Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ВИКТОР ЕФИМОВИЧ БАРАНОВ
СОТСКОВ ЮРИЙ ВЛАДИМИРОВИЧ
ШМАКОВ ВЛАДИМИР ЭДУАРДОВИЧ

ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

Учебное пособие

Санкт-Петербург
2019

681. 3 (075.8)
ББК 32. 97я 73
Б241
Баранов В. Е., Сотсков Ю.В., Шмаков В.Э. Дискретная оптимизация: Учебное пособие/ В.Е.Баранов, Ю.В.Сотсков, В.Э.Шмаков. – СПб., 2018. - 165 с.

Пособие соответствует ФГОС ВО по направлениям подготовки 27.03.03 «Системный анализ и управление» и 09.03.02 «Информационные системы и технологии» (уровень бакалавриата).

В учебном пособии рассмотрены основные понятия булевой алгебры. Основное внимание уделено автоматизированной и ручной минимизации логических функций. Показана связь между выражениями логических функций и их техническими аналогами. Отражены основные вопросы организации структурных и информационных связей для цифровых ЭВМ и систем. В пособии приведены все необходимые материалы для синтеза и анализа структур.

Настоящее пособие предназначено студентам Высшей Школы Киберфизических систем и Управления (программы «Системный анализ и управление» и «Информационные системы и технологии») в качестве пособия по выполнению практических заданий и освоению теоретического курса «Дискретная оптимизация».

Ключевые слова: Дискретная оптимизация, цифровые технологии, булева алгебра, синтез, анализ, системный подход, карта Карно, метод Моррeля, структурная оптимизация,

ОГЛАВЛЕНИЕ

Введение

Постановка и решение задачи оптимизации (поиск целевой функции) включает в себя:

1. Условия (для аппаратных и программных составляющих):

- постановка задачи;
- временной и частотный анализ;
- выбор параметров и переменных;
- выбор критериев;
- граничные условия;
- входные и выходные данные;
- функция преобразования;
- внешние факторы;
- и т.д.

2. Конечномерные задачи оптимизации:

- задачи параметрической идентификации нелинейных детерминированных объектов;
- задачи идентификации стохастических объектов;
- задачи экстремального регулирования (вертолет);
- задачи синтеза адаптивных систем управления;
- задачи синтеза стохастических оптимальных систем управления;
- задачи оптимального проектирования.

3. Вычислительные технологии (элементы) для решения вопросов:

- нормализация основных переменных задачи;
- выбор вида критерия оптимальности;
- выбор весовых коэффициентов в критериях метода наименьших квадратов;
- методика перехода с одного критерия оптимальности на другой в процессе решения одной и той же задачи;
- методика реализации среднестепенной аппроксимации минимаксных критериев с целью предотвращения появления «машинных» нулей и построения гладких аппроксимаций исходных негладких функционалов;
- учет ограничений методом замены переменных и с помощью специальных реализаций методом штрафных функций (игры – штрафной круг, биатлон, передача информации и т.д.);
- работа со спецификациями оптимизируемой системы в виде систем неравенств и т.д.

4. Классическая задача оптимизации:

- бесконечная система уравнений (частные производные);
- структурная оптимизация и т.д.

Дискретная оптимизация находит широкое применение в различных областях науки, технике, социологии, образование, медицине и пр. Например: логистика, линейное и нелинейное прораммирование, планирование эксперимента, моделирование и т.д. Будем рассматривать вопросы, в основном связанные с цифровыми технологиями.

Информационные системы и технологии используются в различных областях науки и техники. Например: при решении задач автоматизации производства; в экономике; социологии; психологии; медицине и т.д. В зависимости от поставленных целей строятся сети: ЭВМ, контроллеров (промышленных или для научных исследований) или реализуются сетевые технологии для оптимального прохождения информационных потоков. В данном пособии основное внимание уделяется аппаратной составляющей систем.

Вычислительные машины позволяют решать весьма широкий круг задач, ограниченный способностью человека указывать инструкции для их решения. Подобные инструкции в форме точной и специфической последовательности операторов, подробно определяющих процедуру решения задачи, задаются программой.

На начальном этапе использования современной вычислительной системы мы имеем дело не с самой машиной, а с совокупностями правил, называемых языками программирования, на которых указываются действия, которые должна выполнять ЭВМ. При этом сама вычислительная машина может рассматриваться как аппаратный интерпретатор некоторого конкретного языка, который называется **машинным** языком. Большинство пользователей используют языки высокого уровня (ЯВУ), которые с помощью программ-трансляторов (компиляторов или интерпретаторов) преобразуются в программы на машинном языке.

Таким образом, вычислительная машина состоит не только из вычислительной машины, представляющей собой комплекс оборудования, но и из программ, включая те, которые транслируют программы пользователей с любого из имеющихся языков в программы на машинном языке. То есть речь идет об аппаратной и программной частях ЭВМ.

Исторический обзор.

Механические средства для счета и вычисления были известны еще в далеком прошлом. Одним из древних средств подобного рода являются счеты, которые сохранились до наших дней в качестве простого практического приспособления для вычислений во многих частях света, в особенности на Востоке. (Разновидность счетов - абак - использовалась, видимо, древними египтянами и была знакома китайцам еще в VI в. до нашей эры). При этом

операции умножения и деления выполняются с помощью последовательностей операций сложения и вычитания.

Хотя в ранних устройствах большее внимание уделялось необходимости механизации арифметических операций, хранение промежуточных результатов было также важным. Большинство устройств, подобных счетам, хранили лишь простой текущий результат. В качестве памяти другого типа использовался любой писчий материал, например глиняные дощечки, а позднее бумага.

Первые предвестники управления последовательностями операций появились в областях далеких от вычислительной техники. Например, в ткацком станке Жаккарда (1801г.) использовались перфорированные (пробитые) карты для управления командами переплетения нитей в ткани.

Чарльз Бэббидж (1792-1871) был одним из первых, кто сформулировал идею создания универсальной вычислительной машины. Первым побудительным мотивом была невысокая надежность вычислений. Любопытство привело его к тому, что он обнаружил много ошибок в астрономомических таблицах. Бэббидж детально разработал проект универсальной вычислительной машины, но отсутствие средств не позволило ему реализовать проект.

Перепись населения в США в 1890 г. была революционной с точки зрения обработки ее результатов. Доктор Герман Холлерит применил перфорированные карты и простые машины для обработки данных переписи в 1890 г. С тех пор машины с перфорированными картами получили широкое распространение в деловой и административной сферах.

Первая треть двадцатого века ознаменовалась последовательным развитием и внедрением многих вычислительных устройств. Значительный вклад в это внес математик Алан Тьюринг, опубликовавший в 1937 году описание универсальной гипотетической вычислительной машины (схемы вычислений). Идея привлекла внимание многих ученых, но практического смысла в создание такого рода машины не было никого - она работала бы недопустимо медленно.

С 40-х годов 20-го века началось интенсивное развитие аппаратной и программной частей средств вычислительной техники (ВТ). Аппаратную часть ВТ можно условно разделить на пять поколений, связанных с техническим развитием ее компонентов.

В 1944 году Говард Айкен и группа исследователей из IBM построили электрическую вычислительную машину на релейных логических элементах. Чуть позднее в 1946 году Дж.П.Эккерт и Дж.В.Мочли разработали электронную вычислительную машину (ЭВМ) ENIAC на электронных лампах. Эти машины были предназначены для выполнения научных расчетов. Серийное производство ламповых ЭВМ ENIAC1 ассоциируют с термином "первое поколение". Примерно в это же время в СССР (1949-1951г.г.) были разработаны и реализованы под руководством академика С.А.Лебедева в АН

УССР (Киев) первые ламповые ЭВМ МЭСМ (малая электронная счетная машина).

Появление транзисторов (изобретены в 1948 г.) привело к созданию ЭВМ второго поколения. По сравнению с электронной лампой полупроводниковый транзистор является более эффективным элементом, так как: не требует нагрева источника электронов, имеет более продолжительный период жизни и высокую надежность, затраты на его производство существенно меньше. 1960 год считают годом широкого внедрения и использования ЭВМ второго поколения, используемых в областях: общего назначения; для экономических задач; задач управления производственными процессами.

Третье и четвертое поколения технологии ЭВМ (относящиеся примерно к 1964 и 1970 г.г.) отличаются возрастающим применением методов интегрального изготовления с целью производства большей части ЭВМ в едином автоматическом непрерывном процессе без использования ручного труда. Третье поколение ЭВМ характеризуется широким применением операционных систем (ОС), а четвертое решением новых классов задач: имитационное моделирование; планирование; многопараметрическая оптимизация и т.д.

Пятое поколение ЭВМ связывают с понятием - "искусственного интеллекта". Ориентировочно можно говорить об их появлении в 1980-х годах. В этих машинах значительно увеличивается интеграция программной и аппаратной частей между собой, а также появляются возможности общения с внешней средой.

Успехи в разработке оборудования сопровождались достижениями в программировании. Все языки программирования можно разделить на: машинные, Ассемблеры и языки высокого уровня (ЯВУ). Они реализуются, как правило, в системе, которая может быть более или менее сложной. Цель системы заключается в том, чтобы обеспечить быстрое написание и выполнение программ. При этом, затраты времени на подготовку программ прямо пропорциональны их стоимости. Достижения в области автоматического программирования и операционных систем за последние 40 лет позволили увеличить скорость написания программ в десятки-сотни раз. Это дает возможность пользователю не только писать программы быстрее, но и позволяет решать их с меньшими затратами.

Программное обеспечение вычислительной системы предназначено для упрощения процедуры подготовки пользователями задания для ЭВМ и облегчения процесса его прогона и отладки. На машинном уровне (в машинных кодах - машинном языке) ЭВМ выполняет команды, составленные из цифр. Язык Ассемблера (его разработку приписывают Г.Хопперу) представляет собой набор мнемонических обозначений, соответствующих машинному языку. Используется в основном для операций ввода/вывода информации. Он позволяет не только упростить чтение/запись и написание

программ, но и создавать дополнительные языковые средства, повышающие качество пользования вычислительной машины. Эти дополнительные средства обеспечиваются операционной системой. Для обычного пользователя не имеет значения, являются ли эти дополнительные команды частью аппаратных средств или они относятся к программному обеспечению. Важно предоставить ему более мощную систему программирования высокого уровня.

Основные аппаратные средства обеспечивают выполнение последовательности команд, находящихся в памяти ЭВМ. Задача пользователя и системы заключается в том, чтобы ввести программу в память и начать ее выполнение с нужной команды. При этом сами команды ввода должны находиться в ЭВМ.

По сравнению с машинным языком или Ассемблером языки высокого уровня (ЯВУ) значительно упрощают программирование и позволяют уменьшить количество ошибок в программах. Программа, написанная на ЯВУ для одной машины, с небольшими изменениями или вообще без изменений может быть использована на других. Однако программы на языках высокого уровня выполняются медленнее и занимают больше памяти, чем программы на машинных языках или Ассемблере.

Первым языком высокого уровня был Фортран, разработанный в 1954 году Джоном Бэкусом и примененный в 1956 году. Основное назначение - решение математических и научных задач. Существует очень большое количество ЯВУ. В данном курсе не предусматривается их разбор. Остановимся на их общих особенностях.

Языки высокого уровня являются средством, позволяющим вычислять заданную функцию или решать некоторую задачу. Команды, написанные на ЯВУ, должны быть преобразованы в машинные коды с помощью вычислительной системы, на которой реализована программа. Программа преобразования называется транслятором, а программа, написанная на ЯВУ, исходной программой. Результатом трансляции исходной программы будет программа на машинном языке - объектная программа. Трансляторы подразделяются на компиляторы и интерпретаторы. Первые транслируют всю программу целиком (требуют больше памяти и более производительны), а вторые - команды исходной программы по одной.

Развитие элементной базы ЭВМ стимулировало разработку более эффективных систем программирования, предназначенных для решения различного рода задач. Общая тенденция создания мощных баз программного обеспечения заключается в поисках возможностей использования разнородных программных продуктов в едином целом.

1. Теоретические основы вычислительной техники.
1.1. Представление информации в ЭВМ.
1.1.1. Системы счисления.

Система счисления (СС) – это способ представления любого числа с помощью принятого набора символов (цифр, букв, …).

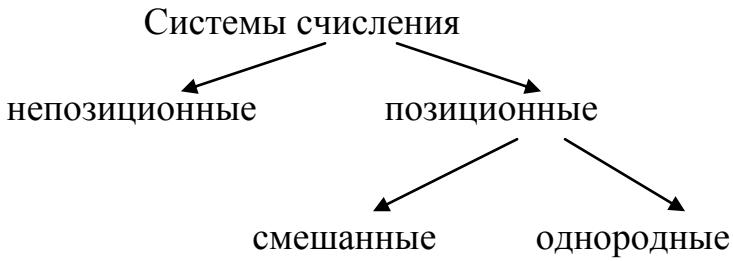
Представление чисел в различных системах счисления.

В различных странах и в различное время исторически сложились и использовались следующие СС: римская; 5 (Африка); 12 (1фут=12дюймов, 12 месяцев…); 20 (ацтеки, майя); 60 (Вавилон – время, углы, …); 2 (Австралия, Полинезия); 10 (Индия → арабы → Европа).

Выбор системы счисления основывается в каждом конкретном случае на ряде критериев. Наиболее распространёнными из них являются:

- эффективность вычислений (обеспечивает простые алгоритмы);
- наглядность (определяется физическими свойствами субъекта);
- реализуемость (в данном случае – техническая).

Системы счисления классифицируются следующим образом:



Различные СС связаны между собой, например: римская и десятичная.

Римская СС	10-я СС
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Пример. Представить число MDCCCXII в десятичной СС.
1000(M) + 500(D) + 100(C) + 100(C) + 100(C) + 10(X) + 1(I) + 1(I) = 1812

Пример. Представить число 1988 в римской СС.
1988 = 1000+900+80+8 = 1000+(1000-100)+50+10+10+10+5+1+1+1=
=MCMLXXXVIII

MDCCCXII – 1812

XXX – 30
XL – 40
LXXX – 80
XC – 90
CCC – 300
CD – 400
DC – 600
CM – 900

Позиционная СС (ПСС) определяется конфигурацией и местом (позицией) каждого элемента, а характеризуется: основанием (числом символов n) и алфавитом.

Позиционная СС может быть представлена в виде полинома
EMBED Equation.3 ,
где n – основание СС, a_i – любой символ из алфавита СС

Примеры ПСС (см. табл. 1.1.).

Таблица 1.1.

СС	основание	алфавит
десятичная	10	0 1 2 3 4 5 6 7 8 9
двоичная	2	0 1
восьмеричная	8	0 1 2 3 4 5 6 7
шестнадцатеричная	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Однородной СС является система, у которой в каждом разряде одинаковое число цифр. С теоретической точки зрения оптимальной ЭВМ является та, у которой основание системы счисления – е (ближайшее число 3), а с практической используется основание кратное 2 (это определено существующей технологической базой: триггеры – 2 состояния).

Задание 1. Представить число 1539 в римской СС.

Задание 2. Представить число MMDCXLVII в 10-ной СС.

Задание 3. Представить число $15.394_{(10)}$ полиномом.

Задание 4. Представить число 11010100,001 полиномом.

1.1.2. Методы перевода чисел из одной позиционной системы счисления в другую.

$A_{(n)}$=>$A_{(k)}$, где A – число, n и k – основания двух СС.

Алгоритм перевода при n<k:

- представить $A_{(n)}$ полиномом основания n,
- все коэффициенты, основание n-ичной СС и все степени основания полинома записать в k-ичной СС,
- выполнив все предусмотренные действия получить $A_{(k)}$.

Пример. Перевести число 1011,01 из двоичной СС в десятичную СС.

Здесь n = 2, k = 10, т. е. n<k.

$$1011{,}01_{(2)} = 1\cdot 10^{11} + 0\cdot 10^{10} + 1\cdot 10^{1} + 1\cdot 10^{0} + 0\cdot 10^{-1} + 1\cdot 10^{-10} =$$

$$= 1\cdot 2^{3} + 0\cdot 2^{2} + 1\cdot 2^{1} + 1\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} = 8 + 2 + 1 + 0{,}25 = 11{,}25_{(10)}$$

Задание 5. Перевести число $10001{,}011_{(2)}$ в десятичную СС.

Задание 6. Перевести число $0{,}10011_{(2)}$ в десятичную СС.

Алгоритм перевода при n>k. Преобразуются отдельно и по-разному целая $A_{ц(n)}$ и дробная $A_{д(n)}$ части числа $A_{(n)}$.

$A_{(n)} = A_{ц(n)} + A_{д(n)}$.

Преобразование $A_{ц(n)}$=>$A_{ц(k)}$ осуществляется последовательным делением $A_{ц(n)}$ на k, выделением остатков от деления и составлением из них числа $A_{ц(k)}$.

Пример. Перевести число 37,75 из десятичной СС в двоичную СС. Здесь n=10, k=2, т. е. n>k.

$A_{(10)}=A_{ц(10)}+A_{д(10)}=37_{(10)}+0{,}75_{(10)}$. Нужно $A_{ц(10)}$=>$A_{ц(2)}$ и $A_{д(10)}$=>$A_{д(2)}$.

37	|2					=> $A_{ц(2)} = a_5a_4a_3a_2a_1a_0 = 100101_{(2)}$
36	18	|2				
1	18	9	|2			
a_0	0	8	4	|2		
	a_1	1	4	2	|2	
		a_2	0	2	1	
			a_3	0	a_5	
				a_4		

	x0,	75
		2
a_{-1}=1,		50
		2
a_{-2}=1,		00
		2
a_{-3}=0		00

=> $A_{д(2)}=0,a_{-1}a_{-2}a_{-3}\ldots=0,11_{(2)}$

В итоге: $37{,}75_{(10)}=100101{,}11_{(2)}$

Проверка: $1\cdot 2^{5} + 0\cdot 2^{4} + 0\cdot 2^{3} + 1\cdot 2^{2} + 0\cdot 2^{1} + 1\cdot 2^{0} + 1\cdot 2^{-1} + 1\cdot 2^{-2} = 37{,}75_{(10)}$

Задание 7. Перевести число $29{,}8125_{(10)}$ в двоичную СС.

Задание 8. Перевести число $100{,}3_{(10)}$ в двоичную СС.

Задание 9. Перевести число $364_{(7)}$ в пятеричную СС через десятичную СС.

Перевод чисел в СС с основанием кратным двум.

Пусть $n=2^l$, а k=2. Тогда каждая цифра n-ичной СС представляется *l* – разрядным двоичным числом.

Пример. Перевести число $472,54_{(8)}$ в двоичную и шестнадцатеричную СС.

Сначала выполним перевод в двоичную СС, а далее в шестнадцатеричную, где $n = 16 = 2^4$, а l = 4.

$472,54_{(8)}$= 1 0011 1010 , 1011 = 1 3 A , $B_{(16)}$.

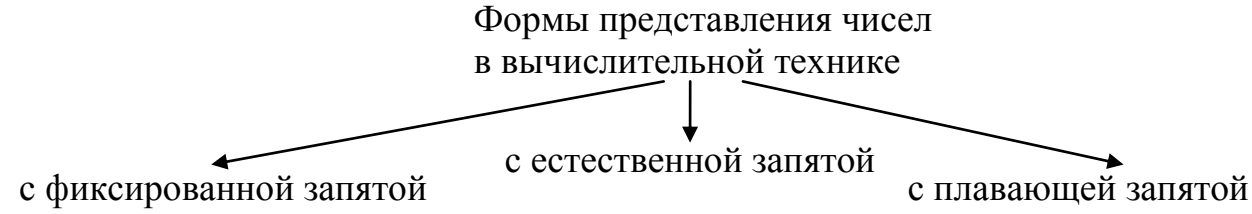
Задание 10. Перевести число $3103,12_{(4)}$ в двоичную СС.

Задание 11. Перевести число $1011010101,11_{(2)}$ в восьмеричную СС и шестнадцатеричную СС.

Задание 12. Перевести число $F0A_{(16)}$ в четвертичную СС.

Задание 13. Перевести число $27,59375_{(10)}$ в шестнадцатеричную СС.

1.1.3. Формы представления чисел.



Представление чисел в форме с фиксированной запятой (ФЗ)

Формат представления чисел с запятой, фиксированной перед старшим разрядом:

,	знак	2^{-1}	2^{-2}	2^{-3}		$2^{-(n-2)}$	$2^{-(n-1)}$
	0	1	2	3		n-2	n-1

<= веса разрядов

<= номера разрядов

↑ n – разрядная сетка ЭВМ

В этом формате диапазон представляемых чисел $2^{-(n-1)} \le |A| \le 1 - 2^{-(n-1)}$

Формат представления чисел с запятой, фиксированной после младшего разряда:

знак	2^{n-2}	2^{n-3}	2^{n-4}		2^1	2^0
n-1	n-2	n-3	n-4		1	0

В этом формате диапазон представляемых чисел $1 \le |A| \le 2^{n-1} - 1$.

Представление знаков: “+” <==> 0; “-“ <==> 1.

Пример. Представить число $-0,0225_{(10)}$ в форме с ФЗ в 16-разрядной сетке ЭВМ.

$-0,0225_{(10)}=-0,000001011100001_{(2)}$.

,	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	1

Формат представления чисел в форме с естественной запятой.

(с произвольным расположением запятой в пределах n-разрядной сетки ЭВМ)

знак	2^{l-1}	2^{l-2}		2^1	2^0	,	2^{-1}		$2^{-(m-1)}$	2^{-m}
	целая часть числа (*l* разрядов)						дробная часть числа (m разрядов)			
n – разрядная сетка ЭВМ (n=*l*+m+1)										

Задание 14. Представить число $1141_{(10)}$ в форме с ФЗ в 16-разрядной сетке.

Задание 15. Представить число $-0{,}4_{(10)}$ в форме с ФЗ в 8-разрядной сетке.

Задание 16. Представить число $-88{,}8125_{(10)}$ в форме с естественной запятой в 16-разрядной сетке.

Представление чисел в форме с плавающей запятой (ПЗ).

Число представляется в виде: $\boldsymbol{A = \pm S^{\pm P} \cdot M}$, где A – исходное число, M – мантисса (/M/<1), S^P – характеристика числа A, S – основание СС, P – порядок.

знак порядка				знак числа		
	разряды P				разряды M	
l разрядов				m разрядов		

При таком представлении чисел необходимо проведение их нормализации. Число A называется нормализированным, если мантисса удовлетворяет условию $\boldsymbol{1/S \le |M| \le 1 - 2^{-(m-1)}}$, т. е. если старший разряд мантиссы отличен от нуля.

Пример. Представить число $-27{,}375_{(10)}$ в форме с ПЗ в 16-разрядной сетке ЭВМ.

$-27{,}375_{(10)} = -11011{,}011_{(2)}$. Проведя нормализацию, получим $\boldsymbol{-0{,}11011011 \cdot 10^{+101}}$. Пусть под порядок отведено 4 разряда. Тогда содержимое разрядной сетки будет:

0	0	1	0	1	1	1	1	0	1	1	0	1	1	0	0

Задание 17. Представить число $59{,}625_{(10)}$ в форме с ПЗ в 16-разрядной сетке.

Задание 18. Представить число $-0{,}0625_{(10)}$ в форме с ПЗ в 8-разрядной сетке.

Задание 19. Представить число $-6{,}025_{(10)}$ в форме с ПЗ в 32-разрядной сетке.

Задание 20. Представить число $0{,}1E_{(16)}$ в форме с ПЗ в 16-разрядной сетке.

Представление чисел в форме с ПЗ и смещённым порядком.

Смещённый порядок $P_{см} = (\pm P) + N$. Здесь $\pm P$ - истинный порядок, а $N=2^k$, где число k – число двоичных разрядов отведённое для представления $\pm P$ (или $P_{см}$). Всегда $P_{см}>0$.

Короткий (32 разряда) формат представления чисел с ПЗ и смещённым порядком в ЭВМ

Знак числа	$P_{см}$	мантисса					
0 β_1 3	4 β_2 7	8 α_1 11	12 α_2 15	16 α_3 19	20 α_4 23	24 α_5 27	28 α_6 31

Здесь $\beta_1, \beta_2, \alpha_1 \div \alpha_6$ - шестнадцатеричные разряды. Следовательно, содержимое разрядной сетки может быть представлено числом $\beta_1\beta_2\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$. При таком представлении нормализация проводится для шестнадцатеричного числа. При определении смещённого порядка число N в шестнадцатеричной СС будет равно $40_{(16)}$.

В ЭВМ существуют также длинный (64 двоичных разряда) и расширенный (128 двоичных разряда) форматы. В них расширяется только поле отводящееся под мантиссу.

Пример. Представить число $-0{,}01_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЭВМ.

Переведём число в шестнадцатеричную СС $-0{,}01_{(10)} \approx -0{,}028F5C3_{(16)}$. Проведя нормализацию этого числа, получим мантиссу $(-0{,}28F5C3)_{(16)}$, а порядок $(-1)_{(16)}$. Определим смещённый порядок.

$$P_{см} = N + (\pm P) = 40_{(16)} + (-1)_{(16)} = 1000000_{(2)} - 1_{(2)} = 0111111_{(2)}.$$

Расположим число в 32-х разрядной сетке

знак числа	$P_{см}$		M					
1	0 1 1	1 1 1 1	0 0 1 0	1 0 0 0	1 1 1 1	0 1 0 1	1 1 0 0	0 0 1 1
B		F	2	8	F	5	C	3

Следовательно в памяти ЭВМ число $-0{,}01_{(10)}$ будет представлено шестнадцатеричным словом BF28F5C3.

Задание 21. Представить число $-43{,}375_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЭВМ. Содержимое разрядной сетки представить 16-ричным словом.

Задание 22. Представить число $17{,}1_{(10)}$ в форме с ПЗ и смещённым порядком в длинном формате ЭВМ.

Задание 23. Представить число $-0{,}0625_{(10)}$ в форме с ПЗ и смещённым порядком в коротком формате ЭВМ. Содержимое разрядной сетки представить 16-ричным словом.

Задание 24. Содержимое 32-х разрядной сетки ЭВМ представлено словом $C3180000_{(16)}$. Определить: какое десятичное число записано в разрядной сетке?

1.1.4. Представление алфавитно-цифровой информации.

Используются различные системы соответствия символов (цифр, букв, математических и служебных знаков) их двоичному представлению. Это ДКОИ – двоичный код для обработки информации (8 разрядное представление символа), КОИ-8 – код обмена информацией (8 разрядов), КОИ-7 – код обмена информацией (7 разрядов).

байт	байт		байт
символ	символ		символ

алфавитно-цифровая информация

Соответствующие таблицы кодирования символов даны в [1].

Пример. Представить запись «СМ-4» в ДКОИ.

байт	байт	байт	байт
11000011	11010100	01100000	11110100
С	М	-	4

Система представления данных состоящих только из десятичных цифр использует двоично-десятичное кодирование. Соответствие десятичных цифр их 2-10-м кодам дано в таблице 1.2. Не используемые в таблице 1.2 комбинации (1010 – 1111) применяются для представления значков чисел и зоны в соответствии с табл. 1.3.

Таблица 1.2.

десятичная цифра	2-10 код
0	0000
1	0001
2	0010
…	…
9	1001

Таблица 1.3.

	ДКОИ	КОИ-8
+	1100	1010
-	1101	1011
зона	1111	1100

Существуют два специальных формата для многоразрядных десятичных чисел:

зонный формат

зона	цифра	зона	цифра		знак	цифра
байт		байт			байт	

упакованный формат

цифра	цифра		цифра	знак
байт			байт	

В упакованном формате число байтов должно быть целым.

Пример. Представить десятичное число -5901 в зонном и упакованном форматах ДКОИ.

1111	0101	1111	1001	1111	0000	1101	0001
зона	5	зона	9	зона	0	знак	1

число в зонном формате

0000 (*)	0101 5	1001 9	0000 0	0001 1	1101 знак

число в упакованном формате
(*) – дополнение до целого числа байтов

Задание 25. Представить запись «A/B-I» в ДКОИ и КОИ-8 используя таблицы в [1].

Задание 26. Представить последовательность десятичных чисел -17+0-10 в зонном формате ДКОИ.

Задание 27. Представить последовательность десятичных чисел +10-187-4 в упакованном формате КОИ-8.

1.2. Арифметические основы вычислительной техники.

Так как двоичная СС и десятичная СС принадлежат к одному классу позиционных аддитивных СС с естественным порядком весов, то правила выполнения арифметических операций над числами в этих СС одинаковы.

Пример. Выполнить четыре арифметических операции над числами $A=1001_{(2)}$ и $B=11_{(2)}$.

$+1001$	-1001	$\times 1001$	$-1001 \mid 11$
11	11	11	$11 \quad 11$
1100	0110	1001	011
		1001	11
		11011	00

1.2.1. Прямой, обратный и дополнительный коды чисел.

Коды целых и дробных чисел, представленных в форме с ФЗ, формируются одинаково. Вопрос о том, какое число закодировано – целое или дробное, решается по формату с которым работает ЭВМ.

Прямой код представляет собой запись самого числа с соответствующим знаком. Знак «+» кодируется 0, а знак «–» кодируется 1 и записывается крайним слева (для наглядности будем выделять знаковый разряд рамкой).

Обратный код положительного числа совпадает с его прямым кодом. Обратный код отрицательного числа формируется из прямого путём инвертирования цифр во всех разрядах за исключением знакового.

Дополнительный код положительного числа совпадает с его прямым кодом. Дополнительный код отрицательного числа формируется добавлением единицы к младшему разряду обратного кода этого числа.

Задание 28. Представить число $-0{,}111_{(2)}$ в прямом, обратном и дополнительных кодах.

Задание 29. Представить число $-10000_{(2)}$ в прямом, обратном и дополнительных кодах.

1.2.2. Сложение и вычитание двоичных чисел.

Сложение и вычитание чисел представленных в форме с ФЗ.

Сложение и вычитание чисел А и В в форме с ФЗ сводится в ЭВМ к суммированию обратных или дополнительных кодов этих чисел с учётом знаков.

При суммировании дополнительных кодов чисел А и В сумма S получается в дополнительном коде: $A_{доп}+B_{доп}=S_{доп}$.

При суммировании обратных кодов чисел А и В сумма S получается в обратном коде: $A_{обр}+B_{обр}=S_{обр}$ всегда, за исключением случая, когда B<0. В этом случае, если есть перенос из знакового разряда, нужна коррекция результата суммирования S’ путём добавления 1 к его младшему разряду, т. е. при B<0: $A_{обр}+B_{обр}=S'+1=S_{обр}$. Для этого используется перенос из знакового разряда.

Пример. Выполнить суммирование чисел $A=\pm 9_{(10)}=\pm 1001_{(2)} uB=\pm 3_{(10)}=\pm 0011_{(2)}$ в обратном и дополнительном кодах при всех возможных комбинациях знаков.

Представим результаты вычислений таблицей 1.4. и в каждой графе проведём проверку результата переводом его в прямой код $S_{пр}$.

Таблица 1.4.

A=9 $(A\geq 0)$ B=3 $(B\geq 0)$	A=-9 $(A<0)$ B=3 $(B\geq 0)$	A=9 $(A\geq 0)$ B=-3 (B<0)	A=-9 (A<0) B=-3 (B<0)
$+A_{обр}$= 0 1001 $B_{обр}$= 0 0011	$+A_{обр}$= 1 0110 $B_{обр}$= 0 0011	$+A_{обр}$= 0 1001 $B_{обр}$= 1 1100	$+A_{обр}$= 1 0110 $B_{обр}$= 1 1100
$S_{обр}$= 0 1100	$S_{обр}$= 1 1001	S’= 0 0101 +1	S’= 1 0010 +1
		$S_{обр}$= 0 0110	$S_{обр}$= 1 0011
$S_{пр}=S_{обр}=+12_{(10)}$	$S_{пр}$= 1 0110=$-6_{(10)}$	$S_{пр}=S_{обр}=+6_{(10)}$	$S_{пр}$= 1 1100=-12
$+A_{доп}$= 0 1001 $B_{доп}$= 0 0011	$+A_{доп}$= 1 0111 $B_{доп}$= 0 0011	$+A_{доп}$= 0 1001 $B_{доп}$= 1 1101	$+A_{доп}$= 1 0111 $B_{доп}$= 1 1101
$S_{доп}$= 0 1100	$S_{доп}$= 1 1010	$S_{доп}$= 0 0110	$S_{доп}$= 1 0100
$S_{пр}=S_{доп}=+12_{(10)}$	$S_{пр}$= 1 0110=$-6_{(10)}$	$S_{пр}=S_{доп}=+6_{(10)}$	$S_{пр}$= 1 1100=-12

Задание 30. Выполнить суммирование чисел -7 и 5 в обратном и дополнительном кодах.

