.12>,ВЫХ<C=0.123>#
  АЛГОРИТМ:
    ¬(ВХ1!ВХ2)=>ВЫХ(R0=3.4,R1=13,TZ01=2,TZ10=0.5,
                    KP=0.09,TF01=2,TF10=1.5)#
КОНЕЦ#
```

Таким образом, разработана алгоритмическая модель типа ЭЛЕМЕНТ с именем 2ИЛИ-НЕ, имеющая три внешних вывода (ВХ1, ВХ2, ВЫХ) и помещенная в BANK.BMK. Для контроля подсистемой ФЛМ (обозначена как FLM) нагрузочной способности входных и выходных контактов в модели заданы два символьных параметра. В директиве ЦЕПЬ заданы значения собственных входных и выходных емкостей внешних контактов моделей: $C_{вх1} = C_{вх2} = 0{,}12$ пФ; $C_{вых} = 0{,}23$ пФ. В директиве АЛГОРИТМ описан алгоритм функционирования модели 2ИЛИ-НЕ и указаны ее динамические параметры.

3. Запишем структурную и алгоритмическую модели ячейки W16, необходимые для работы подсистемы ФЛМ, и ее топологическую модель для работы подсистемы TOPBIS:

```
ЯЧЕЙКА /BANK.BMK/:
  W16(K2=R,K8=S,K3=Q,K4=QI),
      (FLM:
         ДОП_НАГР_ВЫХ='0,0,2,2';
         ЭКВ_НАГР_ВХ='1,1,0,0';
       TOPBIS:
         ИМЯ_ЯЧЕЙКИ      ='W16';
         ТИП_ЯЧЕЙКИ      ='W';
         ПОКРЫВ_ТОЧ_ПРИВ='W:(0,0)';
         КОНТ_ЯЧ_СЛ_ВЕРХ='1:(-2,-4),(-2,10);
                          2:(-3,-2),(-1,8);
                          3:(-7,-2),(-9,8);
                          4:(-8,-4),(-8,10)';
         ЗАПР_ЗОН_СЛ_ВЕРХ=
           'ТОЧКИ:(-1,-2)<>(-1,-2),(-4,-4)<>(-4,-4),
                  (-5,-2)<>(-5,-2),(-6,-4)<>(-6,-4),
```

```
                    (-9,-2)<>(-9,-2),(-3,8)<>(-3,8),
                    (-4,10)<>(-4,10),(-5,8)<>(-5,8),
                    (-6,10)<>(-6,10),(-7,8)<>(-7,8)')#
ЭЛЕМЕНТ /BANK.BMK/:
  D1=>2ИЛИ-НЕ(ВХ1=R;ВХ2=QI;ВЫХ=Q);
  D2=>2ИЛИ-НЕ(ВХ1=S;ВХ2=Q;ВЫХ=QI)#

АЛГОРИТМ:
  -(R!QI)=>Q(R0=3.4,R1=13,TZ10=0.5,TZ01=2,
            KP=0.09,TF01=2,TF10=1.5);
  -(S!Q)=>QI(R0=3.4,R1=13,TZ01=2,TZ10=0.5,
            KP=0.09,TF01=2,TF10=1.5)#
КОНЕЦ#
```

Сформированные модели ячейки W16 помещены в BANK.BMK. Они содержат четыре внешних контакта (K2, K8, K3, K4), причем для каждого задан соответствующий синоним (R, S, Q, QI). В моделях указаны параметры для двух подсистем: ФЛМ и TOPBIS. Для подсистемы ФЛМ заданы параметры нагрузочной способности ячейки, а для подсистемы TOPBIS на специальном подъязыке описаны ее топологические параметры.

Пример 3.4. Записать в библиотеку типовых функциональных решений банка данных САПР ФЛП-3000 структурную и алгоритмическую модель детектора фронта сигнала со сбросом (рис. 3.21), собранного на ячейках БМК типов W16, W11, V3 (соответственно *D*2, *D*3, *D*1).

Решение:

```
ТЕКСТ: ПQ ='R0=3.4,R1=15.5,KP=0,TZ10=1.5,TZ01=6.6,
                TF10=4.5,TF01=8.7';
       ПQ1='R0=3.4,R1=5.1,KP=0.09,TF=1.5,TZ=1.5';
ТФУ /BANK.TFU.BMK/:
  DPFS(C,R,Q,QI),
     (FLM:
        ДОП_НАГР_ВЫХ='0,0,1,5';
        ЭКВ_НАГР_ВХ='1,2,0,0')#
  ЯЧЕЙКА /BANK.BMK/:
    D1=>V3 (K2=Q;K4=C;K3=D1_3);
    D2=>W16(R=D1_3;S=R;Q=D2_Q;K7=*);
    D3=>W11(K8=D1_3;K6=D3_Q;K4=R;K5=Q;K2=Q;K1=QI)#
```

```
АЛГОРИТМ:
  ПАМЯТЬ TR;
  ЕСЛИ (R) ТО
    0=>Q(%ПQ);Q=>QI(%ПQI);1=>TR;
  ИНАЧЕ
    ЕСЛИ (C=0) ТО
      ЕСЛИ (Q=0) ТО
        0=>TR;
      ВСЕ_ЕСЛИ;
    ИНАЧЕ
      ЕСЛИ (TR=0) ТО
        1=>Q(%ПQ);Q=>QI(%ПQI);
      ВСЕ_ЕСЛИ;
  ВСЕ_ЕСЛИ#
КОНЕЦ#
```

Модель имеет имя DPFS, относится к моделям типа ТФУ (типовые функциональные узлы) и помещена в библиотеку BANK.TFU. ВМК. Структурное описание схемы состоит из трех ячеек БМК (V3, W16, W11), т. е. для нормального функционирования модели DPFS требуется предварительно разработать модели ячеек V3 и W11. Для сокращения повторяющихся частей текста в описании модели используются две текстовые константы ПQ и ПQ1, описанные в директиве ТЕКСТ и содержащие динамические параметры выходов алгоритмической модели. В алгоритмической модели используется элемент памяти TR, основное назначение которого — фиксировать лог. 0 на входе C при наличии разрешения на ожидание фронта. При записи алгоритма использовано несколько вложенных друг в друга операторов ЕСЛИ. Для наглядности записи используются отступы, т. е. оператор, вложенный в другой оператор, располагается правее на две позиции.

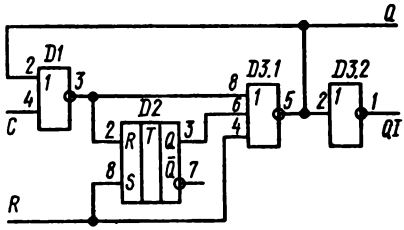


Рис. 3.21. Логическая схема детектора фронта сигнала со сбросом

Таким образом, уже на этапе настройки САПР на БМК создаются алгоритмические и структурные логические модели на трех уровнях представления (базовые ЛЭ, ячейки БМК, типовые функциональные решения), которые, естественно, различаются по точности и быстродействию моделирования. Применение моделей соответствующего уровня определяется разработчиком БИС при раскрытии схемы.

Глава 4

МЕТОДИКА РАЗРАБОТКИ ЛОГИЧЕСКОГО ПРОЕКТА БИС

В состав логического проекта БИС входят логическая схема и таблица проверки ее работоспособности. Разработка исходного варианта логического проекта БИС чаще всего осуществляется вручную, а его верификация (проверка) и доводка до требуемого уровня качества — с помощью САПР. Методика отладки логического проекта БИС определяется в основном функциональными возможностями программных средств автоматизации ФЛП (реализацией технологии многоуровневого иерархического проектирования, точностью моделирования, оценкой тестируемости схемы, аттестацией качества логического проекта БИС).

§ 4.1. Технология восходящего и нисходящего проектирования

Функциональные возможности САПР ФЛП-3000, определяемые наличием в системе банка данных и возможностью создания иерархии алгоритмических и структурных моделей БИС на различных уровнях представления, позволяют реализовать обе стратегии разработки логического проекта БИС — снизу вверх и сверху вниз (рис. 4.1).

Технология восходящего проектирования. Рассмотрим последовательность проведения работ при функционально-логическом проектировании БИС на БМК согласно стратегии восходящего проектирования (см. рис. 4.1, *а*). Для разработки логического проекта БИС должен быть задан тип БМК. В качестве исходных данных может также выступать логическая схема цифрового устройства в базисе элементов различных серий ИС или логическая схема БИС в базисе библиотечных элементов другого БМК. Кроме того, для реализации восходящего проектирования необходимо, чтобы САПР была настроена на библиотечные элементы БМК, на которых требуется проектировать логическую схему БИС. В результате настройки САПР на заданный тип БМК в соответствующей библиотеке БД системы должны храниться алгоритмиче-

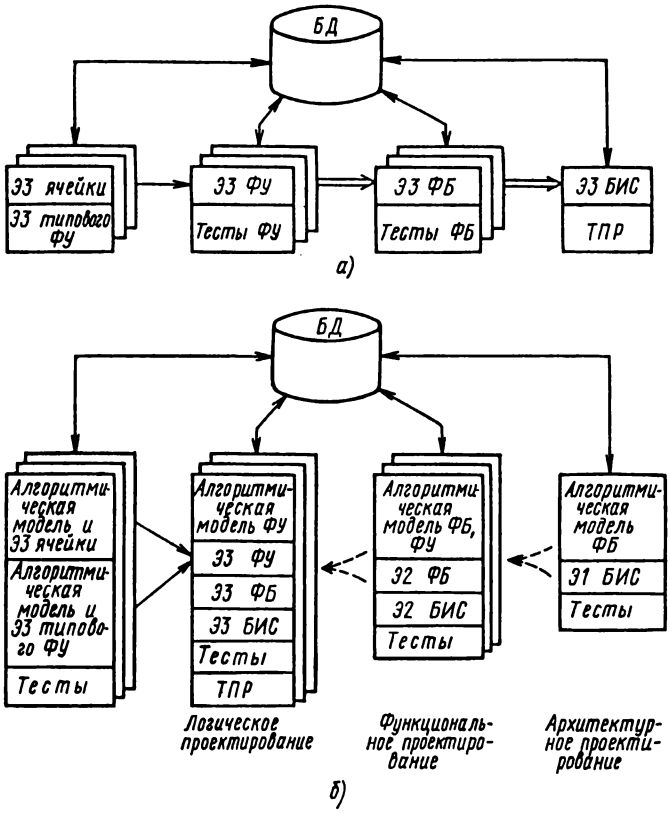


Рис. 4.1. Стратегии разработки логического проекта БИС:
а — восходящее; *б* — нисходящее проектирование

ские и структурные модели всех ячеек БМК, а в библиотеке типовых функциональных узлов банка данных могут храниться логические схемы узлов и блоков, которые могут применяться во вновь проектируемых БИС, и поэтому являются типовыми. На основе такой исчерпывающей исходной информации выполняется комплекс работ восходящего проектирования по непосредственной разработке логического проекта БИС, включающей следующие этапы.

Этап 1 — декомпозиция исходной логической схемы цифрового устройства на отдельные функциональные блоки и узлы. Из библиотеки типовых узлов и блоков банка данных САПР выбираются элементы, которые мо-

гут быть использованы в проектируемой БИС. Затем определяется совокупность функциональных блоков и узлов, подлежащих разработке.

Этап 2 — логическое проектирование на ячейках БМК функциональных узлов, входящих в установленную на первом этапе совокупность: 1) исходная логическая схема узла покрывается соответствующими ячейками БМК; 2) разрабатываются функциональные тесты на каждый узел; 3) на языке ОЦИС-РП составляются структурные модели каждого узла; 4) проводится логическое моделирование функционального узла на САПР ФЛП-3000 с целью отладки его логической схемы и тестов для проверки работоспособности, определения технических характеристик и принятия решения о возможности использования какого-либо разработанного узла в качестве типового; 5) в банке данных САПР накапливается информация о структуре отлаженных функциональных узлов и их функциональных тестов. Структурные схемы узлов хранятся в БД на уровне ячеек БМК или логических элементов.

Этап 3 — функциональное проектирование отдельных блоков: 1) логическая схема блока покрывается разработанными функциональными узлами и ячейками БМК; 2) разрабатываются функциональные тесты на каждый блок; 3) на языке ОЦИС-РП составляется структурная модель каждого блока; 4) проводится логическое моделирование блоков, при этом разработанные узлы могут быть раскрыты до уровня ячеек БМК или логических элементов, а типовые узлы представляются структурными моделями, раскрытыми до уровня ячеек БМК или логических элементов, либо алгоритмическими моделями. Допускается представлять структуру разными по точности и сложности моделями узлов; 5) в банке данных накапливается информация о разработанных функциональных блоках.

Этап 4 — разработка логического проекта БИС: 1) описывается логическая схема БИС; 2) разрабатываются функциональные тесты на БИС для проверки связей между блоками; 3) проводится логическое моделирование; 4) разрабатывается ТПР и оценивается качество тестов БИС; 5) в банке данных накапливается информация о логическом проекте БИС.

Стратегию функционально-логического проектирования снизу вверх целесообразно применять при разработке сравнительно несложных БИС (до трех тысяч элементов) при условии, что заранее известна логическая схема

(ЭЗ) БИС. При разработке более сложных БИС в условиях полной неопределенности (отсутствие логической схемы БИС) следует применять нисходящее проектирование, реализуемое в такой последовательности: 1) архитектурное проектирование цифрового устройства; 2) архитектурное проектирование БИС; 3) функциональное проектирование БИС; 4) логическое проектирование БИС (см. рис. 4.1, *б*).

Технология нисходящего проектирования. Рассмотрим порядок решения задач при разработке логического проекта БИС.

1. *Архитектурное проектирование цифрового устройства* — подготовительная стадия разработки проекта БИС, когда устанавливают состав и технические требования на БИС, входящие в устройство.

Этап 1 — разработка укрупненной схемы устройства. Выполняется декомпозиция устройства на отдельные БИС. Априорно устанавливаются исходные технические требования на каждую БИС на основании технических характеристик на устройство в целом.

Этап 2 — уточнение алгоритма функционирования и технических требований на каждую БИС. На основе исходного алгоритма функционирования отдельных БИС и языка ОЦИС-РП составляются алгоритмические модели и функциональные тесты для их проверки.

Проводится отладка каждой алгоритмической модели БИС на соответствующих функциональных тестах, затем отлаженные модели собираются в структурную схему устройства. Разрабатываются функциональные тесты для проверки правильной работы устройства. Выполняется совместное моделирование устройства на моделях БИС. По результатам моделирования уточняются связи между БИС и корректируются технические требования к каждой БИС. Результаты моделирования накапливают в БД САПР.

Разработка и изготовление различных БИС, входящих в устройство, могут происходить неравномерно — одни БИС могут быть спроектированы и изготовлены, другие могут находиться на стадии проектирования. Поэтому всегда можно вернуться на этап архитектурного проектирования и провести совместное моделирование устройства на моделях БИС различной точности.

2. *Архитектурное проектирование БИС.*

Этап 1 — разработка и отладка алгоритма функционирования БИС. В соответствии с установленными в

результате моделирования устройства техническими требованиями на БИС составляются алгоритмическая модель БИС и функциональные тесты для ее проверки, затем выполняется алгоритмическое моделирование БИС. Результаты моделирования хранятся в банке данных. Уточняются алгоритм ее функционирования и технические характеристики БИС.

Этап 2 — разработка и отладка укрупненной схемы БИС: 1) составляется укрупненная схема БИС (Э1), элементами которой являются функциональные блоки; 2) составляются алгоритмические модели каждого блока и функциональные тесты для их проверки; 3) проводится отладка алгоритмических моделей блоков, затем они собираются и образуют структурную модель БИС; 4) при необходимости функциональные тесты дорабатываются и проводится моделирование схемы Э1 БИС на регистровом уровне; 5) полученные результаты сравниваются с результатами алгоритмического моделирования БИС на предыдущем этапе. Если результаты совпадают, это означает, что алгоритмическая и структурная модели БИС полностью отлажены. В противном случае требуется доработка либо алгоритмической модели БИС, либо алгоритмических моделей функциональных блоков.

3. Функциональное проектирование БИС.

Этап 1 — разработка и отладка функциональных схем блоков: 1) составляются функциональные схемы блоков, элементами которых являются функциональные узлы, типовые функциональные узлы и отдельные ячейки; 2) составляются алгоритмические модели узлов и функциональные тесты, проводится их отладка; 3) на основе алгоритмических моделей составляется структурная модель схемы Э2 блоков и средствами моделирования выполняется верификация; 4) полученные результаты сравниваются с результатами алгоритмического моделирования блоков, проведенного на предыдущем этапе; 5) если результаты не совпадают, то требуется доработать либо структурную модель узла и повторить операции 1) — 4) данного этапа, либо функциональные тесты и вновь провести алгоритмическое моделирование блоков, а затем сравнить полученные данные с результатами моделирования схемы Э2 блоков. Если результаты совпадают, то это означает, что алгоритмическая и структурная модели каждого блока полностью отлажены и реализуют один и тот же алгоритм.

Этап 2 — разработка и отладка функциональной схемы

Э2 БИС: 1) на основе отлаженных схем Э2 функциональных блоков составляется схема Э2 БИС; 2) при необходимости функциональные тесты дорабатываются и осуществляется моделирование схемы Э2 БИС; 3) результаты моделирования сравниваются с результатами, полученными при моделировании схемы Э1 БИС. Если они не совпадают, то требуется доработать алгоритмические модели узлов и/или блоков.

4. Логическое проектирование БИС.

Этап 1 — разработка и отладка логических схем Э3 функциональных узлов: 1) на ячейках БМК разрабатываются тестопригодные логические схемы Э3 узлов; 2) составляются функциональные тесты для проверки работоспособности схем Э3 узлов; 3) проводится логическое моделирование и одновременно анализируется качество функциональных тестов. Если качество не достаточно, то выполняется доработка функциональных тестов; 4) автоматически сравниваются результаты моделирования алгоритмических моделей функциональных узлов и схемы Э3. Если они не совпадают, то необходимо доработать алгоритмические модели узлов. Их верификацию следует проводить на функциональных тестах, разработанных при функциональном проектировании БИС. Совпадение результатов моделирования узлов в статическом режиме по алгоритмической и структурной моделям свидетельствует о завершении отладки логических схем узлов и о возможности дальнейшего их использования при моделировании блоков и БИС.

Этап 2 — разработка и отладка логических схем Э3 блоков: 1) на основе отлаженных схем Э3 узлов составляется схема Э3 блока и функциональные тесты для ее проверки; 2) проводится логическое моделирование и оценивается тестируемость. Если она соответствует заданной, то процесс отладки логических схем Э3 блоков и их функциональных тестов прекращается. В противном случае составляются дополнительные тесты (при возможности этого) и на них продолжается проверка работоспособности схемы Э3 блоков с одновременным анализом тестируемости. Процесс доработки функциональных тестов и логического моделирования схемы Э3 блоков является итерационным. Критерием окончания этого процесса является соответствие фактической тестируемости каждого блока некоторой заданной (желаемой).

Этап 3 — разработка и отладка логической схемы Э3 БИС: 1) на основе отлаженных схем Э3 ФБ составля-

ется схема ЭЗ БИС и контролирующие тесты в виде таблицы логической проверки работоспособности; 2) проводится логическое моделирование для верификации схемы ЭЗ БИС и оценивается тестируемость. Если она соответствует заданной (желаемой), то процесс отладки схемы ЭЗ и ТПР прекращается. В противном случае в ТПР добавляются дополнительные тесты, на которых продолжается верификация схемы ЭЗ БИС и оценка тестируемости. Процесс доработки ТПР и логического моделирования схемы является итерационным.

Таким образом, в процессе разработки логического проекта БИС по методике восходящего и нисходящего проектирования создаются: 1) совокупность алгоритмических моделей узлов, блоков и БИС в целом (в случае нисходящего проектирования); 2) логическая схема БИС, раскрытая до уровня ячеек БМК и представленная в виде иерархического пофрагментного описания взаимно вложенных друг в друга многополюсников (ячеек БМК, узлов и блоков); 3) иерархия функциональных тестов для узлов, блоков и БИС. Использование при функционально-логическом проектировании иерархии функциональных тестов, алгоритмических и структурных моделей различных взаимовложенных фрагментов БИС существенно упрощает процесс отладки и позволяет оперативно проводить итерации по доработке на любом уровне представления. Следует также отметить, что на практике обе методики могут реализоваться одновременно, взаимно дополняя друг друга. Например, параллельно с отладкой алгоритмов функционирования БИС и ее блоков могут разрабатываться логические схемы узлов, расширяющие библиотеку типовых ФУ с учетом специфики конкретной БИС. Иногда целесообразно сразу перейти к отработке логической схемы некоторых блоков, минуя этап разработки алгоритма функционирования.

После разработки логического проекта БИС выполняется синтез ее топологии. Вопросы синтеза топологии БИС выходят за рамки настоящего пособия. Поэтому отметим только некоторые особенности их реализации в САПР ФЛП-3000. Исходной информацией для синтеза топологии служит логическая схема БИС, хранимая в БД системы. В зависимости от процента заполнения поля БМК могут применяться различные методы: полностью автоматический синтез (60...70 %); автоматический синтез с ручной доразводкой неразведенных трасс (не более 85 %); полностью ручная разводка топологий на

планшете, содержащем условное графическое изображение поля матрицы БМК (более 85 %).

Подсистема разработки топологии в САПР объединяет подсистемы синтеза и верификации (контроля) топологии БИС. Подсистема синтеза топологии осуществляет размещение, макро- и микротрассировку межсоединений ячеек на поле БМК. Подсистема верификации топологии выполняет: контроль конструкторско-топологических и технологических ограничений; контроль соответствия логической схемы восстановленной из топологии исходной логической схеме; расчет задержек распространения сигналов в межсоединениях ячеек на поле БМК.

Качественное решение задач контроля топологии особенно актуально при ручной разводке или доразводке топологии БИС, так как при этом неизбежны ошибки разработчика, которые устраняются за несколько итераций контроля и корректировки топологии. Рассчитанные задержки в межсоединениях ячеек необходимы для аттестации проекта БИС. Результатом работы подсистемы верификации топологии является логическая схема БИС, восстановленная из топологии и помещенная в БД в виде раздела типа ФРАГМЕНТ.