Задание 31. Выполнить суммирование чисел 14 и -10 в обратном и дополнительных кодах.

Задание 32. Выполнить суммирование чисел -8 и -6 в обратном и дополнительных кодах.

При выполнении операции суммирования кодов чисел, представленных в форме с ФЗ, возможно переполнение разрядной сетки ЭВМ. Существуют следующие признаки переполнения:

- одинаковые знаковые разряды слагаемых отличаются от знака результата;

- в процессе суммирования значение переносов из старшего и из знакового разрядов не совпадают.

Пример. Определить факт переполнения разрядной сетки ЭВМ при суммировании чисел $A = \pm 9_{(10)} = \pm 1001_{(2)}$ и $B = \pm 10_{(10)} = \pm 1010_{(2)}$, представленных в обратном и дополнительном кодах при всех возможных комбинациях знаков.

Переполнение возможно лишь при равенстве знаков суммируемых чисел поэтому рассмотрим лишь пары чисел A>0, B>0 и A<0, B<0.

A=+9 (A≥0) B=+10 (B≥0)	A=-9 (A<0) B=-10 (B<0)	A=-9 (A<0) B=-10 (B<0)
$_{+}A_{обр}=A_{доп}=$ 0 1001 $B_{обр}=B_{доп}=$ 0 1010	$_{+}A_{обр}=$ 1 0110 $B_{обр}=$ 1 0101	$_{+}A_{доп}=$ 1 0111 $B_{доп}=$ 1 0110
S= 1 0011	S'= 0 1011 +1 $S_{обр}=$ 0 1100	$S_{доп}=$ 0 1101

В обоих случаях произошло переполнение, выявляемое по любому из приведённых выше признаков. Например, знак результата в каждом случае отличается от знаковых разрядов слагаемых, а значения переносов не совпадают.

Задание 33. Определить, возникнет ли переполнение разрядной сетки при суммировании чисел $7_{(10)}$ и $11_{(10)}$.

Задание 34. По какому признаку можно определить факт переполнения разрядной сетки при суммировании чисел $-13_{(10)}$ и $-8_{(10)}$.

Сложение и вычитание чисел, представленных в форме с ПЗ.

Выполнение операций сложения и вычитания двоичных чисел $A = M_A \cdot 2^{P_A}$ и $B = M_B \cdot 2^{P_B}$ в форме с ПЗ аналогично соответствующим операциям над числами в форме с ФЗ, если выполняется условие $P_A=P_B$. Если $P_A \neq P_B$, то необходимо:

- провести предварительное выравнивание порядков, что приводит к денормализации одной из мантисс;

- провести нормализацию результата операции, если мантисса результата $M_S \geq 1$ или $M_S<0{,}5$. При $M_S \geq 1$ нужна нормализация вправо, а при $M_S<0{,}5$ – нормализация влево.

Признаком необходимости нормализации вправо является переполнение поля разрядной сетки, отводимое под M_S. Это возможно лишь при суммировании чисел с одинаковыми знаками.

Признаком необходимости нормализации влево является совпадение цифр в знаковом разряде и старшем разряде мантиссы. Это возможно лишь при суммировании чисел с разными знаками, то есть при представлении мантисс в обратном или дополнительных кодах.

Пример. Выполнить суммирование двоичных чисел $A = +0{,}1011 \cdot 10^{+100} = 11_{(10)}$ и $B = +0{,}11 \cdot 10^{+10} = 3_{(10)}$.

В данном случае $P_A \neq P_B$, следовательно, проведём предварительное выравнивание порядков.

$A = +0{,}1011 \cdot 10^{+100}$, $B = +0{,}0011 \cdot 10^{+100}$.

Так как A>0 и B>0, то выполним суммирование в прямом коде (мантиссы суммируются в прямом коде)

	порядок	знак	мантисса
$_+$A	100	0	1011
B	100	0	0011
S	100	0	1110

Так как $0{,}5<M_S<1$, то нормализация результата не требуется и получим

$S = +0{,}1110 \cdot 10^{+100} = 14_{(16)}$.

Пример. Выполнить суммирование двоичных чисел $A = +0{,}111 \cdot 10^{+100} = 14_{(10)}$ и $B = \pm 0{,}1011 \cdot 10^{+100} = \pm 11_{(10)}$ с представлением мантиссы шестью разрядами.

Так как $P_A=P_B$, то выравнивание порядков не требуется. Для случая B>0 нахождение суммы S проведём, используя прямой код.

	порядок	знак	мантисса
$_+$A	100	0	111000
B	100	0	101100
	100	1	100100

Так как возникло переполнение поля разрядной сетки, отведённого под мантиссу, проведём нормализацию результата вправо на один разряд, соответственно увеличив значение порядка на единицу, т. е.

100	1	100100
101	0	110010

В итоге получим результат $S = +0{,}11001 \cdot 10^{+101} = 25_{(10)}$.

Для случая B<0 нахождение суммы S проведём используя дополнительный код (мантиссы суммируются в дополнительном коде).

	порядок	знак	мантисса в доп. коде
$_+$A	100	0	111000
B	100	1	010100
S'	100	0	001100

Так как возникло совпадение цифр в знаковом разряде и старшем разряде мантиссы, то проведём нормализацию результата влево на два разряда, соответственно уменьшив значение порядка на два, т. е.

S'	100	0	⟵	001100
S	010	0		110000

В итоге получим результат $S = +0{,}11 \cdot 10^{+10} = 3_{(10)}$.

Задание 35. Выполнить суммирование двоичных чисел $+0{,}1011 \cdot 10^{+11} = +5{,}5_{(10)}$ и $+0{,}11 \cdot 10^{+10} = +3_{(10)}$.

Задание 36. Выполнить суммирование двоичных чисел $+1101{,}11$ и $-1000{,}01$ в форме с ПЗ.

Задание 37. Выполнить суммирование двоичных чисел $-0{,}111 \cdot 10^{+11}$ и $-0{,}1001 \cdot 10^{+100}$.

1.2.3. Сложение и вычитание чисел в двоично-десятичных кодах (Д-кодах).

Таблица 1.5.

0	0000
1	0001
2	0010
…	….
9	1001

Соответствие десятичных цифр их двоично-десятичным кодам дано таблицей.

Такое кодирование применяется при записи десятичных чисел в зонном и упакованном формате.

Сложение чисел в Д-кодах сводится к суммированию тетрад, соответствующих их разрядам, по правилам двоичной арифметики и коррекции тетрад суммы. Для тетрад суммы, значение которых превышают 1001, необходимо организовать перенос в следующую тетраду и выполнить коррекцию. Коррекция представляет собой вычитание 1010, что соответствует сложению тетрады с 0110 без переноса (т. к. 0110 есть дополнительный код числа 1010).

Пример. Определить сумму чисел $A=753_{(10)}$ и $B=439_{(10)}$ в двоично-десятичном коде.

Представим числа A и B в Д-коде и выполним суммирование:

+A	=	0111	0101	0011
B	=	0100	0011	1001
S’	=	1011	1000	1100
выполним коррекцию		+0110		+0110
S	= 1	0001	1001	0010

,что соответствует $1192_{(10)}$.

Вычисление чисел в Д-кодах выполняется так же, как и вычисление двоичных чисел, т. е. вычисляемое представляется в дополнительном коде, а операция вычисления сводится к сложению. Дополнительный код числа, представленного Д-кодом, формируется дополнением каждой тетрады до 1001 и добавлением к младшей тетраде единицы.

Пример. Определить разность чисел A=$753_{(10)}$ и B=$439_{(10)}$ в двоично-десятичном коде.

Представим число B в дополнительном коде

¯1001	¯1001	¯1001	дополнение каждой тетрады до 1001
0100	0011	1001	
0101	0110	0000	добавление к младшей тетраде единицы
		+1	
$B_{доп}$= 0101	0110	0001	

Теперь выполним операцию S=A-B=A+$B_{доп}$.

+A	0111	0101	0011
$B_{доп}$	0101	0110	0001
S’	1100	1011	0100
коррекция	+0110	+0110	
S	0011	0001	0100

, что соответствует $314_{(10)}$.

Задание 38. Выполнить сложение чисел $280_{(10)}$ и $157_{(10)}$ в Д-коде.

Задание 39. Определить разность чисел $1092_{(10)}$ и $956_{(10)}$ в Д-коде.

1.2.4. Умножение и деление двоичных чисел.

Ограничимся рассмотрением выполнения операций умножения и деления над двоичными числами в прямом коде с учётом их знаков. Эти операции выполняются с многократным применением операций сложения и сдвига и могут быть реализованы схемой показанной на рисунке 1.1.

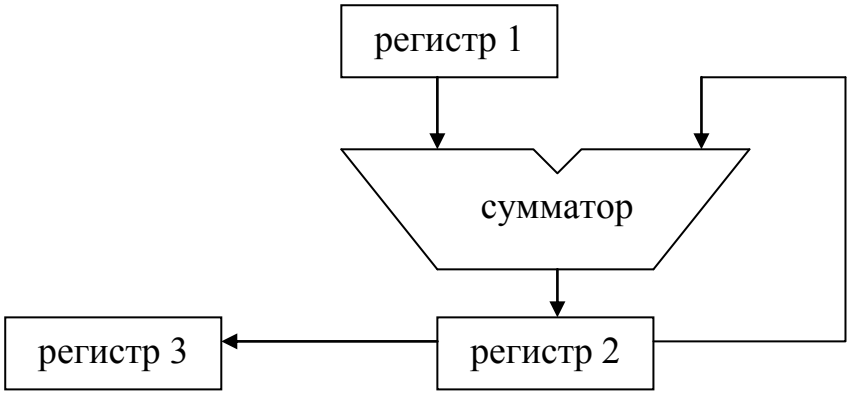


Рис. 1.1.

Операция умножения чисел A и B выполняется в несколько тактов. Множимое A записывается в Рег.1, а множитель B в Рег.3. Рег.2 предварительно устанавливается в 0. На каждом такте анализируется содержимое старшего (левого) разряда Рег.3. Если в нём находится 1-ца, то производится общий сдвиг влево содержимого Рег.3 и Рег.2 (с переносом между ними) и суммирование содержимого регистров Рег.1 и Рег.2 с

занесением результата в Рег.2. Если в старшем разряде Рег.3 находится 0, то производится лишь сдвиг влево содержимого Рег.3 и Рег.2. Число тактов равно числу разрядов множителя В. В результате произведение будет располагаться в Рег.3 и Рег.2.

Код знака результата операции умножения получаем суммированием по mod2 кодов знаков сомножителей (логическая функция сложения по mod2 показана далее).

Пример. Выполнить умножение чисел $A=11_{(10)}=1011_{(2)}$ и $B=13_{(10)}=1101_{(2)}$.

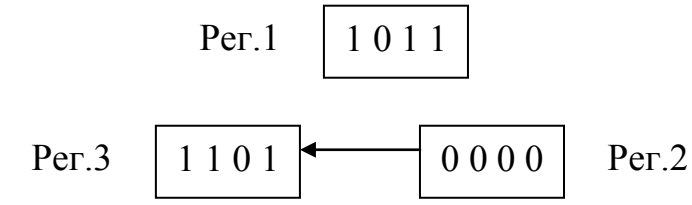
Для наглядности последующих действий предварительно умножим числа «в столбиках».

Теперь рассмотрим выполнение операции умножения схемой, представленной на рис. 1.

Операция проводится в 4 такта (число В четырёхразрядное).

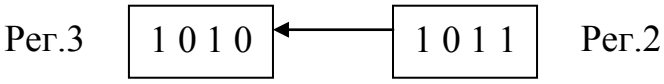
Начальное содержимое регистров:

$$\begin{array}{r} {}_{x}1011 \\ 1101 \\ \hline 1011 \\ 1011 \\ 1011 \\ \hline 10001111 \end{array}$$

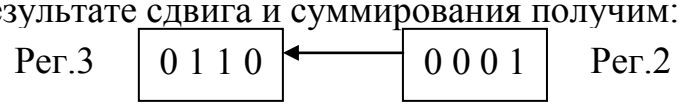


Такт 1. Так как в старшем разряде Рег.3 находится 1-ца, то производится сдвиг содержимого Рег.3 и Рег.2 и суммирование содержимого Рег.1 и Рег.2 с записью результата в Рег.2.

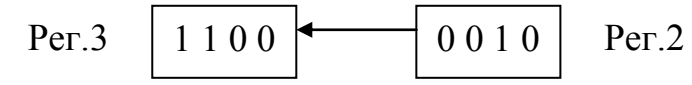
Получим:



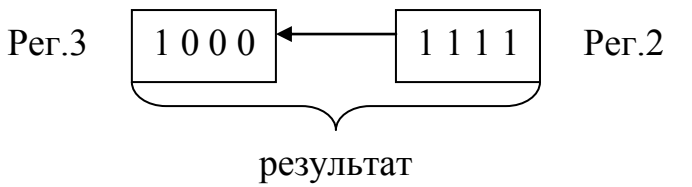
Такт 2. В старшем разряде Рег.3 опять 1-ца, следовательно, в результате сдвига и суммирования получим:



Такт 3. В старшем разряде Рег.3 находится 0, поэтому производится только сдвиг.



Такт 4. В старшем разряде Рег.3 опять 1-ца. Выполняются сдвиг и суммирование.



Проверим правильность выполнения умножения.

$10001111_{(2)}=143_{(10)}$, т. е. $\boldsymbol{13_{(10)} \cdot 11_{(10)} = 143_{(10)}}$.

Задание 40. Выполнить операцию умножения чисел $14_{(10)}$ и $-9_{(10)}$ реализуя алгоритм работы схемы рис. 1.

Задание 41. Выполнить операцию умножения чисел $-19_{(10)}$ и $-7_{(10)}$, реализуя алгоритм работы схемы рис. 1.

Операция деления чисел A и B выполняется схемой рис. 1 также за несколько тактов. Делимое A располагается одновременно в Рег.3 и Рег.2, а делитель B в Рег.1. На каждом такте сравнивается содержимое Рег.3 и Рег.1. Если число в Рег.3 больше числа в Рег.1, то производится:

- вычитание из содержимого Рег.3 содержимого Рег.1 с записью результата в Рег.3;
- общий сдвиг влево содержимого Рег.3 и Рег.2 (с переносом между ними);
- запись 1-цы в младший разряд Рег.2.

Если число в Рег.3 меньше числа в Рег.1, то производится лишь сдвиг влево содержимого Рег.3 и Рег.2. Итоговый результат операции деления будет располагаться в Рег.2.

Код знака результата операции деления получается аналогично коду знака результата операции умножения.

Пример. Выполнить деление числа $143_{(10)}=10001111_{(2)}$ на число $11_{(10)}=1011_{(2)}$.

Для наглядности последующих действий предварительно разделим числа «в столбик».

```
 1 0 0 0 1 1 1 1 | 1 0 1 1
 1 0 1 1         | 1 1 0 1
-------
 0 1 1 0 1
   1 0 1 1
---------
 - 0 1 0 1 1
     1 0 1 1
   -------
   0 0 0 0
```

Теперь рассмотрим выполнение операции деления схемой, представленной на рис. 1.1. Начальное содержимое регистров:

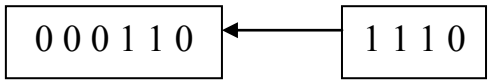
Рег.1 | 1 0 1 1 |

Рег.3 | 0 0 1 0 0 0 | ← | 1 1 1 1 | Рег.2

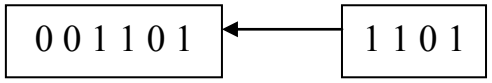
Такт 1. Так как число в Рег.3 меньше числа в Рег.1 то производится общий сдвиг влево содержимого Рег.3 и Рег.2. Получим

Рег.3 | 0 1 0 0 0 1 | ← | 1 1 1 0 | Рег.2

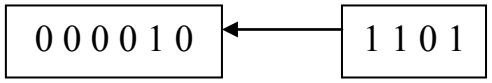
Такт 2. Теперь число в Рег.3 больше числа в Рег.1, поэтому производится вычитание из содержимого Рег.3 содержимого Рег.1 с записью результата в Рег.3.



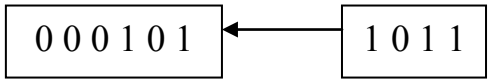
а затем выполняется общий сдвиг всего содержимого Рег.3 и Рег.2 и запись 1-цы в младший разряд Рег.2. Получим



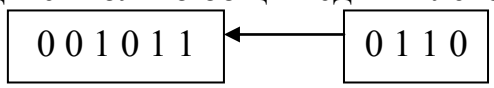
Такт 3. Так как сейчас число в Рег.3 больше числа в Рег.1, то действия предыдущего такта повторяются. После вычитания получим



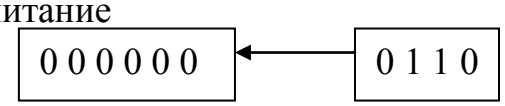
а после сдвига влево в Рег.3 и Рег.2 и записи 1-цы в младший разряд Рег.2 будем иметь



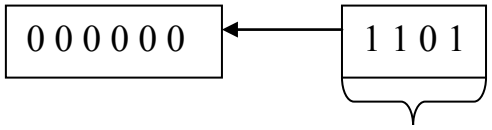
Такт 4. В этом случае число в Рег.3 стало меньше числа в Рег.1, поэтому производится только общий сдвиг влево Рег.3 и Рег.2



Такт 5. Теперь опять число в Рег.3 больше числа в Рег.1. Поэтому выполняется вычитание



сдвиг и запись 1-цы в младший разряд Рег.2



Проверим правильность выполнения деления:

$1101_{(2)}=13_{(10)}$, т. е. $143_{(10)}:11_{(10)}=13_{(10)}$.

Замечание. Чтобы реализовать операцию вычитания в сумматоре (рис.1) нужно заменить её операцией сложения с числом в дополнительном коде, т. е. к содержимому Рег.3 прибавлять содержимое Рег.1 взятое в дополнительном коде. В рассмотренном примере вычиталось число 1011, следовательно нужно прибавлять число 0101. (Проверить самостоятельно).

Задание 42. Выполнить операцию деления чисел $126_{(10)}$ и $14_{(10)}$, реализуя алгоритм работы схемы рис.1.1.

Задание 43. Выполнить операцию деления чисел $123_{(10)}$ и $-7_{(10)}$, реализуя алгоритм работы схемы рис.1.1.

Замечание. При выполнении операции умножения и деления чисел в форме с ПЗ необходимо выполнить:

- умножение или деление мантисс исходных чисел (по алгоритмам рассмотренным выше);
- определение порядка результата (получается соответственно сложением или вычитанием порядков исходных чисел с учётом их знаков);
- нормализацию результата с соответствующей коррекцией порядка.

1.3. Логические основы вычислительной техники.

1.3.1. Переключательные (булевы, логические) функции.

I. Основные понятия и определения.

Булевой функцией называется $f(x_1, x_2, \dots x_n)$, аргументы которой $x_i (i = \overline{1,n})$ и сама функция принимают значение из множества $E^2=\{0,1\}$. Они могут быть заданы таблицей. Полная таблица функции (см. табл.1.6.) от n переменных имеет 2^n строк и n+1 столбцов. Упорядоченная таблица называется таблицей истинности. Другая форма – булевы выражения.

Таблица 1.6.

x_1	x_2	…	x_n	$f(x_1 \dots x_n)$
0	0	…	0	$f(0,0,\dots 0) \in \{0,1\}$
1	0	…	0	
…	…	…	…	…
1	1	…	1	$f(1,1,\dots 1) \in \{0,1\}$

Теорема: число различных переключательных функций n-аргументов равно 2^{2^n}.

Определение: Две переключательных функции называются равными, если на всех наборах аргументов они принимают одинаковые значения.

Определение: Переключательная функция f называется существенно зависимой от x_i, если выполняется неравенство: $f(x_1,\dots, x_{i-1}, 0, x_{i+1},\dots, X_n) \neq f(x_1,\dots, x_{i-1}, 1, x_{i+1},\dots, X_n)$. Не все переключательные f существенно зависимы.

Определение: Переключательная функция f независящая ни от одного аргумента и равная 0 (1) на всех наборах называется нулевой (единичной).

II. Элементарные логические функции.

Элементарными называются функции 1 или 2-х аргументов.

Логическая функция 1-ой переменной (см. табл. 1.7.):

Таблица 1.7.

x	0	1
$f_1(x)$	0	0
$f_2(x)$	0	1
$f_3(x)$	1	0
$f_4(x)$	1	1

$f_1(x) \equiv 0$ - нулевая функция;

$f_2(x) = x$ - функция повторения аргумента

$f_3(x) = \overline{x}$ - функция инверсии аргумента

$f_4(x) \equiv 1$ - единичная функция

Логические функции 2-х переменных (см. табл. 1.8.).

Таблица 1.8.

функции	аргументы x_1 0011 x_2 0101	обозначение функции	название функции и комментарии
$f_0(x_1,x_2)$	0 0 0 0	0	Нулевая
$f_1(x_1,x_2)$	0 0 0 1	$x_1 \wedge x_2$	Конъюнкция
$f_2(x_1,x_2)$	0 0 1 0	$x_1 \Delta x_2$	запрет по x_2 $(x_1 \overline{x_2})$
$f_3(x_1,x_2)$	0 0 1 1	x_1	повторение x_1
$f_4(x_1,x_2)$	0 1 0 0	$x_2 \Delta x_1$	запрет по x_1
$f_5(x_1,x_2)$	0 1 0 1	x_2	повторение x_2
$f_6(x_1,x_2)$	0 1 1 0	$x_1 \oplus x_2$	сумма по mod2
$f_7(x_1,x_2)$	0 1 1 1	$x_1 \vee x_2$	Дизъюнкция
$f_8(x_1,x_2)$	1 0 0 0	$x_1 \downarrow x_2$	стрелка Пирса $(\overline{x_1 \vee x_2})$
$f_9(x_1,x_2)$	1 0 0 1	$x_1 \sim x_2$	эквивалентность (равнозначность)
$f_{10}(x_1,x_2)$	1 0 1 0	$\overline{x_2}$	отрицание x_2
$f_{11}(x_1,x_2)$	1 0 1 1	$x_2 \rightarrow x_1$	импликация от x_2 к x_1 $(x_1 \vee \overline{x_2})$
$f_{12}(x_1,x_2)$	1 1 0 0	$\overline{x_1}$	отрицание x_1
$f_{13}(x_1,x_2)$	1 1 0 1	$x_1 \rightarrow x_2$	импликация от x_1 к x_2 $(x_2 \vee \overline{x_1})$
$f_{14}(x_1,x_2)$	1 1 1 0	$x_1 \backslash x_2$	штрих Шеффера $(\overline{x_1 \wedge x_2})$
$f_{15}(x_1,x_2)$	1 1 1 1	1	Единичная

III. Системы элементарных логических функций.

Определение: Набор элементарных логических функций, используемый для записи любых логических выражений, называется системой логических функций (СЛФ).

Определение: Набор элементарных логических функций достаточный для записи любых логических выражений называется функционально полной СЛФ (ФП СЛФ).

Определение: Если исключение любой элементарной логической функции из ФП СЛФ приводит к функциональной неполноте, то исходная ФП СЛФ называется безызбыточной. Иначе ФП СЛФ называется избыточной.

Теорема о функциональной полноте СЛФ (Постникова, Яблонского).

СЛФ называется функционально полной в том и только том случае, если она включает хотя бы одну функцию, не принадлежащую к 5 важнейшим замкнутым классам:

1) – класс функций сохраняющих 0,
2) – класс функций сохраняющих 1,
3) – класс самодвойственных функций,
4) – класс монотонных функций,
5) – класс линейных функций.

Рассмотрим классы:

1) $f(0, 0, \dots 0) = 0$;
2) $f(1, 1, \dots 1) = 1$;
3) $f(x_1, x_2, ..., x_n) = \overline{f(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)}$;
4) Пусть $x_{1i}=1$, а $x_{1j}=0$. Тогда $x_{1i}>x_{1j}$. Функция $f(x_1, x_2, \dots x_n)$ является монотонной, если для всех случаев, когда $x_{1i} \le x_{1j}; x_{2i} \le x_{2j}; ...; x_{ni} \le x_{nj}$. $\overline{i, j = 0, 2^n - 1}$, во всех наборах выполняется неравенство $f(x_{1i}, x_{2i}, ..., x_{ni}) \le f(x_{1j}, x_{2j}, ..., x_{nj})$.

Например:

x_1	$0(x_{10})$	$1(x_{11})$	$1(x_{12})$
x_2	$0(x_{20})$	$0(x_{21})$	$1(x_{22})$

видно, что $(x_{10}) \le (x_{11}) \le (x_{12})$ и $(x_{20}) \le (x_{21}) \le (x_{22})$, то выполняется: $f(x_{10}, x_{20}) \le f(x_{11}, x_{21}) \le f(x_{12}, x_{22})$;

5) Если $f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_x$, где $a_i \in \{0,1\}, i \in \overline{0,n}$, то $f(x_1, x_2, \dots, x_n)$ – называется линейной логической функцией

Примеры функционально полных СЛФ:

Таблица 1.9.

Наименование	Обозначение	Сохр. «0»	Сохр. «1»	самодвойственные	монотонные	линейные
и	$\wedge$	+	+	-	+	-
или	$\vee$	+	+	-	+	-
инверсия	$\neg$	-	-	+	-	+
штрих Шеффера	\	-	-	-	-	-

Из таблицы 1.9. на основании теоремы о ФП СЛФ:

1) $x_1 \wedge x_2$ $x_1 \vee x_2$ $\bar{x}$ - избыточная СЛФ, но наиболее распространённая (каноническая СЛФ)

2) $x_1 \wedge x_2$ $\quad \bar{x}$

3) $x_1 \vee x_2$ $\quad \bar{x}$

4) $x_1 \backslash x_2$

} безызбыточные СЛФ, есть другие безызбыточные ФП СЛФ

Технический аналог булевой функции – комбинационная схема.

Вывод: для синтеза любой комбинационной схемы достаточно иметь лишь технические устройства, соответствующие ФП СЛФ.

Основные законы и правила.

А. Преобразование выражений в булевой алгебре (см. табл.1.10).

Таблица 1.10.

№	логическое сложение	логическое умножение	название закона
1	$x+0=x$	$x \cdot 1 = x$	-
2	$x+1=1$	$x \cdot 0 = 0$	-
3	$x+x=x$	$x \cdot x = x$	закон идемпотентности
4	$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$	-
5	$\bar{\bar{x}} = x$		закон двойного отрицания
6	$x_1+x_2=x_2+x_1$	$x_1x_2=x_2x_1$	коммутативный закон
7	$x_1+x_1x_2=x_1$	$x_1(x_1+x_2)=x_1$	закон поглощения
8	$\overline{(x_1 + x_2)} = \bar{x}_1\bar{x}_2$	$\overline{(x_1x_2)} = \bar{x}_1 + \bar{x}_2$	закон Де Моргана (правило)
9	$(x_1+x_2)+x_3=$ $=x_1+(x_2+x_3)=$ $=x_1+x_2+x_3$	$(x_1x_2)x_3=x_1(x_2x_3)=$ $=x_1x_2x_3$	ассоциативный закон
10	$x_1+x_2x_3=$ $=(x_1+x_2)(x_1+x_3)$	$x_1(x_2+x_3)=$ $=x_1x_2+x_1x_3$	дистрибутивный закон (распределительный)

Пример. Упростить логическое выражение

$$(((\bar{x}_2 \oplus 1)x_2x_3) \downarrow (x_2 \vee x_2\bar{x}_3) \vee x_3\bar{x}_1)) \backslash (\overline{x_2x_1}(x_3 \vee \bar{x}_3))$$

используя законы и правила булевой алгебры.

$$(((\underbrace{\bar{x}_2}_{x_2} \oplus 1)x_2x_3) \downarrow (\underbrace{x_2 \vee x_2\bar{x}_3}_{x_2}) \vee x_3\bar{x}_1)) \backslash (\overline{x_2x_1}(\underbrace{x_3 \vee \bar{x}_3}_{1})) =$$

$$= ((x_2x_3 \downarrow x_2) \vee x_3\bar{x}_1) \backslash \overline{x_2x_1} =_{\text{функцию}\downarrow}^{\text{раскрываем}} = (\underbrace{\overline{\overline{x_2x_3 \vee x_2} \vee x_3\bar{x}_1}}_{x_2}) \cdot \overline{x_2x_1} = (\overline{\bar{x}_2 \vee x_3\bar{x}_1}) \cdot \overline{x_2x_1} =$$

$$=_{\text{Де\_Моргана}}^{\text{применим\_правило}} = \overline{\bar{x}_2 \vee x_3\bar{x}_1} \vee \overline{x_2x_1} =_{\text{применяется\_правило\_Де\_Моргана}}^{\text{последовательно\_дважды}} = \bar{\bar{x}}_2 \cdot \overline{x_3\bar{x}_1} \vee x_2x_1 =$$

$$= x_2(\overline{x_3} \vee x_1) \vee x_2 x_1 = x_2(\underbrace{\overline{x_3} \vee x_1 \vee x_1}_{x_1}) = x_2(\overline{x_3} \vee x_1)$$

Задание 44. Упростить выражение

$$\overline{((x_1 \vee x_3) \cdot \overline{x_2(\overline{x_1 \vee x_3})}} \cdot \overline{(x_3 x_1 \vee x_1(x_2 \vee \overline{x_2}))} \cdot \overline{x_3}$$

Задание 45. Упростить выражение

$$(x_2(\overline{x_1} \oplus 1) \vee \overline{(x_1 \vee \overline{x_2})}) \cdot (x_1 \downarrow (\overline{x_2} \vee \overline{x_2} x_3)) \vee \overline{(x_1 \vee \overline{x_3}} \vee \overline{(\overline{x_1} \setminus 1)})$$

Задание 46. Привести выражение

$((\overline{x_1} x_3 x_2) \setminus \overline{(x_1 \overline{x_2} \vee \overline{x_3})}) \downarrow x_1$ к виду содержащему лишь операции инверсии и дизъюнкции.

1.3.2. Формы представления логических функций.

Задать логическую функцию можно в различных формах: а)словесно, б)таблицей истинности, в)алгебраически (формульно), г)картами Карно, д)в цифровой форме, е)в строчной форме и др.

Словесное представление логических функций.

Покажем на примере функции конъюнкции. Словесно эту функцию можно представить так: логическая функция принимает значение равное 1 лишь в том случае, когда все её аргументы равны 1.

Табличное представление логических функций.

Каждому набору аргументов ставится в соответствие значение функции и представляется таблицей. Если наборы аргументов упорядочены по возрастанию, то такая таблица называется таблицей истинности (ТИ).

Алгебраическое представление логических функций (в СДН и СКН формах).

Примем обозначение $x_i^{\alpha_i} = \begin{cases} x_i, при\ \alpha = 1 \\ \overline{x_i}, при\ \alpha = 0 \end{cases}$, $i = \overline{1,n}.$,

где n – количество аргументов логической функции.