Завершается процесс разработки аттестацией проекта БИС и получением магнитной ленты для карты заказа на изготовление.

В заключение кратко рассмотрим технологические маршруты решения основных задач проектирования матричных БИС с указанием используемых разделов банка данных САПР ФЛП-3000 (рис. 4.2). Основные операции при технологии восходящего и нисходящего проектирования выполняются путем моделирования схем Э1—Э3 БИС на алгоритмических и структурных моделях. Разработка и отладка схем Э1—Э3 БИС проводятся в такой циклической последовательности блоков: 1—2—3—6—11—16—15—10 (см. рис. 4.2).

Программа раскрытия (блок 6) преобразует схему из исходного иерархического пофрагментного описания в иное описание, когда элементы являются моделями меньшей сложности.

Разработка и отладка контролирующих тестов в виде ТПР производятся следующим образом. Функциональные тесты, на которых отлажена схема Э3 БИС, дорабатываются до ТПР (блок 2). После этого из банка данных системы (блок 11) извлекается схема Э3 БИС и на доработанных до ТПР тестах (блок 2) проводится логиче-

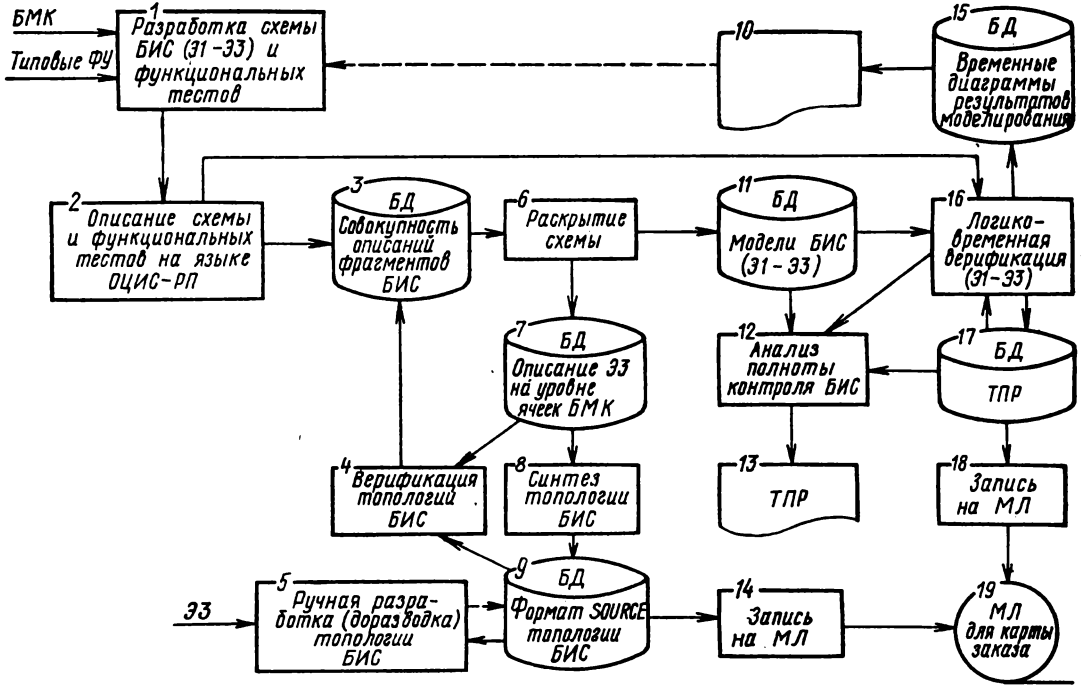


Рис. 4.2. Маршруты решения задач восходящего и нисходящего проектирования БИС с использованием банка данных САПР ФЛП-3000

ское моделирование (блок 16) с одновременным анализом качества тестов (блок 12). По окончании логического моделирования все элементарные проверки отлаженной ТПР запоминаются в банке данных (блок 17), а результаты оценки полноты ТПР выводятся на распечатку для принятия решения о завершении отладки ТПР или необходимости их доработки. В последнем случае вновь повторяется последовательность операций 2—16—17—12—13 (см. рис. 4.2).

Исходной информацией для синтеза топологии является логическая схема БИС, раскрытая до уровня ячеек БМК. Для реализации автоматического синтеза топологии БИС включается следующий маршрут: 3—6—7—8—9 (см. рис. 4.2). Результаты синтеза топологии в виде массивов размещения ячеек и трассировки соединений хранятся в БД системы (блок 9).

Для разработки топологии вручную или ручной доработки топологии БИС (блок 5) в САПР предусмотрено автоматизированное рабочее место, на котором выполняются «сколка», коррекция топологии и запись на магнитную ленту (МЛ) информации о топологии БИС в формате SOURCE. Далее осуществляются чтение данной магнитной ленты и запись топологической информации в БД системы (блок 9).

Контроль топологии БИС выполняется независимо от метода ее разработки (автоматический, вручную или в режиме автомат — ручная доработка). В результате верификации топологии (блок 4) проверяется соблюдение конструкторско-технологических ограничений, из топологии восстанавливается логическая схема и сравнивается с исходной схемы ЭЗ БИС, хранящейся в блоке 7. Таким образом, маршрут контроля топологии БИС в общем случае представляется двумя последовательностями: 3—6—7 и 9—4—3 (см. рис. 4.2). Первая последовательность устанавливает раскрытие схемы ЭЗ БИС до уровня ячеек БМК (в случае ручной разработки топологии), вторая — непосредственно контроль топологии БИС и запись восстановленной логической схемы БИС в БД (блок 3).

Для аттестации проекта БИС необходимо прежде восстановленную из топологии логическую схему БИС (блок 3) раскрыть до логических элементов (блок 6) и получить схему ЭЗ БИС (блок 11). Затем проводится логико-временное моделирование схемы ЭЗ на отработанных ранее тестах (блок 17) с учетом влияния дестабили-

зирующих факторов на параметры моделей ЛЭ. На основе анализа полученных результатов (блоки 16, 10) принимается заключение о работоспособности БИС. Таким образом, в маршрут аттестации проекта БИС включаются следующие блоки: 3, 6, 11, 16, 17, 15, 10 (см. рис. 4.2).

§ 4.2. Тестирование БИС

На основе проекта БИС, разработанного на этапах логического и топологического проектирования, создаются реальные образцы БИС. Каждый образец должен затем пройти функциональный контроль, устанавливающий правильность его работы. Вопросы функционального контроля БИС сложны и трудоемки. С повышением степени интеграции БИС решение задач функционального контроля значительно усложняется. Например, повышение степени интеграции на порядок приводит к увеличению времени ручной разработки тест-программ в три раза. Для обеспечения функционального контроля образцов БИС требуется уже на ранних этапах проектирования учитывать вопросы тестирования.

В общем случае при тестировании на математической моделе или реальном образце обнаруживаются неисправности БИС путем анализа состояний ее выходов на определенных наборах входных сигналов. Успешное решение задачи тестирования БИС на всех стадиях проектирования и изготовления определяет в конечном итоге ее важнейшие характеристики, такие, как бездефектность проектирования, надежность и устойчивость работы, стоимость образцов и др.

Различают два вида тестирования БИС: 1) функциональное тестирование, осуществляемое на всех этапах разработки логической схемы; 2) функциональный контроль правильности работы образцов БИС после их изготовления.

Цель функционального тестирования — проверить на этапе ФЛП правильность функционирования спроектированной схемы и соответствие требованиям технического задания на БИС. Особенность функционального тестирования заключается в том, что процесс разработки схемы во многом остается эвристическим и сильно зависит от опыта и интуиции разработчика. Отсюда — полная непредсказуемость возможных ошибок и их проявлений. Положение усугубляется еще и тем, что БИС,

как правило, является сложным цифровым автоматом последовательностного типа, имеющим огромное число внутренних состояний, перебрать которые в процессе тестирования практически невозможно.

Для успешного решения задачи функционального тестирования на этапе ФЛП необходимо создать иерархию функциональных тестов в соответствии с иерархией фрагментов, из которых состоит БИС. Каждый отдельно взятый функциональный узел (например, дешифратор, счетчик, регистр сдвига, сумматор) представляет собой относительно простое устройство, для которого возможно построить полный тест для проверки его функционирования. На основе узла разрабатываются блоки, логическая функция которых значительно сложнее. Однако при разработке тестов для блоков нет необходимости выполнять полное функциональное тестирование каждого узла, входящего в блок, так как оно уже было выполнено ранее при его разработке. Особое внимание при тестировании необходимо уделить проверке корректности связей функциональных узлов внутри блока и проверке собственных внутренних состояний блока. Аналогично при разработке функциональных тестов на БИС в целом особое внимание уделяется проверке корректности связей между блоками в БИС.

Цель функционального контроля — констатировать отсутствие или наличие неисправностей в конкретном образце БИС, нарушающих его функционирование, т. е. установить соответствие изготовленного образца проекту БИС. Неисправность БИС — это ее состояние, вызванное неработоспособностью (отказом) одного или нескольких ее элементов. Причинами отказов КМОП БИС могут быть: обрыв или замыкание проводников; пробой подзатворного диэлектрика; замыкание *p-n*-перехода; наличие паразитных связей. Однако подавляющее большинство неисправностей приводит к отказам логического типа, когда в той или иной цепи устанавливается и не изменяется состояние, тождественное лог. 0 или лог. 1, что нарушает нормальное функционирование БИС. Поэтому функциональный контроль обычно проводится на отсутствие в БИС неисправностей логического типа, называемых «одиночными тождественными неисправностями».

Функциональный контроль на стадии производства БИС осуществляется на специальном автоматизированном контрольно-измерительном оборудовании. Для про-

ведения контроля на этапе ФЛП разрабатываются специальные контролирующие тесты в виде ТПР, ориентированные на обнаружение одиночных тождественных неисправностей.

Метод контроля состоит в следующем. По разработанной на основе ТПР специальной программе контроля на входы исследуемой БИС подаются входные воздействия. В заданный момент времени определяются состояния выходов БИС, которые сравниваются с эталонными (ожидаемыми по ТПР) состояниями на данной элементарной проверке. По результатам сравнения делается вывод о работоспособности БИС на данном входном наборе. Испытание БИС проводится последовательно на всех элементарных проверках (тестах), входящих в таблицу ТПР. При обнаружении несоответствия фактического и ожидаемого состояний выходов БИС процесс контроля прекращается. При этом БИС помечается как неработоспособная. В противном случае БИС считается годной.

Несмотря на кажущуюся простоту организации процесса, функциональный контроль содержит одно очень серьезное противоречие — между качеством теста и временем тестирования БИС, что в конечном итоге определяет стоимость испытаний. БИС имеет огромное число внутренних состояний. При функциональном контроле требуется разработать совокупность контролирующих тестов, предназначенных для проверки всех состояний БИС. В этом случае время тестирования может достигать нескольких часов, дней и даже недель. Поэтому по мере увеличения сложности БИС ощущается необходимость в разработке и развитии приемлемых методов тестирования.

Возможными путями эффективного решения задачи функционального контроля БИС являются следующие: 1) разработка методов проектирования тестопригодных БИС, направленных на повышение уровня тестируемости БИС; 2) развитие методов встроенного самоконтроля БИС; 3) развитие методов автоматического синтеза и минимизации контролирующих тестов; 4) разработка методов анализа качества тестов, т. е. оценки полноты контроля БИС с помощью заданных тестов.

Методы проектирования тестопригодных БИС. БИС относятся к схемам, не допускающим зондирования и восстановления работоспособности. Поэтому оценку их работоспособности можно проводить только сравнением

совокупности логических нулей и единиц на внешних выводах, полученной в ответ на входной тест.

При анализе тестируемости БИС используют два важных понятия — наблюдаемость и управляемость узла схемы. Наблюдаемый узел связан с внешним выводом БИС, логическое состояние которого легко определить. Для управляемого узла обеспечивается возможность принудительной установки в заданное логическое состояние. Для повышения наблюдаемости и управляемости внутренних узлов схемы используют различные методы проектирования тестопригодных БИС.

Наибольшее распространение получил *метод сканирующего пути* и его варианты, когда обычные элементы памяти БИС могут быть либо реконфигурированы в сдвиговый регистр, либо замещены сдвиговым регистром (рис. 4.3). В результате упрощается процесс ввода/вывода данных в БИС. Каждый триггер схемы при этом может функционировать в двух режимах: 1) как обычный триггер; 2) как триггер, функционирующий в составе сдвигового регистра, образующего путь сканирования. Условное графическое обозначение триггера, предназначенного для организации пути сканирования, показано на рис. 4.3, *а*, где D — информационный вход, DS — вход пути сканирования, C — синхронизация нормального режима работы, CS — синхронизация режима сканирования, Q — информационный выход, QS — выход сканирования. Сдвиговый регистр (рис. 4.3, *б*) может быть проверен отдельно путем загрузки произвольной последовательности нулей и единиц через вход сканирования и последующего считывания этой последовательности через выход сканирования.

В общем случае возможны три режима тестирования: 1) тестирование самого сканирующего пути; 2) задание исходного состояния схемы на основе сканирующего пути для последующего ее тестирования через внешние выводы; 3) считывание состояния комбинационной схемы.

Разновидностью метода сканирующего пути является *метод LSSD*, разработанный фирмой «IBM» для проектирования кристаллов БИС и СБИС. В методе используются специфичные элементы памяти, изменение состояния которых производится уровнем сигнала синхронизации, а не фронтом. Схема триггера-защелки (рис. 4.4) состоит из двух D-триггеров: D1 и D2. Триггер D1 — это обычный элемент памяти с информационным входом

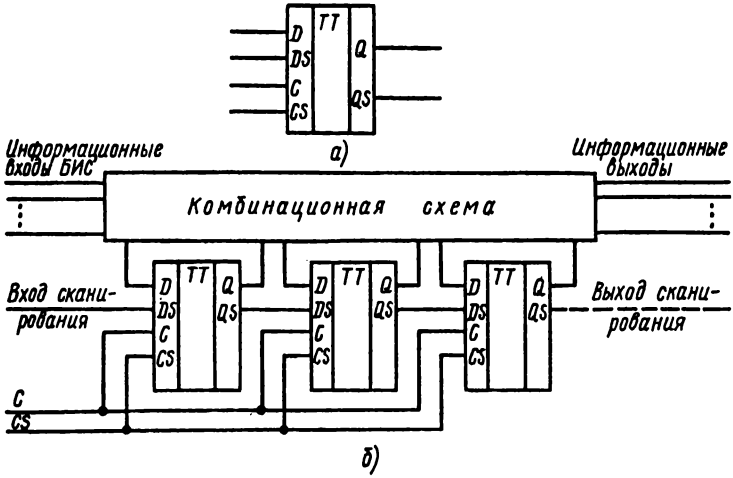


Рис. 4.3. Схема объединения триггеров в сдвиговый регистр для организации пути сканирования

D, синхровходом C и выходом Q. В нормальном режиме работы под управлением системного синхросигнала C сигналы управления сдвигом CS1 и CS2 равны нулю, триггер D2 и вход DS не используются. При отсутствии синхронизации, когда на входе CS1 устанавливается единица, информация из D1 переписывается в D2 и сохраняется в нем при CS2 = 0. Сигналы управления сдвигом CS1 и CS2 не перекрываются и подаются на все элементы памяти через специальные выводы БИС. Информация с выхода QS подается на вход DS следующего элемента памяти, образуя путь сканирования. Метод LSSD более пригоден для МОП-схем, поскольку триггер-защелка D2 может быть реализован всего на одном транзисторе.

При использовании описанных методов тестирования все входы триггеров доступны для управления, а все выходы — наблюдаемы. Составление тестов существенно упрощается, так как вместо решения трудной задачи генерации тестов для последовательностной схемы требуется решить более простую задачу генерации тестов для комбинационных схем. Тестирование при помощи средств сканирования позволяет оценить работоспособность БИС с вероятностью не более чем 98 %, а также снизить стоимость и повысить качество тестирования.

Однако методы имеют и недостатки. Наличие цепей сканирования приводит к ухудшению динамических характеристик БИС, удвоению числа триггеров необходимости выделения значительных площадей кристалла под разводку цепей управления сканированием (до 5 % для схем токовой логики и до 50 % для произвольных логических КМОП БИС), дополнительно требуется три или четыре внешних вывода.

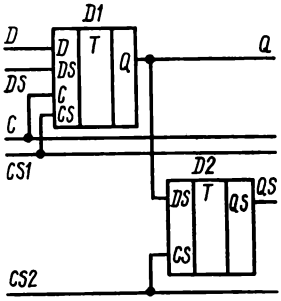


Рис. 4.4. Логическая схема памяти для реализации пути сканирования методом LSSD

На практике более целесообразно применять неполный сдвиговый регистр, объединяющий 15...30 % наиболее сложно тестируемых триггеров.

Суть задачи синтеза (генерации) тестов заключается в разработке ТПР, с помощью которой можно обнаружить в БИС все или большую часть неисправностей. В настоящее время получили распространение несколько методов синтеза тестов, наиболее известными из которых являются D-алгоритм Рота, вероятностный метод, метод эквивалентных нормальных форм.

Существующие системы синтеза тестов достаточно эффективны для комбинационных схем, однако практически все системы синтеза тестов плохо работают на произвольных последовательностных схемах. Это связано с тем, что характер проявления неисправности на выходах последовательностных схем зависит не только от входных сигналов (как в случае комбинационных схем), но и от внутреннего состояния схемы. Выход из создавшегося положения видится в переходе на методы проектирования тестопригодных БИС (метод LSSD, метод сканирующего пути и др.). Экспериментальные версии систем проектирования тестов, ориентированных на такие методы, уже существуют. При этом порядок разработки тестов включает следующие шаги:

1) отладка логической схемы БИС;

2) оценка тестируемости схемы путем поиска дерева управляемости (наблюдаемости);

3) прогнозирование качества тестов, стоимости тестирования и стоимости пропуска ошибки;

4) при неудовлетворительной оценке тестируемости схемы выделение триггерных схем — претендентов на

включение в сдвиговый регистр; доработка логической схемы БИС; переход к шагу 1;

5) при удовлетворительной оценке тестируемости выполнение автоматической генерации тестов на основе результатов анализа тестируемости схемы.

Для большинства БИС нереально пытаться получить 100 %-ную гарантию правильности функционирования, поэтому целесообразным становится построение приемлемых тестов. Действительно, если в результате сокращения теста время тестирования уменьшится в 100 раз и при этом будет покрыто (обнаружено) 95 % неисправностей, то это вполне приемлемо для широкого класса БИС, так как стоимость контроля оставшихся 5 % неисправностей может значительно превышать стоимость обнаружения и замены неисправной схемы в процессе эксплуатации. Отсюда актуальность задачи оценки качества тестов, предназначенных для контроля БИС.

Наибольшее распространение для оценки качества тестов получили *методы моделирования неисправностей на ЭВМ*, сущность которых заключается в сравнении результатов моделирования исправной схемы и множества неисправных схем, полученных путем поочередного введения в исправную схему заданных неисправностей. Неисправность считается покрытой, если в тестовой последовательности существует хотя бы один тест, который выявляет несоответствие логических состояний выходов исправной и неисправной схем. Наряду с процентом обнаруживаемых неисправностей в результате моделирования может быть построен список непроверенных неисправностей, что является очень важной информацией для повышения качества тестов. Однако данный подход к оценке качества тестов не может быть эффективно применен для БИС, так как время моделирования неисправностей имеет кубическую зависимость от числа элементов в схеме. В настоящее время активно разрабатываются методы, учитывающие вероятностные факторы.

Для оценки качества тестов в САПР ФЛП-3000 разработан и реализован *метод инверсии чувствительности*. В процессе логического моделирования схемы составляется полный список неисправностей. В методе использован принцип обратного прохода (от выходов схемы к ее входам). В процессе обратного прохода выполняется анализ чувствительности входов элементов, по результатам которого строятся пути транспортировки

неисправностей и пути транспортировки инверсии (исчезновения) чувствительности входов.

В данном методе введены следующие понятия и определения. Чувствительный вход — это вход элемента, изменение логического состояния которого на противоположное вызывает изменение логического состояния его выхода. Нечувствительный вход — вход элемента, изменение логического состояния которого на противоположное не вызывает изменения состояния его выхода.

Путь транспортировки неисправностей — последовательность элементов, соединенных между собой таким образом, что выход предыдущего элемента подключен к чувствительному входу последующего элемента пути. Выход последнего элемента пути является внешним выводом БИС. Таким образом, если имеет место неисправность на каком-либо чувствительном входе элемента пути, то эта неисправность вызовет инверсию состояния выходов всех последующих элементов пути, т. е. в конечном итоге вызовет изменение логического состояния соответствующего выхода БИС.

Инверсно-чувствительный вход образуется в случае, когда нечувствительный вход элемента становится чувствительным при появлении неисправности на его чувствительном входе или при транспортировке неисправности через данный элемент. Понятие пути транспортировки инверсии чувствительности аналогично понятию пути транспортировки неисправности, но данный путь оканчивается на инверсно-чувствительном входе элемента, принадлежащего пути транспортировки неисправности.

Метод инверсии чувствительности основан на том факте, что вход элемента не может одновременно принадлежать пути транспортировки неисправности и пути транспортировки инверсии чувствительности, так как при появлении неисправности на данном входе реализуется путь инверсии чувствительности, что в конечном итоге вызывает блокировку пути транспортировки данной неисправности. В свою очередь, появление неисправности на любом чувствительном входе (инверсия его состояния) вызовет изменение логического состояния всех последующих элементов пути транспортировки неисправности и изменение логического состояния соответствующего внешнего вывода БИС. Таким образом, данная неисправность является обнаруживаемой.

Приведем алгоритм формирования списков обнару-

живаемых неисправностей по методу инверсии чувствительности:

```
ЦИКЛ ПО ВСЕМ ЭЛЕМЕНТАРНЫМ ПРОВЕРКАМ ТАБЛИЦЫ ТПР;
  ОПРЕДЕЛЯЕМ СОСТОЯНИЕ УЗЛОВ СХЕМЫ БИС НА ОЧЕРЕДНОЙ
  ЭЛЕМЕНТАРНОЙ ПРОВЕРКЕ;
  ЦИКЛ ПО ВСЕМ ВЫХОДНЫМ КОНТАКТАМ БИС;
    ЗАНОСИМ НОМЕР ЦЕПИ В СПИСОК ЧУВСТВИТЕЛЬНЫХ ВХОДОВ, СООТВЕТСТВУЮЩИЙ
    ОЧЕРЕДНОМУ ВЫХОДНОМУ КОНТАКТУ БИС;
    ЦИКЛ ПОКА ЕСТЬ НОМЕРА ЦЕПЕЙ В СПИСКЕ ЧУВСТВИТЕЛЬНЫХ ВХОДОВ;
      ДЛЯ ОЧЕРЕДНОГО НОМЕРА ЦЕПИ ЧУВСТВИТЕЛЬНОГО ВХОДА ОПРЕДЕЛЯЕМ
      НОМЕР ЛЭ, ЯВЛЯЮЩЕГОСЯ ИСТОЧНИКОМ СИГНАЛА В ДАННОЙ ЦЕПИ;
      ПО НОМЕРУ ЭЛЕМЕНТА ВЫЗЫВАЕМ ЕГО МОДЕЛЬ ДЛЯ АНАЛИЗА
      ЧУВСТВИТЕЛЬНОСТИ ЕГО ВХОДОВ;
      ЦИКЛ ПО ВСЕМ ВХОДАМ ЭЛЕМЕНТА;
        ЕСЛИ ЧУВСТВИТЕЛЬНЫЙ ВХОД ОБНАРУЖЕН ТО
          ЕСЛИ ВХОД НЕ ПРИНАДЛЕЖИТ ПУТИ ТРАНСПОРТИРОВКИ
                  ИНВЕРСИИ ЧУВСТВИТЕЛЬНОСТИ ТО
            ЛЭ ЗАНОСИТСЯ В СПИСОК ЭЛЕМЕНТОВ ПУТИ ТРАНСПОРТИРОВКИ
            НЕИСПРАВНОСТЕЙ;
            ВХОД ЗАНОСИТСЯ В СПИСОК ЧУВСТВИТЕЛЬНЫХ ВХОДОВ;
            ВЫПОЛНЯЕМ АНАЛИЗ МОДЕЛИ ЭЛЕМЕНТА НА НАЛИЧИЕ ИНВЕРСНО-
            ЧУВСТВИТЕЛЬНЫХ ВХОДОВ;
            ДЛЯ КАЖДОГО ОБНАРУЖЕННОГО ИНВЕРСНО-ЧУВСТВИТЕЛЬНОГО ВХОДА
            СТРОИМ ПУТЬ ТРАНСПОРТИРОВКИ ИНВЕРСИИ ЧУВСТВИТЕЛЬНОСТИ;
          ВСЕ ЕСЛИ;
        ВСЕ ЕСЛИ;
      ВСЕ ЦИКЛ;
    ВСЕ ЦИКЛ;
    СОСТАВЛЯЕМ СПИСОК ОБНАРУЖИВАЕМЫХ НЕИСПРАВНОСТЕЙ;
  ВСЕ ЦИКЛ;
ВСЕ ЦИКЛ;
```

Расчет чувствительности каждого входа элемента схемы к неисправностям типа тождественный нуль ($\equiv 0$) и тождественная единица ($\equiv 1$) осуществляется по соответствующим логическим уравнениям. Если элемент реализует некоторую логическую функцию $Y = f(x_1, \ldots, x_i, \ldots \ldots, x_n)$ от n независимых переменных, данную функцию относительно x_i можно привести к виду

$$Y = x_i f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \vee f''(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n). \quad (4.1)$$

Тогда можно определить чувствительность входа x_i к неисправностям согласно выражению

$$\bar{Y}\,[f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \vee \times] \vee [f''(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \& \times] \to x_i, \qquad (4.2)$$

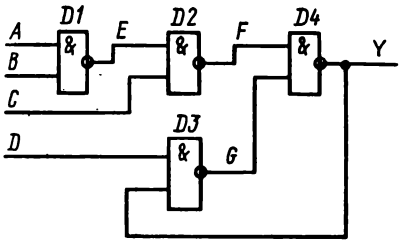


Рис. 4.5. Логическая схема

где $\times$ — переменная, имеющая логически неопределенное состояние. Подставив в выражение (4.2) конкретные значения переменных $Y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ на основе правил выполнения операций И, ИЛИ, НЕ в троичном алфавите, получим результат в символах 0, 1, $\times$.

Нуль (0) означает, что вход x_i является чувствительным к обнаружению неисправности $\equiv 0$; 1 — что вход x_i является чувствительным к обнаружению неисправности $\equiv 1$; $\times$ — что вход x_i является нечувствительным, т. е. неисправности на этом входе не обнаруживаются.

Пример 4.1. Записать уравнения определения чувствительности входов для логических элементов, реализующих следующие функции: а) $Y = x$, б) $Y = \bar{x}$, в) $Y = x_1 \& x_2$, г) $Y_1 = \overline{x_1 \vee x_2}$, д) $Y = \overline{(x_1 \& x_2) \vee (x_2 \& x_3)}$.

Решение. Используя выражения (4.1) и (4.2), получим:

а) $\bar{Y} \to x$;

б) $Y \to x$;

в) $\bar{Y} \& (x_2 \vee \times) \to x_1$, $Y \& (x_1 \vee \times) \to x_2$;

г) $Y \vee (x_2 \& \times) \to x_1$, $Y \vee (x_1 \& \times) \to x_2$;

д) $Y \& (x_2 \vee \times) \vee (x_2 \& x_3 \& \times) \to x_1$, $Y \& (x_2 \vee \times) \vee (x_1 \& x_2 \& \times) \to x_3$, $Y \& (x_1 \vee x_3 \vee \times) \to x_2$.

Пример 4.2. Для функционального контроля логической схемы, представленной на рис. 4.5, предложены семь входных наборов, указанных в левой части табл. 4.1. Необходимо определить список обнаруживаемых неисправностей и коэффициент тестируемости данной схемы.

Решение. Согласно (4.1) и (4.2) составляем уравнения определения чувствительности входов логических элементов схемы:

для элемента D1 $E(B \vee \times) \to A$, $E(A \vee \times) \to B$,

для элемента D2 $F(C \vee \times) \to E$, $F(E \vee \times) \to C$,

для элемента D3 $G(Y \vee \times) \to D$, $G(D \vee \times) \to Y$,

для элемента D4 $Y(G \vee \times) \to F$, $Y(F \vee \times) \to G$.

Используя принцип обратного прохода, определяем для каждого входного набора пути транспортировки неисправностей и возможности их блокировки (инверсии чувствительности). На нулевом входном наборе (0000), т. е. при $A = B = C = D = 0$, получим $E = F = G = 1$, $Y = 0$. Отмечаем выход схемы Y как контролируемый и в табл. 4.1 помечаем

Таблица 4.1
Таблица истинности логической схемы (рис. 4.5)

Входы	Выходы ЛЭ
A B C D	E F G Y
0 0 0 0	1 1 1 0
0 0 0 1	1 1 1 0
0 0 1 0	1 0 1 1
0 0 1 1	1 0 0 1
0 1 0 0	1 1 1 0
0 1 0 1	1 1 1 0
0 1 1 0	1 0 1 1

его прямоугольником. Определяем чувствительность входов F и G элемента D4:

$$Y(G \vee \times) = 0(1 \vee \times) = 0 \to F,\ Y(F \vee \times) = 0(1 \vee \times) = 0 \to G.$$

Нули на входах F и G элемента D4 означают, что входы чувствительные, и это свидетельствует о возможности обнаружения неисправности типа тождественный нуль. Таким образом, входы F и G должны быть включены в путь транспортировки неисправности и помечены в таблице прямоугольником.

Далее определяем чувствительность входов E и C для элемента D2:

$$F(C \vee \times) = 1(0 \vee \times) = 1 \cdot \times = \times \to E,\ F(E \vee \times) = 1(1 \vee \times) = 1 \to C. \quad (4.3)$$

Значение $\times$ на входе E характеризует его как нечувствительный, а единица на входе C — как чувствительный к обнаружению неисправности типа тождественная единица. Далее для элемента D2 необходимо проверить вход E на инверсию чувствительности. Для этого в формуле (4.3) заменяем C на $\bar{C}$: $F(\bar{C} \vee \times) = 1(1 \vee \times) = 1 \to E$. Полученное значение $E = 1$ свидетельствует о возможности блокировки пути транспортировки неисправности, т. е. инверсии чувствительности. Принадлежность выхода E пути инверсии чувствительности помечена в таблице истинности кружком.

Продолжим построение пути инверсии чувствительности. Для этого определим чувствительность входов A и B элемента D1:

$$E(B \vee \times) = 1(0 \vee \times) = \times \to A,\ E(A \vee \times) = 1(0 \vee \times) = \times \to B.$$

Так как на входах A и B получили значения $\times$, то данные входы являются нечувствительными (построение пути инверсии чувствительности завершается).

Продолжаем построение пути транспортировки неисправности типа $\equiv 0$ по входу G элемента D4. Для этого определяем чувствительность входов D и Y элемента D3: $G(Y \vee \times) = 1(0 \vee \times) = 1 \cdot \times = \times \to D$, $G(D \vee \times) = 1(0 \vee \times) = 1 \cdot \times = \times \to Y$. Полученные значения X на входах D и Y означают, что эти входы являются нечувствительными к неисправностям.

Выполняя аналогичную процедуру для остальных входных наборов (см. табл. 4.1), получаем список всех обнаруживаемых неисправностей. Из табл. 4.1 можно сделать вывод, что цепи B и D в схеме не тестируются (т. е. не обнаруживаются: $\equiv 0$, $\equiv 1$) на данных наборах, цепь A тестируется только на неисправность типа $\equiv 0$. Таким образом, в логической схеме рис. 4.5 на заданных тестах могут быть обнаружены 9 неисправностей из 16 возможных, что соответствует коэффициенту тестируемости 56 %. Для повышения коэффициента тестируемости заданные в табл. 4.1 тесты необходимо дорабатывать.

§ 4.3. Аттестация проекта БИС

Для проектирования БИС с помощью САПР необходима иерархия математических моделей БИС и среды ее функционирования. Однако точность используемых математических моделей ограничена возможностями ЭВМ и нашими знаниями о процессах, происходящих в БИС. Значения многих параметров элементов схемы имеют большой разброс из-за нестабильности технологии изготовления, кроме того, работоспособность БИС в значительной степени определяется воздействием внешних факторов. На практике недостаточно доказать, что разработанная логическая схема БИС выполняет указанную в техническом задании функцию, а ее топология соответствует логической схеме. Необходимо, чтобы проект БИС имел запас прочности, т. е. запас устойчивости к разбросу задержек распространения сигналов в схеме из-за различной длины и удельного сопротивления топологических цепей, к разбросу технологических параметров и к воздействию факторов внешней среды.

Таким образом, необходимо провести аттестацию проекта БИС путем моделирования на ЭВМ основных видов испытаний. При аттестации проверяется готовность проекта к реализации его в производстве. Успешное решение данной задачи позволяет исключить возвраты на доработку проекта БИС после изготовления и испытаний образцов и тем самым существенно сократить сроки и затраты на разработку БИС.

Традиционно подобные задачи решаются средствами схемотехнического анализа, однако сложность аттестации БИС заключается в том, что даже при использовании макромоделей логических элементов время анализа схемы на ЭВМ, например с учетом задержек распространения сигналов в топологических цепях, может достигать многих часов и даже десятков часов. Поэтому в связи с необходимостью решения таких задач активно

развиваются методы логического моделирования повышенной точности отображения процессов в схеме.

Средства системы ФЛП-3000 позволяют выполнять логическое моделирование схем с учетом задержек в топологических цепях БИС, оценить устойчивость проекта БИС к разбросу технологических параметров (удельного сопротивления слоев разводки топологии, порогового напряжения срабатывания транзисторов), напряжения источников питания, влиянию температуры и других внешних факторов.

Оценка влияния задержек распространения сигналов в топологических цепях на работоспособность логической схемы проводится следующим образом: 1) каждая цепь представляется в виде RC-дерева с одним (или несколькими) источником и множеством приемников; 2) рассчитываются задержки распространения сигнала от каждого источника до всех приемников в цепи методом оценки верхней и нижней границ задержки и их последующего усреднения; 3) полученные значения задержки учитываются в процессе логического моделирования при расчете длительностей фронтов переключения сигналов ($\Delta T_ф$) в схеме согласно формуле (3.3). Затем значения $\Delta T_ф$ используют при расчете времени активизации логических элементов $t_{акт\ лэ}$ по формуле (3.2).

Учет влияния параметров топологии на работоспособность БИС особенно необходим для схем, в которых для разводки межсоединений элементов используют слои поликремния и алюминия. Так как удельное сопротивление поликремния значительно превышает удельное сопротивление алюминия, задержки распространения сигнала в некоторых межсоединениях, где используется поликремний, могут достигать десятков наносекунд и значительно превышать собственные задержки ЛЭ. Это приводит к искажениям временной диаграммы и может нарушить работоспособность БИС.

Аналогично организованы и различные виды «испытаний проекта БИС». Учет влияния на работу БИС того или иного дестабилизирующего внешнего фактора осуществляется путем изменения значений соответствующих параметров [см. формулы (3.2) и (3.3)]. Например, при температурных испытаниях следует учитывать изменения порогов срабатывания ($\alpha_{пор}$) логических элементов, сопротивлений источников сигнала ($R_{вых}$) на выходах логических элементов, удельных сопротивлений ($\rho_{\square}$) слоев разводки топологии.

Отметим, что при обнаружении в процессе аттестации отклонений от нормальной работы БИС может потребоваться существенная доработка проекта, поэтому целесообразно проводить имитацию испытаний не только по завершении разработки проекта, но и на более ранних стадиях проектирования. Например, при разработке логической схемы БИС целесообразно проверить ее устойчивость к изменениям температуры, напряжения питания и влиянию других факторов. После размещения ячеек схемы на поле БМК и предварительной оценки задержек в межэлементных связях следует провести аттестацию, проверив тем самым работоспособность логической схемы при данном варианте размещения ячеек. После трассировки соединений выполняется окончательная аттестация проекта БИС.

Глава 5

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ КОМБИНАЦИОННЫХ ФУНКЦИОНАЛЬНЫХ УЗЛОВ БИС

При проектировании комбинационных функциональных узлов (КФУ) могут применяться эвристические и формальные методы. Наибольшее распространение получили эвристические методы, основанные на изобретательской деятельности разработчика. Однако не следует отрицать положительной роли формальных методов проектирования, позволяющих разработать логическую схему КФУ на основе заданного алгоритма ее функционирования. Вместе с тем нельзя и преувеличивать их возможности из-за того, что качество получаемых формальных решений логических схем уступает качеству аналогичных эвристических решений.

С развитием автоматизированных систем логического моделирования задача синтеза КФУ достаточно эффективно решается в интерактивном режиме. Разработчик при этом предлагает исходный вариант логической схемы КФУ и с помощью САПР ФЛП выявляет и устраняет все его недостатки. Поэтому основная задача состоит в разработке исходного варианта схемы КФУ. Рассмотрим методику формализованного синтеза КФУ и примеры синтезированных логических схем.

§ 5.1. Методика проектирования и особенности работы комбинационных функциональных узлов

Исходными данными для проектирования являются описание алгоритма функционирования КФУ, требования к основным электрическим параметрам, библиотека элементов и конструктивно-топологические особенности построения логических схем. На пути от исходного описания алгоритма функционирования до логической схемы КФУ можно выделить несколько основных этапов:

1. Словесное описание алгоритма функционирования КФУ.
2. Оценка размерности задачи и решение вопроса о разделении КФУ на части, если это необходимо.
3. Переход к формализованному заданию алгоритма функционирования КФУ.
4. Минимизация логических функций.
5. Преобразование минимальных логических функций

для рациональной реализации логической схемы в заданном базисе.

6. Построение структуры логической схемы.

7. Разработка функциональных тестов для проверки работоспособности спроектированной логической схемы.

8. Описание и совместная отладка логической схемы и функциональных тестов средствами САПР ФЛП.

Первые три этапа могут отсутствовать, если алгоритм функционирования КФУ задан в табличной или аналитической форме, т. е. в виде, пригодном для минимизации. На втором этапе решается вопрос о проектировании КФУ в целом или разбивке его на отдельные части (модули). Например, мультиплексор «16—1» может быть спроектирован и как КФУ, реализующий заданный алгоритм функционирования, и на основе модуля (мультиплексора «4—1»). Процесс проектирования во втором случае существенно упрощается, однако при этом не достигается максимальное быстродействие КФУ, так как критический путь прохождения сигналов здесь обычно больше.

На четвертом этапе выполняются процедуры минимизации логических функций методом Квайна и Мак-Класки или с помощью карт Карно, описанных в гл. 2. Основные традиционные критерии минимизации — минимальное число букв в реализуемой функции или логических операций в данной функции хороши для ИС малой степени интеграции, в которых логические элементы являются достаточно дорогими. Для комбинационных узлов БИС критерием сложности является не только число элементов на кристалле, необходимых для реализации функций узла. Большое значение приобретают морфологические свойства реализуемых схем (регулярность структуры, повторяемость элементов и связей, занимаемая площадь, минимальная длина межсоединений и т. п.). В ряде случаев полезность минимизации логических функций в традиционном смысле значительно снижается, поскольку минимальность функций может противоречить морфологическим критериям. Примером может служить использование ПЗУ для воспроизведения логических функций, задаваемых в СДНФ. Если функции минимизировать и задавать в МДНФ, то потеряется однородность структуры, повторяемость ее элементов, что усложняет производство БИС. Следует заметить, что методы проектирования схем, наилучших с точки зрения

морфологических критериев, только начинают развиваться.

На пятом этапе производится выбор состава логических элементов и преобразование функций, подлежащих схемной реализации, в наиболее целесообразном логическом базисе. Все проектируемые логические схемы реализованы на ячейках библиотеки БМК (см. приложение). При проектировании функциональных узлов следует учитывать нагрузочную способность ячеек БМК: для инвертора V1 она равна пяти, для усилителя V2 — 10, усилителя W3 — 15, двух- и трехвходовых логических элементов — трем, триггеров — трем. Библиотека БМК построена на двух типах топологических ячеек: V и W. Размеры ячейки W в два раза больше размеров ячейки V. В столбцах БМК топологические ячейки V и W попеременно следуют друг за другом. Цифровые ячейки библиотеки БМК имеют обозначения: VN, WN, V1WN, V2VN, V4VN, V6VN, которые определяют их топологические размеры. Например, обозначение VN/WN указывает на то, что ячейка БМК содержит одну топологическую ячейку V/W, буква N определяет порядковый номер ячейки данного типа в библиотеке БМК. Обозначение VMVN означает, что данная ячейка БМК начинается с топологической ячейки V и оканчивается топологической ячейкой такого же типа V, которые разделяются ячейками типа V, W и M поликремниевыми шинами. При топологической реализации логической схемы необходимо, чтобы число ячеек типов V и W было примерно одинаковое. В этом случае вероятность получения минимальных топологических размеров КФУ повышается.

На седьмом этапе проводится разработка функциональных тестов для проверки работоспособности спроектированной логической схемы. Эта процедура обычно реализуется разработчиком вручную на основе знания алгоритма функционирования спроектированной схемы. Для КФУ функциональные тесты могут соответствовать входным рабочим наборам таблицы истинности.

Процедуры кодирования и методика отладки логической схемы и функциональных тестов на САПР ФЛП-3000 подробно изложены в гл. 3 и 4.

Влияние задержек в элементах и межсоединениях на работу КФУ. Задержки в элементах логической схемы и межсоединениях ограничивают быстродействие схемы и могут явиться причиной появления на выходе

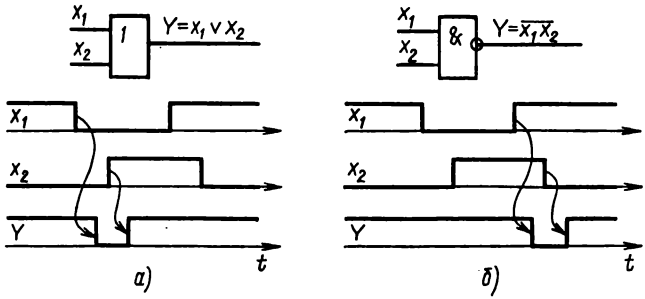


Рис. 5.1. Статический 1-риск сбоя:
а — на элементе ИЛИ; *б* — на элементе И-НЕ

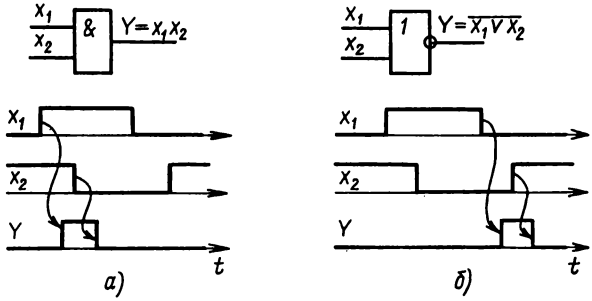


Рис. 5.2. Статический 0-риск сбоя:
а — на элементе И; *б* — на элементе ИЛИ-НЕ

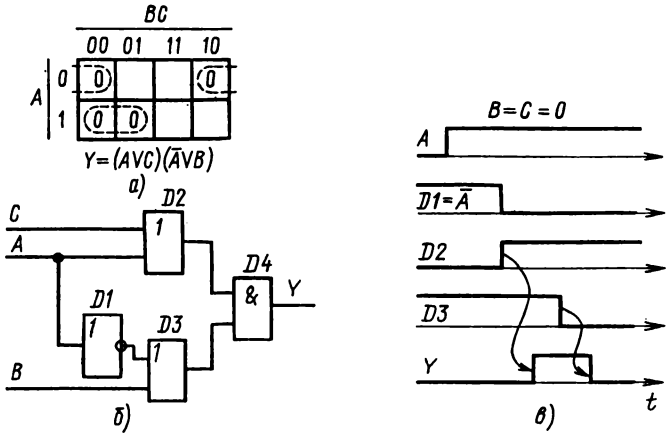


Рис. 5.3. Обнаружение статического 0-риска сбоя

КФУ кратковременных ложных сигналов, называемых рисками сбоя. Различают статические и динамические риски сбоя.