Любую переключательную функцию n аргументов можно представить в следующем виде:

$$f(x_1, x_2, ..., x_n) = \bigvee_{(\alpha_1, \alpha_2, ..., \alpha_k)} x_1^{\alpha_1} \cdot ... \cdot x_k^{\alpha_k} \cdot f(\alpha_1, \alpha_2, ..., \alpha_k, x_{k+1}, ..., x_n)$$

Следствие:

$$f(x_1, ..., x_n) = \overline{x}_i f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee x_i f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)$$

Теорема о СДНФ (совершенная дизъюнктивная нормальная форма):
Всякую переключательную функцию можно представить:

$$f(x_1,...,x_n) = \bigvee_{f=1}(x_1^{\alpha_1} \cdot ... \cdot x_n^{\alpha_n})$$

Доказать самостоятельно.

Теорема о СКНФ (совершенная конъюнктивная нормальная форма):

$$f(x_1,..., x_n) = \bigwedge_{f=0}(\bar{x}_1^{\alpha_1} \vee ... \vee \bar{x}_n^{\alpha_n})$$

Доказательство:

$$f(...) = \bigvee_{f=1}(x_1^{\alpha_1}...x_n^{\alpha_n}) = \overline{\overline{\bigvee_{f=1}(x_1^{\alpha_1}...x_n^{\alpha_n})}}^{\text{закон\_двойственности}} = \overline{\bigwedge_{f=1}\overline{x_1^{\alpha_1}...x_n^{\alpha_n}}}^{\text{з\_Де\_Морг}} = \overline{\bigwedge_{f=1}(\bar{x}_1^{\alpha_1} \vee ... \vee \bar{x}_n^{\alpha_n})}^{\text{закон\_Де\_Моргана}} = \bigwedge_{f=0}(\bar{x}_1^{\alpha_1} \vee ... \vee \bar{x}_n^{\alpha_n})$$

ч. т. д.

Пример: Пусть переключательная функция $f(x_1,x_2,x_3)$ задана таблицей 1.11.

Таблица 1.11.

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$f(x_1,x_2,x_3)_{СДНФ} = \underset{\text{где\_f=1}}{\text{из\_строк}} = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2x_3$$

$$\mathbf{f(x_1,x_2,x_3)_{СКНФ} = \underset{\text{иверсий\_строк\_f=0}}{из} = (x_1 \vee x_2 \vee x_3)\cdot(\bar{x}_1 \vee x_2 \vee x_3)\cdot(\bar{x}_1 \vee x_2 \vee \bar{x}_3)\cdot(\bar{x}_1 \vee \bar{x}_2 \vee x_3)}$$

таблица истинности избыточна более чем в 2 раза.

Отсюда следует, что любую логическую функцию (кроме F=0) можно представить в совершенной дизъюнктивной нормальной форме (СДНФ).

$$F(x_1,...,x_n) = \bigvee_{F=1}(x_1^{\alpha_1} \cdot,...,\cdot x_n^{\alpha_n})$$

или в совершенной конъюнктивной нормальной форме (СКНФ)

$$F(x_1,..., x_n) = \bigwedge_{F=0}(\bar{x}_1^{\alpha_1} +,...,+\bar{x}_n^{\alpha_n}).$$

Для логической функции, представленной ТИ, СДНФ строится следующим образом:

- выделяются наборы аргументов (строки ТИ) при которых функция равны 1;

- из аргументов каждого выделенного набора формируется соответствующая конъюнкция (аргументы, значения которых в наборе равны 0, входят в конъюнкцию с инверсией);

- сформированные конъюнкции соединяются знаком дизъюнкций.

При построении СКНФ порядок действий следующий:

- выделяются наборы аргументов при которых функция равна 0;

- из аргументов каждого выделенного набора формируется соответствующая дизъюнкция (аргументы, значения которых в наборе равны 1, входят в дизъюнкцию с инверсией);

- сформированные дизъюнкции соединяются знаками конъюнкций.

Пример. Представить логическую функцию $F_1(x_1,x_2,x_3)$, заданную таблицей 1.12. истинности, в СДНФ и СКНФ.

Таблица 1.12.

x_1	x_2	x_3	F_1	F_2
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$$F_1(x_1,x_2,x_3)_{СДНФ} = \overline{x_1}\,\overline{x_2}x_3 \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1\overline{x_2}x_3 \vee x_1x_2x_3$$

$$F_1(x_1,x_2,x_3)_{СКНФ} = (x_1 \vee x_2 \vee x_3)\cdot(x_1 \vee \overline{x_2} \vee x_3)\cdot(x_1 \vee \overline{x_2} \vee \overline{x_3})\cdot(\overline{x_1} \vee \overline{x_2} \vee x_3)$$

Задание 47. Логическая функция $F_2(x_1,x_2,x_3)$ задана ТИ. Представить функцию в СДНФ и СКНФ.

Задание 48. Представить логическую функцию $(x_1 \cdot \overline{x_2}) \oplus (x_2 \vee \overline{x_3})$ таблицей истинности, а также в СКНФ и СДНФ.

Представление логических функций картами Карно.

Карта Карно – это графическое представление таблицы истинности. Между строками ТИ и клетками карты Карно (КК) существует взаимно однозначное соответствие. Каждая клетка КК заполняется нулём или единицей в соответствии со значением функции, вычисленной при значениях аргументов на пересечении которых расположена данная клетка.

Например, для 2-х переменных ТИ и КК будут выглядеть следующим образом:

x_1	x_2	$F(x_1,x_2)$
0	0	F(0,0)
0	1	F(0,1)
1	0	F(1,0)
1	1	F(1,1)

x_1 \ x_2	0	1
0	F(0,0)	F(0,1)
1	F(1,0)	F(1,1)

Карты Карно для 3-х и 4-х переменных выглядят так:

КК для 3-х переменных

x_1 \ x_2x_3	00	01	11	10
0	F(0,0,0)	F(0,0,1)	F(0,1,1)	F(0,1,0)
1	F(1,0,0)	F(1,0,1)	F(1,1,1)	F(1,1,0)

КК для 4-х переменных

x_1x_2 \ x_3x_4	00	01	11	10
00	…	…	…	…
01	…	…	…	…
11	…	…	…	…
10	…	…	…	…

Следует обратить внимание на расположение наборов аргументов по координатным осям КК. Последовательность наборов определяется изменением значения только одной переменной. Поэтому расположение 00 01 10 11 является неправильным, т. к. при переходе от 01 к 10 изменяются сразу две переменные. Правильным будет расположение 00 01 11 10.

Задание 49. Представить логическую функцию $\overline{x}_2x_3 + \overline{x}_1\overline{x}_3$ картой Карно.

Задание 50. Представить логическую функцию $x_1\overline{x}_4 \downarrow \overline{x}_2\overline{x}_3x_4$ картой Карно.

Цифровая форма представления логических функций.

Если аргумент x_i в таблице истинности располагать в порядке возрастания индекса i, то наборы аргументов в таблице можно сокращённо обозначать десятичными числами. Тогда СДНФ в цифровой форме будет представлять сумму чисел, соответствующих наборам аргументов на которых функция принимает значение равное единице.

СКНФ представляется произведением десятичных чисел, соответствующих наборам аргументов на которых функция принимает значение равное нулю. Но в этом случае при получении чисел, соответствующих наборам аргументов, значения аргументов нужно заменить на обратное.

Десятичные числа в цифровой форме располагаются в порядке возрастания.

Пример. Представить логическую функцию, заданную таблицей 1.13., в цифровых формах.

Таблица 1.13.

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

При построении $F_{СДНФ}$ в цифровой форме следует выделять в ТИ наборы аргументов 001, 010, 100, 110, 111 и представлять их десятичными числами 1, 2, 4, 6, 7. Тогда $F_{СДНФ} = \sum_{0}^{7}(1,2,4,6,7)$.

При построении $F_{СКНФ}$ в цифровой форме выделяем наборы 000, 011, 101, заменяем их на обратные, т. е. на 111, 100, 010 и представляем их десятичными числами 7, 4, 2, располагая по возрастанию: $F_{СКНФ} = \prod_{0}^{7}(2,4,7)$.

Возможен переход от цифровой формы к алгебраической. Для рассмотренного примера переход будет следующим:

$$\sum_{0}^{7}(1,2,4,6,7) = \sum_{0}^{7}(001,010,100,110,111) = \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} + x_1x_2\overline{x_3} + x_1x_2x_3$$

$$\prod_{0}^{7}(2,4,7) = \prod_{0}^{7}(010,100,111) = (\overline{x_1} + x_2 + \overline{x_3}) \cdot (x_1 + \overline{x_2} + \overline{x_3}) \cdot (x_1 + x_2 + x_3)$$

При цифровых формах представления логических функций переход от СДНФ к СКНФ или наоборот выполняется по следующим правилам.

Пусть задана функция $F_{СДНФ} = \sum_{0}^{N-1}(\alpha_1,...,\alpha_t)$.

Перейдём к обратной функции $\overline{F}_{СДНФ} = \sum_{0}^{N-1}(\beta_1,...,\beta_k)$, $t + k = N$, где N – полное количество наборов аргументов функции. Обратная функция $\overline{F}_{СДНФ}$ получается путём записи в неё десятичных чисел, отсутствующих в представлении $F_{СДНФ}$. Используя $\overline{F}_{СДНФ}$ можно получить $F_{СКНФ} = \prod_{0}^{N-1}(\varphi_1,...,\varphi_k)$, где $\varphi_1 = (N-1) - \beta_k,..., \varphi_i = (N-1) - \beta_i,..., \varphi_k = (N-1) - \beta_1$.

Пример. Для функции, рассмотренной в предыдущем примере, выполнить переход от $F_{СДНФ}$ к $F_{СКНФ}$ в цифровой форме.

Задана $F_{СДНФ} = \sum_{0}^{7}(\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5) = \sum_{0}^{7}(1,2,4,6,7)$.

Перейдём к обратной функции: $\overline{F}_{СДНФ} = \sum_{0}^{7}(\beta_1,\beta_2,\beta_3) = \sum_{0}^{7}(0,3,5)$. Так как N=8, то $\varphi_1 = (N-1) - \beta_3 = 7 - 5 = 2$, $\varphi_2 = (N-1) - \beta_2 = 7 - 3 = 4$, $\varphi_3 = (N-1) - \beta_1 = 7 - 0 = 7$.

Таблица 1.14

Следовательно, $F_{СКНФ} = \prod_{0}^{7}(2,4,7)$.

x_1	x_2	x_3	x_4	F_1	F_2
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

Задание 51. Представить логическую функцию F_1, заданную таблицей истинности (табл. 1.14.), в цифровых формах.

Задание 52. Представить логическую функцию F_2, заданную таблицей истинности (табл. 1.14.), в цифровых формах.

Задание 53. Представить $F = x_1 \cdot \overline{x_2} \cdot (x_3 \oplus 1) + \overline{x_2 \cdot x_3}$ в цифровых формах.

Задание 54. Представить $F_{СДНФ} = \overline{x_1}x_2\overline{x_3}x_4 + x_1\overline{x_2}\overline{x_3}\overline{x_4} + x_1\overline{x_2}x_3x_4 + x_1x_2x_3\overline{x_4}$ в цифровой форме и выполнить переход к $F_{СКНФ}$.

Задание 55. Представить $F_{СКНФ} = (\overline{x}_1 + x_2 + x_3)\cdot(x_1 + \overline{x}_2 + \overline{x}_3)\cdot(x_1 + x_2 + x_3)$ в цифровой форме и выполнить переход к $F_{СДНФ}$.

Строчная форма представления логических функций.

Строчное представление целесообразно при обработке логических функций на ЭВМ. При таком представлении столбец значений функции из таблицы истинности записывается горизонтально – строчкой и выделяется рамкой.

Пример. Представить функцию F_1 (таблица 1.14) в строчной форме.

$F_1 = \sum_{4}$ 0001001111110000

Переход от цифровой формы представления функции к строчной форме и наоборот выполняется просто. Покажем на примере.

Пример. Представить функцию, заданную цифровыми формами $\sum_{0}^{7}(1,2,4,6,7)$ и $\prod_{0}^{7}(2,4,7)$ в строчных формах.

$F_{СДНФ} = \sum_{0}^{7}(1,2,4,6,7) = \sum_{3}$ 01101011

$F_{СКНФ} = \prod_{0}^{7}(2,4,7) = \prod_{3}$ 00101001

01234567 – номера_разрядов

Как видно из примера, единицы при строчной записи расположены в разрядах, соответствующих десятичным числам при цифровой записи.

Задание 56. Представить функцию F_2 (табл. 1.14) в строчной форме и выполнить переход к цифровой форме представления.

Задание 57. Представить функцию $F = x_1 \cdot \overline{x_2} \cdot (x_3 \oplus 1) + \overline{x_2 \cdot x_3}$ в строчной форме.

1.3.3. Минимизация логических функций.

Задача минимизации заключается в сведении логической функции к виду, позволяющему выполнить наиболее простую её техническую реализацию.

Для минимизации применяется различные методы: геометрический метод, метод алгебраических преобразований, метод Карно, метод Морреля и другие.

Геометрическая интерпретация задачи упрощения и минимизации логических функций.

Таблица 1.15.

x_3	x_2	x_1	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0
таблица истинности			

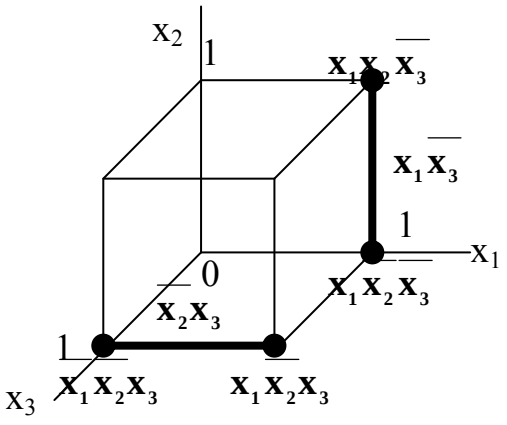


Рис. 1.2.

Из_таблицы 1.15. $\rightarrow$ СДНФ

$$f = x_1\overline{x_2}\,\overline{x_3} \vee x_1x_2\overline{x_3} \vee \overline{x_1}\,\overline{x_2}x_3 \vee x_1\overline{x_2}x_3$$

Если 2 смежные вершины куба заменить ребром, то получим упрощение: $f = x_1\overline{x_3} \vee \overline{x_2}x_3$.

Ниже рассматриваются два метода: метод Карно, который эффективен при ручной минимизации и метод Морреля, обычно используемый при минимизации функции на ЭВМ.

Минимизация методом Карно (рассмотрим три варианта: А, Б, В).

А. Минимизация в дизъюнктивно нормальной форме (получение МДНФ).

Метод минимизации логических выражений по карте Карно состоит в выделении групп клеток содержащих 1-цы и в составлении минимизированного логического выражения по этим группам. При этом каждая 1-ца в карте Карно должна войти хотя бы в одну группу.

Группой клеток называется совокупность соседних клеток, составляющих прямоугольник размерности $2^a \times 2^b$, где a и b = 0, 1, 2, … .

Каждой группе размерности $2^a \times 2^b$ соответствует конъюнкция состоящая из $n-a-b$ переменных, где n – полное число аргументов логической функции. В конъюнкцию включаются все аргументы, значения которых не изменяются в пределах соответствующей группы. Если переменная в пределах группы равна 0, то она входит в конъюнкцию с инверсией, а если 1 – без инверсии.

Составление минимизированного выражения заключается в объединении дизъюнкциями конъюнкций, соответствующих всем выделенным группам.

Два принципа формирования группы:

- каждая группа должна быть как можно больше, т. е. max(a+b);
- число групп должно быть как можно меньше.

<u>Пример.</u> Дана карта Карно для функции 4-х переменных. Получить МДНФ этой функции.

$$f(x_1,x_2,x_3,x_4)=\underbrace{\overline{x_3}\,\overline{x_4}}_{\text{группа 1}}\vee\underbrace{x_1x_2\overline{x_3}}_{\text{группа 2}}\vee\underbrace{\overline{x_1}x_3}_{\text{группа 3}}$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	1	1
01	1	0	1	1
11	1	1	0	0
10	1	0	0	0

1 группа 1

группа 2

группа 3

Например, конъюнкция соответствующая 1-ой группе состоит из двух аргументов, т. к. $n-a-b=4-2-0=2$, а для 2-ой группы $4-1-0=3$.

Следует обратить внимание, что группы могут пересекаться, т. е. одна и та же 1-ца может входить в разные группы.

В КК для 3-х и 4-х переменных клетки могут объединяться в так называемые разорванные группы, например:

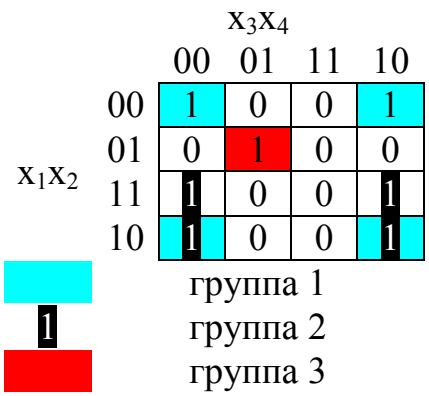
1	0	0	1
1	0	0	1

0	0	1	0
1	0	0	1
1	0	0	1
0	0	1	0

1	0	0	1
0	0	0	0
0	0	0	0
1	0	0	1

Пример. Дана карта Карно для функций 4-х переменных. Получить МДНФ этой функции.

$$f(x_1,x_2,x_3,x_4)=\underset{1гр}{\overline{x}_2\overline{x}_4}\vee\underset{2гр}{x_1\overline{x}_4}\vee\underset{3гр}{\overline{x}_1x_2\overline{x}_3}x_4$$



Задание 58. Получить МДНФ для функции F_1 заданной в таблице 1.16.

Задание 59. Получить МДНФ для функции F_2 заданной в таблице 1.16.

Таблица 1.16.

x_1	x_2	x_3	x_4	F_1	F_2	F_3	F_4	F_5	F_6
0	0	0	0	0	1	1	1	0	0
0	0	0	1	1	0	1	1	0	1
0	0	1	0	0	1	0	0	1	-
0	0	1	1	0	1	0	1	1	0
0	1	0	0	1	1	1	0	1	1
0	1	0	1	1	0	0	0	1	0
0	1	1	0	1	1	1	1	1	1
0	1	1	1	1	0	1	0	0	1
1	0	0	0	1	1	0	0	-	-
1	0	0	1	1	0	1	1	0	-
1	0	1	0	1	1	0	0	-	0
1	0	1	1	0	1	0	1	1	0
1	1	0	0	0	0	0	1	-	1
1	1	0	1	0	0	1	0	1	1
1	1	1	0	0	0	0	1	0	0
1	1	1	1	0	0	1	0	1	1

Б. Минимизация в конъюнктивной нормальной форме (получение МКНФ).

Здесь возможны 2 способа.

1. Перейти от МДНФ к МКНФ при помощи правила Де Моргана.

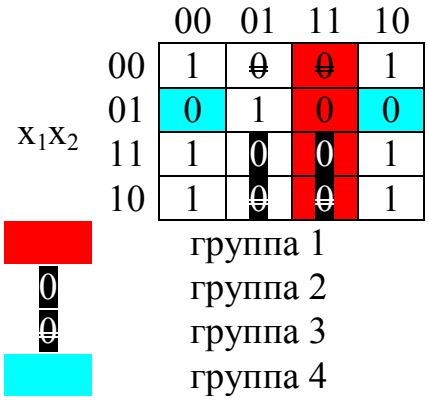
2. Использовать клетки КК содержащие нули.

Составление МКНФ заключается в объединении конъюнкциями дизъюнкций, соответствующих всем выделенным группам. В дизъюнкцию включаются все переменные, значения которых не изменяются в пределах группы. Причём, если переменная в пределах группы равна 0, то она входит в дизъюнкцию без инверсии, а если равна 1, то с инверсией.

x_3x_4

Пример. Дана карта Карно для функции 4-х переменных. Получить МКНФ этой функции.

$$F(x_1,x_2,x_3,x_4) = $$
$$= (\overline{x_3} \vee \overline{x_4}) \cdot (\overline{x_1} \vee \overline{x_4}) \cdot (x_2 \vee \overline{x_4}) \cdot (x_1 \vee \overline{x_2} \vee x_4)$$



Задание 60. Получить МКНФ для функции F_4 заданной в таблице 1.16.

Задание 61. Получить МКНФ для функции F_3 заданной в таблице 1.16.

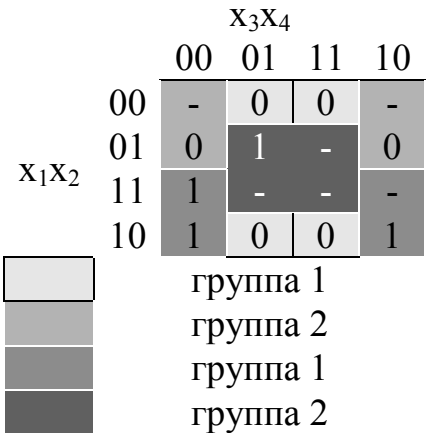
В. Минимизация неполностью определённых функций.

Неполностью определённые функции это такие функции, которые определены не на всех 2^n наборах аргументов.

На карте Карно неопределённое условие обозначается прочерком (-). Клетки с прочерком могут произвольным образом включаться в группы, как при построении МДНФ, так и при построении МКНФ. Более того, их можно вообще никуда не включать (игнорировать).

Пример. Дана карта Карно для функции 4-х переменных. Получить МДНФ и МКНФ этой функции.

$$F(x_1,x_2,x_3,x_4)_{МДНФ} = x_1\overline{x_4} \vee x_2x_4$$
$$F(x_1,x_2,x_3,x_4)_{МКНФ} = (x_2 \vee \overline{x_4}) \cdot (x_1 \vee x_4)$$



Задание 62. Получить МДНФ для функции F_5 заданной в таблице 1.16.

Задание 63. Получить МДНФ и МКНФ для функции F_6 заданной в таблице 1.16.

Минимизация методом Морреля.

В основе метода лежит теорема разложения.

Теорема разложения. Любую логическую функцию от n переменных можно свести к двум функциям от (n-1) переменных, проведя разложение вида

$$F(x_1,...,x_n) = \overline{x_i} \cdot F_0(x_1,...,x_{i-1},0,x_{i+1},...,x_n) + x_i \cdot F_1(x_1,...,x_{i-1},1,x_{i+1},...,x_n)$$

Сущность метода заключается в последовательном разложении функции по переменным и в анализе остаточных функций на равенство их нулю или единице.

$$\underbrace{F(x_1,\ldots,x_n)}_{\text{минимизируемая функция}} = \overline{x_1}\cdot \underbrace{F_0(0,x_2,\ldots,x_n)}_{\text{остоточная функция}} + x_1 \cdot \underbrace{F_1(1,x_2,\ldots,x_n)}_{\text{остаточная функция}}$$

Если остаточная функция равна нулю, то член, её включающий, равен нулю и исключается из разложения. Если остаточная функция равна единице, то переменная или конъюнкция переменных, связанных с ней, входит в тупиковую форму, а член из дальнейшего рассмотрения исключается. Все оставшиеся после анализа остаточные функции разлагаются по следующей переменной и снова анализируются и т. д. Минимизация заканчивается, когда все остаточные функции будут равны 0 или 1.

Для получения формы, наиболее близкой к минимальной, Моррель предложил в формулу разложения добавить так называемый «терм Морреля», который добавляется всякий раз при разложении по любой переменной.

Формула разложения примет вид:

$$F(\ldots) = \overline{x_1}\cdot F_0(\ldots) + x_1 F_1(\ldots) + \underbrace{F_0(\ldots)\cdot F_1(\ldots)}_{\text{терм Морелля}}$$

Покажем, что терм Морелля не изменит функцию:

$$F(\ldots) = \overline{x_1}\cdot F_0(\ldots) + x_1\cdot F_1(\ldots) + F_0(\ldots)\cdot F_1(\ldots)\underbrace{(\overline{x_1} + x_1)}_{1} =$$

$$= \overline{x_1}\cdot F_0(\ldots) + x_1\cdot F_1(\ldots) + F_0(\ldots)\cdot F_1(\ldots)\cdot \overline{x_1} + F_0(\ldots)\cdot F_1(\ldots)\cdot x_1 =$$

$$= \overline{x_1}\cdot F_0(\ldots)\cdot \underbrace{(1 + F_1(\ldots))}_{1} + x_1\cdot F_1(\ldots)\cdot \underbrace{(1 + F_0(\ldots))}_{1} = \overline{x_1}\cdot F_0(\ldots) + x_1\cdot F_1(\ldots).$$

При анализе остаточных функций на каждом этапе разложения проверяются условия поглощения термом Морреля других остаточных функций.

Минимизацию методом Морреля удобно проводить над функцией представленной в строчной форме (т. к. метод реализуем на ЭВМ). Разложение функции, представленной в строчной форме, по переменной x_i означает её разбиение на две части. В первую часть входят те наборы, на которых $x_i=0$, а во вторую – на которых $x_i=1$.

Пример. Пусть задана функция F(A,B,C,D) в строчной форме.

$F(A,B,C,D) = \sum_{A,B,C,D} 0000000100010111$. Выполнить разложение функции по переменной A.

$$F(A,B,C,D) = \sum_{A,B,C,D} 0000000100010111 = \overline{A}\cdot \sum_{B,C,D} 00000001 + A\cdot \sum_{B,C,D} 00010111$$

Теперь получим терм Морреля и добавим в разложение.

Получение терма Морреля поразрядным умножением

```
00000001
00010111
--------
00000001   -терм Морреля
```

Следовательно,

$$F(A,B,C,D)=\overline{A}\cdot\sum_{B,C,D}\boxed{00000001}+A\cdot\sum_{B,C,D}\boxed{00010111}+\sum_{B,C,D}\boxed{00000001}.$$

Пример. Минимизировать методом Морреля функцию

$$F(A,B,C)=\sum_{0}^{7}(1,3,4,5,6,7).$$

$$F=\sum_{A,B,C}\boxed{01011111}=\overline{A}\cdot\sum_{B,C}\boxed{0101}+A\cdot\sum_{B,C}\boxed{1111}+\sum_{B,C}\boxed{0101}=\cdot\sum_{B,C}\boxed{0101}+A=$$

$$=\overline{B}\sum_{C}\boxed{01}+B\sum_{C}\boxed{01}+\sum_{C}\boxed{01}+A=\sum_{C}\boxed{01}+A=C\sum\boxed{0}+C\sum\boxed{1}+$$

$$+\sum\boxed{0}+A=C+A$$

Таким образом, получим F(A,B,C)=A+C.

Задание 64. Выполнить разложение функции F_1 (таблица 1.14.) по переменной x_1.

Задание 65. Минимизировать функцию F_2 (таблица 1.14.) методом Морелля.

Задание 66. Минимизировать функцию $F_{СДНФ}=\sum_{0}^{15}(2,4,5,6,8,9,10,12,14,15)$ методом Морреля.

1.4. Синтез и анализ комбинационных схем.

Устройство, осуществляющее дискретное преобразование информации ЭВМ, называется автоматом. Существуют два основных вида дискретных автоматов: комбинационные автоматы (схемы) и конечные автоматы. Конечные автоматы рассматриваются в разделе 1.5.

Введём: $X(t)=\{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ – множество входных дискретных сигналов; $x_i \in \{0,1\}, i=\overline{1,n}$.

$Y(t)=\{y_1, y_2, \ldots, y_j, \ldots, y_m\}$ – множество выходных дискретных сигналов; $y_j \in \{0,1\}, j=\overline{1,m}$.

Здесь t – дискретное время, определяется моментами перехода автомата из одного состояния в другое (рис. 1.3.).

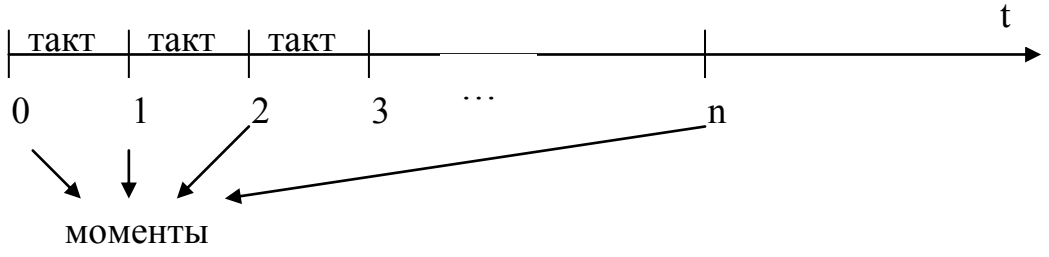


Рис. 1.3.

Определение. Если совокупность выходных сигналов (выходного слова) Y(t) зависит только от совокупности входных сигналов (входного слова) X(t) и не зависит от внутренних состояний автомата, то такой автомат называется комбинационным автоматом (комбинационной схемой).

Структурная схема комбинационного автомата представлена на рисунке:

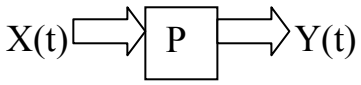


Рис. 1.4.

Здесь P – функция преобразования.

Закон функционирования комбинационной схемы определяется заданием соответствия между её входными и выходными словами Y(t)=P[X(t)] и может быть дан таблицей истинности, в аналитический форме и др.

1.4.1. Алгоритм синтеза комбинационной схемы.

1. Задать закон функционирования комбинационной схемы.

2. Провести минимизацию функции преобразования P, получить $P_{МДНФ}$ и $P_{МКНФ}$.

3. Преобразовать $P_{МДНФ}$ и $P_{МКНФ}$ в соответствии с заданным базисом логических функций (используемыми логическими элементами) и выбрать наиболее компактную форму представления P.

4. Построить функциональную схему устройства.

5.Выполнить схемотехническую коррекцию схемы с учётом характеристик и параметров используемых логических элементов:

-коэффициента объединения (количество входов элемента);

-коэффициента разветвления (характеризует нагрузочную способность – максимально допустимое количество элементов подключаемых к входу);

- уровня логических 0 и 1;

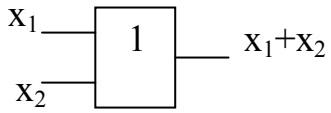
- помехозащитности и др.

6. Провести тестирование полученной схемы на соответствие закону функционирования. Этот этап является уже частью анализа схемы и ставит задачей убедится в её работоспособности.

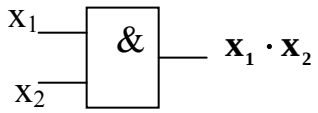
1.4.2. Функциональные схемы логических элементов.