Статический риск сбоя возникает, когда состояние выхода по логике работы схемы должно оставаться неизменным, но происходит его кратковременное изменение (рис. 5.1, 5.2). Кратковременное изменение состояния выхода типа 1—0—1 называется статическим 1-риском сбоя, а кратковременное изменение состояния выхода типа 0—1—0 — статическим 0-риском сбоя. Возможность появления в проектируемой схеме ложных переходов типа 1—0—1 или 0—1—0 может быть обнаружена на этапе минимизации логической функции.

Пример 5.1. Функция $Y = f(A, B, C)$ задана на карте Карно (рис. 5.3, *а*). Спроектировать схему, реализующую функцию Y.

Решение. Выполнив склеивание указанных на карте ячеек, получим функцию Y в форме МКНФ

$$Y = (A \vee C)(\bar{A} \vee B). \tag{5.1}$$

Из анализа выражения (5.1) видим, что если $B = C = 0$, то $Y = A\bar{A}$. Выражение $Y = A\bar{A}$ является условием появления в схеме 0-риска сбоя, что подтверждается временнóй диаграммой работы (рис. 5.3, *в*) синтезированной логической схемы (рис. 5.3, *б*), реализующей логическую функцию (5.1).

Чтобы спроектировать логическую схему, гарантированную от появления в ней статического 0-риска сбоя, необходимо при минимизации исходной функции склеивать все соседние группы ячеек, как это показано на рис. 5.4, *а*. В результате появляется дополнительный сомножитель $(B \vee C)$ в ранее полученной МДНФ вида (5.1), что не приводит к изменению значений функции, но гарантирует от появления 0-риска сбоя. Действительно, при $B = C = 0$ получим

$$Y = (A \vee C)(\bar{A} \vee B)(B \vee C) = A\bar{A} \cdot 0 = 0.$$

Таким образом, приведенные на рис. 5.4, *б, в* синтезированная логическая схема и временнáя диаграмма ее работы иллюстрируют отсутствие 0-риска сбоя.

Пример 5.2. Функция $Y = f(A, B, C)$ задана на карте Карно (рис. 5.5, *а*). Спроектировать схему, реализующую функцию Y.

Решение. Выполнив минимизацию исходной функции, получим МДНФ вида

$$Y = AB \vee \bar{A}C. \tag{5.2}$$

Подставив в (5.2) значения $B = C = 1$, получим $Y = A \vee \bar{A}$. Это выражение является условием появления статического 1-риска сбоя. Рис. 5.5, *б, в* иллюстрируют обнаружение в синтезированной схеме статического 1-риска сбоя. Для его устранения необходимо провести минимизацию исходной функции, как показано на рис. 5.6, *а*. В результате получим

$$Y = AB \vee \bar{A}C \vee BC. \tag{5.3}$$

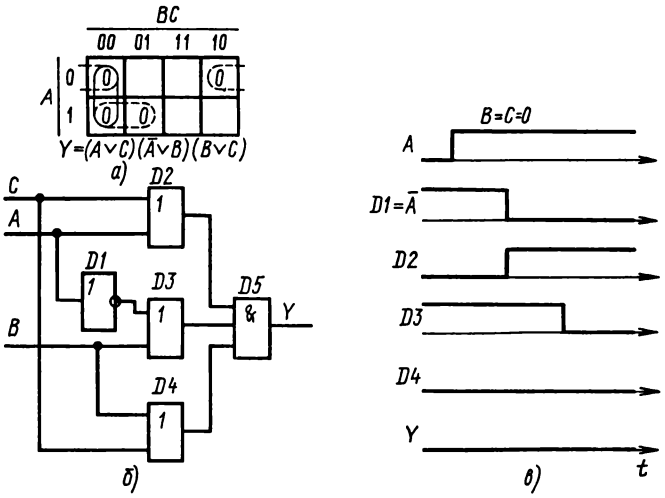


Рис. 5.4. Устранение статического 0-риска сбоя

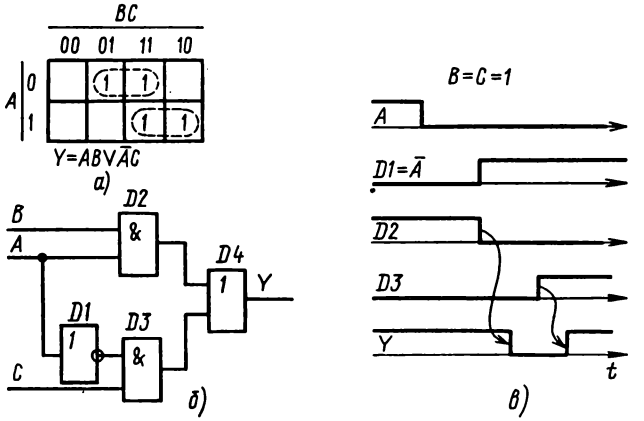


Рис. 5.5. Обнаружение статического 1-риска сбоя

Логическая схема, реализующая уравнение (5.3), и временна́я диаграмма ее работы представлены на рис. 5.6, *б, в*. Анализ работы вновь синтезированной схемы и выражения (5.3) подтверждает устранение 1-риска сбоя. Действительно, при $B = C = 1$ получаем

$$Y = AB \vee \bar{A}C \vee BC = A \vee \bar{A} \vee 1 = 1,$$

что свидетельствует об отсутствии 1-риска сбоя.

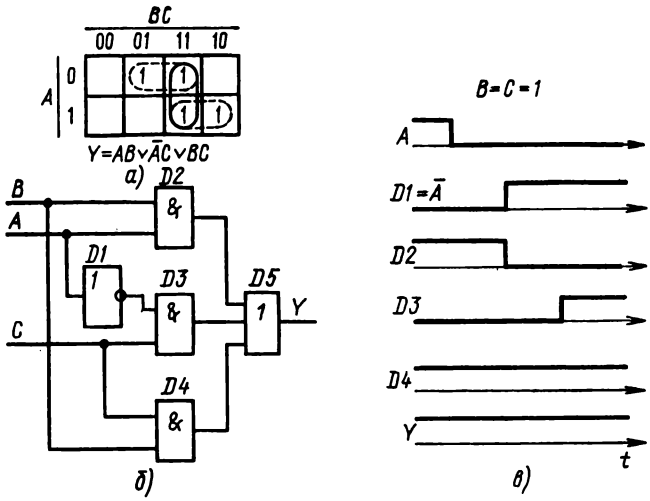
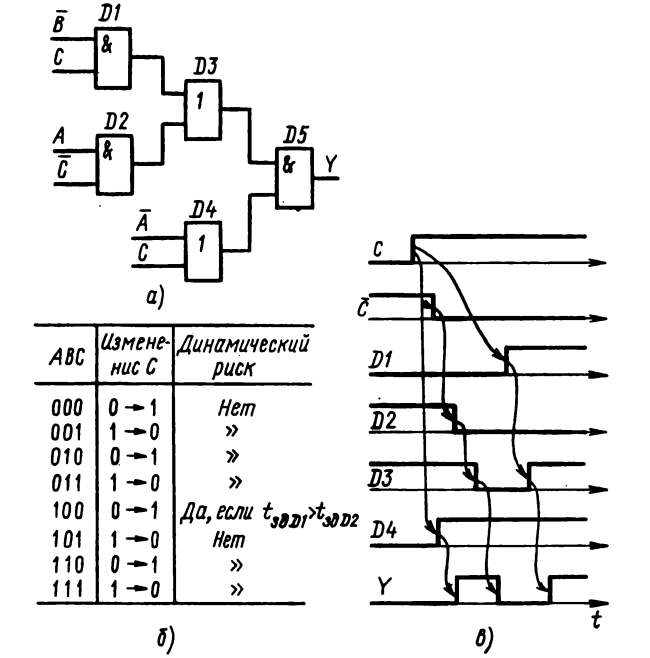


Рис. 5.6. Устранение статического 1-риска сбоя



ABC	Изменение C	Динамический риск
000	0 → 1	Нет
001	1 → 0	»
010	0 → 1	»
011	1 → 0	»
100	0 → 1	Да, если $t_{зD1} > t_{зD2}$
101	1 → 0	Нет
110	0 → 1	»
111	1 → 0	»

б)

Рис. 5.7. Выявление динамического риска в схеме

Динамический риск сбоя возникает в тех случаях, когда состояние выхода по логике работы схемы должно измениться на противоположное, однако вместо однократного перехода с одного уровня на другой происходят многократные переходы (0—1—0—1 вместо 0—1; 1—0—1—0 вместо 1—0). Такие ситуации могут появиться в схемах, где один какой-либо сигнал проходит по трем и более путям и каждый путь имеет различные задержки распространения.

Пример 5.3. Проверить, есть ли в схеме, изображенной на рис. 5.7,*а*, динамический риск.

Решение. Так как сигнал *C* может проходить по трем различным путям, то в схеме возможен динамический риск. Проведем анализ работы схемы при различных входных воздействиях. Если $A = 0$ (первые четыре комбинации сигналов в таблице рис. 5.7,*б*), то на выходе элемента *D4* сигнал равен единице, на выходе *D2* сигнал равен нулю и, следовательно, на выходе *D5* динамического риска быть не может. Если $B = 1$, то $D1 = 0$ и на выходе *D5* не может быть динамического риска. Таким образом, проверка на шести наборах, когда $A = 0$ или $B = 1$, показала, что динамические риски сбоя в схеме отсутствуют.

Анализ работы схемы при условии $A = 1$, $B = 0$ и при изменении сигналов *C* из нуля в единицу выявляет динамический риск сбоя на элементе *D5* (рис. 5.7,*в*). На других наборах динамического риска сбоя нет.

§ 5.2. Проектирование сумматоров

Сумматор — это комбинационный функциональный узел, предназначенный в основном для выполнения операции суммирования двоичных чисел, а также для вычитания, умножения, деления, преобразования чисел в дополнительный код и других дополнительных операций.

Классификация сумматоров может быть проведена по трем основным признакам: 1) по числу входов (полусумматоры, одноразрядные и многоразрядные сумматоры); многоразрядные сумматоры, в свою очередь, делятся на последовательные и параллельные; последние по способу организации межразрядных переносов подразделяются на схемы с последовательным и параллельным переносом и с групповой структурой; 2) по способу тактирования (синхронные и асинхронные сумматоры); 3) по системе счисления (двоичные, двоично-десятичные и др.).

Проектирование полусумматоров. Полусумматорами называют КФУ с двумя входами (a, b) и двумя выходами (S, P), на которых вырабатываются сигналы суммы

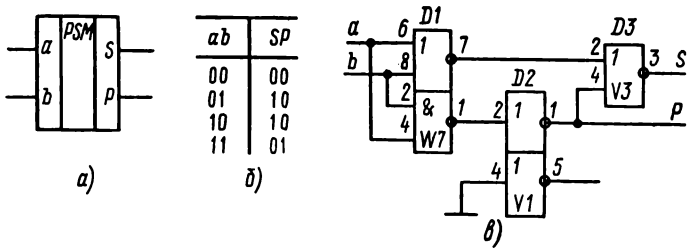


Рис. 5.8. Полусумматор:
а — условное графическое обозначение; *б* — таблица истинности; *в* — схемная реализация на ячейках V_1, V_3, W_7

и переноса. Из таблицы истинности (рис. 5.8, *б*) следует:

$$S = \bar{a}b \vee a\bar{b} = (a \vee b)(\bar{a} \vee \bar{b}), \quad P = ab. \qquad (5.4)$$

Преобразуем уравнения (5.4) к виду, наиболее пригодному для реализации на библиотечных элементах БМК (см. приложение):

$$S = \overline{\overline{a \vee b} \vee ab}; \quad P = ab. \qquad (5.5)$$

Логическая схема полусумматора, реализующего уравнения (5.5), представлена на рис. 5.8, *в*. Описание схемы на языке ОЦИС-РП имеет следующий вид.

```
ФУ:
  PSM(A,B,S,P)#
ЯЧЕЙКА /BANK.BMK/:
  D1=>W7(K6=A;K8=B;K2=B;K4=A;K7=D1_7;K1=D1_1);
  D2=>V1(K2=D1_1;K1=P;K4=ЗЕМЛЯ;K=*);
  D3=>V3(K2=D1_7;K1=P;K3=S)#
КОНЕЦ#
```

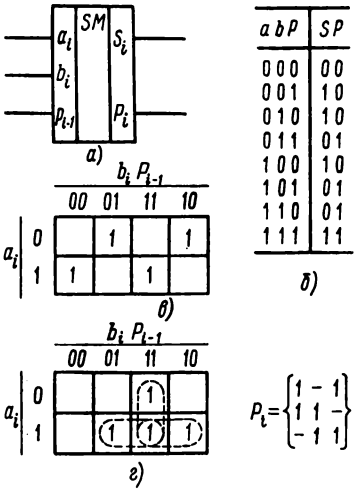
Проектирование одноразрядных сумматоров. Одноразрядным сумматором называют КФУ с тремя входами и двумя выходами. Это основной элемент многоразрядных сумматоров. Он выполняет арифметическое сложение одноразрядных двоичных чисел a_i, b_i и перенос из предыдущего разряда P_{i-1} с образованием на выходах суммы S_i и переноса в старший разряд P_i. Условное графическое обозначение и таблица истинности одноразрядного сумматора показаны на рис. 5.9, *а*, *б*. Для минимизации функций S_i и P_i отобразим их на карте

Карно (соответственно рис. 5.9, *в*, *г*). Функция S_i минимизации не поддается, поэтому запишем ее в форме СДНФ:

$$S_i = \bar{a}_i\bar{b}_iP_{i-1} \vee \bar{a}_ib_i\bar{P}_{i-1} \vee \vee a_i\bar{b}_i\bar{P}_{i-1} \vee a_ib_iP_{i-1}. \quad (5.6)$$

Функцию P_i после минимизации можно записать в виде

$$P_i = a_iP_{i-1} \vee a_ib_i \vee b_iP_{i-1} = = a_ib_i \vee P_{i-1}(a_i \vee b_i). \quad (5.7)$$



a b P	S P
000	00
001	10
010	10
011	01
100	10
101	01
110	01
111	11

б)

Рис. 5.9. Одноразрядный сумматор

Для упрощения схемной реализации функции S_i выполним следующие преобразования. Представим S_i как функцию четырех переменных: a_i, b_i, P_{i-1} и P_i. Для этого составим таблицу истинности (рис. 5.10, *а*), где на нереальных входных наборах функция S_i принимает неопределенные значения (×). После минимизации с помощью карты Карно (рис. 5.10, *б*) запишем

Таблица 5.1

Фрагмент временной диаграммы работы одноразрядного сумматора

Имя узла схемы	Номер испытания 5	6	7
	Время, нс		
	1111122222 02790058903559 04280486202082	11111111222 01600467789117 04802640682044	1 02790 04284
A	11111111111111	11111111111111	11111
B	.....	.....	11111
P0	.....	11111111111111	.....
S	.....<<1	1111111111>>.	.....
P1	111111>.....	.....<<<111111	11111
D1(K5)	....<<<<<11111	111>>>>.....	.....
D2(K1)	11>.....	..<11111111111	11>..
D2(K7)	.<<<<111111111	1>>>.....	.<<<1
D3(K1)	.....	.....	.....
D3(K14)	11111111111111	11111111111111	11111
D4(K3)	.....<<<<11	111111>>>>....	.....
D4(K1)	1111111111>>..	.....<<<11	11111

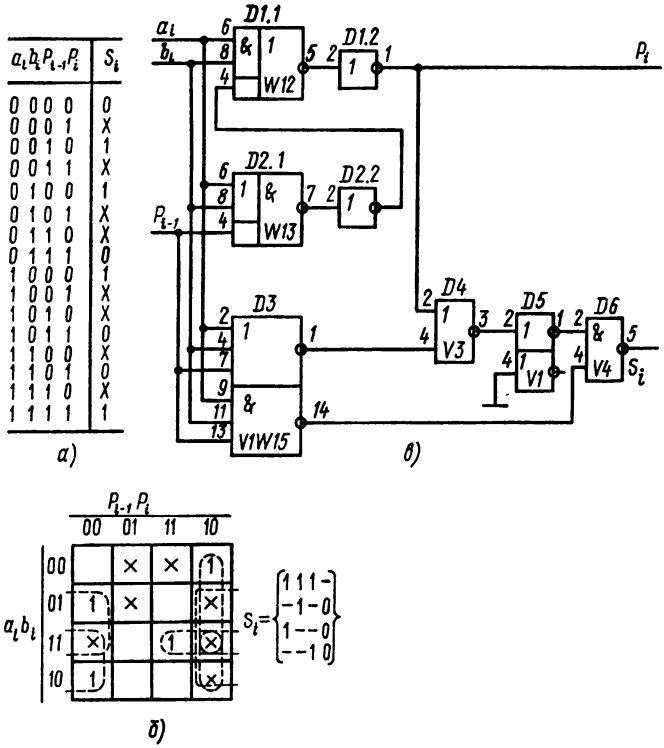


Рис. 5.10. Одноразрядный сумматор:
а — табличное представление функции S_i, *б* — минимизация функции S_i; *в* — логическая схема на ячейках V_1, V_3, V_4, W_{12}, W_{13}, V_1, W_{15}

$$S_i = a_i b_i P_{i-1} \vee b_i \bar{P}_i \vee a_i \bar{P}_i \vee P_{i-1} \bar{P}_i = \\ = a_i b_i P_{i-1} \vee (a_i \vee b_i \vee P_{i-1}) \bar{P}_i . \qquad (5.8)$$

Логическая схема сумматора, построенного по формулам (5.7) и (5.8), показана на рис. 5.10, *в*. Данная схема сумматора имеет топологические размеры (V6V). Фрагмент временнóй диаграммы, работы сумматора, полученной на системе ФЛП-3000, приведен в табл. 5.1. По временной диаграмме могут быть определены различные динамические параметры схемы (задержка распространения сигнала на ЛЭ, длительность фронта переключения и др.). Например, время распространения сигнала от входов A и B до выхода *S* равно 29,2 нс при переключении из 0 в 1 (пятое испытание) и 24,7 нс —

из 1 в 0 (шестое испытание), переключение на выходе К3 элемента $D4$ из 0 в 1 на пятом испытании начинается в 19,2 нс и заканчивается в 25,8 нс, т. е. имеет длительность 6,6 нс.

§ 5.3. Проектирование шифраторов и дешифраторов

Шифрация и дешифрация (сжатие данных и обратное преобразование) являются основными видами преобразования информации. Дешифраторы преобразуют двоичный код в код «1 из N», а шифраторы, наоборот, — код «1 из N» в двоичный код.

Различают полные и неполные дешифраторы. Число выходов полного дешифратора $N_{вых}=2^n$, неполного — $N_{вых}<2^n$, где n — число двоичных разрядов (число входов). На рис. 5.11, *а* приведено условное графическое изображение полного стробируемого дешифратора «1 из 8», а на рис. 5.11, *б* — его таблица истинности. Приведем алгоритмическое описание дешифратора на языке ОЦИС-РП.

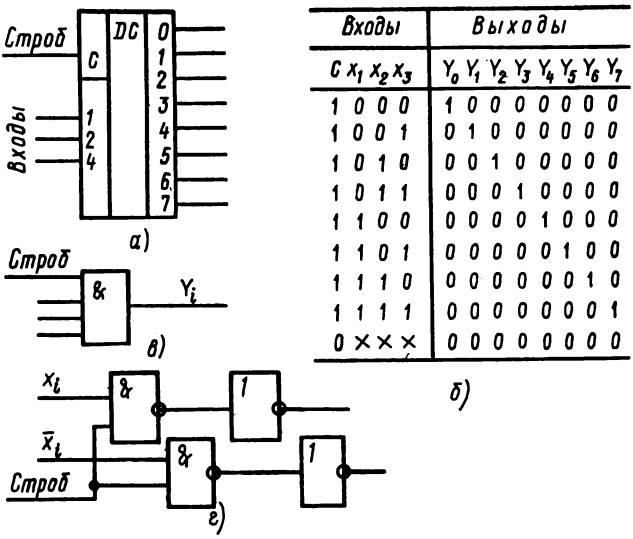
```
ФУ:
  DC1_8(C,X[0:3],Y[0:7])#
  АЛГОРИТМ:
    0=>П[0:7];C=>П[X[0:3]];
    П[0:7]=>Y[0:7](TZ=25,R=3)#
КОНЕЦ#
```

Аналитическое описание дешифратора в форме СДНФ

$$Y_i=m_iC,\ i=0,1,\ldots,7, \qquad (5.9)$$

где m_i — i-й минтерм трех входных переменных; C — сигнал разрешения (строб).

На рис. 5.11, *в, г* соответственно представлены два способа стробирования дешифраторов: путем введения дополнительного входа в каждый элемент (стробирование по выходу) либо путем блокирования всех элементов через одну из входных цепей (стробирование по входу). Таким образом, полный стробируемый по выходу одноступенчатый дешифратор может быть реализован в соответствии с (5.9) на восьми ячейках типа V1W2 и на двух ячейках V 1 и W 1.



Входы				Выходы							
C	x_1	x_2	x_3	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1
0	×	×	×	0	0	0	0	0	0	0	0

Рис. 5.11. Полный стробируемый дешифратор «1 из 8»

Для неполных дешифраторов имеются безразличные входные наборы, которые можно использовать при минимизации выходных функций. Например, при проектировании дешифратора «1 из 10» безразличными являются наборы, отмеченные знаком X на карте Карно (рис. 5.12). После совместной минимизации на рабочих и соседних к ним безразличных наборах получим следующие логические уравнения:

$$\begin{aligned} Y_0 &= \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4, & Y_3 &= \bar{x}_2x_3x_4, & Y_6 &= x_2x_3\bar{x}_4, \\ Y_1 &= \bar{x}_1\bar{x}_2\bar{x}_3x_4, & Y_4 &= x_2\bar{x}_3\bar{x}_4, & Y_7 &= x_2x_3x_4, \\ Y_2 &= \bar{x}_2x_3\bar{x}_4, & Y_5 &= x_2\bar{x}_3x_4, & Y_8 &= x_1\bar{x}_4, \\ & & & & Y_9 &= x_1\bar{x}_4. \end{aligned} \qquad (5.10)$$

Дешифратор «1 из 10» может быть реализован в соответствии с (5.10) на двух ячейках V 1 W 1, шести ячейках W 11, трех ячейках V 1 и двух ячейках V 4. Топологические размеры дешифратора W 14 W. Если при проектировании дешифратора не проводить совместной минимизации рабочих и безразличных наборов, то для схемной реализации потребуется 10 ячеек типа V 1 W 2, а топологические размеры будут значительно большими (V 21 W).