Приведём лишь функциональные схемы, реализующие простейшие логические функции:

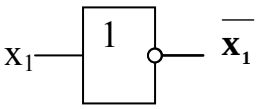
дизъюнкция («или»)



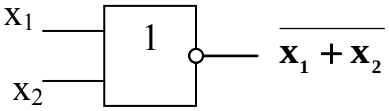
конъюнкция («и»)



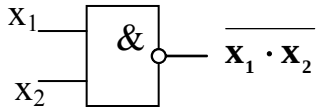
инверсия («не»)



стрелка Пирса («или-не»)

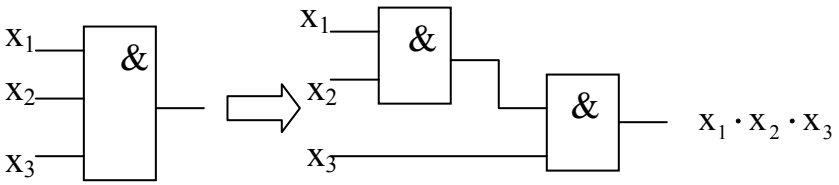


штрих Шеффера («и-не»)

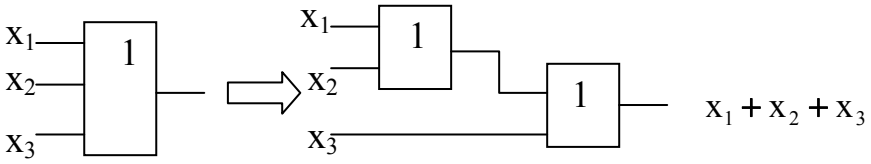


Свести 3-х и 4-х входовые логические схемы к 2-х входовым можно следующим образом:

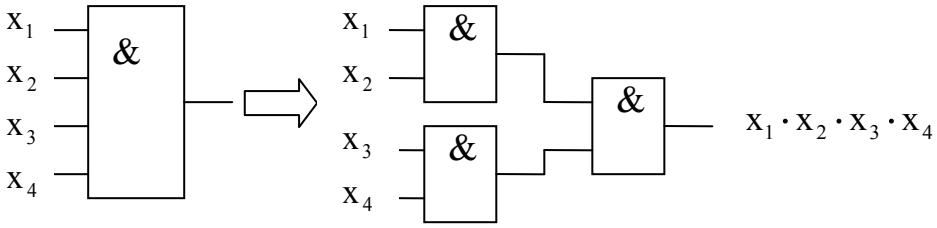
«3 и»



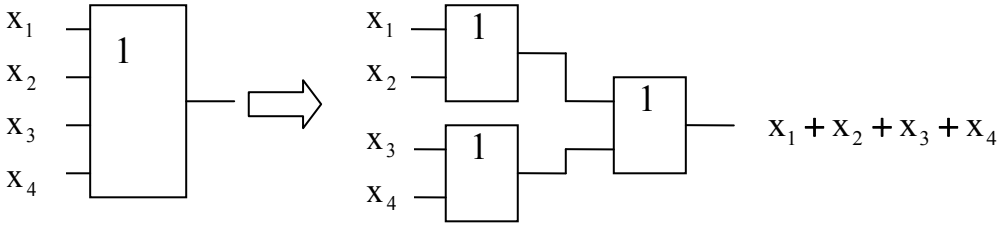
«3 или»



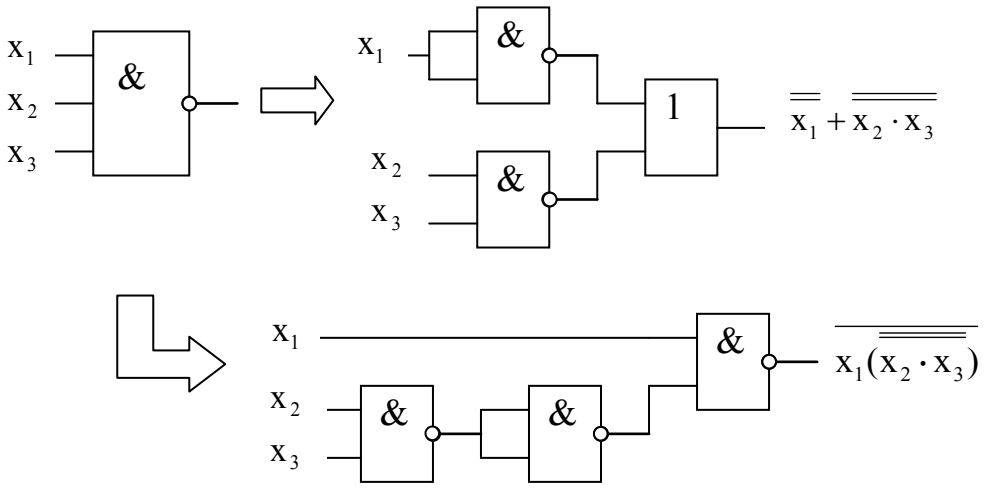
«4 и»



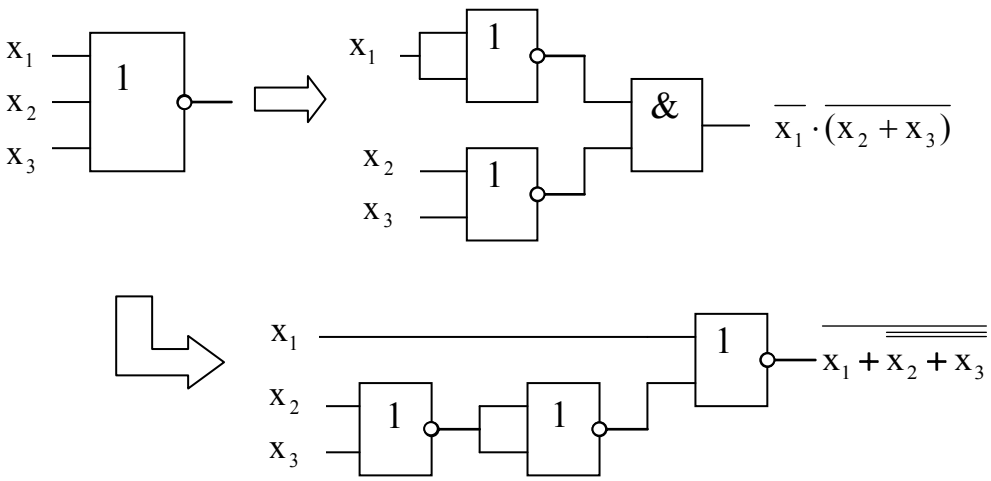
«4 или»

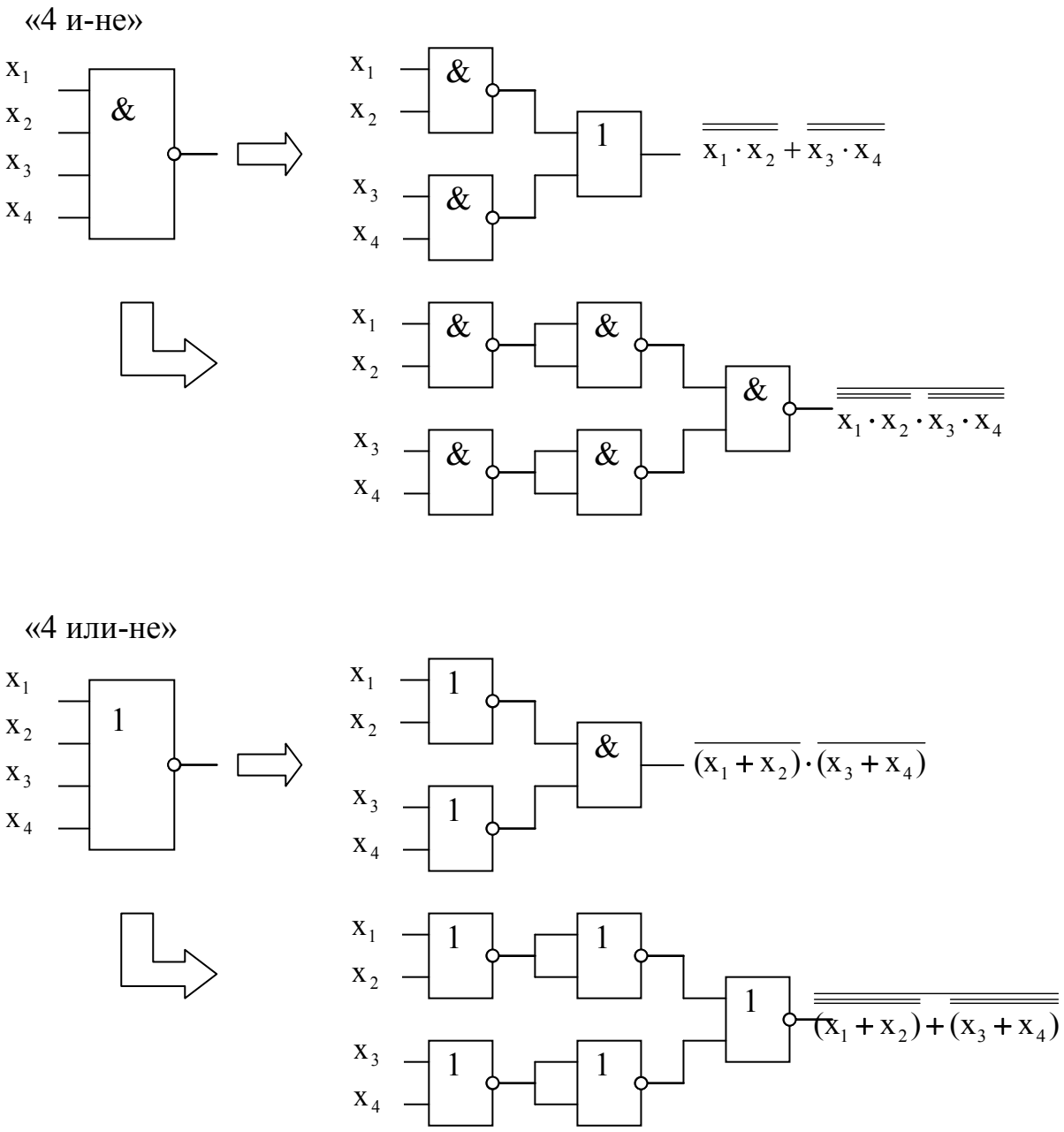


«3 и-не»



«3 или-не»





1.4.3. Пример синтеза полного одноразрядного двоичного сумматора.

Пример. Синтезировать схему полного одноразрядного двоичного сумматора, используя двухвходовые логические элементы, реализующие логическую функцию «и-не».

Представим сумматор в виде (рис. 1.5.):

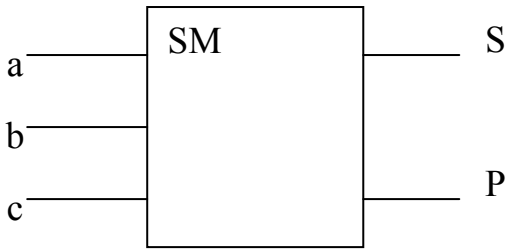


Рис. 1.5.

где: a и b – одноразрядные двоичные слагаемые;

c – перенос из младшего разряда;

S – сумма;

P – перенос в старший разряд;

$S=F_1(a,b,c)$, $P=F_2(a,b,c)$.

При синтезе реализуем последовательно этапы алгоритма.

1. Зададим закон функционирования сумматора в виде ТИ (табл. 1.17.).

Таблица 1.17.

c	b	a	S	P
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2. Проведём минимизацию функций S и P методом Карно. По ТИ составим карты Карно:

c \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

для S

c \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

для P

$$S_{МДНФ} = a \cdot b \cdot c + a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c}$$

$$S_{МКНФ} = (a + b + c) \cdot (a + \bar{b} + \bar{c}) \cdot (\bar{a} + \bar{b} + c) \cdot (\bar{a} + b + \bar{c})$$

$$P_{МДНФ} = a \cdot b + a \cdot c + b \cdot c$$

$$P_{МКНФ} = (a + b) \cdot (a + c) \cdot (b + c)$$

Проверим предварительное построение схем сумматора непосредственно по выражениям $S_{МДНФ}$ и $P_{МДНФ}$ без преобразования их к заданным двухвходовым элементам «и-не» (рис. 1.6.).

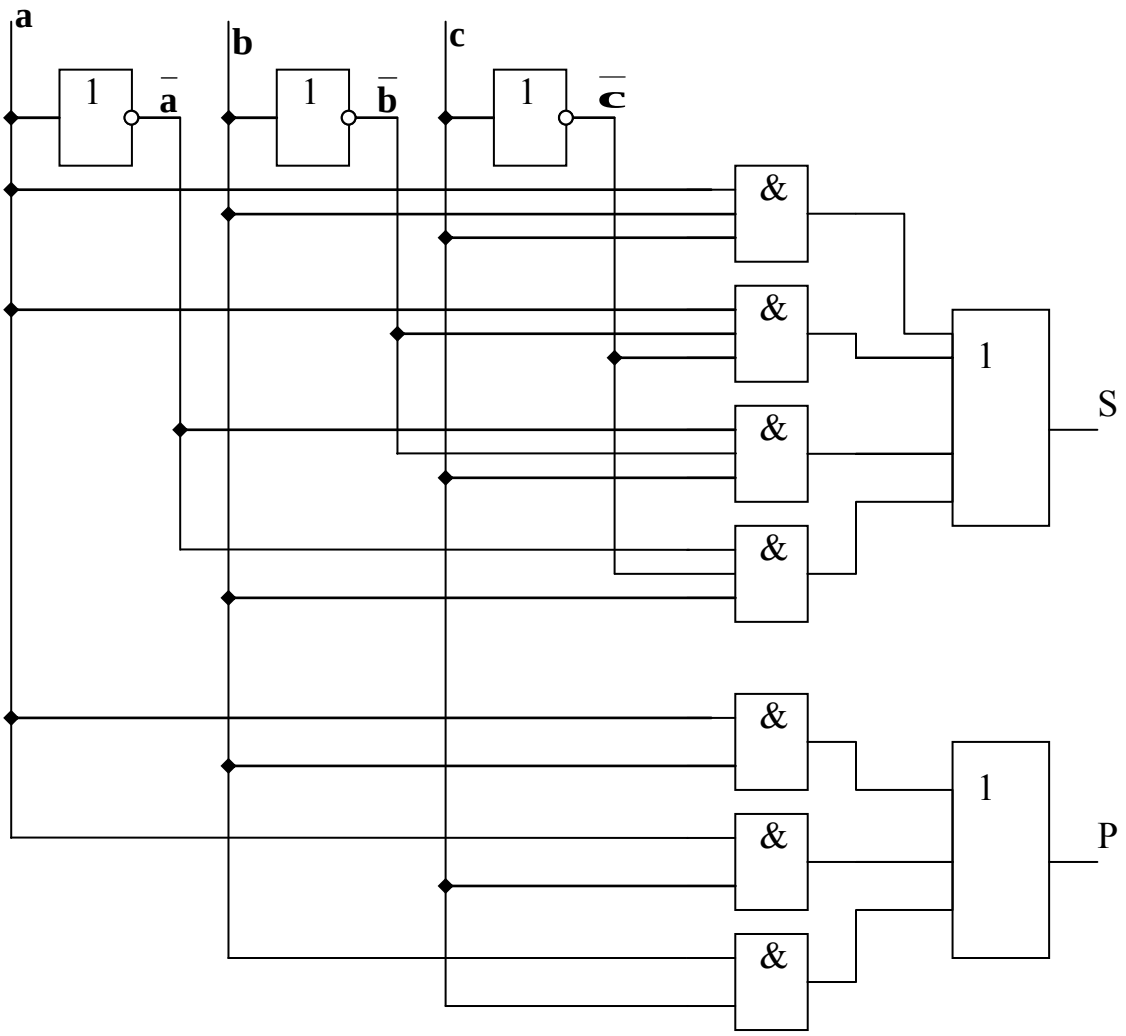


Рис. 1.6.

3. Преобразуем полученные выражения в соответствии с заданными логическими элементами «и-не».

$$S_{МДНФ} = \overline{\overline{a \cdot b \cdot c + a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c}}} = \overline{\overline{(a \cdot b \cdot c)} \cdot \overline{(a \cdot \bar{b} \cdot \bar{c})} \cdot \overline{(\bar{a} \cdot \bar{b} \cdot c)} \cdot \overline{(\bar{a} \cdot b \cdot \bar{c})}},$$

$$S_{МКНФ} = \overline{\overline{(a + b + c)} \cdot \overline{(a + \bar{b} + \bar{c})} \cdot \overline{(\bar{a} + \bar{b} + c)} \cdot \overline{(\bar{a} + b + \bar{c})}} =$$

$$= \overline{\overline{(a \cdot b \cdot c)} \cdot \overline{(\bar{a} \cdot \bar{b} \cdot \bar{c})} \cdot \overline{(\bar{a} \cdot \bar{b} \cdot c)} \cdot \overline{(\bar{a} \cdot b \cdot \bar{c})}} = \overline{(a \cdot b \cdot c)} \cdot \overline{(\bar{a} \cdot \bar{b} \cdot \bar{c})} \cdot \overline{(\bar{a} \cdot b \cdot c)} \cdot \overline{(a \cdot \bar{b} \cdot \bar{c})} \cdot \overline{(a \cdot \bar{b} \cdot c)}.$$

Для построения $S_{МДНФ}$ и $S_{МКНФ}$ на двухвходовых элементах «и-не» потребуется соответственно 20 и 21 логический элемент. Поэтому для построения части схемы, реализующей S, выберем $S_{МДНФ}$.

$$P_{МДНФ} = \overline{\overline{a \cdot b + a \cdot c + b \cdot c}} = \overline{\overline{a \cdot b} \cdot \overline{a \cdot c} \cdot \overline{b \cdot c}},$$

$$P_{МКНФ} = \overline{\overline{(a + b) \cdot (a + c) \cdot (b + c)}} = \overline{\bar{a} \cdot \bar{b}} \cdot \overline{\bar{a} \cdot \bar{c}} \cdot \overline{\bar{b} \cdot \bar{c}}.$$

Для реализации $P_{МДНФ}$ необходимо 6 двухвходовых элементов «и-не», а для $P_{МКНФ}$ – 7 (с учётом того, что $\bar{a}, \bar{b}, \bar{c}$ уже получены в $S_{МДНФ}$).

Для построения части схемы, реализующей P, выберем $P_{МДНФ}$.

4. Построим функциональную схему сумматора, используя выражения, выбранные на третьем шаге алгоритма (рис. 1.7.).

$$S_{МДНФ} = \overline{\overline{(a \cdot b \cdot c)} \cdot \overline{(a \cdot \bar{b} \cdot \bar{c})} \cdot \overline{(\bar{a} \cdot b \cdot \bar{c})} \cdot \overline{(\bar{a} \cdot \bar{b} \cdot c)}}; \quad P_{МДНФ} = \overline{\overline{a \cdot b} \cdot \overline{a \cdot c} \cdot \overline{b \cdot c}}.$$

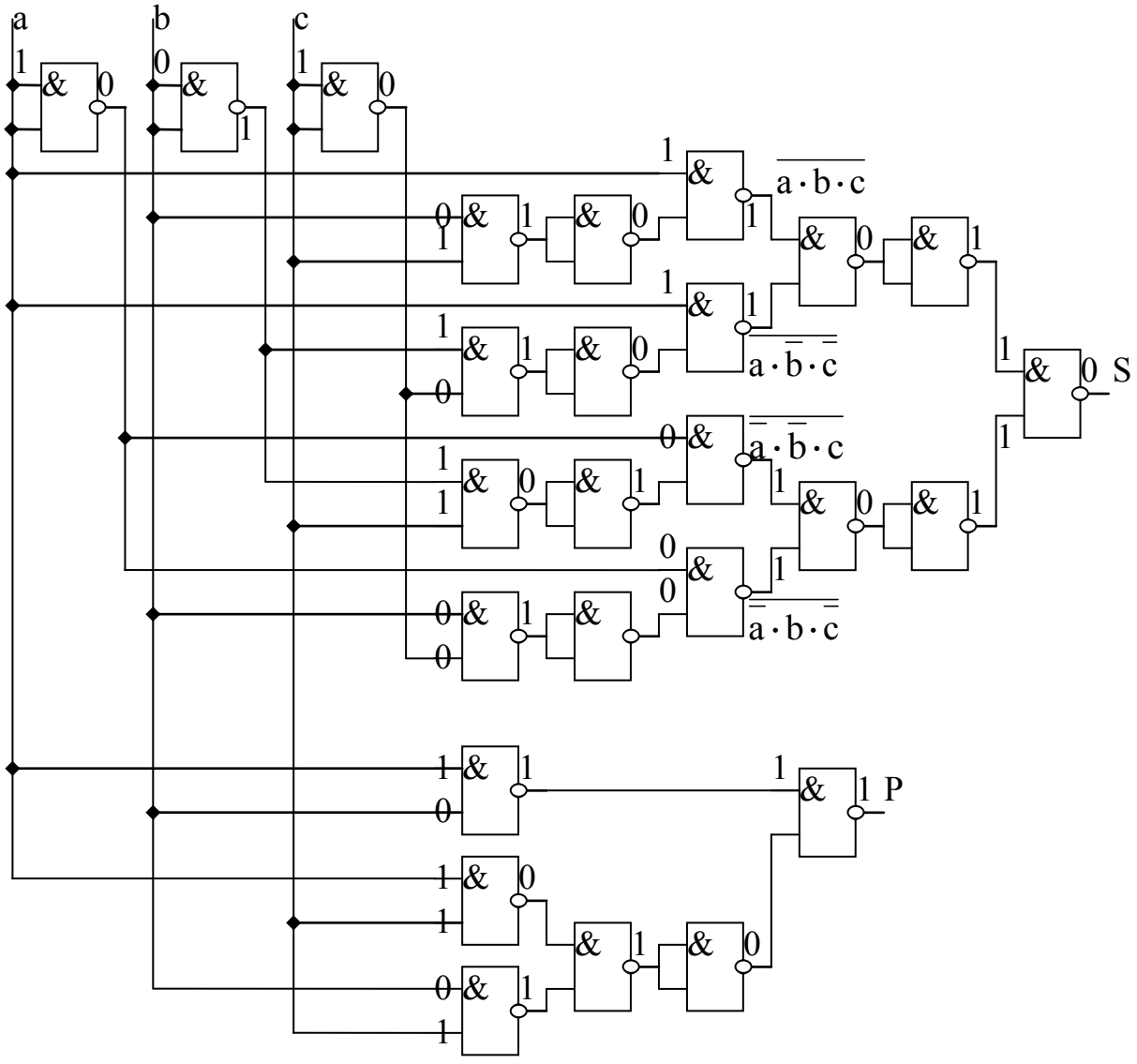


Рис. 1.7.

В данном случае построение частей схемы, реализующих S и P проводилось, независимо друг от друга.

5. На этапе схемотехнической коррекции ограничимся лишь оценкой выполнения требований по коэффициентам объединения $К_о$ и разветвления $К_р$. $К_0$ был задан равным двум, а $К_р$ для логических схем равен $5 \div 20$. Зададим $К_р$=5. Схема показывает, что для всех элементов $К_0$=2, а $К_р$<4 и, следовательно, требования по $К_о$ и $К_р$ удовлетворены.

6. Проведём тестирование схемы сумматора только на одном наборе аргументов. Пусть a=1, b=0, c=1. По ТИ этому набору соответствуют S=0 и P=1. Расположим на всех входах и выходах элементов схемы соответствующие логические значения (0 или 1) и убедимся в правильности реализации функции S и P на данном наборе аргументов (см. рисунок 1.7). Если выполнить аналогичные действия для всех наборов аргументов их ТИ, то тестирование будет полным.

Обратим внимание на то, что полученная схема, в соответствии с заданием, содержит только двухвходовые элементы «2 и-не».

Задание 67. Синтезировать комбинационную схему, закон функционирования которой задан функцией F_4 (таблица 1.16.), используя логические элементы «2 и-не».

Задание 68. Синтезировать комбинационную схему полного одноразрядного сумматора на двухвходовых логических элементах «2 или-не» и протестировать её на наборе a=1, b=1, c=0.

Задание 69. Синтезировать комбинационную схему, реализующую функцию F_1, заданную в таблице 1.18. Использовать двухвходовые логические элементы «2 или-не». Провести тестирование схемы при x_1=0, x_2=1, x_3=1, x_4=0.

Задание 70. Синтезировать комбинационную схему, реализующую функции F_2 и F_5, заданные в таблицах 1.18. и 1.19. Использовать двухвходовые логические элементы «2 и-не». Провести тестирование схем при x_1=0, x_2=1, x_3=0, x_4=1, x_5=1.

Таблица 1.18.

x_1	x_2	x_3	x_4	F_1	F_2	F_3
0	0	0	0	1	0	0
0	0	0	1	1	0	1
0	0	1	0	1	1	-
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	1
1	0	0	0	1	0	-
1	0	0	1	1	0	-
1	0	1	0	0	1	0
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	0	1	1
1	1	1	0	0	0	0
1	1	1	1	0	1	1

Таблица 1.19.

x_1	x_2	x_5	F_4	F_5	F_6
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	1	-
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	-
1	1	0	1	1	0
1	1	1	1	1	1

Задание 71. Синтезировать комбинационную схему, реализующую функции F_3 и F_6, заданные в таблицах 1.18. и 1.19. Использовать логические элементы: «не» и «2и». Провести тестирование при

$x_1=x_3=x_5=0$, $x_2=x_4=1$.

1.4.4. Односторонние булевы уравнения.

Рассмотрим метод решения односторонних булевых уравнений с одним и двумя неизвестными. Уравнения с одним неизвестным:

$$F_1(x_1,...,x_n)\cdot P(x_1,...,x_n)+\overline{F_1}(x_1,...,x_n)\cdot\overline{P}(x_1,...,x_n)=F_2(x_1,...,x_n).$$

Уравнения с двумя неизвестными:

$$F_1(x_1,...,x_n)\cdot Q(x_1,...,x_n)+\overline{F_1}(x_1,...,x_n)\cdot R(x_1,...,x_n)=F_2(x_1,...,x_n),$$

где $F_1(\ldots)$ и $F_2(\ldots)$ – известные (заданные) функции, а $P(\ldots)$, $Q(\ldots)$, $R(\ldots)$ – неизвестные функции.

Решение этих уравнений будем искать методом подбора значений P, Q, R на всех наборах F_1 и F_2. Таких наборов 4. Составим таблицу решений 1.20.

$$F_1\cdot P+\overline{F_1}\cdot\overline{P}=F_2;\ F_1\cdot Q+\overline{F_1}\cdot R=F_2.$$

Таблица 1.20.

F_1	F_2	P	Q	R
0	0	1	-	0
0	1	0	-	1
1	0	0	0	-
1	1	1	1	-

Для получения решений в виде $P=\varphi(x_1,...,x_n)$ для уравнений с одним неизвестным и $Q=\varphi_1(x_1,...,x_n)$, $R=\varphi_2(x_1,...,x_n)$ для уравнения с двумя неизвестными, необходимо сначала перейти от формы представления исходных функций F_1 и F_2 к форме представления функций P, Q, R (для этого используется таблица решений), а затем выполнить минимизацию этих функций.

Пример. Решить булевы уравнения $F_1\cdot P+\overline{F_1}\cdot\overline{P}=F_2$ и $F_1\cdot Q+\overline{F_1}\cdot R=F_2$ при F_1 и F_2, заданных в строчной форме: $F_1=\sum_4(0110110000111001)$, $F_2=\sum_4(1001011001101001)$. Выполнить переход от F_1 и F_2 к P, Q, R. Используем таблицу решений односторонних булевых уравнений.

$$F_1=\sum_4(0110110000111001)$$

$$F_2=\sum_4(1001011001101001)$$

$$P=\sum_4(0000010110101111)$$

$$Q=\sum_4(-00-01----101--1)$$

$$R=\sum_4(1--1--1001---00-)$$

Перейдём от представления P, Q и R к строчной форме к представлению картами Карно и выполним минимизацию:

карта Карно для P

X_1X_2 \ x_3x_4	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	1	1	1
10	1	0	0	1

карта Карно для Q

x_1x_2 \ x_3x_4	00	01	11	10
00	-	0	-	0
01	0	1	-	-
11	1	-	1	-
10	-	-	0	1

Карта Карно для R

x_1x_2 \ x_3x_4	00	01	11	10
00	1	-	1	-
01	-	-	0	1
11	-	0	-	0
10	0	1	-	-

Проведя минимизацию P, Q и R получим:

$P_{МДНФ} = x_2x_4 + x_1\overline{x_4}$, $Q_{МДНФ} = x_2x_4 + x_1\overline{x_4}$, $R_{МДНФ} = \overline{x_2}x_4 + \overline{x_1}\,\overline{x_4}$.

Задание 72. Решить односторонние булевы уравнения с одним и двумя неизвестными, если известные функции F_4 и F_5 заданы ТИ (таблица 1.19.).

Задание 73. Решить односторонние булевы уравнения с одним и двумя неизвестными, если известные функции F_1 и F_2 заданы ТИ (таблица 1.18.).

1.4.5. Методы синтеза комбинационных схем со многими выходами.

После раздельной минимизации булевых функций, описывающих многовыходную схему, применяются различные методы совместной обработки функций.

Рассмотрим два из них:

- метод учёта повторяющихся членов,
- метод последовательного наращивания.

Метод учёта повторяющихся членов.

В данном методе находятся общие для формул нескольких функций члены (функции, отдельные дизъюнкции, конъюнкции и их части). При

построении общей комбинационной схемы, реализующей несколько логических функций, фрагменты, соответствующие общим (повторяющимся) членам, формируются только один раз и используются для получения любой функции, куда они входят.

Пример. Синтезировать комбинационную схему, реализующую функции (рис. 1.8.):

$F_1 = x_1\overline{x_2} + \overline{x_1}x_2\overline{x_3} + \overline{x_1}x_3$, $F_2 = x_1\overline{x_2}\overline{x_3} + \overline{x_1}x_3 + x_2\overline{x_3}$, $F_3 = \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\overline{x_3}$.

Общими членами в этих выражениях являются: $x_1\overline{x_2}$, $x_2\overline{x_3}$, $\overline{x_1}x_3$, $\overline{x_1}x_2\overline{x_3}$.

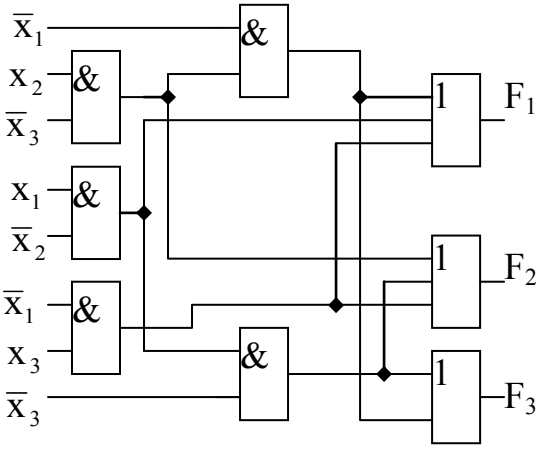


Рис. 1.8.

Задание 74. Синтезировать комбинационную схему, реализующую функции:

$F_1 = AB\overline{C} + \overline{A}\overline{B}C + BC$, $F_2 = \overline{A}B + A\overline{C} + A\overline{B}C$, $F_3 = \overline{A}B\overline{C} + A\overline{B}C$.

Метод последовательного наращивания.

Логическая схема строится сначала для какого-либо одного выхода, затем для другого, третьего и т. д.

При построении схемы последующего выхода, выходные сигналы уже построенной схемы используются на ряду с общими выходными сигналами схемы (рис. 1.9.). Следовательно, функция этого выхода может быть представлена в виде функции от (n+1) переменной (n – входных сигналов плюс выход предыдущей схемы).

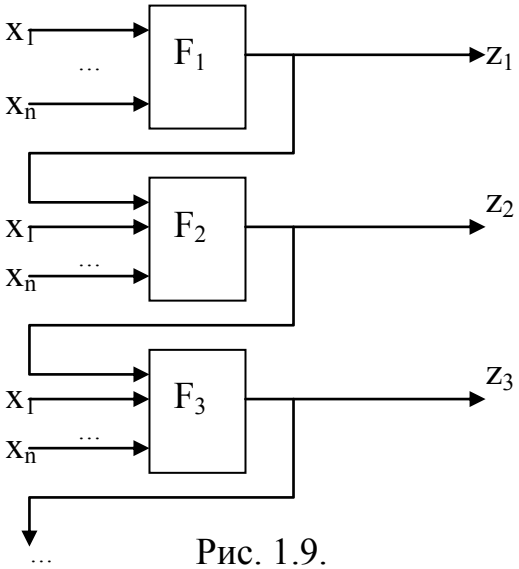


Рис. 1.9.

$$z_i = F_i(x_1,...,x_n) = F_i'(\underbrace{F_{(i-1)}, x_1,...,x_n}_{(n+1)})$$

Реализуя данный метод нужно получить специальный блок, преобразующий функцию F_{i-1} в функцию F_i. Для этого проведём разложение F_i по новой переменной F_i': $F_i'(F_{(i-1)}, x_1,..., x_n) =$ $= F_{(i-1)}F'_{1_i} + \overline{F_{(i-1)}}F'_{0_i} = F_1(x_1,..., x_n)$.

Таким образом, получаем обычное булево уравнение с двумя неизвестными $F_{(i-1)} \cdot Q_i + \overline{F_{(i-1)}} \cdot R_i = F_i$. Следовательно, при построении многовыходной схемы методом последовательного наращивания необходимо найти решение (n-1) булева уравнения и определить все Q_i и R_i.

Рекомендации. За исходную функцию F_1 можно брать любую из заданных, но, как правило, лучшие результаты получаются, если взять ту, у которой проще минимальная форма.

Если одна из исходных функций не доопределена, а вторая определена полностью, то неопределённые значения группируются так, чтобы это было наиболее выгодно при минимализации Q_i и R_i.

Структурная схема, реализующая функцию F'_i может быть представлена следующим образом (рис. 1.10.):

$F_i = F_{i-1} \cdot Q + \overline{F_{i-1}} \cdot R$.

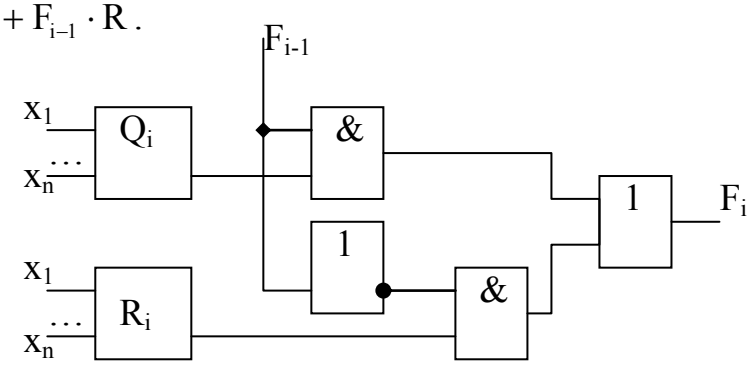


Рис. 1.10.

Пример. Синтезировать комбинационную схему полного одноразрядного двоичного сумматора методом последовательного наращивания. $P = AB + AC + BC$, $S = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$. В качестве F_1 выбираем P, как функцию с простой минимальной формой. $S = P \cdot Q + \overline{P} \cdot R$.

Для P

		AB 00	01	11	10
C	0	0	0	1	0
	1	0	1	1	1

Для S

		AB 00	01	11	10
C	0	0	1	0	1
	1	1	0	1	0

C	AB	00	01	11	10
	0	-	-	0	-
	1	-	0	1	0

Для Q

C	AB	00	01	11	10
	0	0	1	-	1
	1	1	-	-	-

Для R

Q=ABC, R=A+B+C.

Проведём тестирование построенной схемы(см.рис.1.11) на наборе аргументов A=1, B=0, C=1. Этому набору должны соответствовать значения S=0 и P=1. Проставленные логические значения на входах и выходах элементов схемы показывают работоспособность схемы (на данном наборе).

При синтезе многовыходных схем методом последовательного наращивания удаётся, обычно, построить схему с использованием меньшего количества логических элементов, чем при синтезе с независимым построением частей схемы, реализующих функции выходов. Но при этом получается некоторый проигрыш в быстродействии схемы.

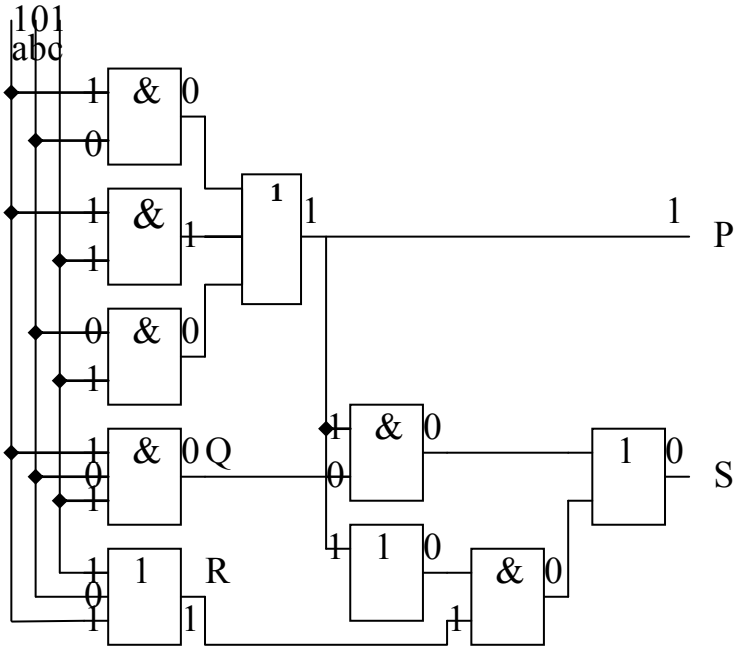


Рис. 1.11.

Задание 75. Реализовать рассмотренную схему только на двухвходовых элементах «и-не» и, проведя сравнение полученной реализации со схемой, определить выигрыш (проигрыш) в количестве элементов схемы и её быстродействии.

Задание 76. Используя условия задания 70 синтезировать двухвыходную комбинационную схему методом последовательного наращивания.

Задание 77. Используя условия задания 71 синтезировать двухвыходную комбинационную схему методом последовательного наращивания.

1.5. Синтез и анализ конечных автоматов.

Определение. Если совокупность выходных сигналов (выходного слова) Y(t) зависит, как от совокупности входных сигналов (входного слова) X(t), так и от внутренних состояний S(t-1), то такой преобразователь информации называется конечным автоматом (автоматом с памятью).

Для описания функционирования конечного автомата (КА) задаются:

$X(t)=\{x_1,...,x_i,...,x_n\}$ - множество входных дискретных сигналов, $x_i \in \{0,1\}, i=\overline{1,n}$;

$Y(t)=\{y_1,\ldots,y_j,\ldots,y_m\}$ – множество выходных дискретных сигналов, $y_j \in \{0,1\}, j=\overline{1,m}$;

$S(t)=\{s_1,\ldots,s_k,\ldots s_r\}$ – множество внутренних дискретных состояний, $s_k \in \{0,1\}, k=\overline{1,r}$;

s_0 – начальное состояние автомата;

FP – функция переходов, позволяющая определить новое внутреннее состояние автомата, если известно предыдущее внутреннее состояние и состояние входных сигналов в данных момент времени S(t+1)=FP[S(t),X(t)];

FV – функция выходов, позволяющая определить состояние выходных сигналов автомата, если известно внутреннее состояние и состояние входных сигналов Y(t)=FV[S(t),X(t)].

Таким образом, функционирования КА можно представить так:

$$\begin{cases} S(t+1)=FP[S(t),X(t)], \\ Y(t)=FV[S(t),X(t)], \text{ где } t=0,1,2\ldots \\ S(t=0)=s_0 \end{cases}$$

Автомат Мили

или так:

$$\begin{cases} S(t+1)=FP[S(t),X(t)], \\ Y(t)=FV[S(t)], \text{ где } t=0,1,2\ldots \\ S(t=0)=s_0 \end{cases}$$

Автомат Мура

Конечные автоматы делятся на 2 части:

Абстрактную – описание полноты FV, X, Y, S автомата, работоспособности

Структурную – отвечает на вопрос как построить КА.

При описании работы конечного автомата используют таблицы состояний или графы (диаграммы состояний).

таблицы состояний:

таблица переходов

	s_0	s_1	…	s_r
x_1	s_3	s_0	…	…
x_2	s_1	s_3	…	…
…	…	…	…	…
x_n	…	…	…	…

таблица выходов

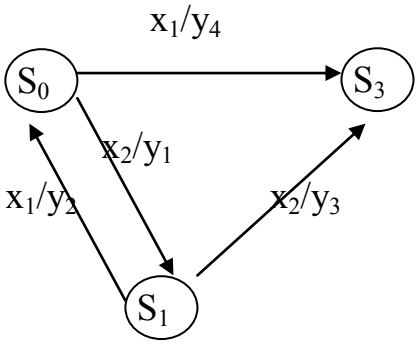
	s_0	s_1	…	s_r
x_1	y_4	y_2	…	…
x_2	y_1	y_3	…	…
…	…	…	…	…
x_n	…	…	…	…

совмещённая таблица

	$s_0(t)$	$s_1(t)$	…
$x_1(t)$	$s_3(t+1) / y_4(t)$	$s_0(t+1) / y_2(t)$	…
$x_2(t)$	$s_1(t+1) / y_1(t)$	$s_3(t+1) / y_3(t)$	…
…	…	…	…

(фрагмент)

Задание КА графом (соответствует содержимому таблиц состояний).



Различают два основных типа КА: автомат Мили и автомат Мура. Различие между ними заключается в том, что в автомате Мура состояние выходных сигналов не зависит от состояния входных сигналов, а определяется лишь внутренним состоянием, т. е. Y(t)=FV[S(t)].

Структурная схема КА (рис. 1.12. и 1.13.)

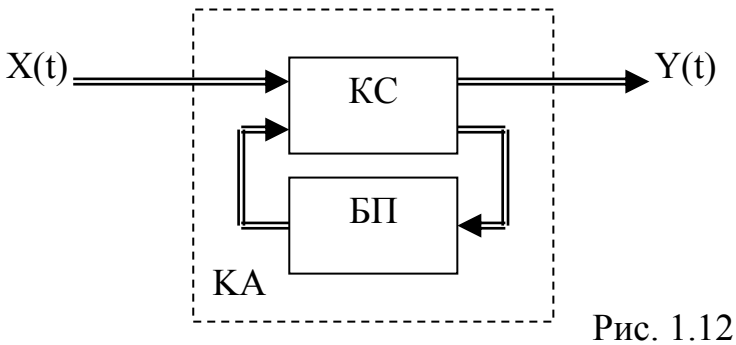


Рис. 1.12.

или несколько подробнее

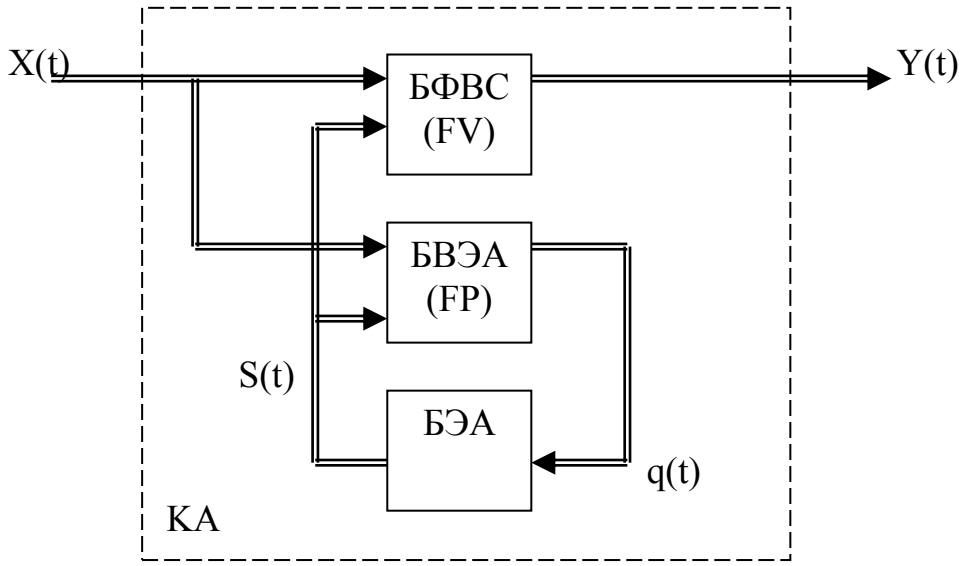


Рис. 1.13.

где:

КС – комбинационная схема;

БП – блок памяти;

БФВС – блок формирования выходных сигналов (комбинационная схема, реализующая FV);

БВЭА – блок возбуждения элементарных автоматов (комбинационная схема, реализующая FP);

БЭА – блок элементарных автоматов (память КА);

q – функция возбуждения.

1.5.1. Элементарные конечные автоматы и их техническая реализация.

Функция памяти КА реализуется на элементарных автоматах. Элементарный автомат (ЭА) – это автомат Мура, имеющий два и только два различных состояния, имеющий 1 или 2 входа, при подаче сигналов на которые возможен переход из одного состояния в другое.

Для построения памяти КА используются, в основном, четыре типа ЭА: два типа одновходовых и два двухвходовых.

Одновходовые ЭА (Табл. 1.21.):
$Q_1(t+1)=q(t)$ – ЭА D-типа,

Таблица 1.21.

q(t)	0 0 1 1
Q(t)	0 1 0 1
$Q_1(t+1)$	0 0 1 1
$Q_2(t+1)$	0 1 1 0

$Q_2(t+1) = q(t) \oplus Q(t)$ - ЭА Т-типа.

Таблица 1.22.

$q_0(t)$	0 0 0 0 1 1 1 1
$q_1(t)$	0 0 1 1 0 0 1 1
Q(t)	0 1 0 1 0 1 0 1
$Q_3(t+1)$	0 1 1 1 0 0 - -
$Q_4(t+1)$	0 1 1 1 0 0 1 0

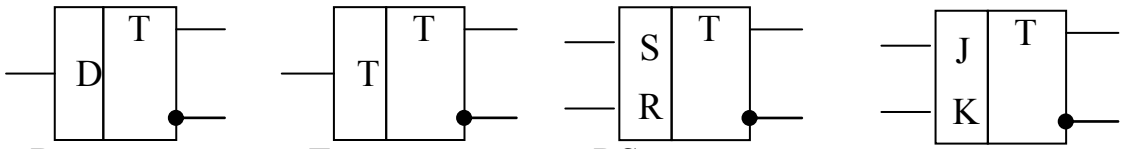
Двухвходовые ЭА (Табл. 1.22.):

$Q_3(t+1) = \overline{q_0(t)} \cdot Q(t) + q_1(t)$ - ЭА RS-типа,

$Q_4(t+1) = \overline{q_0(t)} \cdot Q(t) + q_1(t) \cdot \overline{Q(t)}$ - ЭА JK-типа.

Техническая реализация ЭА:

Автомат D-типа Автомат T-типа Автомат RS-типа Автомат JK-типа



D - триггер T - триггер RS - триггер JK - триггер

$q_1(t)=q_S(t)$, $q_1(t)=q_J(t)$, $q_0(t)=q_R(t)$, $q_0(t)=q_K(t)$.

1.5.2. Алгоритм структурного синтеза конечного автомата.

1. Задать закон функционирования конечного автомата.

2. Определить n – требуемое количество ЭА. $n=\lceil \log_2(r+1) \rceil$, где r+1 – количество внутренних состояний ЭА, $\lceil \ldots \rceil$ - ближайшее большее целое.

3. Выбрать тип ЭА по таблице. Для этого предварительно построить таблицу для требуемой функции возбуждения.

Таблица 1.23.

$Q(t) \to Q(t+1)$	тип ЭА					
	D	T	RS		JK	
	$q_D(t)$	$q_T(t)$	$q_S(t)$	$q_R(t)$	$q_J(t)$	$q_K(t)$
$0 \to 0$	0	0	0	-	0	-
$0 \to 1$	1	1	1	0	1	-
$1 \to 0$	0	1	0	1	-	1
$1 \to 1$	1	0	-	0	-	0

4. Построить функциональную схему БЭА.

5. Провести синтез комбинационной схемы БВЭА.

6. Провести синтез комбинационной схемы БФВС.

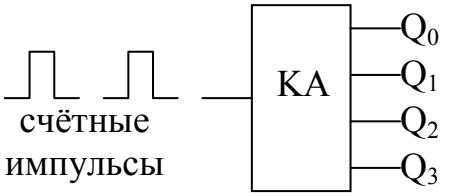
7. Составить из БЭА, БВЭА и БФВС схему конечного автомата.

8. Провести тестирование полученной схемы на соответствие закону функционирования.

1.5.3. Пример синтеза конечного автомата – двоично-десятичного счётчика.

Пример. Синтезировать схему двоично-десятичного счётчика.

Зададим закон функционирования двоично-десятичного счётчика. Этот КА имеет 10 внутренних состояний:



	0	1	2	3	4	5	6	7	8	9
Q_0	0	1	0	1	0	1	0	1	0	1
Q_1	0	0	1	1	0	0	1	1	0	0
Q_2	0	0	0	0	1	1	1	1	0	0
Q_3	0	0	0	0	0	0	0	0	1	1

Так как $n=\lceil \log_2 10 \rceil=4$, то для запоминания всех 10 состояний КА потребуется четыре ЭА.

Выберем тип ЭА. Пусть это будет ЭА Т-типа. Тогда структурная схема синтезируемого КА будет иметь вид (рис. 1.14.):

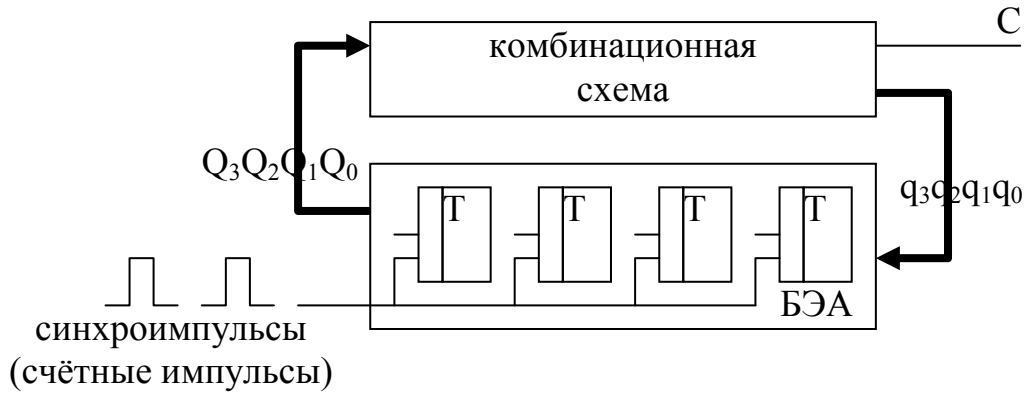


Рис. 1.14.

Выполним синтез БВЭА. Построим либо совмещённую таблицу состояний 1.24., либо диаграмму состояний (рис. 1.15.):

Таблица 1.24.

текущее состояние				следующее состояние				выходы комбинационной схемы					
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	q_3	q_2	q_1	q_0	/	c
0	0	0	0	0	0	0	1	0	0	0	1	/	0
0	0	0	1	0	0	1	0	0	0	1	1	/	0
0	0	1	0	0	0	1	1	0	0	0	1	/	0
0	0	1	1	0	1	0	0	0	1	1	1	/	0
0	1	0	0	0	1	0	1	0	0	0	1	/	0
0	1	0	1	0	1	1	0	0	0	1	1	/	0
0	1	1	0	0	1	1	1	0	0	0	1	/	0
0	1	1	1	1	0	0	0	1	1	1	1	/	0
1	0	0	0	1	0	0	1	0	0	0	1	/	0
1	0	0	1	0	0	0	0	1	0	0	1	/	1

Над дробью в диаграмме ничего нет, т. к. входные сигналы отсутствуют в данной схеме. Под дробью значение выхода С.

Построим карты Карно для выходных сигналов БВЭА (т. е. для сигналов q_0, q_1, q_2, q_3).

Диаграмма состояний

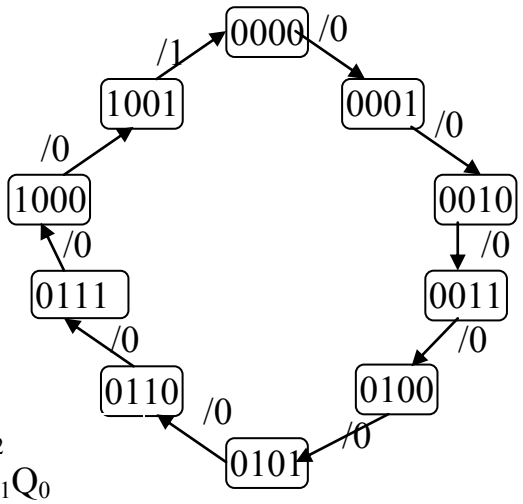


Рис. 1.15.

КК для q_3

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	-	-	-	-
10	0	1	-	-

КК для q_2

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	-	-	-	-
10	0	0	-	-

КК для q_1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	-	-	-	-
10	0	0	-	-

КК для q_0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	-	-	-	-
10	1	1	-	-

$$q_3 = Q_3Q_0 \vee Q_2Q_1Q_0, q_2 = Q_1Q_0, q_1 = \overline{Q_3}Q_0, q_0 = 1.$$

Проведём синтез БФВС. Построим карту Карно для выходных сигналов комбинационной схемы БФВС. (т. е. для сигнала С).

КК для С

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	-	-	-	-
10	0	1	-	-

$C=Q_3Q_0$.

Составим из БЭА, БВЭА и БФВС схему конечного автомата (рис. 1.16.):

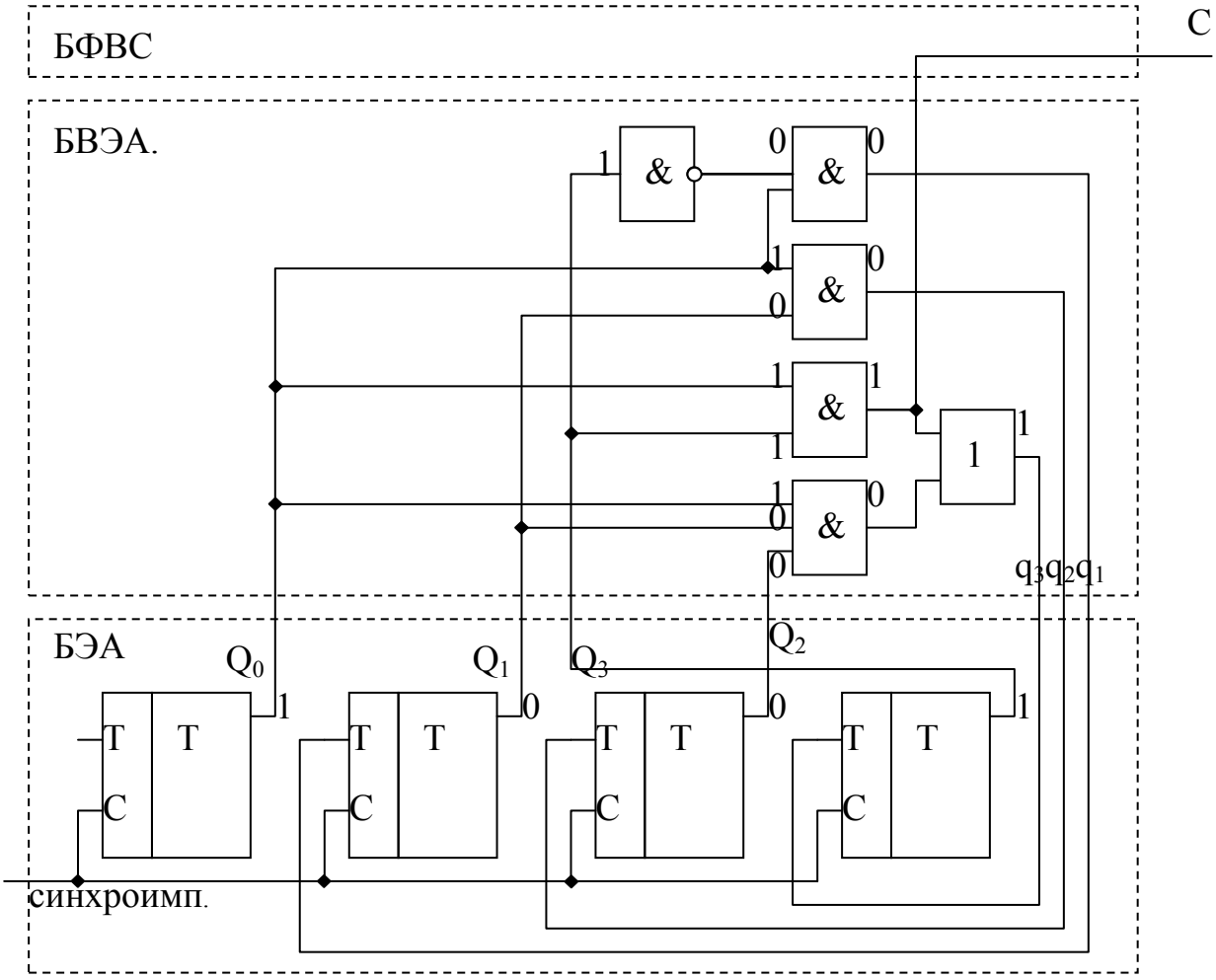


Рис. 1.16.

Вспомним соответствие с общей структурой КА (рис. 1.17.):

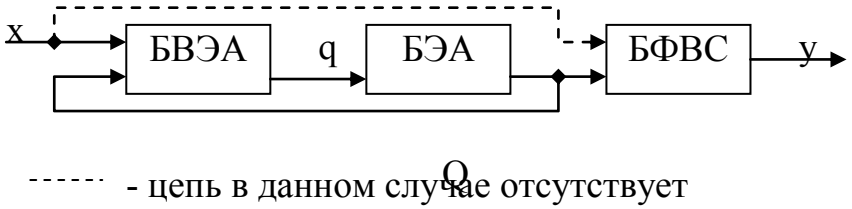


Рис. 1.17.

Проверим работоспособность схемы на примере перехода из состояния 9 в 0 и формирования С. В этом состоянии Q_3=1, Q_2=0, Q_1=0, Q_0=1 (см. 1 и 0 в схеме). При этом сформируются сигналы возбуждения: q_3=1, q_2=0, q_1=0, q_0=1 и, следовательно, следующим состоянием КА будет: Q_3=Q_2=Q_1=Q_0=0 и С=1, что соответствует правильной работе двоично-десятичного счётчика.

Задание 78. Синтезировать схему КА, который при подаче на вход счётных импульсов переходит последовательно в состояния $S_0 \div S_4$ в соответствии с рисунком 1.18.

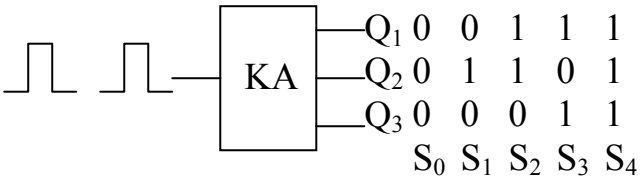


Рис. 1.18.

Задание 79. Синтезировать схему КА, который на входные счётные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 4, 5, 7, 9. В качестве ЭА выбрать Т-триггеры.

Задание 80. Синтезировать схему КА в соответствии с заданием №79, но с использованием в качестве ЭА RS-триггеров.

2. Организация ЭВМ.

2.1. Типовые структурные элементы цифровой техники.

2.1.1. Основные характеристики и классификация цифровых элементов вычислительной техники.

Типовыми структурными элементами называются наименьшие функциональные части, из которых состоят устройства цифровой техники, а именно:

- элементы, реализующие логические функции;
- элементы, преобразующие информацию;
- элементы, запоминающие информацию;
- вспомогательные элементы (генерирующие, формирующие и усиливающие сигналы).

По способу представления информации структурные элементы делятся на:

- импульсные («0» - отсутствие импульса напряжения, «1» - наличие импульса напряжения)
- потенциальные («0» - один уровень напряжения, «1» - другой уровень напряжения)

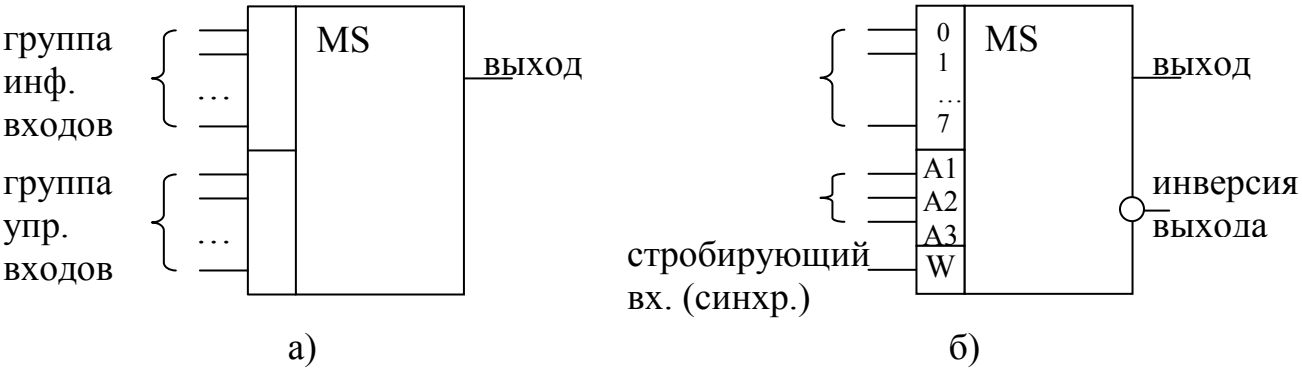
Для систематизации структурных элементов используются условные обозначения и маркировка микросхем. Необходимые сведения о них можно получить в справочной литературе. Например:

- реализуемая логическая функция;
- конструктивные и технологические признаки;

- номера элементов обозначения;
- нагрузочная способность;
- коэффициент объединения по входу;
- средняя задержка передачи сигнала;
- предельная рабочая частота;
- помехоустойчивость;
- потребляемая мощность и т.д.

2.1.2. Мультиплексоры и дешифраторы.

I. Мультиплексор – схема, передающая сигналы с одной из нескольких входных линий в выходную линию (рис. 2.1.). Основное назначение такого рода устройств – формирование сигналов для управления другими устройствами цифровой системы.



а) б)

Рис. 2.1.

В реальных условиях число информационных и управляющих входов ограничено возможностями конкретной схемотехнической элементной базы. Достаточно часто встречаются ситуации, когда использование определенной серии микросхем является наиболее предпочтительной: несмотря на их существенно недостаточные функциональные возможности. Это связано с тем, что расширение числа информационных входов приводит к значительным временным и/или энергетическим затратам. При выборе определенной серии микросхем возникают ограничения с ней связанные (см. рис. 2.1.б.): количество информационных входов не должно превышать допустимого, например, восьми. Принцип функционирования микросхемы заключается в следующем.

При подаче на управляющие входы двоичного кода и на вход W нуля, к выходу подключается только тот вход, номер которого в десятичном изображении совпадает со значением двоичного кода на управляющих входах. Информация на других информационных входах не влияет.

Если необходимо решить задачу построения N-канального мультиплексора на базе (N-1)-го, то возможно следующее решение. Допустим, существует мультиплексор, для которого составлена таблица истинности для функции трех переменных (табл. 2.1.) и представлено схемотехническое решение (рис. 2.2.).

Таблица 2.1.

x_1	x_2	x_3	y
0	0	0	f(0,0,0)
1	0	0	f(1,0,0)
…	…	…	…
1	1	1	f(1,1,1)

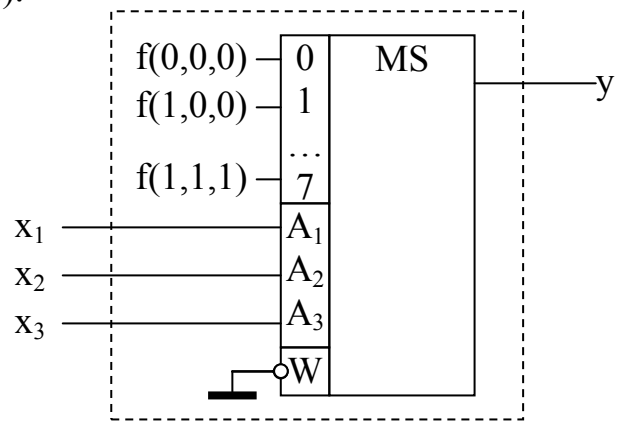


Рис. 2.2.