Рассмотрим некоторые области применения дешифраторов в разрабатываемых устройствах:

1) в качестве преобразователя двоично-десятичного кода в семисегментный код в устройствах визуальной индикации десятичных цифр на световом табло (рис. 5.13, *а*). Проектирование таких дешифраторов осуществляется на основе таблицы истинности (рис. 5.13, *б*) дешифратора «1 из 10» и системы уравнений:

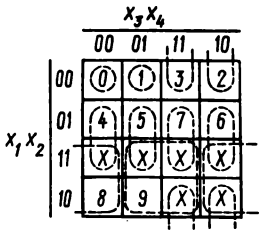


Рис. 5.12. Минимизация функции неполного дешифратора «1 из 10»

$$a = \overline{Y_1 \vee Y_4}, \quad б = \overline{Y_5 \vee Y_6}, \quad в = \bar{Y}_2, \quad г = \overline{Y_1 \vee Y_4 \vee Y_7},$$
$$д = Y_0 \vee Y_2 \vee Y_6 \vee Y_8, \; е = \overline{Y_1 \vee Y_2 \vee Y_3 \vee Y_7}, \; ж = \overline{Y_0 \vee Y_1 \vee Y_7}, \tag{5.11}$$

где Y_0 — Y_9 — выходные шины дешифратора «1 из 10» (рис. 5.13, *в*);

2) для реализации логических функций.

Пример 5.4. Получить на основе дешифратора логическую функцию вида

$$Y = x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3. \tag{5.12}$$

Решение. Исходная функция (5.14) задана в форме ДНФ. Преобразуем ее в СДНФ:

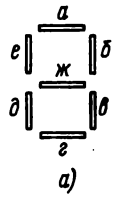
$$Y = x_1\bar{x}_2(x_3 \vee \bar{x}_3) \vee x_2\bar{x}_3(x_1 \vee \bar{x}_1) \vee \bar{x}_1\bar{x}_2x_3 =$$
$$= x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3$$

Полученное выражение реализуется логической схемой, представленной на рис. 5.14;

3) для реализации четырехфазной системы синхронизации согласно временно́й диаграмме (рис. 5.15);

4) для преобразования кодов. В качестве примера приведена таблица истинности преобразователя двоично-десятичного кода в код «3 из 5» (табл. 5.2), из которой получаются уравнения выходных функций для реализации преобразователя:

$$V = \overline{Y_3 \vee Y_4 \vee Y_5 \vee Y_6}, \quad X = \overline{Y_0 \vee Y_2 \vee Y_4 \vee Y_8},$$
$$W = \overline{Y_0 \vee Y_1 \vee Y_3 \vee Y_7}, \quad Y = \overline{Y_1 \vee Y_2 \vee Y_6 \vee Y_9},$$
$$Z = \overline{Y_6 \vee Y_7 \vee Y_8 \vee Y_9}. \tag{5.13}$$



Десятичное число	Входы x_1	x_2	x_3	x_4	DC	Выходы а	б	в	г	д	е	ж
0	0	0	0	0	Y_0	1	1	1	1	1	1	0
1	0	0	0	1	Y_1	0	1	1	0	0	0	0
2	0	0	1	0	Y_2	1	1	0	1	1	0	1
3	0	0	1	1	Y_3	1	1	1	1	0	0	1
4	0	1	0	0	Y_4	0	1	1	0	0	1	1
5	0	1	0	1	Y_5	1	0	1	1	0	1	1
6	0	1	1	0	Y_6	1	0	1	1	1	1	1
7	0	1	1	1	Y_7	1	1	1	0	0	0	0
8	1	0	0	0	Y_8	1	1	1	1	1	1	1
9	1	0	0	1	Y_9	1	1	1	1	0	1	1

б)

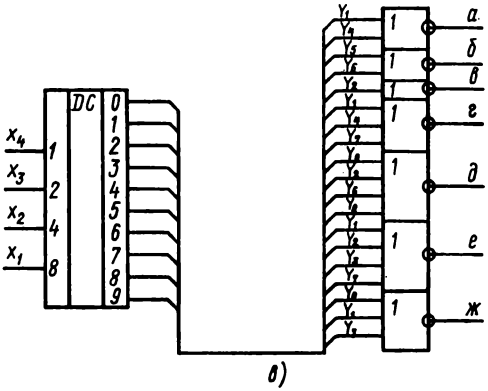


Рис. 5.13. Дешифратор-преобразователь двоично-десятичного кода в семисегментный код

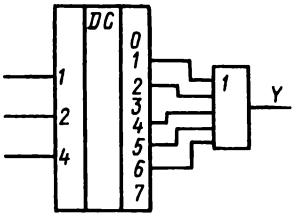


Рис. 5.14. Реализация логической функции $Y = x_1x_2 \vee x_2x_3 \vee \bar{x}_1\bar{x}_2x_3$ на основе дешифратора

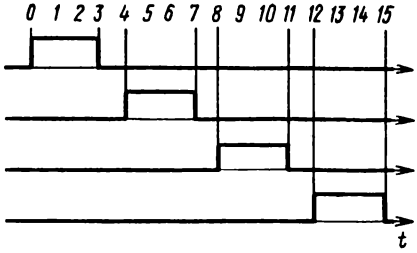


Рис. 5.15. Временна́я диаграмма работы системы четырехфазной синхронизации

Таблица 5.2

Таблица истинности преобразователя двоично-десятичного кода в код "3 из 5"

Десятичное число	Входы $X_1X_2X_3X_4$	DC	Код "3 из 5" V W X Y Z
0	0 0 0 0	Y_0	1 0 0 1 1
1	0 0 0 1	Y_1	1 0 1 0 1
2	0 0 1 0	Y_2	1 1 0 0 1
3	0 0 1 1	Y_3	0 0 1 1 1
4	0 1 0 0	Y_4	0 1 0 1 1
5	0 1 0 1	Y_5	0 1 1 0 1
6	0 1 1 0	Y_6	0 1 1 1 0
7	0 1 1 1	Y_7	1 0 1 1 0
8	1 0 0 0	Y_8	1 1 0 1 0
9	1 0 0 1	Y_9	1 1 1 0 0

Таблица 5.3

Таблица истинности шифратора 10-4

Десятичное число	Входы $X_0X_1X_2X_3X_4X_5X_6X_7X_8X_9$	Выходы $Y_3Y_2Y_1Y_0$
0	1 0 0 0 0 0 0 0 0 0	0 0 0 0
1	0 1 0 0 0 0 0 0 0 0	0 0 0 1
2	0 0 1 0 0 0 0 0 0 0	0 0 1 0
3	0 0 0 1 0 0 0 0 0 0	0 0 1 1
4	0 0 0 0 1 0 0 0 0 0	0 1 0 0
5	0 0 0 0 0 1 0 0 0 0	0 1 0 1
6	0 0 0 0 0 0 1 0 0 0	0 1 1 0
7	0 0 0 0 0 0 0 1 0 0	0 1 1 1
8	0 0 0 0 0 0 0 0 1 0	1 0 0 0
9	0 0 0 0 0 0 0 0 0 1	1 0 0 1

Шифратор — это комбинационный функциональный узел, преобразующий код «1 из *N*» в двоичный код. Полный шифратор имеет 2^n входов и *n* выходов. Одно из основных применений шифратора — ввод данных с клавиатуры, при котором нажатие на клавишу с десятичной цифрой должно приводить к передаче в устройство этой цифры в двоичном коде. Данная функция реализуется неполным шифратором «10—4» (рис. 5.16). Из таблицы истинности (табл. 5.3) получаем логические уравнения для шифратора «10—4»:

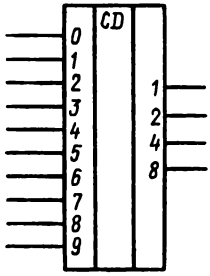


Рис. 5.16. Условное графическое обозначение шифратора «10—4»

$$Y_0 = x_1 \vee x_3 \vee x_5 \vee x_7 \vee x_9, \qquad Y_2 = x_4 \vee x_5 \vee x_6 \vee x_7,$$
$$Y_1 = x_2 \vee x_3 \vee x_6 \vee x_7, \qquad Y_3 = x_8 \vee x_9. \qquad (5.14)$$

Алгоритмическое описание этого шифратора на языке ОЦИС-РП имеет вид

```
ФУ:
  CD10_4(X[0:9],Y[0:3])#
  АЛГОРИТМ:
    X[1]!X[3]!X[5]!X[7]![9] => Y[0](TZ=25,R=5);
    X[2]!X[3]!X[6]!X[7]     => Y[1](TZ=25,R=5);
    X[4]!X[5]!X[6]!X[7]     => Y[2](TZ=25,R=5);
    X[8]!X[9]               => Y[3](TZ=25,R=5);
КОНЕЦ#
```

§ 5.4. Проектирование мультиплексоров и демультиплексоров

Мультиплексор — это комбинационный функциональный узел, служащий для последовательного опроса состояний большого числа переменных и передачи их на

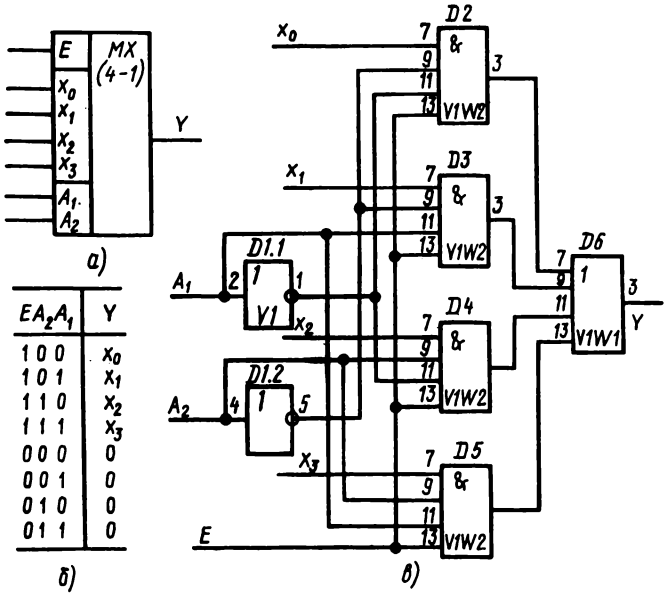


Рис. 5.17. Мультиплексор «4—1»

один выход. Чаще всего используют мультиплексоры на 4, 8, 16 входов. Условное графическое обозначение мультиплексора на четыре входа показано на рис. 5.17, *а*, где E — сигнал, разрешающий мультиплексирование, x_0, x_1, x_2, x_3 — информационные входы; A_1, A_2 — адресные (управляющие) входы; Y — выход. На основе таблицы истинности (рис. 5.17, *б*) получим СДНФ выходной функции

$$Y = x_0\bar{A}_2\bar{A}_1E \vee x_1\bar{A}_2A_1E \vee x_2A_2\bar{A}_1E \vee x_3A_2A_1E = \\ = \bigvee_i x_i\alpha_iE, \quad i = 0, 1, ..., 2^{n-1}, \qquad (5.15)$$

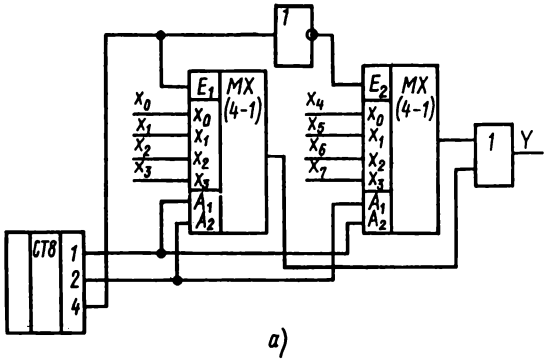
где α_i — минтерм, соответствующий i-му адресному набору. Данная функция далее может быть реализована в виде логической схемы мультиплексора «4—1» в заданном базисе элементов (рис. 5.17, *в*). Алгоритмическое и структурное описание мультиплексора на языке ОЦИС-РП имеет вид

```
ФУ:
  MX4(E,A1,A2,X[0:3],Y)#
  АЛГОРИТМ:
    E&X[A2.A1]=>Y(TZ10=23,TZ01=20,R=3)#
  ЯЧЕЙКА /BANK.BMK/:
    D1=>V1(K2=A1;K1=A1I;K4=A2;K5=A2I);
    D2=>V1W2(K7=X[0];K9=A2I;K11=A1I;K13=E;K3=D2_3);
    D3=>V1W2(K7=X[1];K9=A2I;K11=A1;K13=E;K3=D3_3);
    D4=>V1W2(K7=X[2];K9=A2;K11=A1I;K13=E;K3=D4_3);
    D5=>V1W2(K7=X[3];K9=A2;K11=A1;K13=E;K3=D5_3);
    D6=>V1W1(K7=D2_3;K9=D3_3;K11=D4_3;K13=D5_3;K3=Y)#
КОНЕЦ#
```

Мультиплексор можно рассматривать как преобразователь параллельной информации в последовательную. Мультиплексор на большое числов входов, как правило, приходится строить из мультиплексоров меньшей размерности. На рис. 5.18 и 5.19 показаны примеры реализации мультиплексоров «8—1» и «16—1» на основе МХ «4—1».

Рассмотрим некоторые области применения мультиплексоров в разрабатываемых устройствах:

1) *для генерации повторяющейся двоичной последовательности цифр.* Например, двоичная последователь-



EA_1A_2	Y
1 0 0	x_0
1 0 1	x_1
1 1 0	x_2
1 1 1	x_3
0 0 0	x_4
0 0 1	x_5
0 1 0	x_6
0 1 1	x_7

Рис. 5.18. Мультиплексор «8—1»:
а — логическая схема; *б* — таблица истинности

$EA_4A_3A_2A_1$	Y
1 0 0 0 0	x_0
1 0 0 0 1	x_1
1 0 0 1 0	x_2
1 0 0 1 1	x_3
.....	.....
1 1 1 0 0	x_{12}
1 1 1 0 1	x_{13}
1 1 1 1 0	x_{14}
1 1 1 1 1	x_{15}
0 0 0 0 0	0
0 0 0 0 1	0
.....	.....
0 1 1 1 1	0

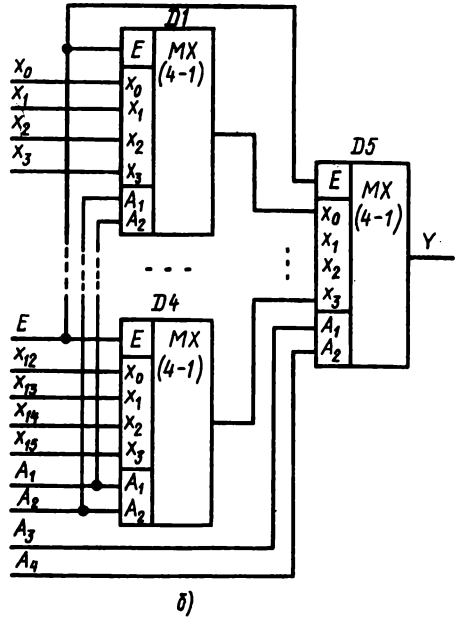


Рис. 5.19. Мультиплексор «16—1»:
а — таблица истинности; *б* — логическая схема

ность, состоящая из восьми цифр: 10101110, может быть реализована на мультиплексоре типа «8—1» (рис. 5.20). Информационные входы мультиплексора в соответствии с заданной последовательностью соединяются с шинами земли (0) и питания (1);

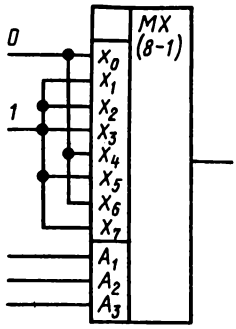


Рис. 5.20. Мультиплексор «8—1», используемый для генерации кода 10101110

2) *для реализации логических функций.* Рассмотрим, каким образом с помощью мультиплексора «4—1» реализуется функция трех переменных $Y = f\ (A,\ B,\ C)$. Сначала выбирается любое сочетание двух переменных AB, AC, BC, которые являются управляющими и подаются на адресные входы мультиплексора. На информационные входы в этом случае могут быть поданы четыре функции одной (третьей) переменной. Например, если в качестве управляющих сигналов выбраны переменные A и B, то на четыре входа мультиплексора могут поступать переменные C, $\bar{C}$, 0 и 1.

Рис. 5.21 иллюстрирует соответствие информационных входов x_0, x_1, x_2, x_3 определенным адресным (управляющим) входом мультиплексора «4—1». Если в качестве управляющих принять сигналы A и B, то входу x_0 будут соответствовать две ячейки карты Карно, для которых $A = B = 0$, входу x_1 — две ячейки с координатами $A = 0$, $B = 1$, x_2—$A = 1$, $B = 0$, x_3 — ячейки с $A = B = 1$ (рис. 5.21, *а*). Таким образом, карта Карно на три переменные разбивается как бы на четыре двухъячеечные карты на одну переменную. Затем минимизируется набор из четырех функций одной переменной и получаются необходимые значения сигналов на информационных входах для реализации заданной логической функции.

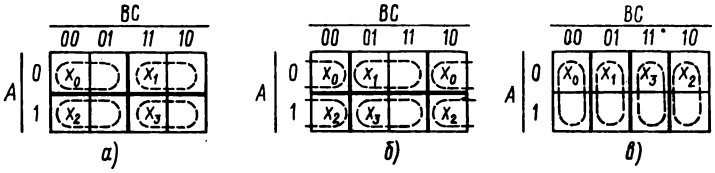


Рис. 5.21. Соответствие информационных входов мультиплексора «4—1» управляющим сигналам A и B (*а*), A и C (*б*), B и C (*в*)

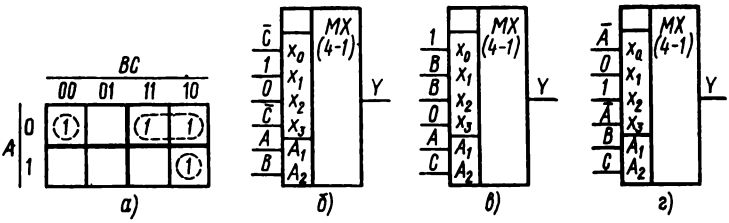


Рис. 5.22. Применение мультиплексора для реализации функции $Y=\bar{A}\bar{B}\bar{C}\vee\bar{A}B\bar{C}\vee\bar{A}BC\vee AB\bar{C}$

Пример 5.5. Реализовать на мультиплексоре «4—1» логическую функцию трех переменных:

$$Y=\bar{A}\bar{B}\bar{C}\vee\bar{A}B\bar{C}\vee\bar{A}BC\vee AB\bar{C}. \qquad (5.16)$$

Решение. В качестве управляющих сигналов произвольно выбираются переменные A и B. Функция (5.16) представляется на четырех двухъячеечных картах Карно (рис. 5.22, *а*). Затем минимизируется набор из четырех функций от одной переменной C, соответствующих каждому информационному входу (x_0, x_1, x_2, x_3). Обе ячейки, соответствующие x_1, помечены единицей, и, следовательно, на вход x_1 подается сигнал $C\vee\bar{C}=1$. Оставшиеся входные функции получаем также при помощи карты Карно (рис. 5.22, *а*): $x_0=\bar{C}$, $x_2=0$, $x_3=\bar{C}$. Подавая на входы найденные значения x_i, реализуем заданную функцию (рис. 5.22, *б*). Если в качестве управляющих выбрать сигналы A и C или B и C, то получатся новые реализации (соответственно рис. 5.22, *в*, *г*).

Пример 5.6. Реализовать на мультиплексоре «4—1» логическую функцию четырех переменных A, B, C, D:

$$Y=\vee\,(0,1,6,7,9,10,11,12,13). \qquad (5.17)$$

В качестве управляющих сигналов принять переменные A и B.

Решение. Покажем на карте Карно для функции четырех переменных (A, B, C, D) расположение информационных сигналов (x_0, x_1, x_2, x_3 мультиплексора «4—1» (рис. 5.23, *а*). Для каждого из четырех x_i получили свою четырехъячеечную карту для двух переменных. Например, x_0 соответствуют четыре ячейки с координатами $A=B=0$ (рис. 5.23, *а*).

Отобразим на карте Карно функцию (5.17) и проведем для каждого x_i минимизацию (рис. 5.23, *б*). Получим $x_0=\bar{C}$, $x_1=C$, $x_2=C\vee\vee D$, $x_3=\bar{C}$. Это означает, что для реализации заданной функции на мультиплексоре «4—1» необходимо подать на информационные входы соответствующие значения переменных C и D (рис. 5.23, *в*).

Демультиплексор — это комбинационный функциональный узел, выполняющий функцию, противоположную функции мультиплексора, — распределение входного сигнала x в соответствии с адресом на одну из N выходных шин. Если входной сигнал x может быть распределен

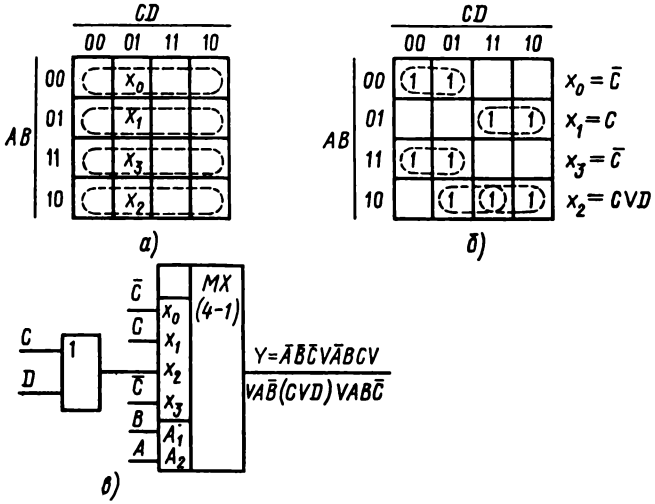


Рис. 5.23. Реализация функции Y=(0, 1, 6, 7, 9, 10, 11, 12, 13) на мультиплексоре «4—1»

на одну из четырех выходных шин, то имеем демультиплексор «1—4», функционирование которого определяется следующими логическими уравнениями:

$$Y_0 = x\bar{A}_1\bar{A}_2, \qquad Y_2 = xA_1\bar{A}_2,$$

$$Y_1 = x\bar{A}_1A_2, \qquad Y_3 = xA_1A_2. \qquad (5.18)$$

Приведем алгоритмическую модель демультиплексора «1—4» с параметрами TZ= 15нс, R=5 кОм:

(5.19)

```
ФУ:
  DMX4(X,A1,A2,Y[0:3])#
  АЛГОРИТМ:
    X=>Y[A2.A1](TZ=15,R=5
КОНЕЦ#
```

§ 5.5. Проектирование компараторов и преобразователей кода

Компаратор — это комбинационный функциональный узел, предназначенный для сравнения двух чисел (A, B) по различным признакам: равно (=), не равно ($\neq$),

больше (>), меньше (<), больше или равно (⩾), меньше или равно (⩽). Выходные функции компаратора определяются следующими выражениями:

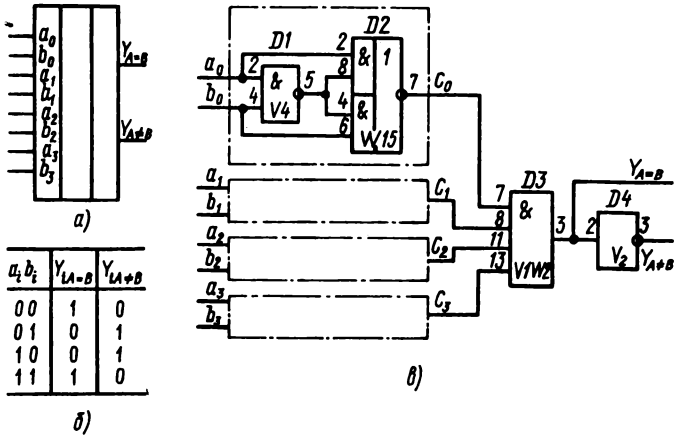
$$Y_{A\neq B} = \bar{Y}_{A=B}, \quad Y_{A\geqslant B} = Y_{A=B} \vee Y_{A>B} = \bar{Y}_{A<B},$$
$$Y_{A<B} = \bar{Y}_{B>A}, \quad Y_{A\leqslant B} = Y_{A=B} \vee Y_{A<B} = \bar{Y}_{A>B}. \qquad (5.20)$$

Условное графическое обозначение четырехразрядного компаратора для функций равнозначности $Y_{A=B}$ и неравнозначности $Y_{A\neq B}$ показано на рис. 5.24, *а*. Алгоритм функционирования такого компаратора на языке ОЦИС-РП имеет вид

```
ФУ:
  КОМП4(A[0:3],B[0:1],Y)#
  АЛГОРИТМ:
    (A=B)=>Y(TZ=15,R=5)#
КОНЕЦ#
```

Синтезируем четырехзначный компаратор для функций $Y_{A=B}$ и $Y_{A\neq B}$. Равенство чисел A и B имеет место при равнозначности цифр всех разрядов. Значения функций $Y_{A=B}$ и $Y_{A\neq B}$ приведены в таблице истинности (рис. 5.24, *б*), из которой следует:

$$Y_{iA=B} = \bar{a}_i\bar{b}_i \vee a_ib_i, \qquad Y_{iA\neq B} = \bar{a}_ib_i \vee a_i\bar{b}_i. \qquad (5.21)$$



a_ib_i	$Y_{iA=B}$	$Y_{iA\neq B}$
00	1	0
01	0	1
10	0	1
11	1	0

Рис. 5.24. Четырехразрядный компаратор

Функцию разнозначности $Y_{A=B}$ для двух четырехразрядных чисел можно записать в виде

$$Y_{A=B} = (\bar{a}_0\bar{b}_0 \vee a_0b_0)(\bar{a}_1\bar{b}_1 \vee a_1b_1)(\bar{a}_2\bar{b}_2 \vee a_2b_2)$$

$$(\bar{a}_3\bar{b}_3 \vee a_3b_3) = \mathop{\&}_{i=0}^{3}(\bar{a}_i\bar{b}_i \vee a_ib_i). \qquad (5.22)$$

Аналогично функция неравнозначности ($Y_{A\neq B}$) имеет вид

$$Y_{A\neq B}=(a_0\bar{b}_0 \vee \bar{a}_0b_0)(a_1\bar{b}_1 \vee \bar{a}_1b_1)(a_2\bar{b}_2 \vee \bar{a}_2b_2)$$

$$(a_3\bar{b}_3 \vee \bar{a}_3b_3) = \mathop{\&}_{i=0}^{3}(a_i\bar{b}_i \vee \bar{a}_ib_i). \qquad (5.23)$$

Так как $Y_{A=B}$ и $Y_{A\neq B}$ взаимодополняющие функции, то для схемной реализации может быть использовано любое из уравнений (5.22), (5.23).

В ряде случаев требуется построить компаратор, не используя инверсии аргументов. С этой целью выражения (5.21) преобразуются к виду:

$$Y_{iA=B} = \overline{a_i\bar{b}_i \vee \bar{a}_ib_i} = \overline{a_i(\bar{a}_i \vee \bar{b}_i) \vee b_i(\bar{a}_i\bar{b}_i)} =$$

$$= \overline{a_i\overline{a_ib_i} \vee b_i\overline{a_ib_i}}, \qquad (5.24)$$

$$Y_{iA\neq B} = a_i\overline{a_ib_i} \vee b_i\overline{a_ib_i}. \qquad (5.25)$$

Полученные соотношения приводят к логической схеме, представленной на рис. 5.24, *в*, структурная модель которой на языке ОЦИС-РП имеет вид

```
ФУ /BANK.BIS/:
  КОМП1(A,B,C)#
  ЯЧЕЙКА /BANK.BMK/:
    D1=>V4(K2=A;K4=B;K5=D1_5);
    D2=>W15(K2=A;K8.K4=2*D1_5;K6=B;K7=C)#
КОНЕЦ#
ФБ /BANK.BIS/:
  КОМП4(A[0:3],B[0:3],Y,YI)#
   ФУ /BANK.BIS/:
    D1=>КОМП1(A=A[0];B=B[0];C=D1_C);
```

```
D2=>КОМП1(A=A[1];B=B[1];C=D2_C);
D3=>КОМП1(A=A[2];B=B[2];C=D3_C);
D4=>КОМП1(A=A[3];B=B[3];C=D4_C);
ЯЧЕЙКА /BANK.BMK/:
D5=>V1W2(K7=D1_C;K9=D2_C;K11=D3_C;K13=D4_C;K3=Y);
D6=>V2(K2=Y;K3=YI)#
КОНЕЦ#
```

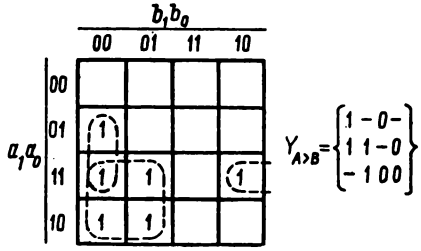
Рассмотрим процедуры синтеза двухразрядного компаратора для функций $Y_{A>B}$ и $Y_{A<B}$. Итак, необходимо сравнить два двухразрядных двоичных числа: $A=a_1a_0$ и $B=b_1b_0$. Запишем функционирование данного компаратора в виде таблицы истинности (рис. 5.25, *а*). Как следует из таблицы, $A>B$, если $a_2>b_2$ или при $a_2=b_2$, $a_1>B_1$. В свою очередь, $A<B$, если $a_2<b_2$ или $a_1<b_1$ при $a_2=b_2$.

Проведем минимизацию функций $Y_{A>B}$ и $Y_{A<B}$ с помощью карты Карно (рис. 5.25, *б, в*):

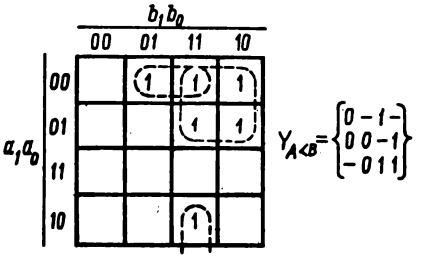
$$Y_{A>B} = a_1\bar{b}_1 \vee a_1a_0\bar{b}_0 \vee a_0\bar{b}_1\bar{b}_0 = a_1\bar{b}_1 \vee a_0\bar{b}_0(a_1 \vee \bar{b}_1),$$
$$Y_{A<B}=\bar{a}_1b_1 \vee \bar{a}_1\bar{a}_0b_0 \vee \bar{a}_0b_1b_0=\bar{a}_1b_1 \vee \bar{a}_0b_0(a_1 \vee b_1). \quad (5.26)$$

a_1	a_0	b_1	b_0	$Y_{A>B}$	$Y_{A<B}$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	0

а)



б)



в)

Рис. 5.25. Двухразрядный компаратор

При сравнении n двухразрядных двоичных чисел уравнения (5.26) можно записать в виде:

$$Y_{A>B} = a_{n-1}\bar{b}_{n-1} \vee a_{n-2}\bar{b}_{n-2}(a_{n-2} \vee \bar{b}_{n-1}) \vee \ldots \vee a_i\bar{b}_i(a_{i\pm1} \vee \vee \bar{b}_{i=1})(a_{i+2} \vee \bar{b}_{i+2})\ldots(a_{n-1} \vee \bar{b}_{n-1}) \vee \ldots \vee a_0\bar{b}_0(a_1 \vee \bar{b}_1)\ldots \ldots(a_{n-1} \vee \bar{b}_{n-1}),$$

$$Y_{A<B} = \bar{a}_{n-1}b_{n-1} \vee \bar{a}_{n-2}b_{n-2}(\bar{a}_{n-1} \vee b_{n-1}) \vee \ldots \vee \bar{a}_ib_i \times \times(\bar{a}_{i+1} \vee b_{i+1})(\bar{a}_{i+2} \vee b_{i+2})\ldots(\bar{a}_{n-} \vee b_{n-1}) \vee \ldots \vee \bar{a}_0b_0(\bar{a}_1 \vee b_1)\ldots \ldots(a_{n-1} \vee b_{n-1}). \quad (5.27)$$

Логические выражения для функций $Y_{A\geqslant B}$, $Y_{A\leqslant B}$, в соответствии с которыми могут быть построены n-разрядные компараторы, представляются в виде:

$$Y_{A\geqslant B} = Y_{A>B} \vee Y_{A=B} = a_{n-1}\bar{b}_{n-1} \vee a_{n-2}\bar{b}_{n-2} \times \times(a_{n-1} \vee \bar{b}_{n-1}) \vee \ldots \vee a_ib_i(a_{i+1} \vee \bar{b}_{i+1})\ldots(a_{n-1} \vee \vee \bar{b}_{n-1}) \vee \ldots \vee a_0\bar{b}_0(a_1 \vee \bar{b}_1)\ldots(a_{n-1} \vee \bar{b}_{n-1}) \vee (a_{n-1}\bar{b}_{n-1} \vee \vee a_{n-1}b_{n-1})(a_{n-2}\bar{b}_{n-2} \vee a_{n-2}b_{n-2})\ldots(a_0\bar{b}_0 \vee a_0b_0),$$

$$Y_{A\leqslant B} = Y_{A<B} \vee Y_{A=B} = \bar{a}_{n-1}b_{n-1} \vee \bar{a}_{n-2}b_{n-2}(\bar{a}_{n-1} \vee \vee b_{n-1}) \vee \ldots \vee \bar{a}_ib_i(\bar{a}_{i+1} \vee b_{i+1})\ldots(\bar{a}_{n-1} \vee b_{n-1}) \vee \ldots \vee \bar{a}_0b_0(\bar{a}_1 \vee \vee b_1)\ldots(\bar{a}_{n-1} \vee b_{n-1}) \vee (\bar{a}_{n-1}\bar{b}_{n-1} \vee a_{n-1}b_{n-1}) \times \times(\bar{a}_{n-2}\bar{b}_{n-2} \vee a_{n-2}b_{n-2})\ldots(\bar{a}_0\bar{b}_0 \vee a_0b_0). \quad (5.28)$$

Таблица 5.4

Таблица истинности преобразователей кодов

Десятичное число	Код						
	8 4 2 1	2 4 2 1	обратный	дополнительный	с избытком 3	Грея
	$X_4X_3X_2X_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$
0	0 0 0 0	0 0 0 0	1 1 1 1	0 0 0 0	0 0 1 1	0 0 0 0
1	0 0 0 1	0 0 0 1	1 1 1 0	1 1 1 1	0 1 0 0	0 0 0 1
2	0 0 1 0	0 0 1 0	1 1 0 1	1 1 1 0	0 1 0 1	0 0 1 1
3	0 0 1 1	0 0 1 1	1 1 0 0	1 1 0 1	0 1 1 0	0 0 1 0
4	0 1 0 0	0 1 0 0	1 0 1 1	1 1 0 0	0 1 1 1	0 1 1 0
5	0 1 0 1	1 0 1 1	1 0 1 0	1 0 1 1	1 0 0 0	0 1 1 1
6	0 1 1 0	1 1 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 1 1 1
7	0 1 1 1	1 1 0 1	1 0 0 0	1 0 0 1	1 0 1 0	1 1 1 0
8	1 0 0 0	1 1 1 0	0 1 1 1	1 0 0 0	1 0 1 1	1 0 1 0
9	1 0 0 1	1 1 1 1	0 1 1 0	0 1 1 1	1 1 0 0	1 0 0 0

Преобразователь кода — комбинационный функциональный узел, предназначенный для изменения кода информации. В ЭВМ и других устройствах используют различные коды информации: двоичные (прямой, обратный и дополнительный коды), десятичные, двоично-десятичные взвешенные (8421, 2421), с избытком 3, «2 из 5», циклический и др.

Табличное описание преобразователей двоично-десятичного кода 8421 в код 2421, обратный и дополнительный коды, код с избытком 3, код Грея представлено в табл. 5.4.

Рассмотрим для примера проектирование преобразователя двоично-десятичного кода 8421 в циклический код Грея, условное графическое обозначение которого пока-

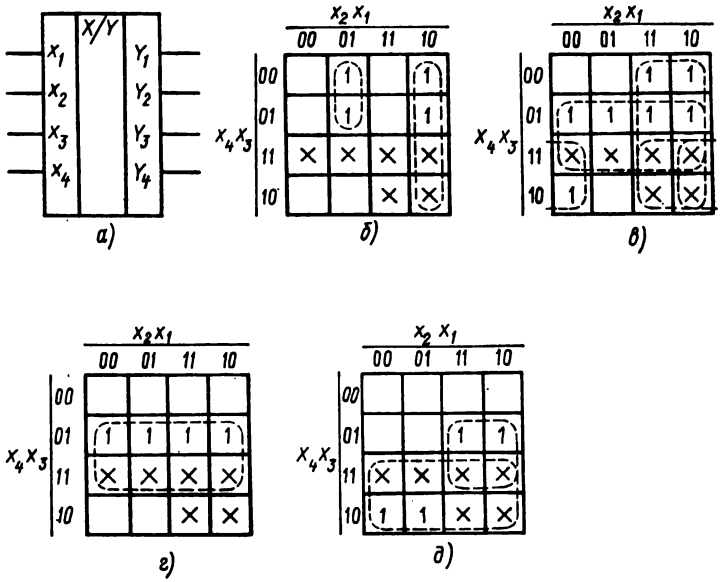


Рис. 5.26. Преобразователь кода 8421 в код Грея

зано на рис. 5.26, *а*. С целью минимизации доопределим значения выходных функций на некоторых избыточных входных наборах, которые показаны на картах Карно знаком × (рис. 5.26). После минимизации выходные функции могут быть записаны в виде системы уравнений:

$$Y_1 = \bar{x}_1 x_2 \vee x_1 \bar{x}_2 \bar{x}_4 \text{ (рис. 5.26, } б\text{)},$$

$$Y_2 = x_2 \vee x_3 \vee \bar{x}_1 x_4 \text{ (рис. 5.26, } в\text{)},$$

$$Y_3 = x_3 \text{ (рис. 5.26, } г\text{)},$$

$$Y_4 = x_4 \vee x_2 x_3 \text{ (рис. 5.26, } д\text{)}. \qquad (5.29)$$

Далее полученная система уравнений реализуется в заданном базисе логических элементов.

Глава 6

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНОСТНЫХ ФУНКЦИОНАЛЬНЫХ УЗЛОВ БИС

Последовательностные функциональные узлы (ПФУ) содержат комбинационную часть схемы и элементы памяти. В отличие от комбинационных схем появление нового элемента — памяти приводит к необходимости введения дополнительной переменной — времени, которая должна учитываться при решении многих задач логического проектирования. Состояния выходов в последовательностных схемах зависят не только от значений входных переменных в данный момент времени, но и от значений внутренних переменных в тот же момент времени. Вопросы математического описания работы последовательностных схем рассмотрены в § 2.4. При синтезе ПФУ (счетчиков, регистров и др.) решаются задачи на уровне абстрактного синтеза (составление автоматного описания, минимизация числа состояний) и структурного синтеза (кодирование состояний, минимизация функций возбуждения и выходов, создание структуры).

§ 6.1. Методика проектирования и особенности работы последовательностных функциональных узлов

Исходными данными для логического проектирования ПФУ являются: описание алгоритма его функционирования, библиотека логических элементов и элементов памяти, требования к электрическим параметрам ПФУ и конструктивно-топологические особенности реализации выбранного элементного базиса. Процесс проектирования ПФУ включает следующие этапы:

1) словесное описание алгоритма функционирования;

2) оценка сложности задачи и решение вопроса о разбиении ПФУ на части (каскады), если это необходимо;

3) переход к формализованному заданию алгоритма функционирования объекта — узла или его части;

4) минимизация состояний объекта проектирования;

5) кодирование внутренних состояний объекта;

6) состояние таблиц переходов (смены состояний), функций возбуждения и функций выхода;

7) минимизация функций возбуждения и выхода;

8) преобразование минимальных функций возбуждения и выхода для рациональной реализации логической схемы в заданном базисе элементов;

9) построение структуры логической схемы;

10) разработка функциональных тестов для проверки работоспособности спроектированной логической схемы;

11) описание и отладка на САПР ФЛП-3000 логической схемы и функциональных тестов.

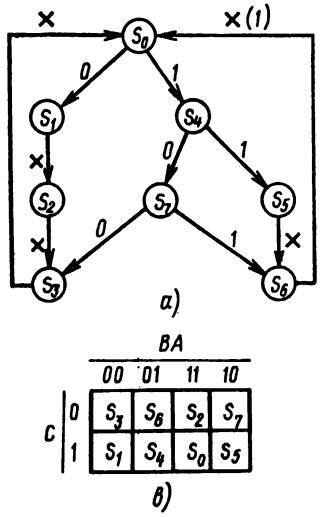
На первом этапе формулируется задача, детализируются внешние входы и требуемые выходы объекта, составляется его условное графическое обозначение. Точно определить сразу все условия задачи очень трудно. Для этого может потребоваться несколько консультаций проектировщика с заказчиком. В ряде случаев возникает необходимость составления алгоритмической модели объекта на языке ОЦИС-РП, моделирования на САПР ФЛП-3000 и получения временных диаграмм входных и выходных сигналов. Анализ полученных временных диаграмм позволяет проектировщику и заказчику уточнить условия задачи и алгоритм функционирования разрабатываемого объекта.

На втором этапе решается вопрос, следует ли проектировать объект как единое целое или его необходимо разбить на части и проектировать их отдельно. Например, если требуется спроектировать счетчик по модулю 64, то его проще построить, соединив последовательно два счетчика по модулю 8. Поэтому при создании счетчика по модулю 64 проектированию подлежит один из его каскадов, а именно счетчик по модулю 8. Вполне очевидно, что спроектировать счетчик по модулю 8 значительно проще, чем счетчик по модулю 64.

На третьем этапе осуществляется переход от словесного алгоритма функционирования объекта или его части к формализованному описанию (автоматному графу, таблицам смены состояний и выходов).

На четвертом этапе минимизация числа состояний проектируемого объекта выполняется в основном с целью сокращения числа элементов памяти. Процесс минимизации числа состояний может осуществляться различными методами: разбиений, импликаций, с помощью правил слияния Колдуэлла и таблиц состояний. В соответствии с правилами слияния Колдуэлла две строки в совмещенной таблице состояний и выходов можно объединить, если совпадают значения желаемых состояний и выходов, указанные в соответствующих ячейках каждой строки.

Различают три основных вида кодирования состояний автомата: случайное, экономичное и противогоночное. При экономичном кодировании выбирают такие значения переменных, характеризующие состояния, чтобы схемное решение ПФУ было проще решений, полученных при случайном кодировании. Критерием качества экономичного кодирования состояний является простота схемной



Состояние	Вид кодирования			
	случайное			экономичное
	1	2	3	
S_0	000	000	000	111
S_1	001	001	001	100
S_2	011	010	011	011
S_3	010	011	010	000
S_4	110	100	100	101
S_5	111	101	101	110
S_6	101	110	111	001
S_7	100	111	110	010
Число ЛЭ И-НЕ, шт	22	18	16	10

б)

Рис. 6.1. Кодирование состояний:
а — исходный граф состояний; *б* — варианты кодирования и оценка их эффективности (по числу ЛЭ); *в* — таблица экономичного кодирования состояний

реализации, т. е. минимальное число логических элементов. Для иллюстрации эффективности экономичного кодирования обратимся к рис. 6.1. Алгоритм функционирования некоторого ПФУ представлен в виде графа состояний (рис. 6.1, *а*). На рис. 6.1, *б* приведены три варианта случайного кодирования состояний и экономичное кодирование одного и того же ПФУ. Результаты различных вариантов кодирования — по числу логических элементов И-НЕ — указаны в нижней строке таблицы (рис. 6.1, *б*). Наилучшим вариантом следует признать экономичное кодирование. Этот вид кодирования является эффективным для синхронных ПФУ, поскольку при наличии цепей синхронизации здесь решается проблема состязаний сигналов в этих схемах.