При необходимости реализации пятивходового мультиплексора на данной элементной базе (см. рис. 2.2.) следует построить таблицы истинности для 4-х (табл. 2.2.) и 5-ти (табл. 2.3.) переменных. Схемотехническое решение

Таблица 2.2.

x_1	x_2	x_3	$f(x_4)$
0	0	0	$f(0,0,0,x_4)$

Таблица 2.3.

x_1	x_2	x_3	x_4	$y(x_5)$
0	0	0	0	$f(0,0,0,0,x_5)$
0	0	0	1	$f(0,0,0,1,x_5)$

будет выглядеть в следующем виде (рис. 2.3.):

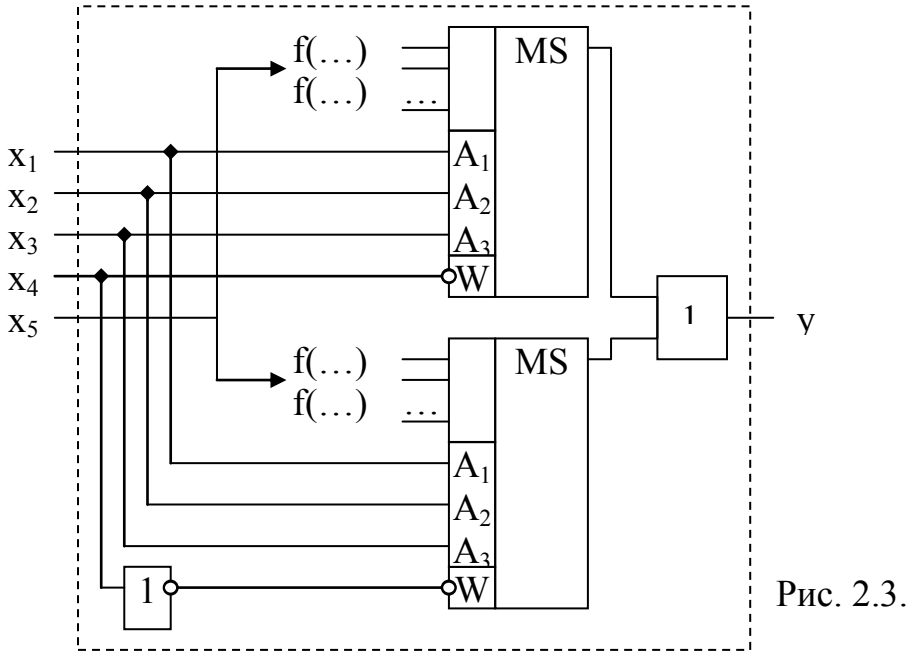


Рис. 2.3.

По аналогии – можно построить мультиплексоры с произвольным числом входов и выходов.

II. Дешифратор – КС, преобразующая код, подаваемый на входы, в сигнал на одном из выходов. Если на входы дешифратора подается двоичный код числа, то функционирование двоичного дешифратора можно описать с помощью выражений:

$$\begin{cases} y_0 = \overline{x_n}\,\overline{x_{n-1}}\ldots\overline{x_2}\,\overline{x_1} \\ y_1 = \overline{x_n}\,\overline{x_{n-1}}\ldots\overline{x_2}\,x_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_{2^n-1} = x_n x_{n-1}\ldots x_2 x_1 \end{cases}, \qquad (2\text{-}1)$$

где $x_1, x_2, \ldots, x_n$ – сигналы на входах дешифратора (n-входов); $y_0, y_1, \ldots$ – сигналы на выходах дешифратора (2^n-выходов). В общем виде дешифратор можно представить в следующем виде (рис. 2.4.):

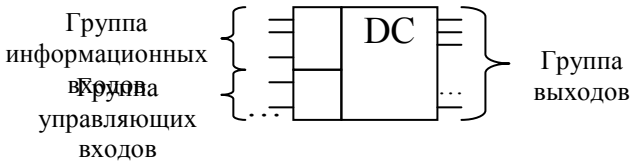


Рис. 2.4.

Из системы уравнений (2-1) видно, что дешифратор представляет собой совокупность схем совпадений, формирующих управляющий (рабочий) сигнал только на одном из выходов (см. рис. 2.5.).

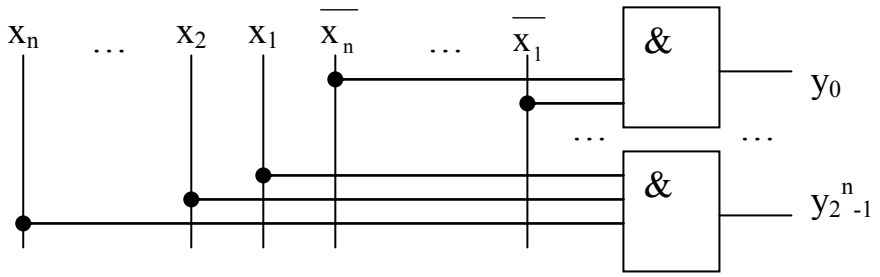


Рис. 2.5.

В зависимости от разрядности дешифрируемого кода и функциональных возможностей ряда логических интегральных схем дешифратор может быть выполнен на основе одноступенчатой (линейной) и многоступенчатой схем дешифрации (см. рис. 2.6.). Среди

многоступенчатых схем выделяют прямоугольные (матричные) и пирамидальные схемы построения дешифрации.

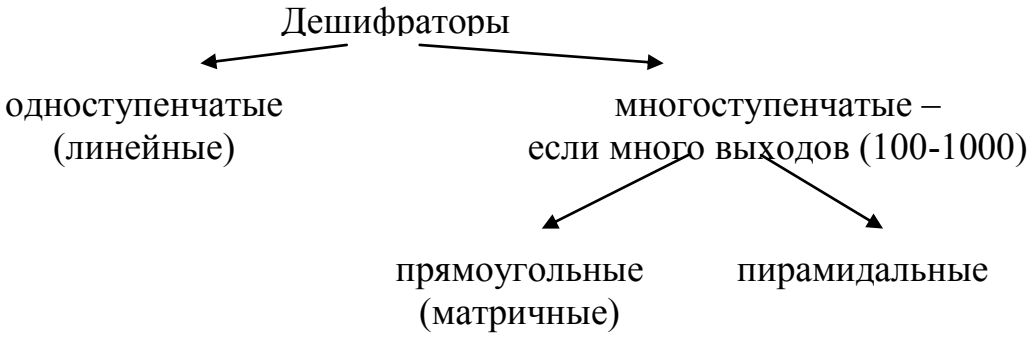


Рис. 2.6.

Линейные дешифраторы выполняются прямой схемной реализацией выражения вида (2-1), что соответствует схемотехническому решению, представленному на рис. 2.5.

Необходимость реализации многоступенчатых схем дешифрации в ЭВМ обусловлена решением задач идентификации: для устройств, процессов, программ, приоритетов и т.д. В целом, данная задача решается как построение многоступенчатых схем.

Прямоугольные (матричные) дешифраторы.

Процесс дешифрации разбивается на этапы. Первоначально, линейные дешифраторы (ЛД) анализируют входную последовательность, а далее проводится анализ совпадения выходных сигналов по матричной схеме пар линейных дешифраторов на двухвходовых элементах. Примеры такого рода реализаций представлены на рис. 2.7. (для двухступенчатой схемы) и на рис. 2.8. (для 3-х ступенчатой схемы, где МД – матричный дешифратор).

Синтез матричного дешифратора при n=4, n^4=16 выходов:

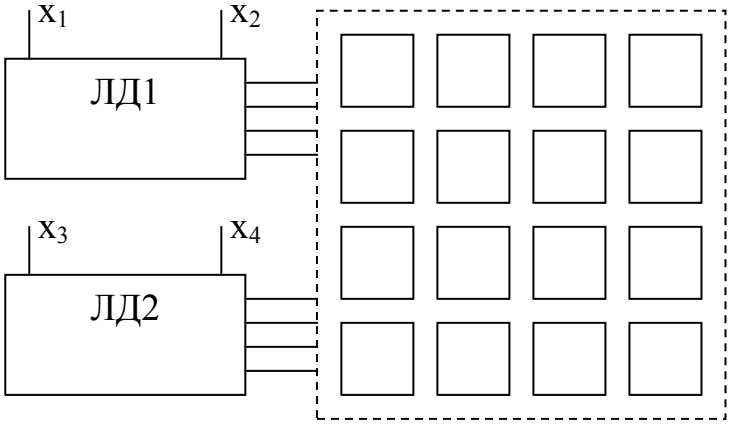


Рис. 2.7.

Пример 3-х ступенчатого матричного дешифратора при n=8:

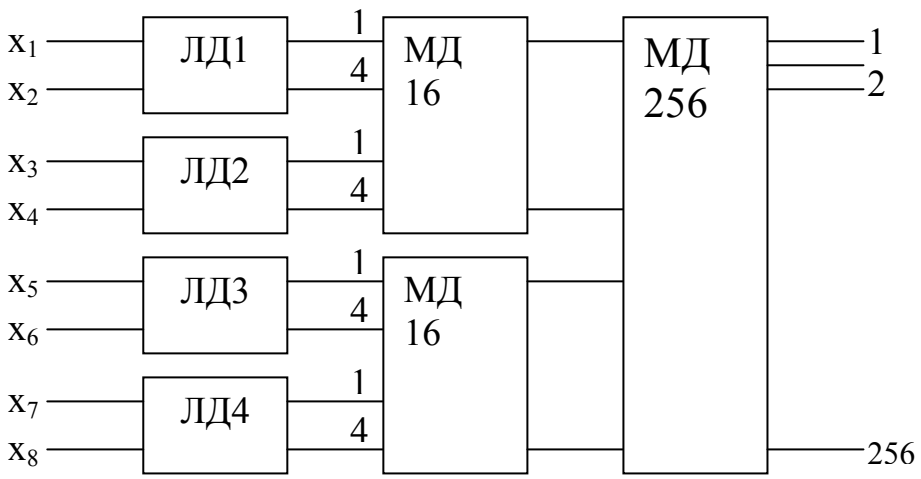


Рис. 2.8.

Пирамидальные дешифраторы.

Пирамидальные дешифраторы относятся к разряду многоступенчатых дешифраторов. Их специфической особенностью является применение во всех ступенях дешифрации двухвходовых элементов с обязательным подключением выхода элемента i-той ступени ко входам только двух элементов (i+1)-ой ступени (см. рис. 2.9.).

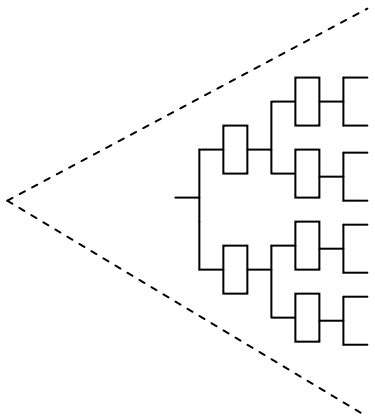


Рис. 2.9.

Следует обратить внимание на то, что такого рода схемотехнические решения позволяют существенно экономить на элементной базе, но при этом снижается быстродействие системы в целом.

2.1.3. Сумматоры.

Сумматор является одной из основных частей вычислительных устройств, предназначенной для выполнения арифметических и логических операций над числами. Числа могут быть представлены в двоичном, двоичнодесятичном, троичном и других кодах. В зависимости от СС различаются и сумматоры. В данном пособии речь идет только о двоичной системе счисления, то есть о двоичных сумматорах.

Для суммирования могут быть использованы одноразрядные или многоразрядные сумматоры, а само суммирование осуществляется: последовательно (начиная с младшего разряда) (рис. 2.10.); параллельно (суммируются все разряды одновременно) (рис. 2.11.); параллельно-последовательно (число разбивается на группы, суммирование разрядов в группах осуществляется параллельно, а группы суммируются последовательно) (рис. 2.12.).

Следует различать и помнить, что операция сложения в ЭВМ, в отличие от суммирования, требует различать знаки, проводить нормализацию, выравнивать порядки и т.д., то есть функционально она более насыщена (затратна).

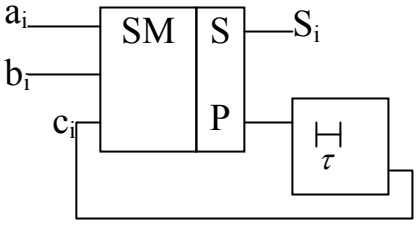


Рис. 2.10.

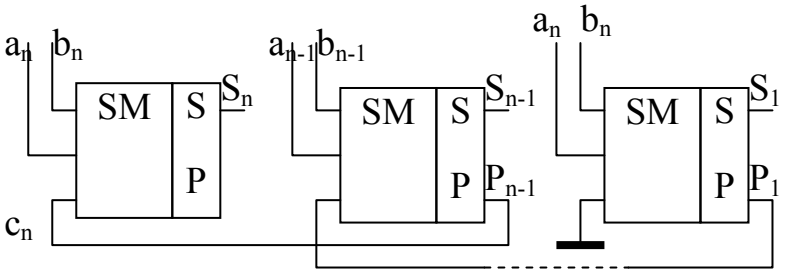


Рис. 2.11.

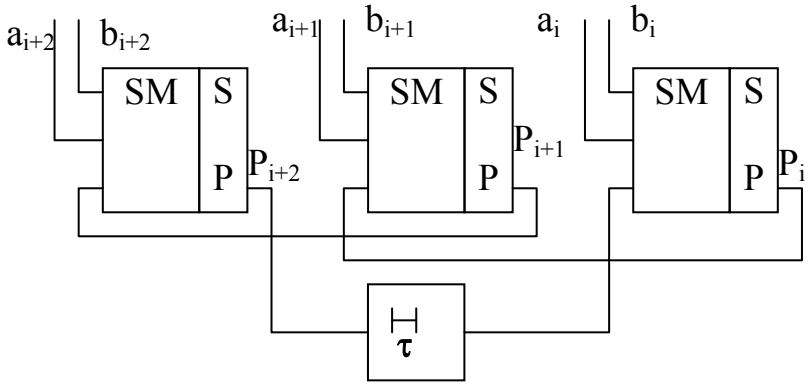


Рис. 2.12.

На вход сумматора (SM) поступают сигналы a_i и b_i. Результат формируется на выходе в виде сигнала S_i. Если при суммировании возникает переполнение, то это отображается на выходе P_i. Работа сумматора описывается с помощью таблицы истинности. Например, для последовательного сумматора необходимо учитывать предшествующее состояние. Сигнал с выхода P_i с задержкой T (один такт) поступает на вход c_i.

2.1.4. Триггеры.

Триггеры – это элементы цифровой техники, позволяющие формировать на выходе два устойчивых состояния. Единица информации, хранимая триггером – один бит. Как технический элемент, триггер состоит из двух частей: регенеративной схемы (собственно триггер) и схемы управления. Существует большая номенклатура интегральных схем, реализующих функциональные возможности триггерных устройств. При этом в качестве базовых используются нижеперечисленные.

Триггер RS-типа

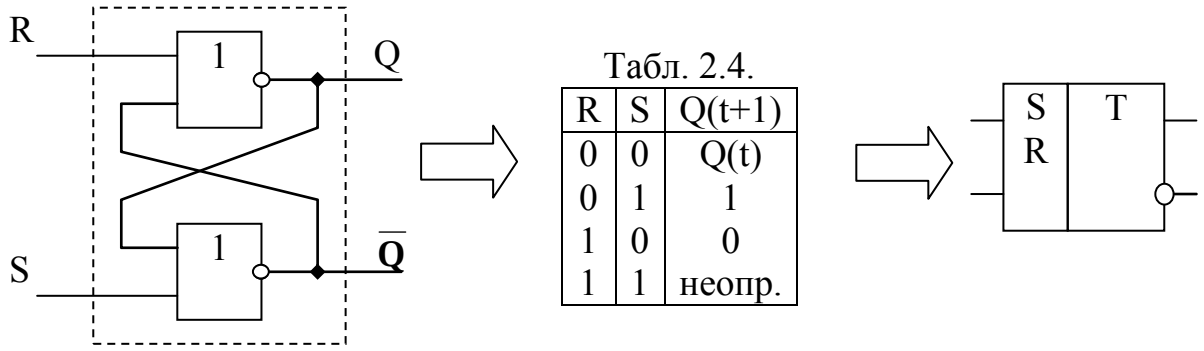


Табл. 2.4.

R	S	Q(t+1)
0	0	Q(t)
0	1	1
1	0	0
1	1	неопр.

Рис. 2.13.

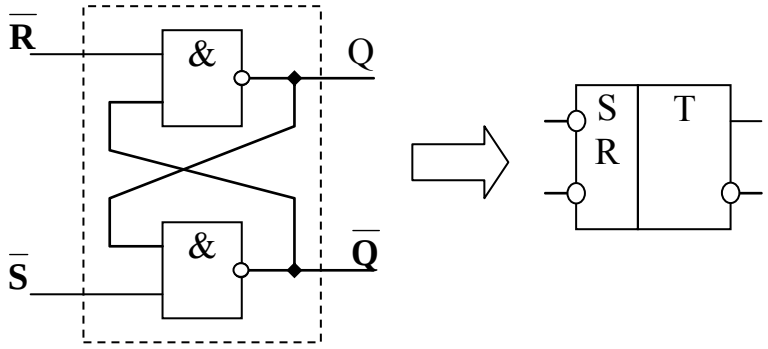


Рис. 2.14.

CRS-триггер (синхронизируемый RS-триггер).

S и R – информационные входы, C – вход синхронизации (clock – времязадающий)

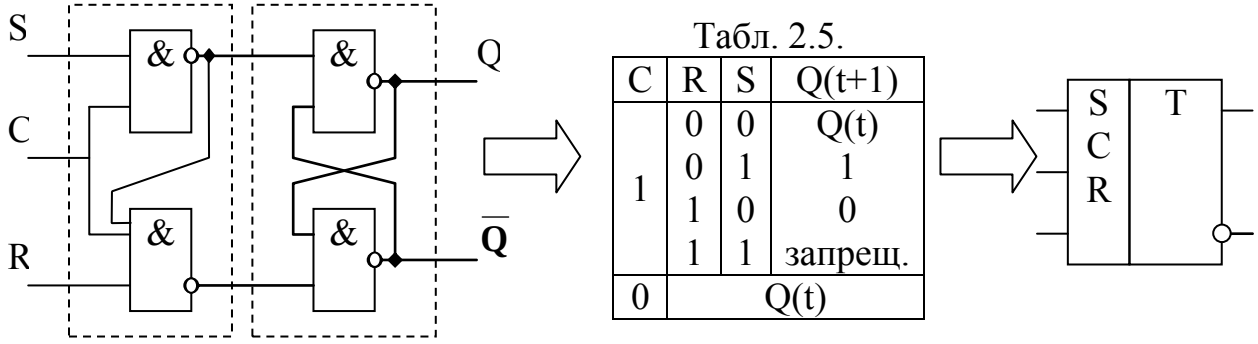


Табл. 2.5.

C	R	S	Q(t+1)
1	0	0	Q(t)
	0	1	1
	1	0	0
	1	1	запрещ.
0	Q(t)		

Рис. 2.15.

D-триггер (“delay” – задержка)

Исключается возможность одновременной подачи 2-х единиц на S и R входы. Отсутствуют запрещенные состояния (см. табл. 2.6. и рис. 2.16.).

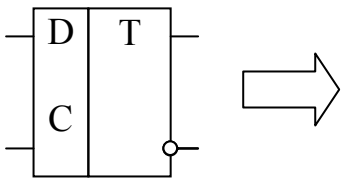


Табл. 2.6.

C	D	Q(t+1)
1	0	0
1	1	1
0	Q(t)	

Рис. 2.16.

Двухтактный RS-триггер.

Этот триггер состоит из двух RS-триггеров и инвертора (см. рис. 2.17.).

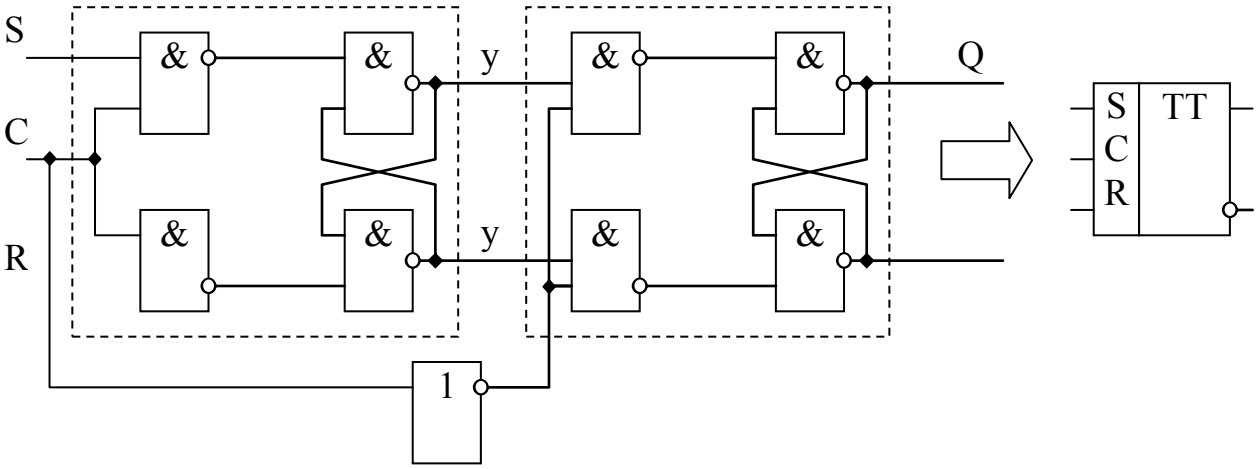


Рис. 2.17.

Временная диаграмма функционирования двухтактного RS-триггера представлена на рис. 2.18.

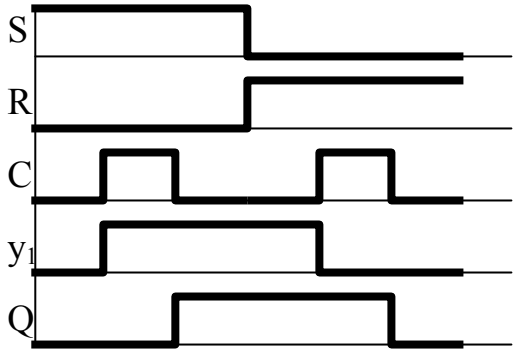


Рис. 2.18.

Т-триггер (счётный)

Алгоритм работы триггера соответствует рис. 2.19.

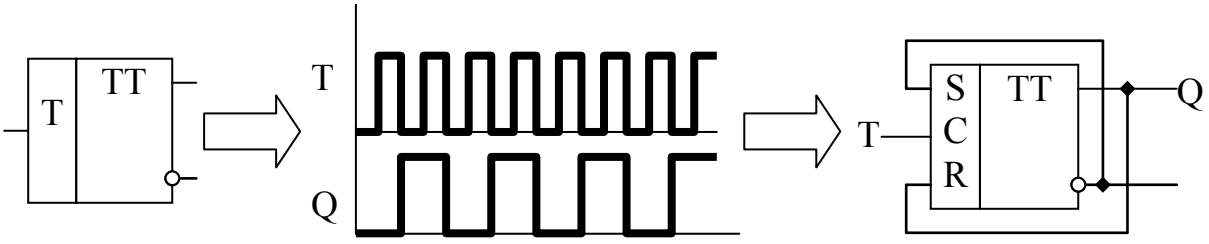


Рис. 2.19.

JK-триггер

Триггер построен на базе двухтактного. Принцип работы соответствует табл. 2.7., а принцип функционирования показан на рис. 2.20.

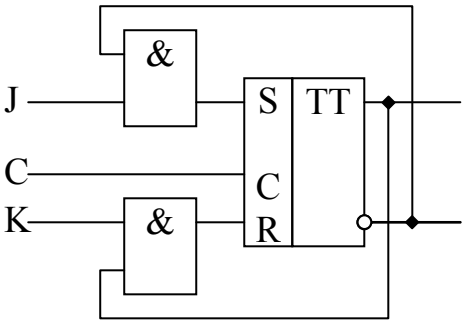
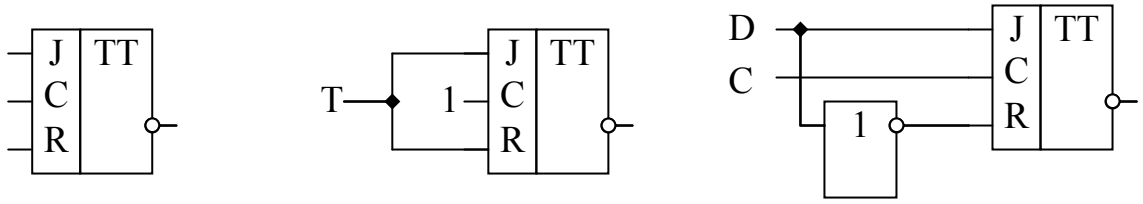


Табл. 2.7.

C	J	K	Q(t+1)
1	0	0	Q(t)
	0	1	0
	1	0	1
	1	1	$\overline{Q(t)}$
0	Q(t)		

Рис. 2.20.

Триггерные устройства позволяют реализовывать различные алгоритмы функционирования на конкретной элементной базе. Например, используя JK-триггер (рис. 2.21.а.), можно построить Т-триггер (рис. 2.21.б.) или D-триггер (рис. 2.21.в.).



а) б) в)

Рис. 2.21.

2.1.5. Регистры и счётчики.

I. Регистры.

Регистрами называются устройства, выполняющие функции приема, хранения и передачи информации. Информация в регистре хранится в виде комбинации сигналов 0 и 1.

Каждый разряд числа записывается в регистр, реализованный на базе триггерных устройств.

Регистры позволяют выполнять следующие преобразования:

- сдвиг слова влево или вправо на требуемое число разрядов;
- преобразование прямого кода в обратный (и наоборот);
- преобразование параллельного кода в последовательный (и наоборот);
- выполнение поразрядных логических операций (конъюнкций, дизъюнкций, сложение mod2 и др.)

Основная функция регистра – запоминание.

По способу записи информации регистры подразделяются на три типа:

- параллельные (статические);
- последовательные (сдвигающие);
- параллельно-последовательные.

В зависимости от числа каналов, по которым поступает информация на входы разрядов регистров, различают регистры парафазного и однофазного видов. У парафазных регистров информация на каждый разряд поступает по прямому и инверсному входам, а у однофазных – только на прямой, либо на инверсный. Наиболее часто парафазные регистры реализуются на базе RS-триггеров, а однофазные – D-триггеров.

Элементная база регистров разбивает их на многотактные и однотактные. Первый тип наиболее часто реализуется на основе тактируемых RS и D-триггеров, а второй – RS_τ , D_τ , JK_τ-триггеров с внутренней задержкой.

А. Статический регистр.

В статическом регистре запись (считывание) числа осуществляется параллельным кодом, то есть во все разряды регистра одновременно.

На рис. 2.22. показан пример записи информации в прямом или обратном кодах (ОК), а на рис. 2.23. – считывания (или преобразования из одного вида в другой).

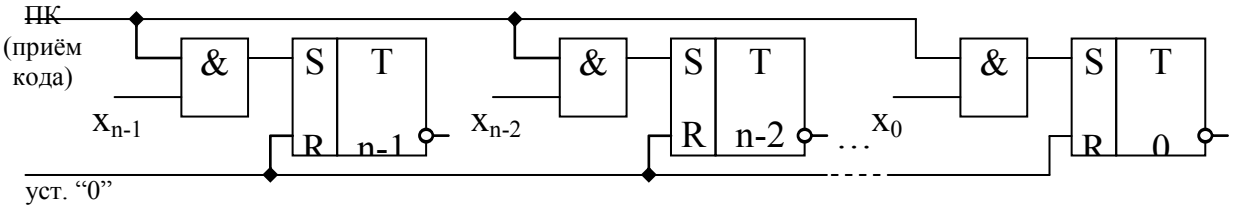


Рис. 2.22.

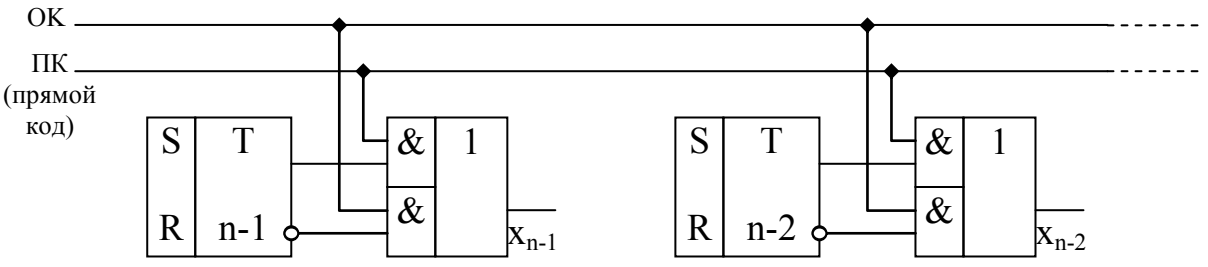


Рис. 2.23.

Выполнение логических функций может быть реализовано в следующем виде (см. рис. 2.24., 2.25. и 2.26., где ПК – прием кода):

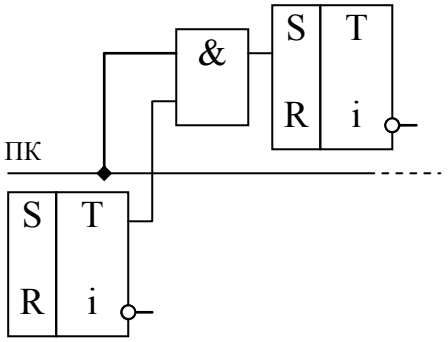


Рис. 2.24. Дизьюнкция

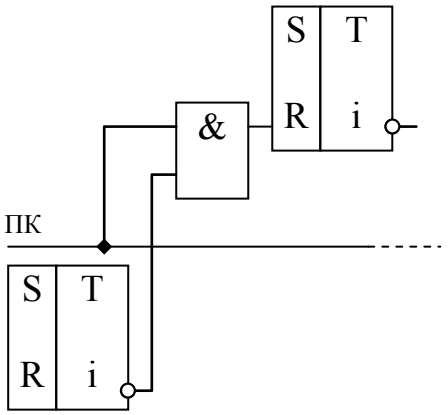


Рис. 2.25. Коньюнкция

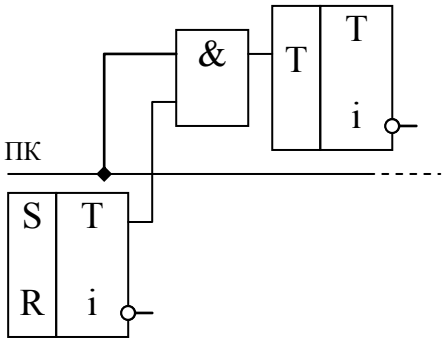


Рис. 2.26. Сложение по mod2

Если используется регистр без предварительной установки в «0», то необходимо его реализовывать на парафазных входах (см. рис.2.27.).

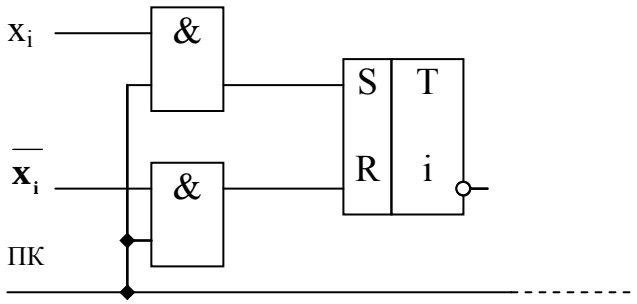
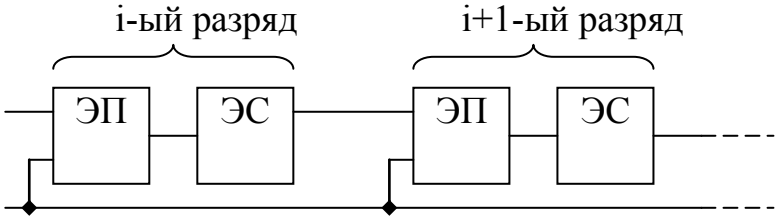


Рис. 2.27.

Б. Сдвигающие регистры.

Сдвигающие регистры (СР) характеризуются последовательной записью кода числа, начиная с младшего или старшего разряда, путем последовательного сдвига кода тактирующими импульсами. В зависимости от возможностей сдвига их подразделяют на нереверсивные (только влево или вправо) и реверсивные (в оба направления). Идея функционирования СР представлена на рис. 2.28.