При экономичном кодировании все логически смежные состояния автомата должны кодироваться таким образом, чтобы их кодовые комбинации отличались друг от друга только одной цифрой. Для определения логически смежных состояний можно использовать два простых правила экономичного кодирования: 1) два состояния, из которых возможны переходы в одно и то же третье состояние, называют логически смежными; 2) два состояния, в которые осуществляются переходы из одного какого-либо состояния, также называют логически смежными. В результате применения этих правил для графа состояний (рис. 6.1, *а*) получим два набора логически смежных состояний: (S_3, S_6), (S_2, S_7), (S_5, S_7) по первому правилу и (S_1, S_4), (S_5, S_7), (S_3, S_6) по второму правилу. На рис. 6.1, *б*, *в* показано, как следует закодировать состояния, чтобы удовлетворить полученным условиям смежности. Если при использовании этих правил невозможно удовлетворить всем условиям смежности, то приоритетным должно быть первое правило.

При кодировании состояний асинхронных ПФУ следует исходить не из простейших реализаций, а из условия отсутствия в проектируемой схеме состязаний сигналов. Из-за задержек сигналов в асинхронных ПФУ наблюдаются их временные состязания, которые могут быть критическими и провести его в другое состояние, не предусмотренное алгоритмом функционирования. Поэтому при проектировании ПФУ критические состязания (гонки сигналов) должны быть устранены на этапе кодирования состояний. Одним из вариантов противогоночного кодирования является *соседнее кодирование*, сущность которого сводится к тому, чтобы при переходе ПФУ из одного состояния в другое кодовые комбинации должны отмечаться только одной цифрой. Это означает, что при различной смене состояний переключаться должен только один элемент памяти (триггер). В противном случае могут возникнуть критические состязания сигналов, которые могут нарушить нормальную работу ПФУ.

Порядок работ, выполняемых на последующих этапах проектирования, рассмотрим на примерах проектирования ПФУ основных типов (счетчиков, регистров и др.).

§ 6.2. Проектирование счетчиков

Счетчик — это функциональный узел последовательностного типа, предназначенный для регистрации числа поступивших на его вход сигналов и деления числа. При работе в счетном режиме счетчик преобразует число или импульсный код, образуемый его каждым разрядом (триггером). При работе в режиме деления частота следования сигналов на выходе каждого разряда в два раза меньше, чем на его входе.

Одним из основных параметров счетчика является модуль счета M — максимальное число импульсов, поступающих на его вход и приведших его в исходное состояние. Модуль счета $M \leqslant 2^n$, где n — число разрядов счетчика.

Счетчики классифицируются по различным признакам: 1) по модулю счета (двоичные, десятичные, с произвольным постоянным модулем, с переменным модулем); 2) по направлению счета (суммирующие — прямого счета, вычитающие — обратного счета и реверсивные — с изменением направления счета); 3) по структурной организации (параллельные, последовательные, параллельно-последовательные); 4) по виду поразрядного переноса (с последовательным сквозным, параллельным и комбинированным переносом).

Проектирование параллельных (синхронных) счетчиков. Счетные импульсы поступают на синхровходы триггера во всех разрядах одновременно (параллельно), что приводит к одновременному переключению всех триггеров. Поэтому в таких счетчиках могут быть временные состязания (гонки) сигналов. Для их исключения используют обычно двухступенчатые JK- и D-триггеры.

Логическую схему счетчика можно получить эвристическим путем или по методике формального синтеза, изложенной в § 6.1. Рассмотрим методику формального синтеза на конкретных примерах.

Пример 6.1. Разработать логическую схему параллельного счетчика прямого счета по модулю 4 на элементах библиотеки БМК (см. приложение).

Решение. 1) составляем граф смены состояний счетчика (рис. 6.2, *а*). Так как $M=4$, то граф имеет четыре вершины и для получения четырех различных состояний достаточно двух триггеров. Проводим взвешенное кодирование состояний счетчика двухразрядным кодом;

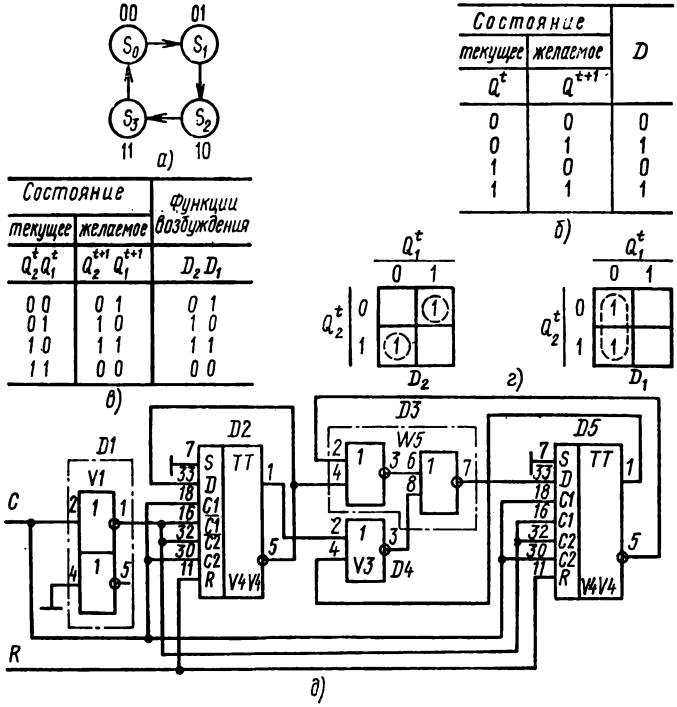
2) в качестве элемента памяти выбираем D-триггер типа V 4 V 4, таблица работоспособности которого приведена на рис. 6.2, *б*;

3) строим таблицу смены состояний и функций возбуждения (рис. 6.2, *в*);

4) проводим минимизацию функций возбуждения триггеров (рис. 6.2, *г*);

5) строим логическую схему счетчика (рис. 6.2, *д*);

6) функциональный тест для проверки работоспособности счетчика должен включать установку счетчика в исходное нулевое состояние и подачу на синхровходы триггеров четырех импульсов;



Состояние		D
текущее	желаемое	
Q^t	Q^{t+1}	
0	0	0
0	1	1
1	0	0
1	1	1

Состояние		Функции возбуждения
текущее	желаемое	
$Q_2^t Q_1^t$	$Q_2^{t+1} Q_1^{t+1}$	$D_2 D_1$
0 0	0 1	0 1
0 1	1 0	1 0
1 0	1 1	1 1
1 1	0 0	0 0

Рис. 6.2. Параллельный счетчик прямого счета по модулю 4 на D-триггерах

7) кодируем логическую схему счетчика и функциональные тесты на языке ОЦИС-РП, проводим логико-временное моделирование. Временная диаграмма работы счетчика приведена в табл. 6.1. Отметим, что на выходе элемента D_3 (контакт 7) обнаружены гонки фронтов, обозначенные на диаграмме символом G (испытания 3 и 8), и 1-риск сбоя длительностью 14,5 нс (испытание 4).

Пример 6.2. Разработать логическую схему параллельного счетчика прямого счета по модулю 4 на JK-триггерах типа V 6 V 4.

Решение: 1) граф смены состояний счетчика аналогичен представленному на рис. 6.2, *а*;

2) таблица работоспособности JK-триггера показана на рис. 6.3, *а*;

Таблица 6.1

Фрагмент временной диаграммы работы параллельного счетчика прямого счета по модулю 4 на D-триггерах

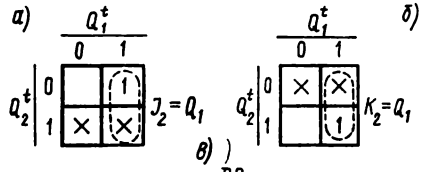
Имя узла схемы	Номер испытания 3	4	5	6	7	8
	Время, нс					
	1 0040 0500	1111122223 092246701351 050555500050	0 . 0	11112 0923470 0550505	0 . 0	11222233 079123503 055050505
R	....	.....	.	.....	.	.....
C	....	111111111111	.	1111111	.	111111111
Q1	1111	111111111>..	.	..<<111	1	11111>...
Q2	....	...<11111111	1	1111111	1	11111>...
NQ1	....	.....<<1111	1	1>>....	.	.<<111111
NQ2	1111	1>.....	.	.....	.	.<1111111
D3(K7)	1>G1	11111>>..<<1	1	11111>.	.	....<<<G.

Состояние		JK
текущее	желаемое	
0	0	0×
0	0	1×
0	1	×1
1	0	×1

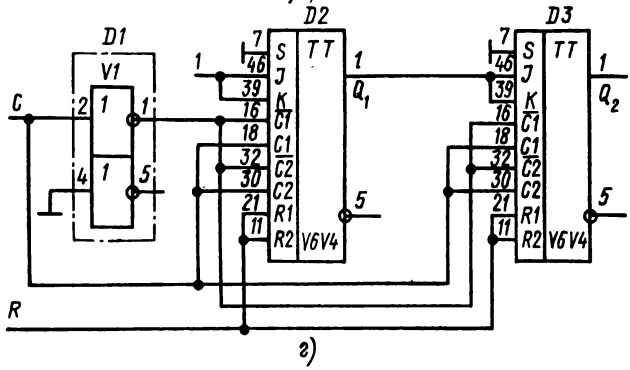
а)

Состояние		Функции возбуждения	
текущее	желаемое		
$Q_2^t Q_1^t$	$Q_2^{t+1} Q_1^{t+1}$	J_2K_2	J_1K_1
0 0	0 1	0×	1×
0 1	1 0	1×	×1
1 0	1 1	×0	1×
1 1	0 0	×1	×1

б)



в)



г)

Рис. 6.3. Параллельный счетчик прямого счета по модулю 4 на JK-триггерах

3) строим таблицу смены состояний и функций возбуждения триггеров (рис. 6.3, *б*) и проводим их минимизацию (рис. 6.3, *в*). В результате минимизации функции возбуждения имеют вид: $J_1 = K_1 = 1$, $J_2 = K_2 = Q_1$;

4) строим логическую схему счетчика (рис. 6.3, *г*);

5) функциональные тесты аналогичны тестам, рассмотренным в примере 6.1;

6) кодируем логическую схему и описываем функциональные тесты на языке ОЦИС-РП, проводим логико-временное моделирование. Временна́я диаграмма работы счетчика на JK-триггерах приведена в табл. 6.2.

Таблица 6.2

Фрагмент временной диаграммы работы параллельного счетчика прямого счета по модулю 4 на JK-триггерах

Имя узла схемы	Номер испытания							
	1	2	3	4	5	6	7	8
	Время, нс							
	11 035912 050000	1111 01347 00500	 0 . 0	11111122 013457936 005055500	 0 . 0	1111 01347 00500	 0 . 0	11222 079356 055000
R	111111	.....	.	.....	.	.....	.	.....
C	.....	11111	.	111111111	.	11111	.	111111
Q1	???>>.	...<1	1	1111111>.	.	...<1	1	111>>.
Q2	???>..	.....	.	...<11111	1	11111	1	111>..
NQ1	?<1111	1>...	.	.....<111	1	1>...	.	.<1111
NQ2	?<1111	11111	1	1>.....	.	.....	.	.<1111

Проектирование параллельных счетчиков прямого счета по модулю 2^n производится так же, как и проектирование счетчиков по модулю $4 = 2^2$. Например, проектирование двух первых разрядов синхронного счетчика прямого счета по модулю 2^3 производится так же, как и проектирование первого и второго разрядов счетчика по модулю 4. Добавление третьего разряда осуществляется так, чтобы не менялись схемы первых двух разрядов. Результаты проектирования на JK-триггерах синхронного счетчика прямого счета по модулю 2^3 получаем в виде

$$J_1 = K_1 = 1,$$

$$J_2 = K_2 = Q_1,$$

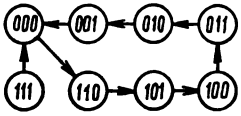
$$J_3 = K_3 = Q_1 Q_2 = J_2 Q_2.$$

Проанализировав эти выражения, делаем вывод, что

уравнения для разрядов 4, 5, ..., n синхронного счетчика по модулю 2^n имеют вид:

$$J_4 = Q_1Q_2Q_3 = J_3Q_3,$$
$$J_5 = K_5 = Q_1Q_2Q_3Q_4 = J_4Q_4,\ldots,$$
$$J_n = Q_1Q_2Q_3\ldots Q_{n-1} = J_{n-1}Q_{n-1}.$$

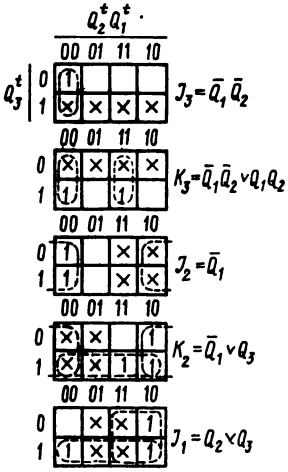
Для проектирования синхронных счетчиков обратного счета используют те же методы, что и для проектирования синхронных счетчиков прямого счета. В результате получаются выражения:



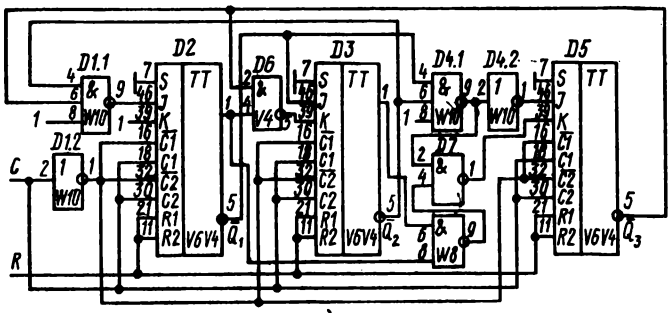
а)

Состояние		Функции возбуждения
текущее	желаемое	
$Q_3^t Q_2^t Q_1^t$	$Q_3^{t+1} Q_2^{t+1} Q_1^{t+1}$	$J_3 K_3 J_2 K_2 J_1 K_1$
0 0 0	1 1 0	1 × 1 × 0 ×
1 1 0	1 0 1	× 0 × 1 1 ×
1 0 1	1 0 0	× 0 0 × × 1
1 0 0	0 1 1	× 1 1 × 1 ×
0 1 1	0 1 0	0 × × 0 × 1
0 1 0	0 0 1	0 × × 1 1 ×
0 0 1	0 0 0	0 × 0 × × 1
1 1 1	0 0 0	× 1 × 1 × 1

б)



в)



г)

Рис. 6.4. Параллельный счетчик обратного счета по модулю 7 на JK-триггерах

$$J_1 = K_1 = 1,\ J_2 = K_2 = \bar{Q}_1,\ J_3 = K_3 = \bar{Q}_1\bar{Q}_2 = J_2\bar{Q}_2,$$
$$J_4 = K_4 = \bar{Q}_1\bar{Q}_2\bar{Q}_3 = J_3\bar{Q}_3 \ldots,\quad J_n = K_n = \bar{Q}_1\bar{Q}_2\bar{Q}_3 \ldots$$
$$\ldots \bar{Q}_{n-1} = J_{n-1}\bar{Q}_{n-1}.$$

Пример 6.3. Разработать логическую схему параллельного счетчика обратного счета по модулю 7 на JK-триггерах.

Решение. Счетчик по модулю 7 (СТ7) имеет семь состояний. Для его реализации необходимо три JK-триггера типа V6V4. На рис. 6.4, *а* изображен граф состояний счетчика. Одно состояние (111) является не рабочим. Для определения функций возбуждения JK-триггеров составляем таблицу состояний и функций возбуждения (рис. 6.4, *б*), а затем проводим минимизацию функций возбуждения с помощью карт Карно (рис. 6.4, *в*). В результате минимизации получаем:

$$J_1 = Q_2 \vee Q_3,\quad K_1 = 1,\quad J_2 = Q_1,\quad K_2 = \overline{Q_1\bar{Q}_3},$$
$$J_3 = \bar{Q}_1\bar{Q}_2,\quad K_3 = \overline{\bar{Q}_1\bar{Q}_2\overline{Q_1Q_2}}. \qquad (6.1)$$

Логическая схема счетчика приведена на рис. 6.4, *г*.

При проектировании параллельного счетчика прямого счета по модулю 7 ветви графа состояний, входящие в замкнутый контур (см. рис. 6.4, *а*), должны иметь проти-

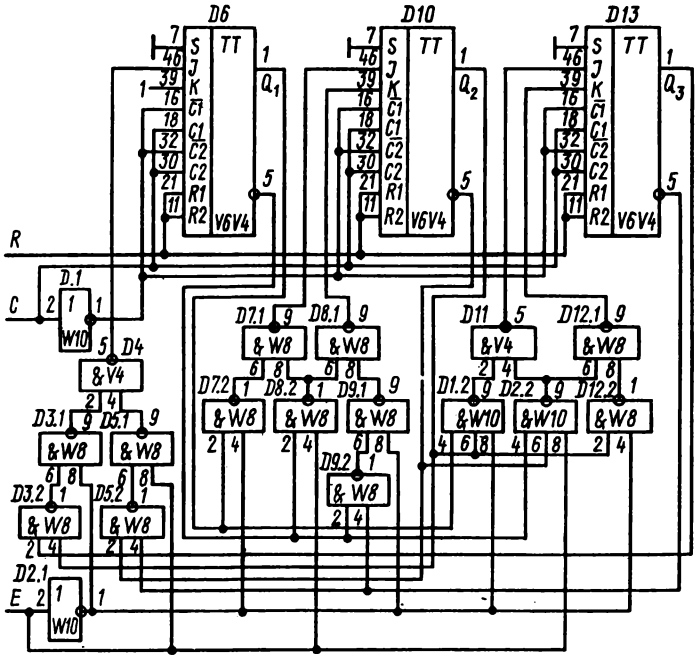


Рис. 6.5. Реверсивный счетчик по модулю 7 на JK-триггерах

Таблица 6.3

Фрагмент временной диаграммы работы параллельного реверсивного счетчика по модулю 7 на JK-триггерах

Имя узла схемы	Номер испытания				
	15	16	17	18	19
	Время, нс				
	222223333 0367893568 0500050055	111 0669135 0055555	1111222 04678156 00050055	. 0 . 0	11222 047347 000500
R	.....	.....	.....	.	.....
E	.....	1111111	11111111	1	111111
C	1111111111	.....	11111111	.	111111
Q3	1>.....	.....	.<111111	1	111111
Q2	1>>.....	.....	.<<11111	1	111>>.
Q1	.....	.....	.....	.	.<1111
D6(K46)	.....<<<1	11>>>..	.....<<1	1	111111
D10(K46)	.....	.<<<111	11111111	1	11>>..
D10(K39)	1111>>>...	.<<<111	11111111	1	11>>..
D13(K46)	.....	...<<<1	1111>>..	.	.....
D13(K39)	11111>>>..	...<<<1	1111>>..	.	.....

воположные направления. В результате проектирования счетчика прямого счета получим следующие функции возбуждения JK-триггеров:

$$K_1 = 1,\ K_2 = Q_1 \vee Q_3 = \overline{\bar{Q}_1 \bar{Q}_3},\ K_3 = Q_2,$$

$$J_1 = \bar{Q}_2 \vee \bar{Q}_3 = \overline{Q_2 Q_3},\ J_2 = Q_1,\ J_3 = Q_1 Q_2. \qquad (6.2)$$

В некоторых случаях необходимо обеспечить работу как в прямом, так и в обратном счете. Уравнения параллельного счетчика обратного счета имеют вид (6.1), а счетчика прямого счета — вид (6.2). Для выбора направления счета обычно используют управляющий сигнал E. Допустим, что когда $E = 1$, то счет ведется в обратном направлении, а когда $E = 0$, — то в прямом. Тогда имеем:

$$J_1 = \bar{E}(\bar{Q}_2 \vee \bar{Q}_3) \vee E(Q_2 \vee Q_3) = \overline{\overline{\bar{E}\overline{Q_2 Q_3}}\,\overline{E\overline{\bar{Q}_2 \bar{Q}_3}}},\ K_1 = 1,$$

$$J_2 = \bar{E} Q_1 \vee E \bar{Q}_1 = \overline{\overline{\bar{E} Q_1}\,\overline{E \bar{Q}_1}},$$

$$K_2 = E(Q_1 \vee Q_3) \vee E \bar{Q}_1 = \overline{\overline{\bar{E}\overline{\bar{Q}_1 \bar{Q}_3}}\,\overline{E \bar{Q}_1}},$$

$$J_3 = \bar{E} Q_1 Q_2 \vee E \bar{Q}_1 \bar{Q}_2 = \overline{\overline{\bar{E} Q_1 Q_2}\,\overline{E \bar{Q}_1 \bar{Q}_2}},$$

$$K_3 = \bar{E} Q_2 \vee E \bar{Q}_1 \bar{Q}_2 = \overline{\overline{\bar{E} Q_2}\,\overline{E \bar{Q}_1 \bar{Q}_2}}.$$

На рис. 6.5 приведена логическая схема параллельного реверсивного счетчика по модулю 7 на JK-триггерах и в табл. 6.3 фрагмент временнóй диаграммы его работы (с 15-го по 19-е испытание).

§ 6.3. Проектирование регистров

Регистр — это последовательностный функциональный узел, служащий для приема, хранения, преобразования и передачи информации. По способу приема и выдачи информации различают следующие группы регистров:

1) с последовательным входом и выходом; при этом информация как на вход, так и с выхода поступает последовательно, бит за битом. Такой регистр называют сдвиговым или последовательным;

2) с последовательным входом и параллельным выходом; регистр загружается последовательно, бит за битом, а считывание проводится одновременно со всех его разрядов;

3) с параллельным входом и последовательным выходом; при этом прием информации осуществляется одновременно сразу во все разряды регистров, а вывод — последовательно, бит за битом, под управлением тактовых импульсов;

4) с параллельным входом и параллельным выходом; при этом информация загружается одновременно, сразу во все триггеры, а когда требуется выводить информацию, то она считывается со всех разрядов регистра также одновременно. Такой тип регистра называется регистром памяти или параллельным регистром;

5) комбинированные с различными способами приема и выдачи информации.

По способу тактирования различают однотактные и многотактные регистры. Однотактные регистры управляются одной последовательностью синхронизирующих сигналов, многотактные — несколькими.

Регистры памяти служат для хранения небольшого объема информации (одного или нескольких байтов). Для записи, хранения и считывания большого объема информации (256 бит и более) используют блоки ОЗУ с адресной организацией записи и считывания. В качестве регистров памяти применяют синхронизируемые уровнем или фронтом D-триггеры (для однофазных сигналов) или RS-триггеры (для парафазных сигналов).

Сдвиговые регистры имеют широкое применение —

для временного хранения данных, для преобразования данных из последовательной формы в параллельную, и наоборот, для сдвига влево или вправо на необходимое число разрядов и др. Они могут работать в качестве счетчиков и генераторов последовательностей. Их используют для построения умножителей и делителей, в центральном процессоре микропроцессорных систем, где выполняются разнообразные функции. Сдвиговый регистр обычно реализуется на D- или RS-триггерах, синхронизируемых фронтом сигнала. Триггеры, управляемые уровнем напряжения, применять нельзя, поскольку, пока действует синхросигнал, передача данных из разряда в

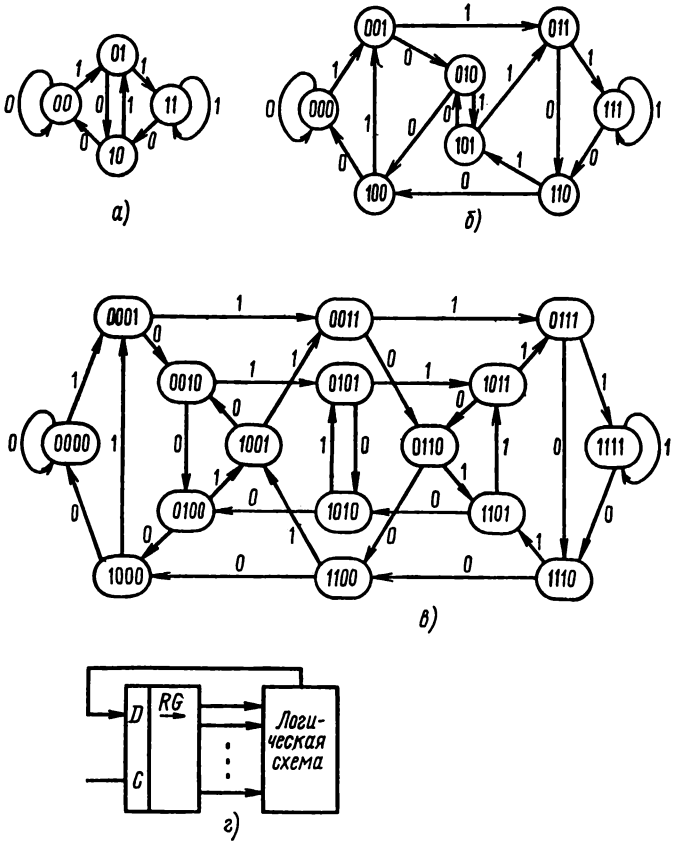


Рис. 6.6. Проектирование счетчиков на основе сдвиговых регистров

разряд проходит безостановочно и триггеры могут переключиться несколько раз, тогда как требуется сдвиг на один разряд.

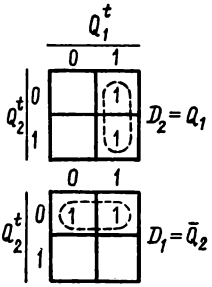
Проектирование сдвиговых регистров. Сдвиговый регистр, состоящий из n последовательно соединенных D-триггеров, имеет следующие функции возбуждения триггеров:

$$D_1 = x,\ D_i = Q_{i-1},\ i = 2,\ 3, \ldots, n. \qquad (6.3)$$

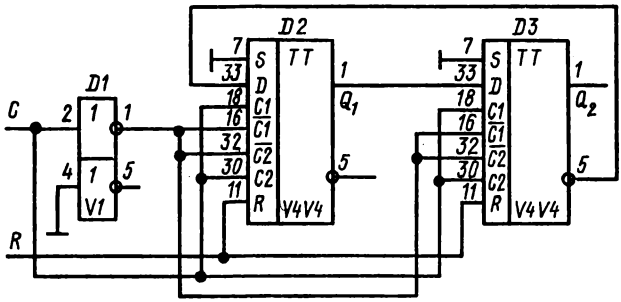
Из выражения (6.3) следует, что информация, хранящаяся в триггере Q_{i-1}, передается в следующем такте в триггер Q_i, т. е. производится сдвиг информации. Входной сигнал x, соответствующий некоторому такту, появляется на выходе сдвигового регистра Q_n через n тактов. Если Q_n — старший разряд, то имеет место сдвиг в

Состояние				Функции возбуждения	
текущее		желаемое			
Q_2^t	Q_1^t	Q_2^{t+1}	Q_1^{t+1}	D_2	D_1
0	0	0	1	0	1
0	1	1	1	1	1
1	1	1	0	1	0
1	0	0	0	0	0

а)



б)



в)

Рис. 6.7. Параллельный счетчик CT4 на D-триггерах, построенный на основе сдвигового регистра

Таблица 6.4

Фрагмент временной диаграммы параллельного счетчика по модулю 4 на D-триггерах, построенного на основе сдвигового регистра

Имя узла схемы	Номер испытания							
	1	2	3	4	5	6	7	8
	Время, нс							
	11 036801 000555	11 025 ... 055	0 . 0	11 0924 0550	0 . 0	22 036 ... 000	0 . 0	122 07035 05500
R	111111	...	.	....	.	...	.	.....
C	.....	111	.	1111	.	111	.	11111
Q2	???>..	...	.	..<1	1	111	1	111>.
Q1	???>>.	.<1	1	1111	1	1>.	.	.....
D3(K5)	?<1111	111	1	1>..	.	...	.	.<111

сторону старших разрядов. Если же Q_n считать младшим разрядом, то происходит сдвиг в сторону младших разрядов.

Используя методику структурного проектирования ПФУ (см. § 6.1), можно спроектировать различные сдвиговые регистры на триггерах того или иного типа.

Использование сдвиговых регистров в качестве счетчиков. Проектирование двух-, трех- и четырехразрядных счетчиков на основе сдвигового регистра осуществляется путем реализации графов состояний, представленных на рис. 6.6, *а—в*. При этом функциональная схема счетчика содержит n-разрядный сдвиговый регистр и комбинационную логическую схему для формирования сигнала возбуждения, подаваемого на триггер младшего разряда регистра (рис. 6.6, *г*).

Пример 6.4. Разработать на основе сдвигового регистра логическую схему параллельного счетчика прямого счета по модулю 4 на D-триггерах.

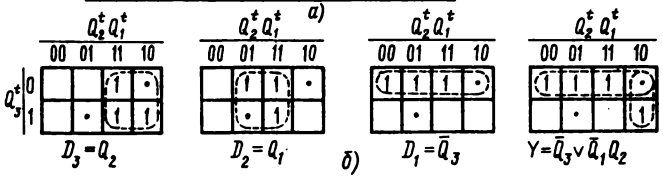
Решение. Для получения четырех различных состояний необходимо использовать два триггера. Алгоритм функционирования проектируемого счетчика должен соответствовать универсальному графу двухразрядного регистра (см. 6.6, *а*). По направлению обхода выбираем на графе контур, включающий смену четырех состояний: 00—01—11—10—00. Составляем таблицу смены состояний и функций возбуждения D-триггеров (рис. 6.7, *а*). Проводим минимизацию функций D_1 и D_2 на картах Карно (рис. 6.7, *б*). В результате минимизации получаем: $D_1 = \bar{Q}_2$, $D_2 = Q_1$. Логическая схема проектируемого счетчика на D-триггерах показана на рис. 6.7, *в*, а временная диаграмма — в табл. 6.4.

Если требуется разработать на основе сдвигового регистра логическую схему параллельного счетчика по модулю 7, то следует воспользоваться универсальным графом трехразрядного регистра сдвига (см. рис. 6.6, *б*). Для этого нужно выбрать по направлению обхода контур, включающий смену семи состояний: 000—001—010—101—011—110—100—000. Далее составляются таблица смены состояний и функция возбуждения для трех триггеров заданного типа, проводится минимизация функций возбуждения и составляется логическая схема счетчика.

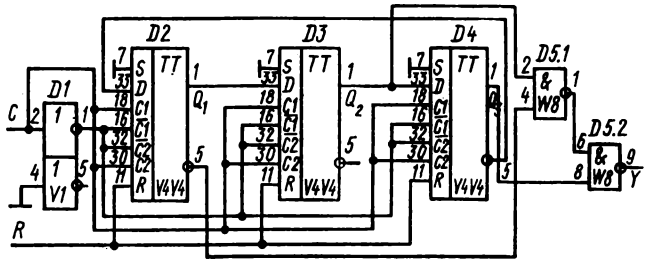
В случае необходимости разработки на основе регистра сдвига параллельного счетчика по модулю 11 следует руководствоваться универсальным графом (рис. 6.6, *в*) и выбрать по направлению обхода контур, включающий

Состояние		Функции возбуждения	Выходная функция
текущее	желаемое		
$Q_3^t Q_2^t Q_1^t$	$Q_3^{t+1} Q_2^{t+1} Q_1^{t+1}$	$D_3 D_2 D_1$	Y
0 0 0	0 0 1	0 0 1	1
0 0 1	0 1 1	0 1 1	1
0 1 1	1 1 1	1 1 1	0
1 1 1	1 1 0	1 1 0	1
1 1 0	1 0 0	1 0 0	0
1 0 0	0 0 0	0 0 0	1

а)



б)



в)

Рис. 6.8. Генератор двоичной последовательности типа 1—1—0—1—1

Таблица 6.5

Фрагмент временной диаграммы генератора двоичной последовательности

Имя узла схемы	Номер испытания							
	1	2	3	4	5	6	7	8
	Время, нс							
	111 0368125 0005000	0 . 0	11 026 ... 050	0 . 0	1112 02561 05000	0 . 0	1112 092483 055550	0 0 . 0
R	1111111	.	...	.	.....	.	.....	.
C	.....	.	111	.	11111	.	111111	.
Q3	???>...	.	...	.	.....	.	..<111	1
Q2	???>>..	.	...	.	.<<11	1	111111	1
Q1	???>>..	.	.<1	1	11111	1	111111	1
Y	????<<1	1	111	1	11>>.	.	....<1	1
D4(K5)	?<11111	1	111	1	11111	1	1>....	.

смену 11 состояний: 0000—0001—0010—0101—1011—0111—1110—1101—1010—0100—1000—0000. Далее определяется функция возбуждения триггеров и составляется логическая схема счетчика.

Использование счетчика на сдвиговом регистре в качестве генераторов двоичных последовательностей. При проектировании генераторов двоичных последовательностей, содержащих до 16 бит, используются соответствующие универсальные графы двух-, трех- и четырехразрядных сдвиговых регистров. Длина двоичной последовательности l зависит от разрядности счетчика. В общем случае для n-разрядного регистра сдвига $l \leqslant 2^n$. Функциональная схема такого генератора должна содержать n-разрядный счетчик на сдвиговом регистре и логическую схему, преобразующую сигналы с выходов триггеров в требуемую двоичную последовательность.

Пример 6.5. Разработать на сдвиговом регистре логическую схему генератора двоичной последовательности 1—1—0—1—0—1.

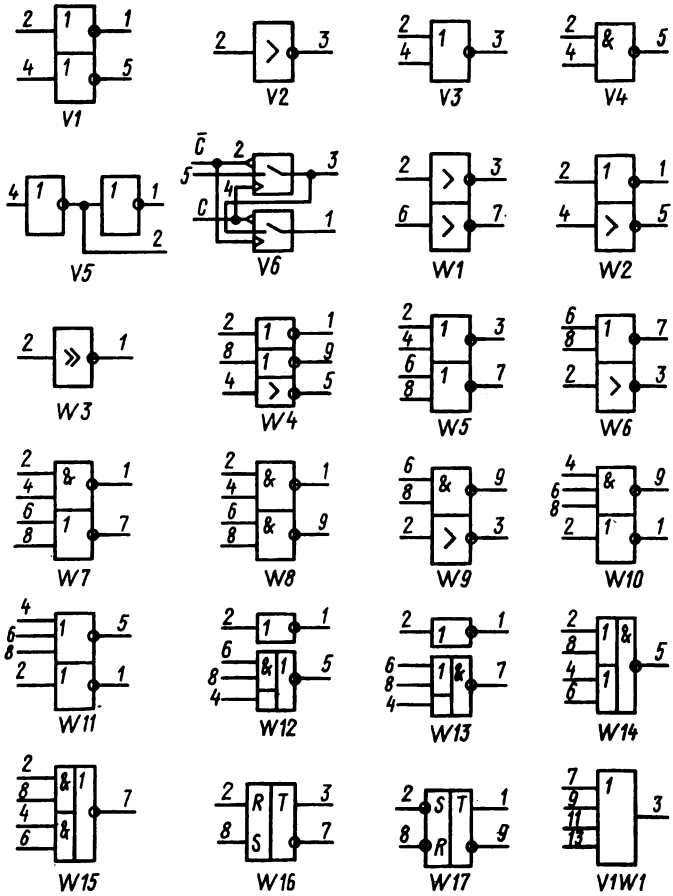
Решение. Так как требуемая последовательность содержит 6 бит, то для схемы генератора необходим трехразрядный сдвиговый регистр. В качестве элемента памяти регистра выбираем D-триггер типа V4V4. Определяем последовательность из шести состояний для трехразрядного сдвигового регистра, полученную из универсального графа (см. рис. 6.6, *б*). Эта последовательность имеет вид 000—001—011—111—110—100—000. Затем составляем таблицу смены состояний с указанием функций возбуждения трех D-триггеров и функции выхода Y, определяющей и требуемую двоичную последовательность (рис. 6.8, *а*). На картах Карно (рис. 6.8, *б*) проводим минимизацию

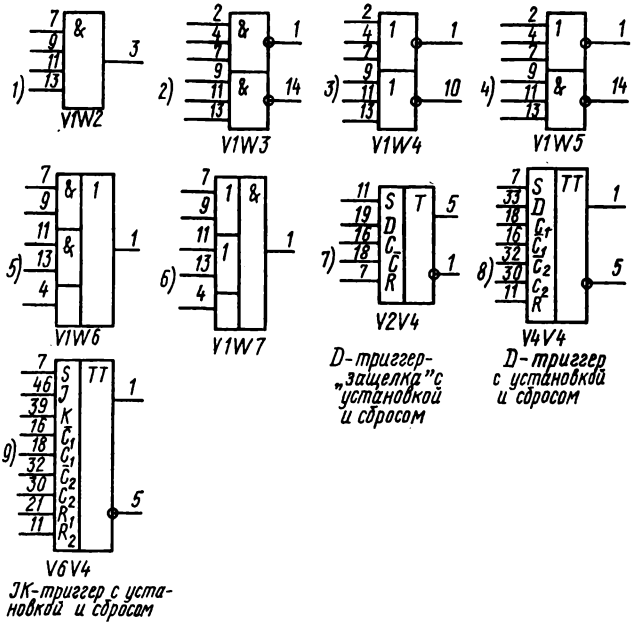
функций D_3, D_2, D_1, Y, получаем функции возбуждения и выхода в виде

$$D_1 = \bar{Q}_3,\ D_2 = Q_1,\ D_3 = Q_2,\ Y = \bar{Q}_3 \vee \bar{Q}_1 Q_2. \qquad (6.4)$$

Далее в соответствии с полученными уравнениями (6.4) синтезируется логическая схема генератора двоичной последовательности (рис. 6.8, *в*). Для проверки работоспособности данного генератора и определения его динамических параметров проводится его логико-временно́е моделирование на САПР ФЛП-3000. Временна́я диаграмма работы генератора типа 1—1—0—1—0—1 приведена в табл. 6.5.

Библиотека цифровых элементов ячеек БМК




1)
V1W2
2)
V1W3
3)
V1W4
4)
V1W5
5)
V1W6
6)
V1W7
7)
V2V4
D-триггер-
„защелка"с
установкой
и сбросом
8)
V4V4
D-триггер
с установкой
и сбросом
9)
V6V4
JK-триггер с уста-
новкой и сбросом

СПИСОК ЛИТЕРАТУРЫ

1. **Алексенко А. Г., Шагурин И. И.** Микросхемотехника/Под ред. И. П. Степаненко. — М.: Радио и связь, 1982. — 418 с.

2. **Автоматизация** схемотехнического проектирования/В. Н. Ильин, В. Т. Фролкин, А. И. Бутков и др.; Под ред. В. Н. Ильина. — М.: Радио и связь, 1987. — 309 с.

3. **Автоматизированное** проектирование СБИС на базовых кристаллах/А. И. Петренко, В. Н. Лешаков, А. Я. Тетельбаум, Б. Л. Шрамченко. — М.: Радио и связь, 1988. — 160 с.

4. **Автоматизированное** проектирование цифровых устройств/С. С. Бадулин, Ю. М. Барнаулов, В. А. Бердышев и др.; Под ред. С. С. Бадулина. — М.: Радио и связь, 1981. — 240 с.

5. **Баталов Б. В., Егоров Ю. Б., Русаков С. Г.** Основы математического моделирования больших интегральных схем на ЭВМ. — М.: Радио и связь, 1982. — 168 с.

6. **Голдсуорт Б.** Проектирование цифровых логических устройств: Пер. с англ./Под ред. Ю. И. Тютчева. — М.: Машиностроение, 1985. — 288 с.

7. **Киносита К., Асада К., Карацу О.** Логическое проектирование СБИС: Пер. с япон. — М.: Мир, 1988. — 309 с.

8. **Савельев А. Я.** Прикладная теория цифровых автоматов. — М.: Высшая школа, 1987. — 272 с.

9. **Угрюмов Е. П.** Проектирование элементов и узлов ЭВМ. — М.: Высшая школа, 1987. — 318 с.

ОГЛАВЛЕНИЕ

Автоматизация проектирования БИС
В шести книгах

Савельев Павел Васильевич,
Коняхин Валерий Вячеславович

Кн. 2

ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БИС

Зав. редакцией *В. И. Трефилов*
Редактор *Е. В. Вязова*
Младшие редакторы *С. А. Пацева, В. В. Пащенкова*
Художник *Ю. Д. Федичкин*
Художественный редактор *Т. М. Скворцова*
Технический редактор *Г. А. Виноградова*
Корректор *В. В. Кожуткина*

ИБ № 8016

Изд. № ЭР-522. Сдано в набор 10.05.89. Подп. в печать 28.11.89. Т-17667. Формат 84×108/32. Бум. тип. №2. Гарнитура литературная. Печать высокая. Объем 8,40 усл. печ. л. 8,61 усл. кр.-отт. 7,73 уч.-изд. л. Тираж 30 000 экз. Зак. № 311. Цена 40 коп.

Издательство «Высшая школа», 101430, Москва, ГСП-4, Неглинная ул., д. 29/14.

Ярославский полиграфкомбинат Госкомпечати СССР. 150014, Ярославль, ул. Свободы, 97.

Автоматизация проектирования БИС. В 6 кн.:
А 22 Практ. пособие. Кн. 2. П. В. Савельев, В. В. Коняхин. Функционально-логическое проектирование БИС/Под ред. Г. Г. Казеннова. — М.: Высш. шк., 1990. — 156 с.: ил.
ISBN 5-06-000093-1

Изложены основы функционально-логического проектирования цифровых БИС. Особое внимание уделено информационным и лингвистическим проблемам автоматизации функционально-логического проектирования БИС, составляющим ядро интегрированной САПР. Рассмотрена методология разработки логического проекта БИС и его аттестации, приведены примеры проектирования различных функциональных узлов БИС.

ББК 32.844.1

$А \frac{2004060000-086}{001(01)-90} 145-89$

6Ф2

Издательство
«Высшая школа»
выпустит в 1991 году
следующие книги:

Фролов В. Н., Львович Я. Е., Меткин Н. П. **Проектирование технологии и микропроцессорного оборудования с применением САПР:** Учебник для студентов радиотехнических специальностей вузов. — 31 л.: — 1 р. 30 к.

Изложены системные принципы проектирования процессов производства радиоэлектронных средств на базе микропроцессорного технологического оборудования. Рассмотрены вопросы построения технического и информационного обеспечения САПР, ориентированного на разработку технологии и микропроцессорного оборудования, что позволяет уяснить закономерности функционирования инвариантных частей САПР и их связи с объектом проектирования. Изложены принципы, методы и алгоритмы математического моделирования и оптимизации технологических процессов. Большое внимание уделено автоматизированной разработке программных средств микропроцессорного оборудования, прикладным аспектам реализации САПР для основных классов технологии радиоэлектронных средств в условиях роботизированного производства.

Глудкин О. П. **Методы и средства испытаний РЭС:** Учебник для студентов вузов, обучающихся по специальностям «Конструирование и технология радиоэлектронных средств», «Конструирование и технология электронных вычислительных средств». — 21 л.: ил. — 1 р.

Рассмотрены вопросы теории и техники испытаний радиоэлектронных и электронных вычислительных средств, подготовки и методики проведения испытаний при климатических, механических, биологических и радиационных воздействиях. Особое внимание уделяется устройствам для испытаний, воспроизводящим внешние воздействия. Описана автоматизированная система испытаний. Обосновывается необходимость проведения испытаний на стадиях исследования и проектирования изделий с целью повышения их качества, а на стадии изготовления — с целью оценки качества.

Учебник будет полезен при выполнении курсовых и дипломных проектов, при изучении специальных дисциплин, определяющих профиль будущих специалистов. Может быть использован аспирантами и инженерно-техническими работниками.

Уважаемые читатели!

По вопросам приобретения литературы просим обращаться в местное отделение Книготорга или книжный магазин по месту жительства.