ЭП – элементы памяти

ЭС – элементы связи (задержки)

Рис. 2.28.

На рис. 2.29., 2.30. и 2.31. представлены варианты реализаций регистров сдвига на различной элементной базе.

Сдвиг регистра на CRS-триггерах (парафазный 2-х тактный сдвиг):

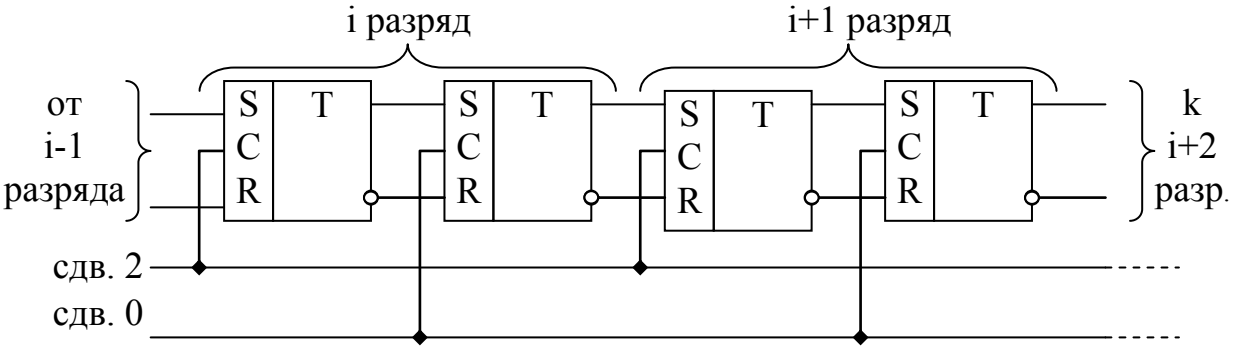


Рис. 2.29.

Сдвиг регистра на D-триггерах:

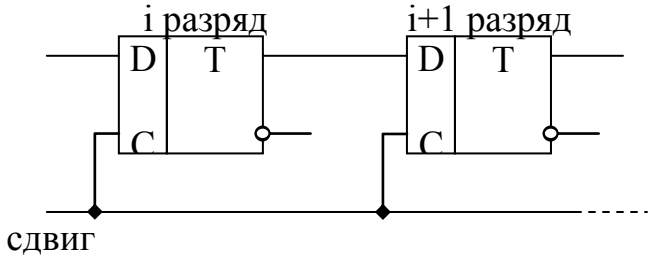


Рис. 2.30.

Сдвиг регистра на JK-триггерах:

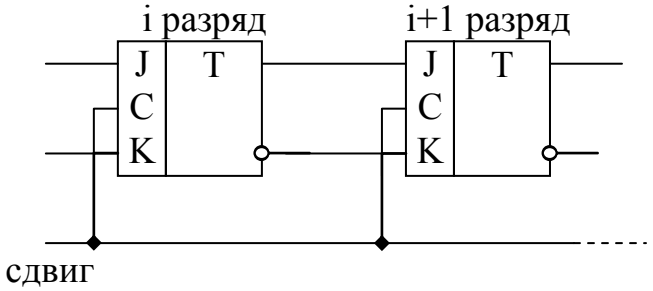


Рис. 2.31.

В качестве конкретного примера рассмотрим однотактную микросхему К155ИР1 (см. рис. 2.32.),

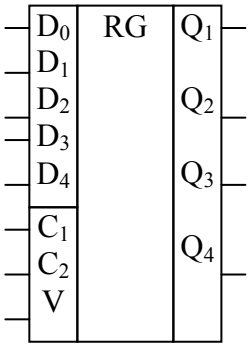
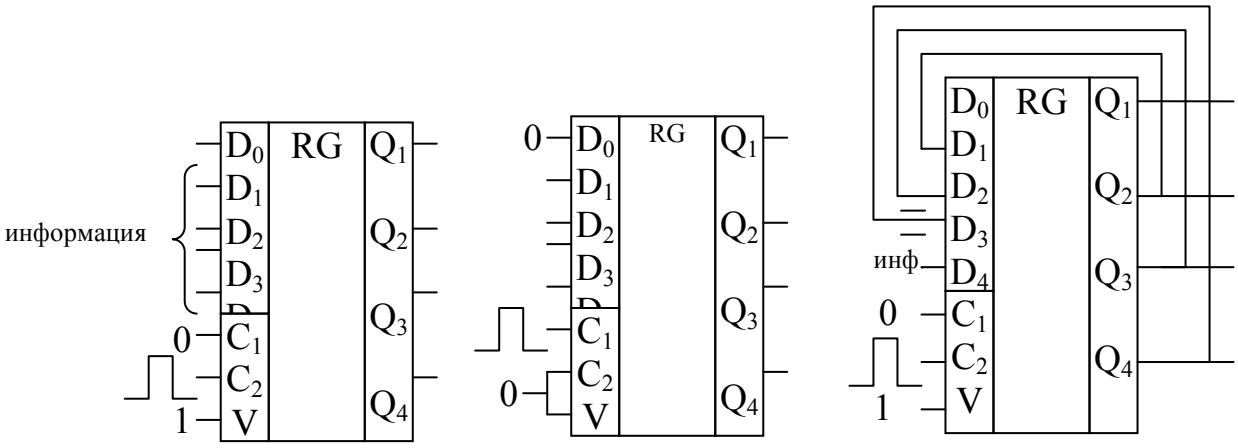


Рис. 2.32.

где D_0 – вход последовательной записи; D_1-D_4 – запись параллельным кодом; C_1 и C_2 – входы сдвига вправо и влево; V – вход управления; Q_1-Q_4 – прямые выходы. На базе данной микросхемы возможна реализация параллельной записи (рис. 2.33.а.), сдвига вправо (рис. 2.33.б.), сдвига влево (рис. 2.33.в.), сдвига по кольцу (самостоятельно). При необходимости работы с 8-ми разрядным кодом используют две микросхемы и т.д.



а) режим параллельной записи б) режим сдвиг вправо с) режим сдвиг влево

Рис. 2.33.

II. Счётчики.

Счетчик – это КА, выполняющий функцию подсчёта количества импульсов за некоторое время, а также формирования и запоминания двоичного кода, соответствующего этому количеству. С точки зрения пользователя, счетчик является преобразователем числа импульсов в двоичный код.

По целевому назначению счетчики подразделяются на простые и реверсивные. Простые счетчики выполняют счет в прямом направлении (сложение), либо в обратном (вычитание), а реверсивные могут работать как в режиме сложения, так и вычитания.

Соединяя несколько счетных триггеров определенным образом, можно получить схему многоразрядного счетчика. В этом случае необходимо учитывать сигнал переноса. Перенос может быть организован следующим образом:

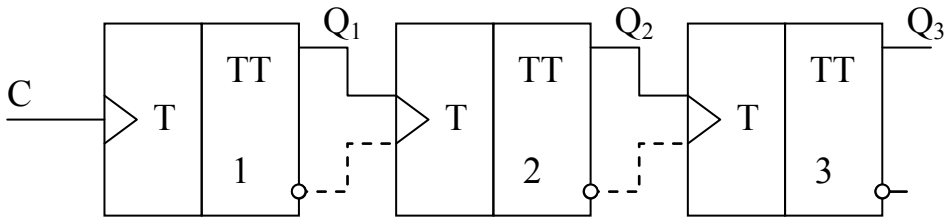
- последовательным;
- сквозным;
- групповым.

Рассмотрим вариант последовательного переноса.

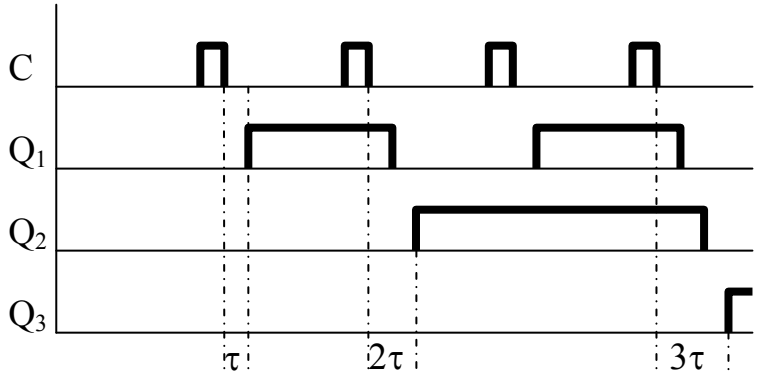
В последовательных счетчиках каждый последующий (i+1)-й разряд запускается от информационных выходов (Q, $\overline{Q}$) предыдущего i-го разряда, а счетный импульс поступает на вход первого разряда.

А. Простые счётчики с последовательным переносом.

На рис. 2.34.а. представлена структурная схема 3-х разрядного счетчика, а на рис. 2.34.б. временная диаграмма его функционирования,



а)



б)

Рис. 2.34.

где: _____ – сложение; - - - - - – вычитание; $t_{установки\ max} = n \cdot \tau$; n=1,2,3 – номер устройства, а τ – его время преобразования.

Б. Реверсивные счётчики с последовательным переносом (см. рис. 2.35.).

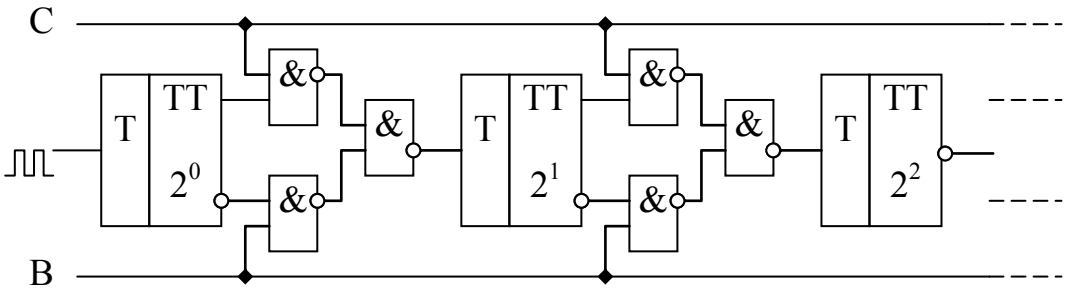


Рис. 2.35.

Работа счетчика организована следующим образом. Режим работы определяется постоянными сигналами на линиях С (сложение) и В (вычитание). Например, если на В подается «1», а на С – «0», то реализуется режим вычитания. При этом на вход первого триггерного устройства подается требуемое число счетных импульсов, а для

последующих они формируются с инверсного выхода предыдущих по аналогии с рис. 2.34.б.

2.2. Структура и принципы организации ЭВМ.
2.2.1. Принципы действия и основные характеристики ЭВМ.

Принцип действия ЭВМ представлен на рис. 2.36. в виде общей структуры.

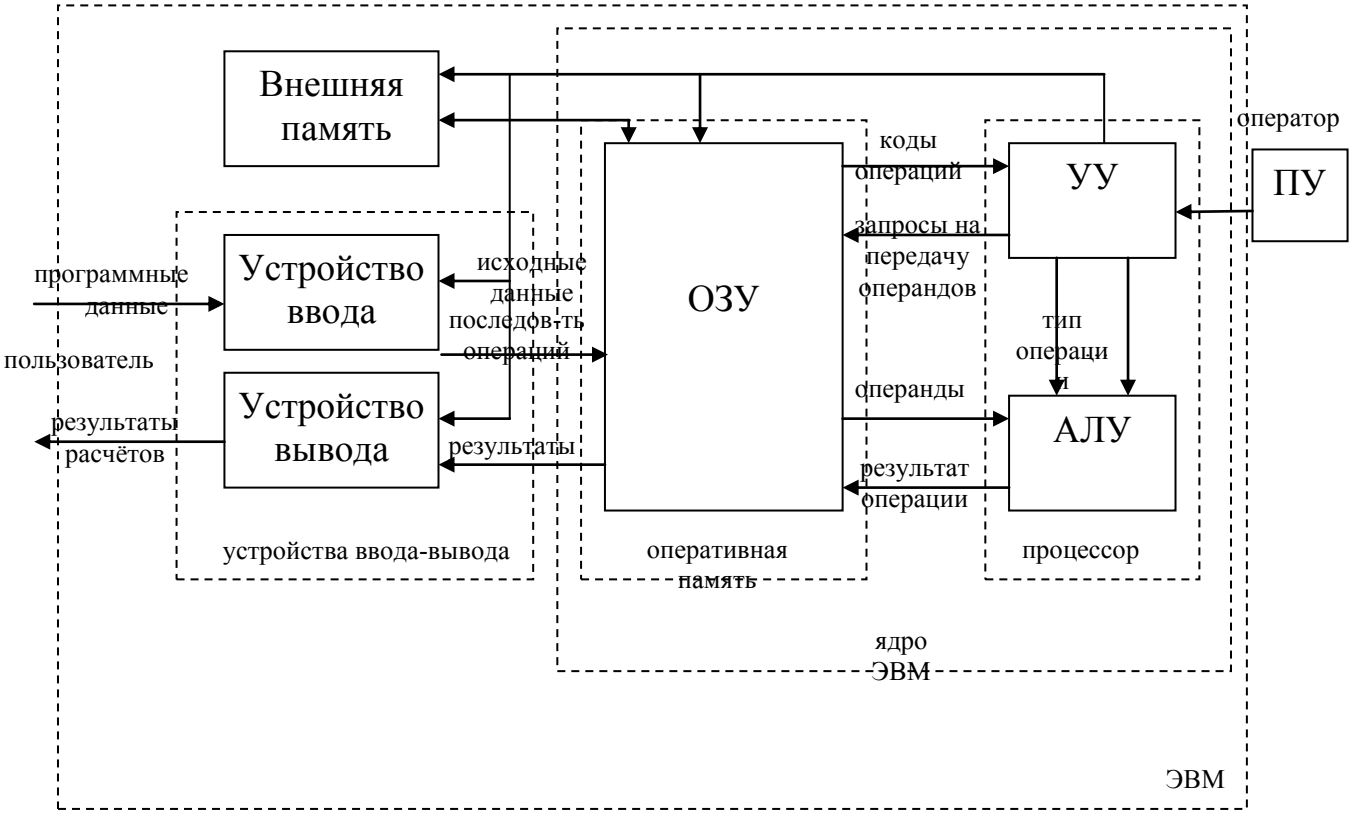


Рис. 2.36.

ЭВМ содержит следующие основные устройства:

АЛУ – производит арифметические и логические преобразования над операндами;

УУ – производит автоматически управление вычислительным процессом посредством сигналов управления, посылаемых всем устройствам ЭВМ;

ОП или ОЗУ – производит запись, хранение и выдачу информации, непосредственно участвующей в вычислительном процессе;

УВвода – производит считывание программ и исходных данных с носителей информации;

УВывода – производит выдачу результатов путём отображения её на печатающие устройства или экране дисплея;

ПУ – позволяет оператору проводить пуск, останов, тестирование ЭВМ и, если надо, вмешиваться в вычислительный процесс.

Структура, представленная на рис. 2.36., является обобщенной и не учитывает специфики конкретных ЭВМ. Например, не указаны: постоянная и перепрограммируемая память; шинная организация (управления, данных, адреса) и т.д.

Основные характеристики ЭВМ.

1) Производительность (быстродействие). Оценивается по смесям команд.

«Смесь Гибсона» - для оценки производительности решения научно-технических задач. Включает: сложение, вычитание, умножение, деление (с фиксированной и плавающей запятыми), логические операции, команды передачи управления, операции с различными коэффициентами.

Производительность определяется следующим образом:

$$P = \sum_{i=1}^{n} k_i \Big/ \sum_{i=1}^{n} k_i t_i$$

k – коэффициент, t – время, n – число команд в смеси.

2) Эффективность.

$Э = P \Big/ (S_{ЭВМ} + S_{Экспл.})$, где $S_{ЭВМ}$ – стоимость ЭВМ, $S_{Экспл.}$ – стоимость эксплуатации (помещения, энергии, обслуживания).

3) Объём оперативной памяти.

Оценивается в Кбайтах или Мбайтах. 1Кбайт=1024байта.

4) Разрядность машинного слова.

5) Надёжность.

Например, коэффициент готовности:

$K_Г = T \Big/ T + T_В$, где T – время наработки на отказ, $T_В$ – время наработки на восстановление. Существует значительно большее количество характеристик надежности.

6) Дополнительные характеристики.

Габариты. Вес. Энергопотребление. Климатические условия. Стоимость. Программное обеспечение и др.

2.2.2. Устройства памяти.

I. Основные понятия и определения.

Память ЭВМ – совокупность устройств для записи, хранения и выдачи информации.

Информация – некоторая совокупность сведений, данных, знаний.

По Шеннону – мера неопределённости определяется как:

$H = -\sum_{k=1}^{n} p_k \log_2 p_k$, где H – энтропия (мера неопределённости), n – количество состояний, p_k – вероятность k-ого состояния.

Измерение информации (количеством) $I=H_{апр.}-H_{апост.}$ есть мера уменьшения неопределённости, где: $H_{априорная}$ – информация до опыта; $H_{апостариорная}$ – информация после опыта.

Пример:

Бросок монеты.

n=2; $p_1=p_2=0,5$ (вероятности p_1 и p_2 соответствуют событиям: «орел», «решка»);

$H_{апр.}=(-0,5\log_2 0,5)+(-0,5\log_2 0,5)=1$

$H_{апост.}=0$ (после броска).

Следовательно: I=1-0=1. Это бит.

Единица информации – это неопределённость любого процесса, имеющая 2 равновероятных исхода.

Характеристики памяти:

а) ёмкость памяти;

б) удельная ёмкость – отношение ёмкости ЗУ к его физическому объёму;

в) быстродействие ЗУ – характеризуется выполнением основных операций: записи и считывания информации:

$t_{счит.}^{обр.}$ - время обращения при считывании;

$t_{зап.}^{обр.}$ - время обращения при записи;

$t_{обр.}=\max(t_{счит.}^{обр.}, t_{зап.}^{обр.})$;

$t_{счит.}^{обр.}=t_{поиска\ (доступа)}+t_{считывания}+t_{записи}$;

г) стоимость хранения 1-цы информации (бита) – зависит от типа ЗУ.

При этом следует учитывать, что увеличение ёмкости ЗУ приводит к уменьшению быстродействия, а увеличение времени обращения – к уменьшению стоимости бита.

Такого рода зависимости представлены на рис. 2.37.

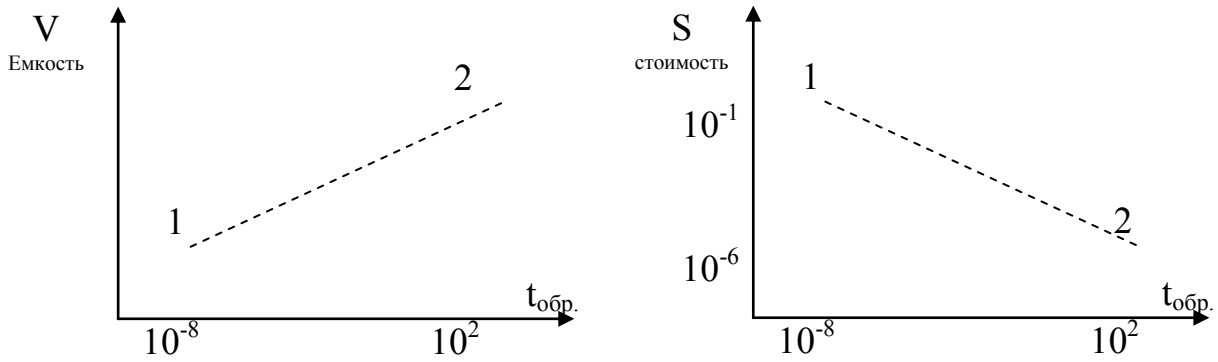


Рис. 2.37.

где: 1 – полупроводниковые ЗУ; 2 – ЗУ на магнитных лентах.

II. Классификация ЗУ.

Запоминающие устройства можно подразделить:

а) по физическому принципу:

- электронные (полупроводниковые);
- магнитные;
- магнитооптические;
- оптические;

б) по способу организации доступа к информации:

- с произвольным доступом (время не зависит от места расположения информации в ЗУ). Цикл обращения 3нСек - 1-2мсек;
- с прямым доступом;
- с последовательным доступом;
- страничная организация памяти (4 Кбайт, 2МБ, 4МБ, 1ГБ и т.д. – внутри станицы прямая адресация);

в) по реализуемым операциям обращения:

- с произвольным обращением (и запись и считывание);
- с обращением только при считывании (записи нет);

г) по функциональному назначению в ЭВМ (см. рис. 2.38.):

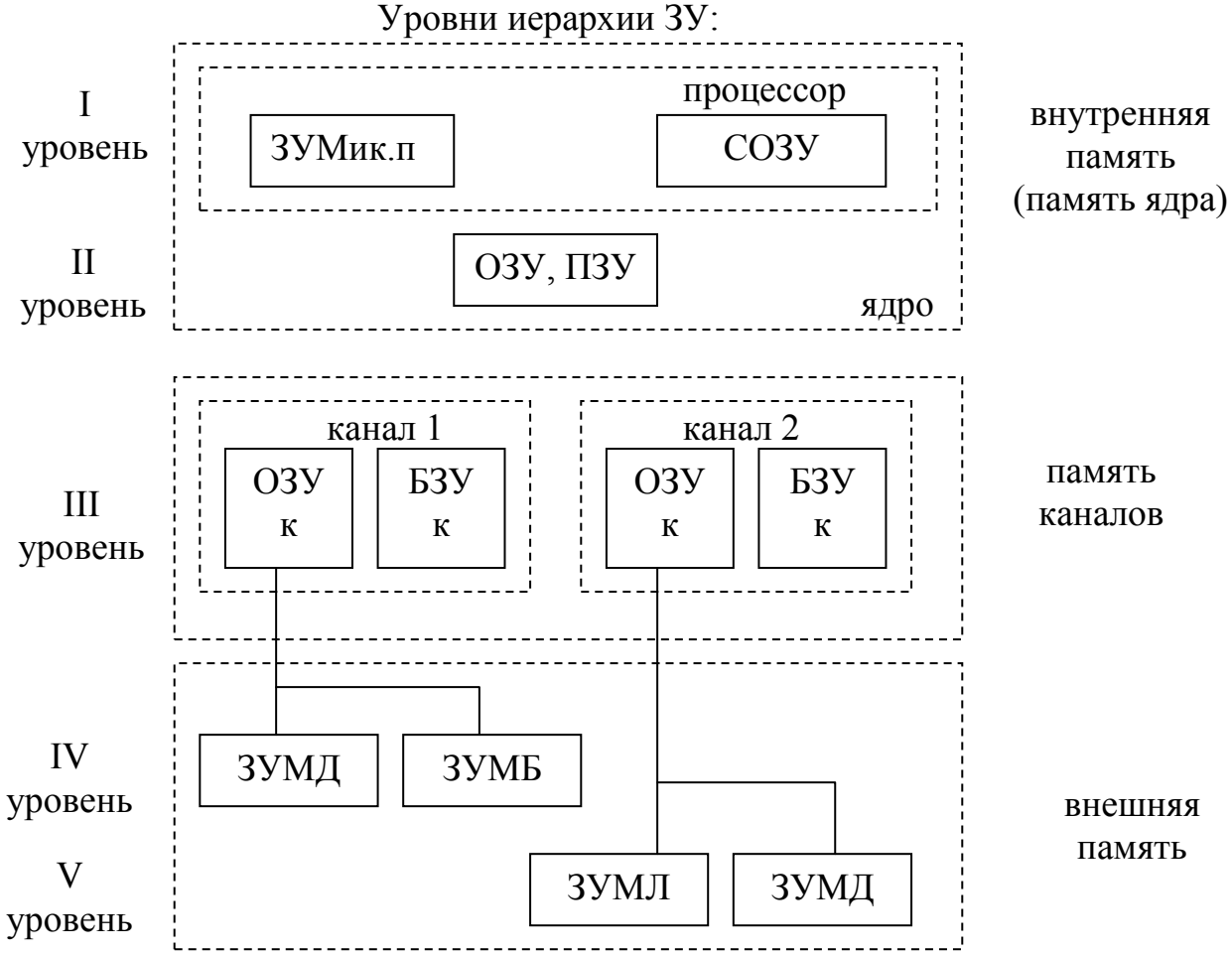


Рис. 2.38.

I уровень содержит: ЗУМик. п – ЗУ микропрограмм и СОЗУ – сверхоперативное запоминающее устройство; II уровень включает в себя ОЗУ – оперативное ЗУ; III уровень состоит из собственной памяти каналов обмена (оперативное ЗУ канала + буферное ЗУ); IV уровень различного рода внешние ЗУ (диски, барабаны и т.д.), предназначенные для хранения основной информации вне ядра; а V уровень необходим для хранения архивных данных внешних ЗУ (диски, ленты и пр.).

III. Адресная организация ЗУ и их структуры.

Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве (ЗМ). Запоминающий массив представляет собой совокупность однотипных запоминающих элементов.

Структура адресной организации ЗУ с произвольным доступом (см. рис. 2.39.).

Каждая ячейка памяти в ЗМ имеет свой адрес.

Поиск слова осуществляется по адресу ячейки.

На принципах адресной организации основано ОЗУ. Адрес ячейки поступает на регистр адреса, т. к. в нём n разрядов то возможна адресация 2^N слов ЗМ. В зависимости от подачи сигнала на «чтение» или «запись» в регистр слова поступит считанное слово, или из регистра поступит в ЗМ.

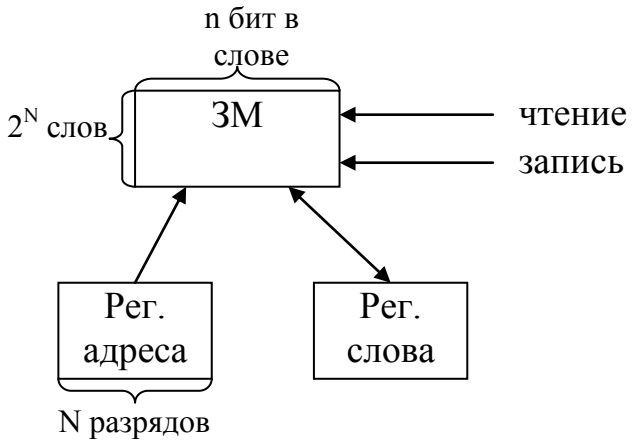


Рис. 2.39.

Более детализировано процедура адресной организации рис. 2.39. представлена на рис. 2.40.

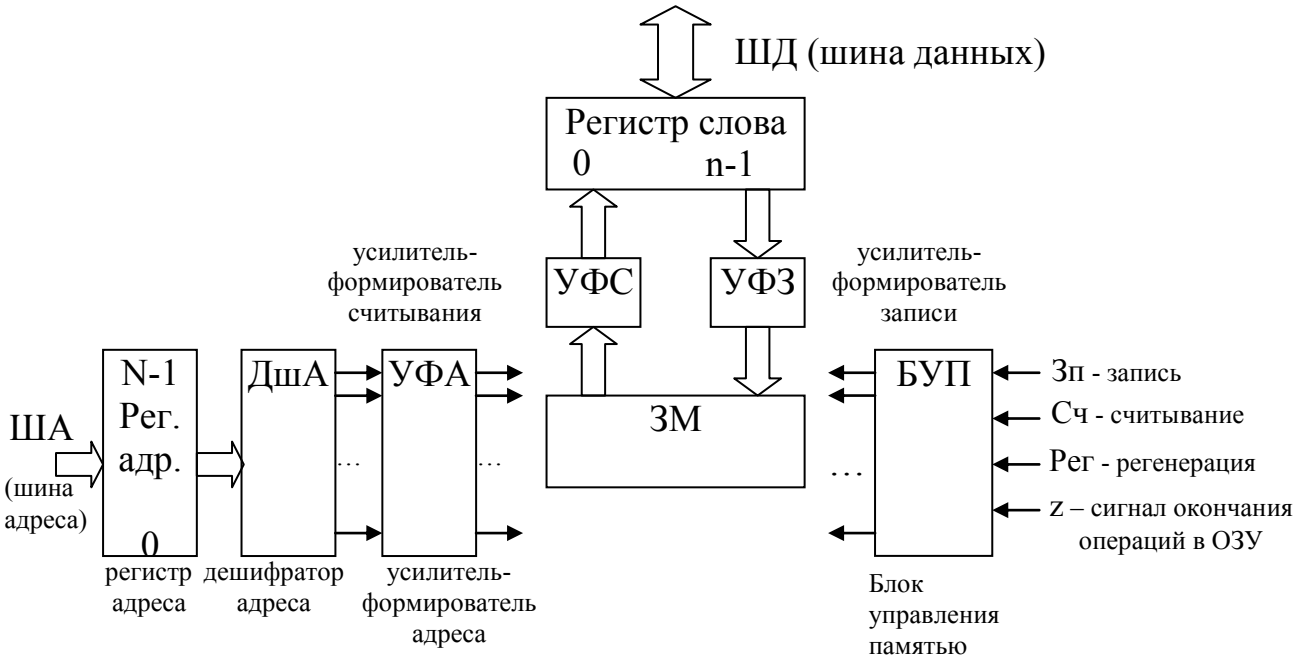


Рис. 2.40.

На рис. 2.40. и 2.41. представлены алгоритмы выполнения процедур записи и считывания, соответствующие рис. 2.40.

Операция записи:

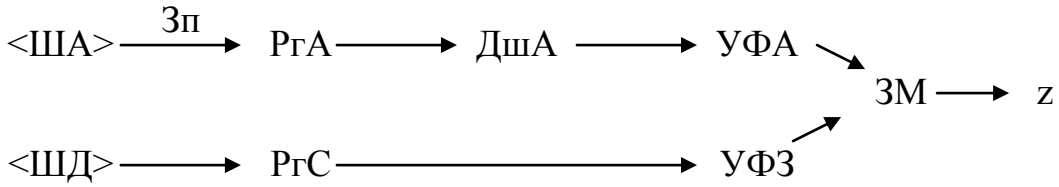


Рис. 2.40.

Операция считывания:

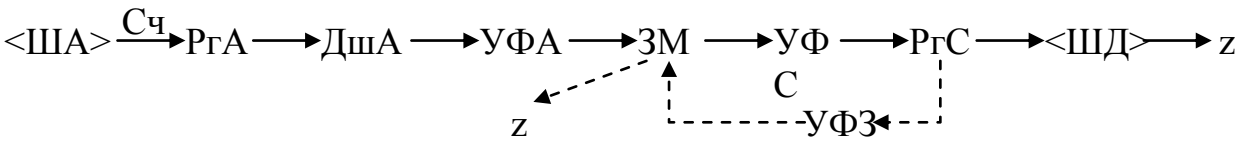


Рис. 2.41.

В зависимости от структуры пространственного размещения ячеек памяти в ЗМ, различают адресные ОЗУ с 2D; 3D; 2,5D; 1D организацией ЗМ (D – размерность).

2D-организация ЗМ (см. рис. 2.42.):

Это двумерная организация (по одному направлению идёт адрес, а по другому – слово), где:

БП – блок памяти; n – разрядный адрес на входе $N=2^n$.

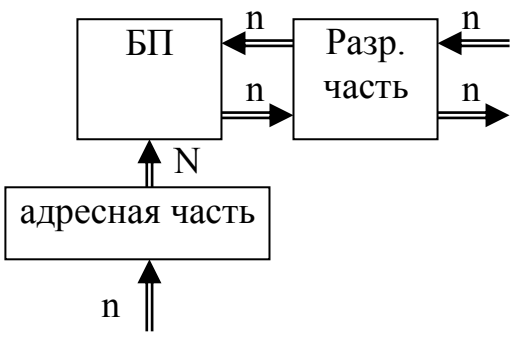


Рис. 2.42.

Достоинством 2D организации ЗМ являются высокое быстродействие и помехоустойчивость (отношение $\frac{\text{сигнал}}{\text{помеха}} = \frac{5}{1}$), а недостатком – сложность адресной части.

3D-организация ЗМ (рис. 2.43.):

Количество выходов из адресной части $= 2\sqrt{N}$.

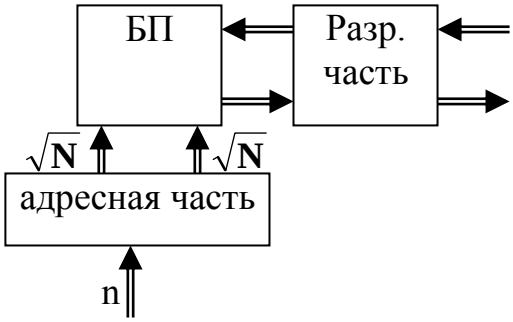


Рис. 2.43.

Выбор ячейки памяти осуществляется совпадением сигналов по двум шинам. Это существенно упрощает дешифратор адреса, но появляются полувыбранные ячейки, которые могут инициировать ложное считывание (запись). Данный недостаток исключают следующим образом:

- строгим стробированием сигналов (УФА выдаёт очень короткие импульсы);
- высокой синхронизацией сигналов с УФА и УФС (или УФЗ);
- жёсткой регенерацией.

Достоинством данной организации является упрощение адресной части, а недостатком – низкая помехоустойчивость = 2.

2,5D – организация ЗМ.

Эта организация является промежуточной. Адрес разбивается на группы, а внутри группы формируется 3D – организация.

1D – организация ЗМ используется для постоянных ЗУ.

Обозначение ОЗУ на схемах представлено на рис. 2.44.

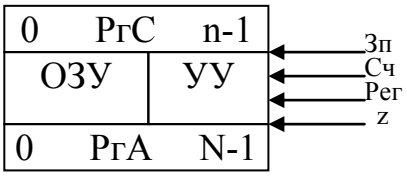


Рис. 2.44.

Во многом характеристики ЗУ зависят от используемой элементной базы. Отдельные характеристики такого рода представлены в таблице 2.8.

Таблица 2.8.

Элементная база	$t_{выб}$ nc	V бит	бит/см^3	Энергопотребление при хранении
Биполярный	40	10^5	200	+
МОП-структ.	250/100	10^6	300	+
Ферритовые сердечники	250	10^8	20	-

IV. Ассоциативные ЗУ.

Поиск по ассоциативному признаку.

структура:

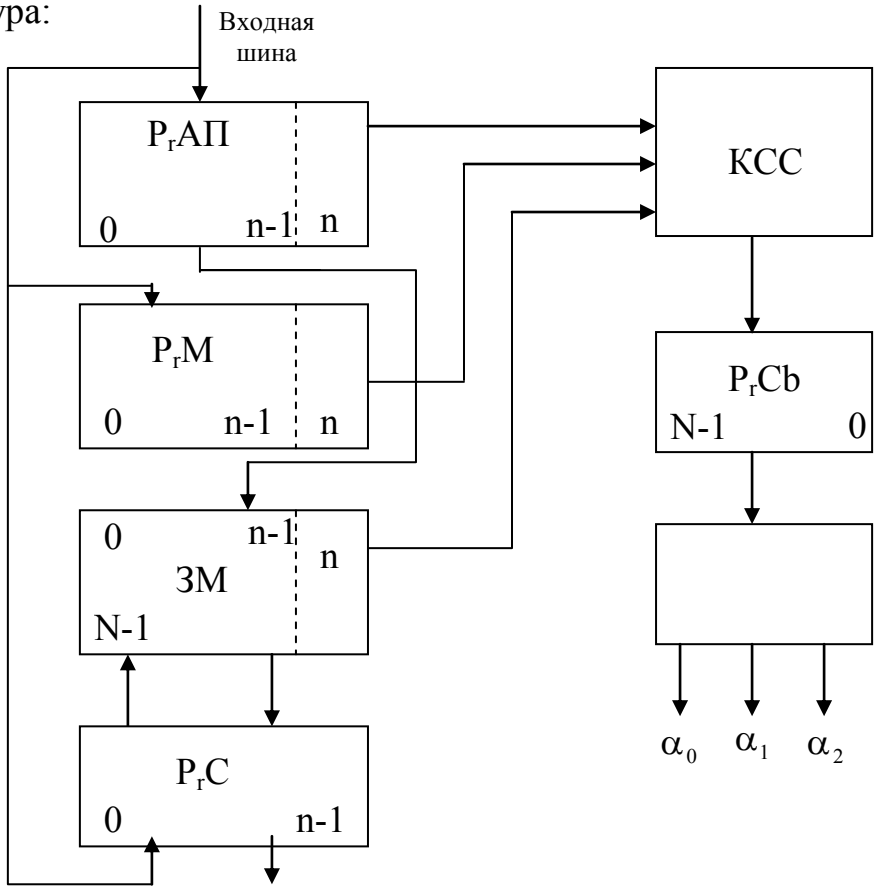


Рис. 2.45.

где: $Р_г$АП – регистр ассоциативного признака;

$Р_г$М – регистр маски;

КСС – комбинационная схема сравнения;

$Р_г$Сb – регистр совпадения;

ФС – формирователь сигналов;

$Р_г$С – регистр слова.

При этом в ЗМ хранятся N (n+1)-разрядных слов. n-ый разряд указывает на занятость ячейки (0 – свободна; 1 – записано слово).

Принцип работы ассоциативных ЗУ заключается в следующем.

По входной шине поступает в $Р_г$АП ассоциативный запрос (в $0 \div n-1$ разряды), а в $Р_г$М – код маски поиска (в $0 \div n-1$). n-ый разряд устанавливается в 0. Ассоциативный поиск проводится по совокупности разрядов $Р_г$АП, которым соответствуют 1-цы в разрядах $Р_г$М (немаскированные разряды). Для слов, у которых цифры в разрядах совпали с незамаскированными разрядами $Р_г$АП, КСС устанавливает 1 в соответствующие разряды $Р_г$Сb и 0 в остальные разряды.

Таким образом, позиции 1-ц в $Р_г$Сb соответствуют адресам слов в ЗМ удовлетворяющих ассоциативному поиску. Комбинационная схема ФС формирует из слова в $Р_г$Сb сигналы $\alpha_0, \alpha_1, \alpha_2$.

V. Стековая и магазинная память.

Стековая память является безадресной.

Стек – совокупность N связанных поразрядно регистров, образующих массив информации (см. рис.2.46.).

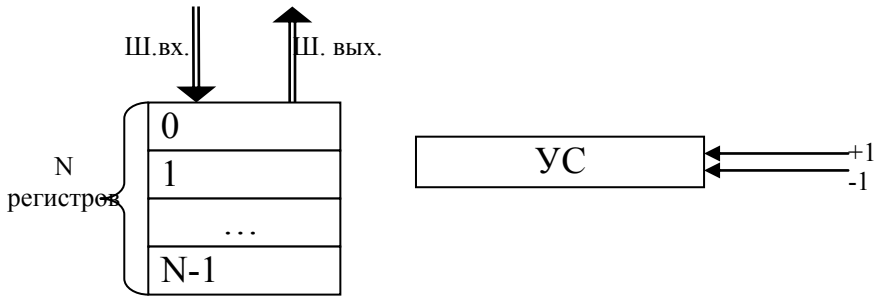


Рис. 2.46.

где: УС – указатель стека (± 1 – указывают направление перемещаемой информации).

Запись и считывание информации осуществляется по принципу: первым пришёл – последним обслужен (LIFO).

Запись только в верхний регистр. При этом все слова вместе с этим сдвигаются на 1 вниз.

Считывание в стеке происходит только из верхнего (0-го) регистра.

УС – счётчик, хранящий количество слов, записанных в стеке или переполнение.

Более сложная организация стека – связанные списки. В этом случае при записи слова занимаются 2 регистра: один для слова и один для № регистра для следующего слова.

Магазинная память (см. рис. 2.47.).

Принцип работы: первым пришёл – первым обслужен (FIFO).

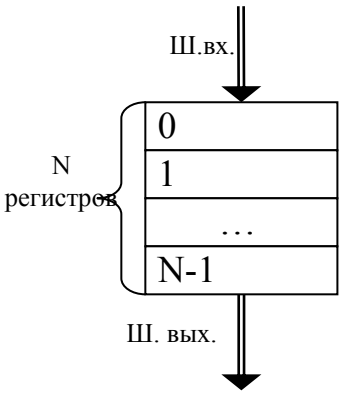


Рис. 2.47.

Микросхемы запоминающих устройств имеют свой классификатор и обозначения. Например: РА – ассоциативные ЗУ; РВ – постоянные ЗУ; РУ или РМ – ОЗУ (см. рис. 2.48.); РР – перепрограммируемые ЗУ.

При этом первая буква (Р) обозначает функциональный класс, вторая (А, В, У или М, Р) – группу.

Для конкретной микросхемы К155РУ1 ее структурная схема представлена на рис. 2.49. При этом основные характеристики и обозначения будут следующие:

$V_{\text{памяти}} = 16 \times 1\text{бит}$ (16 одноразрядных слов); запись информации D_0, D_1 в парафазном виде; считывание F_0, F_1 в парафазном виде; x_1-x_4 и y_1-y_4 шины выборки элемента памяти (адресные шины); CS – вход выбора микросхемы (1); W – вход установки режима: «0» - чтение; «1» - запись;

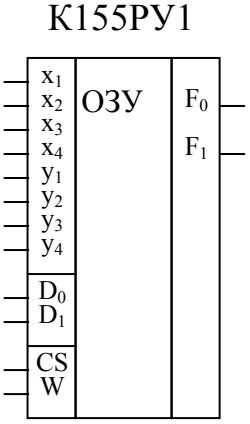


Рис. 2.48.

ЭП – элемент памяти.

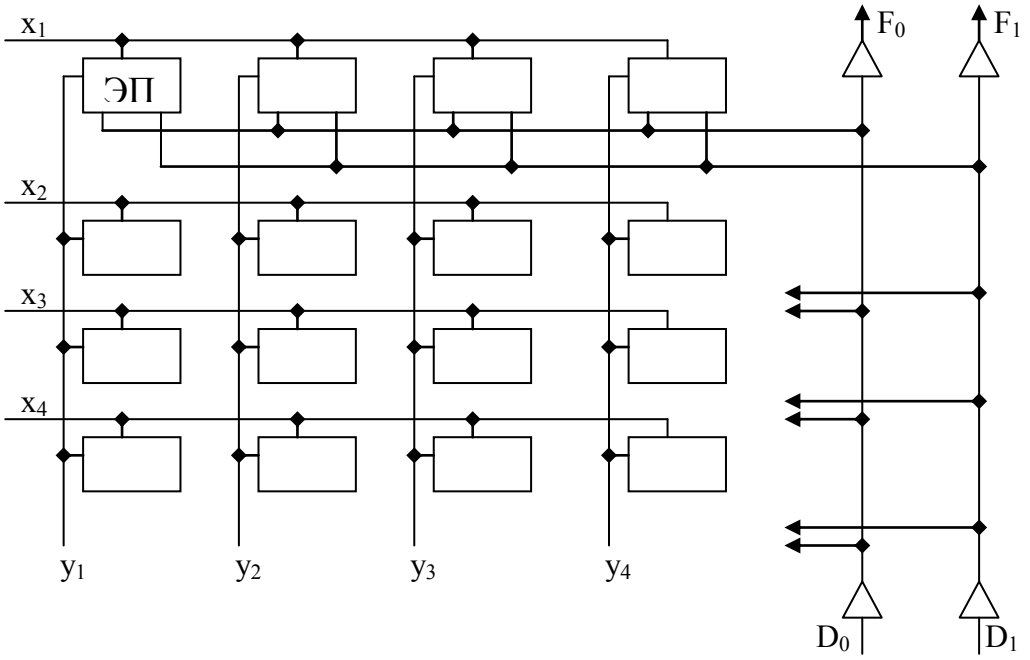


Рис. 2.49.

2.3. Процессоры: архитектура и принципы организации.

Процессор – функциональный блок ЭВМ для арифметической и логической обработки информации на основе программного управления. Его состав: АЛУ, УУ, СОЗУ и пр.

Рассмотрим основные составляющие и характеристики процессора.

I. Система команд ЭВМ.

Рассмотрим 3 части:

А. Классификация команд (операций).

Б. Структура и форматы команд.

В. Способы адресации.

А. Классификация команд (операций).

В современных ЭВМ используется порядка 200 команд. Например:

- арифметические операции: сложение, вычитание, умножение, деление, изменение знака, перенос, заем и др.;
- логические операции: $\neg \vee \wedge \oplus$;
- операции сдвигов: арифметический, циклический, логический;
- пересылочные операции:
  - регистр – регистр (внутри процессора);
  - регистр – память (между процессором и ОЗУ);
  - память – память (внутри ОЗУ);
- операции управления: условные, безусловные переходы, вызов подпрограмм, возврат из подпрограмм, программные прерывания и др.

Б. Структура и форматы команд.

Команда – двоичный код, определяющий операцию выполняемую процессором и данные (операнды), участвующие в операции.

Операнды указываются адресами ячеек памяти, где они содержатся.

Команда в общем виде состоит из (см. рис. 2.50):

КОП	A_1	A_2	A_3	A_4

операционная часть — адресная часть

Рис. 2.50.

A_1 – адрес 1-ого оператора;

A_2 – адрес 2-ого оператора;

A_3 – адрес результата;

A_4 – адрес следующей команды.

Структура команды: - определяется составом, назначением и расположением полей в команде. При этом возникает необходимость оценить размерность команды. Например, если в ЭВМ около 200 команд и до 16Кбайт адресуемой памяти, то это приводит к длине 4-х адресной команды:

$$n_{ком} = n_{коп} + 4 \cdot n_A \geq \log_2 200 + 4 \cdot \log_2 (16 \cdot 1024 \cdot 8) = 8 + 4 \cdot 17 = 76 \text{ дв.разр. (бита)}$$

Отсюда следует, что желательно уменьшать длину команды, поскольку существуют ограничения по разрядной сетке ЭВМ.

Для упрощения аппаратуры процессора и повышения быстродействия нужно укоротить команды до машинного слова (полуслова). Поэтому применяют более простые структуры команд (см. рис. 2.51.):

КОП	A_1	A_2	A_3	КОП	A_1	A_2

КОП	A_1 КОП

- безадресная команда

Рис. 2. 51.

На сегодняшний день используются в основном 2-х и одноадресные команды и безадресные стеки.

2-х адресная команда может быть представлена следующим образом: $A_1 := A_1 * A_2$, где * - знак операции, а 1-но адресная команда – в виде отдельного регистра процессора – аккумулятора: $A_{кк} := A_{кк} *$операнд.

В. Способы адресации.

Решение задачи уменьшения длины команд породило различные способы адресации информации:

1. прямая адресация:

а) абсолютная прямая адресация; б) неявная адресация;

2. косвенная адресация:

а) абсолютная косвенная адресация; б) регистровая косвенная адресация;

3. относительная адресация:

а) адресация по базе; б) индексная адресация;

4. непосредственная адресация.

5. Страничная адресация.

Рассмотрим основные способы адресации на примере одноадресной структуры команд.

Введём обозначения:

О – операнд (число над которым выполняется операция)

$А_К$ – адресный код (адрес операнда)

$А_И$ – исполнительный адрес (номер ячейки памяти, к которой происходит фактическое обращение).

1. Прямая адресация (см. рис. 2.52.):

В этом случае содержимое адресной части прямо указывает на адрес операнда:

- абсолютная прямая адресация: $А_К=А_И$;
- неявная адресация: $А_И=f(КОП, А_К)$, т. е. $А_И$ определяется специальным образом из информации, содержащейся как в КОП, так и в $А_К$.

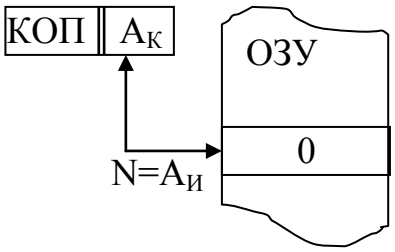


Рис. 2.52.

2. Косвенная адресация (см. рис. 2.53.):

$А_К$ указывает адрес ячейки, в которой находится $А_И$, т. е. адрес операнда.

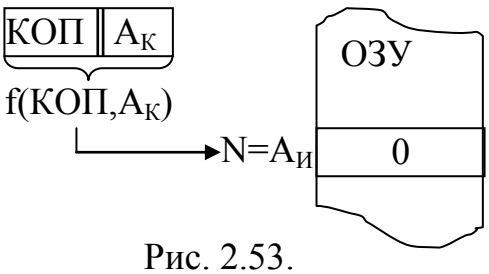


Рис. 2.53.

Варианты модификаций косвенной адресации представлены на рис. 2.54. и 2.55.

Абсолютная косвенная адресация

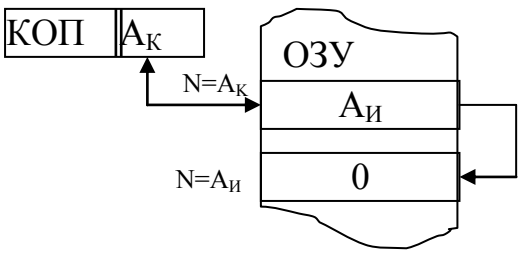


Рис. 2.54.

Регистровая косвенная адресация

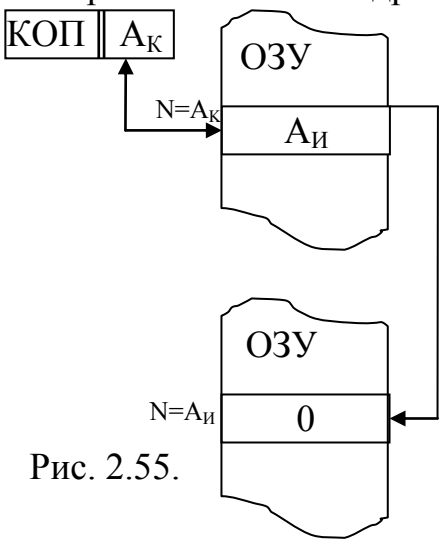


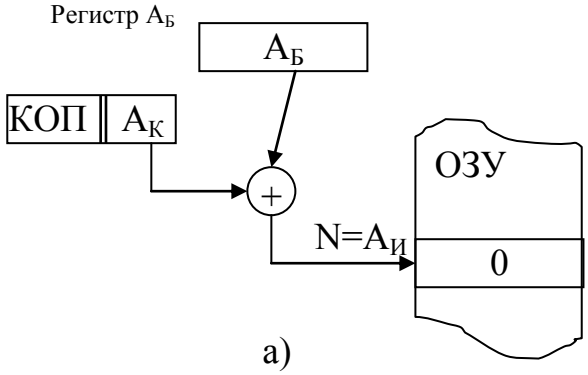
Рис. 2.55.

3. Относительная адресация.

$A_И$ определяется смещением некоторого заданного числа на значение, определяемое адресной частью команды.

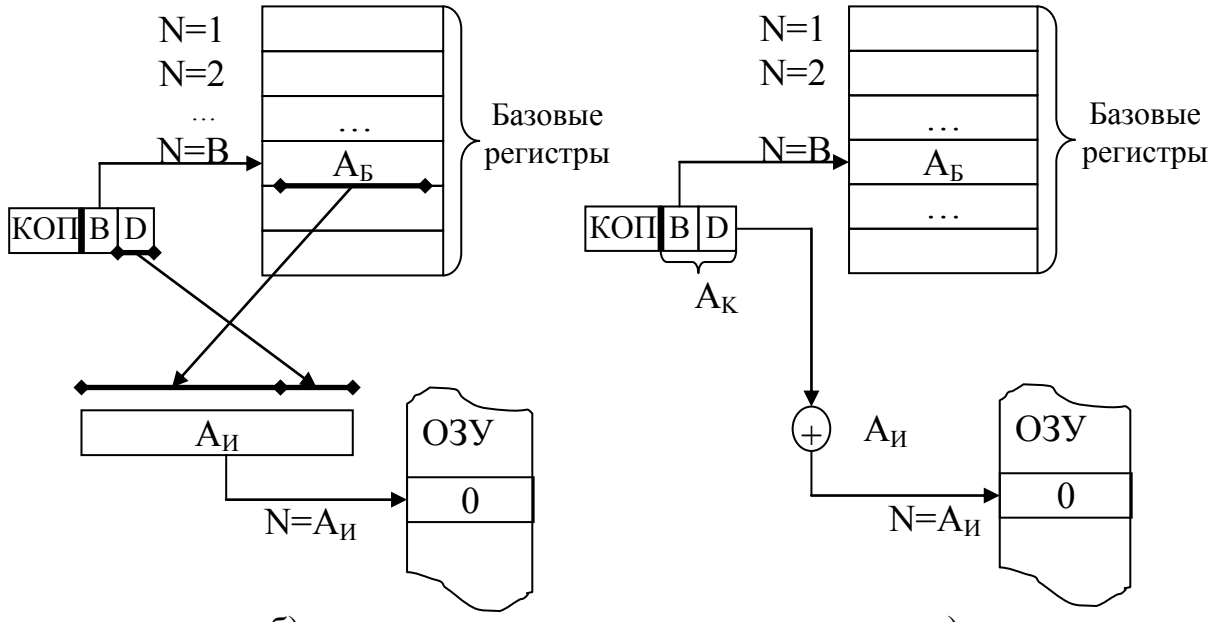
Адресация по базе:

$A_И=A_К+A_Б$, где $A_Б$ – некоторое число, называемое базовым адресом и хранящееся в специальном базовом регистре процессора.



а)

Рис. 2.56.



б) в)

Рис. 2.56.

Схема получения $A_И$ совмещением (конкатенацией) т. е. двоичный код приписывается младшим разрядам к двоичному коду $A_Б$ (рис. 2.56.б.).

Схема получения $A_И$ суммированием (рис. 2.56.в.)

Индексная адресация (см. рис. 2.57.) – операнды являются переменными с индексами, т. е. элементами массивов. Используются индексные регистры, расположенные в процессоре.

X – дополнительное поле, указывающее номер индексного регистра.

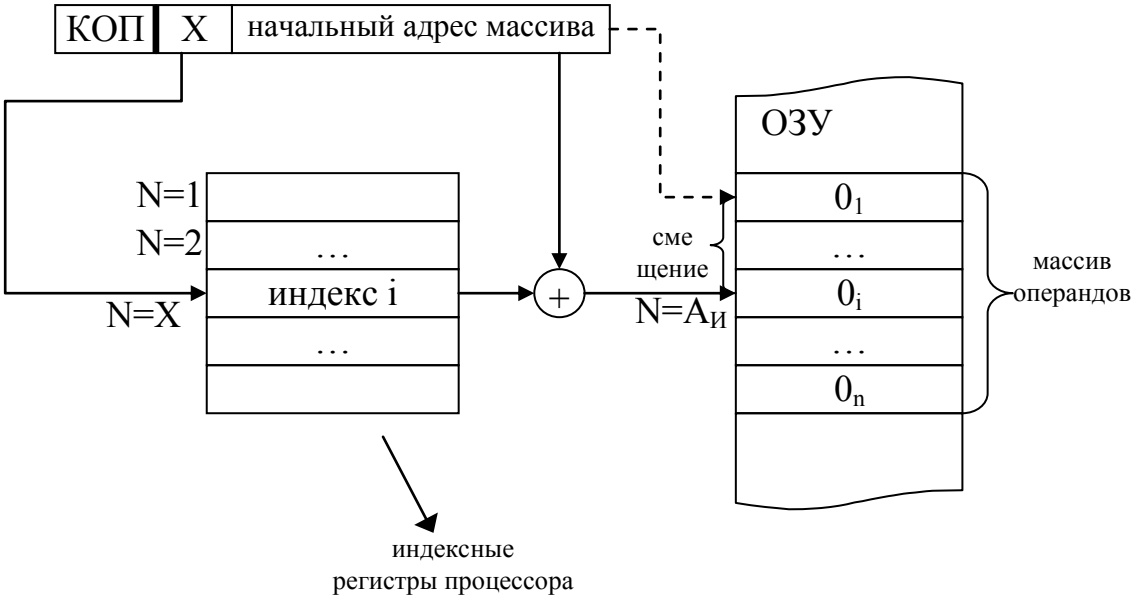


Рис. 2.57.

4. Непосредственная адресация (см. рис. 2.58.).

В команде содержится не адрес операнда A_K, а сам операнд. Используется это при хранении различного рода констант.

Рис. 2.58.

Непосредственная индексация наиболее часто используется в форматах RR-регистр-память или RS-регистр-память. Основное достоинство – минимальные временные затраты.

II. Структура процессора и алгоритм его работы.

Обобщенная блок-схема процессора представлена на рис. 2.59.

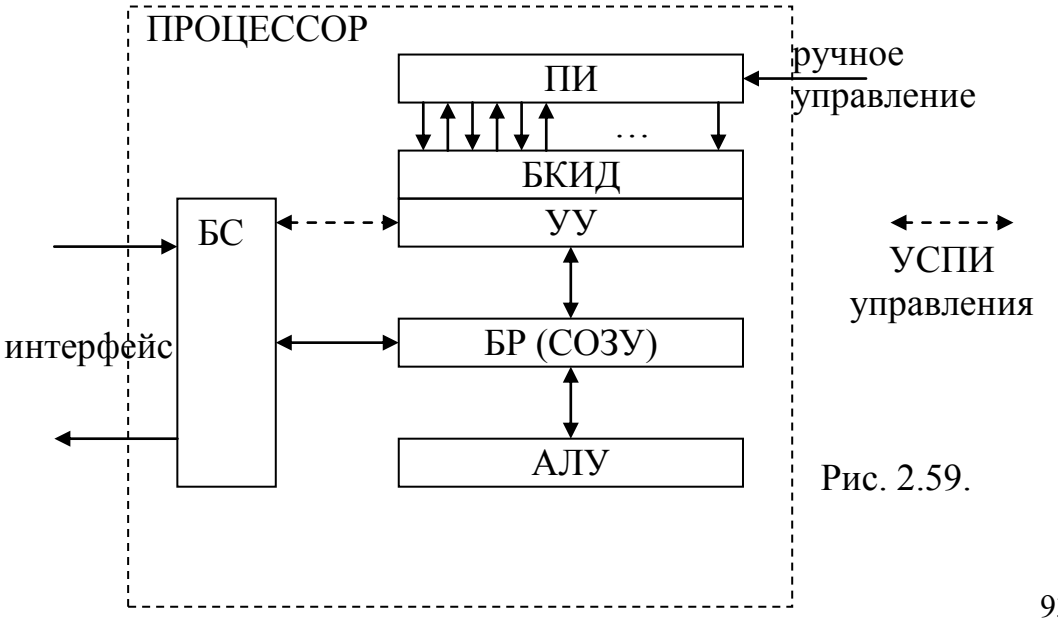


Рис. 2.59.

где: БР – блок регистров (местная память процессора); БС – блок сопряжения с интерфейсом; БКИД – блок контроля и диагностики; ПИ – пульт индикации.

Регистры БР могут жёстко не фиксироваться, а назначаться из 8-32х РОН (регистров общего назначения). Блок регистров для каждой конкретной серии ЭВМ формируется исходя из ее разрядности, функциональных возможностей и т.д.

БР состоит из:

- <u>программно-доступных регистров:</u>
  - аккумулятор;
  - базовые регистры;
  - индексные регистры;
  - указатель стеков;
  - счётчик команд и др.
- <u>программно-недоступных регистров:</u>
  - это рабочие регистры, использующиеся в процессе выполнения одной команды:

    РК – регистр команды;

    РА – регистр адресов;

    РС – регистр слов (операндов) и др.

Блок сопряжения организует обмен информацией между процессором и ОЗУ, а также связь процессора с периферийными устройствами. Согласованность работы процессора с остальными устройствами поддерживается процессорной шиной, которая включает в себя: шину управления, шину адреса, шину данных, шину задания режима ввода, шину задания режима вывода, шину синхронизации.

С точки зрения информационного обмена, в ЭВМ существует только две процедуры: чтение и запись или ввод/вывод слова.

<u>При вводе</u> БС последовательно устанавливает:

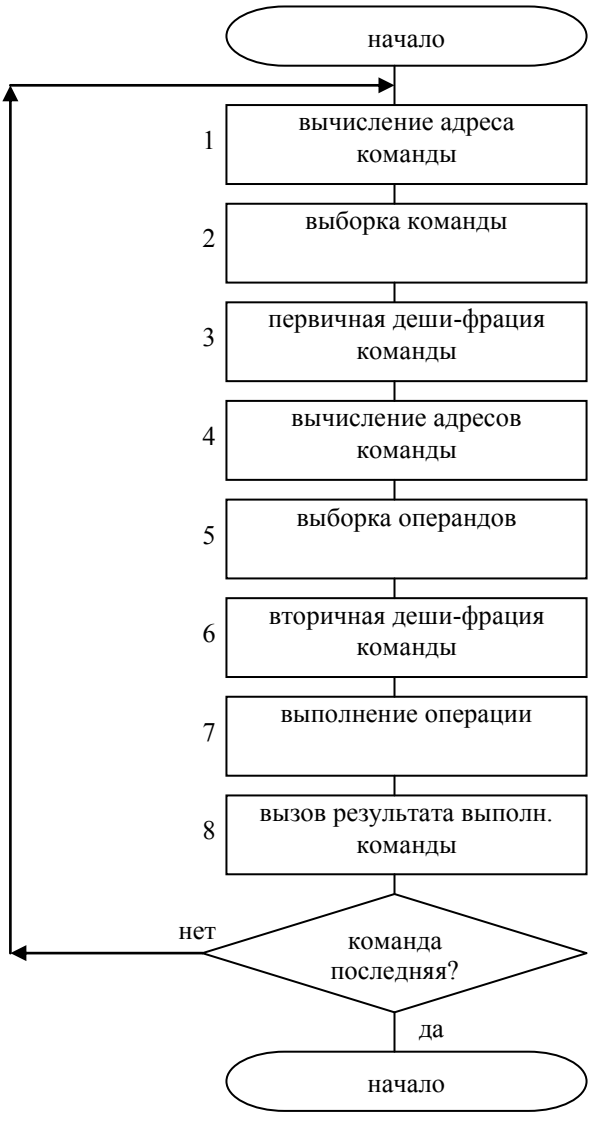


Рис. 2.60.

а) код режима ввода на шину управления;

б) адрес слова в ОЗУ (или периферийного устройств) на шину адреса;

в) выдача сигнала синхронизации по шине синхронизации.

При вводе слова из процессора БС последовательно устанавливаются:

а) код режима вывода на шину управления;

б) адрес слова в ОЗУ на шину адреса;

в) данные на шину данных;

г) выдача сигнала синхрониза-ции по шине синхронизации;

БКИД – служит для обнаружения сбоев и отказов в аппаратуре процессора; восстановление после сбоя и поиск мест неисправности при отказе;

ПИ – обеспечивает индикацию основных РОНов и базовых точек процессора. Обеспечивает режим ручного управления.

Алгоритм работы процессора (см. рис. 2.60.).

Комментарии

1. Адрес команды хранится в счётчике команд. При определении адреса следующей команды к текущему содержимому счётчика прибавляется длина предыдущей команды.

2. Адрес команды посылается из счётчика команд в РА, а БС выполняет операцию ввода слова через интерфейс.

Введённое слово поступает на РК. Если длина команды >1 слова, то на основании анализа 1-го слова вводится остальная часть команды.

3. Определяется группа команды и её адресность.

4. ----------.

5. По вычисленным адресам производится ввод операндов из ОЗУ через интерфейс на регистры слов БР.

6. Подготавливается последовательность действий АЛУ по выполнению данной команды.

7. Операнды подаются в АЛУ. Работой АЛУ управляют сигналы с шага.

8. Результат с выхода АЛУ даётся на место операнда приёмника в соответствующий регистр БР.

8. Результат выводится из процессора. Интерфейс обеспечивает режим вывода.

III. АЛУ.

В общем случае АЛУ обеспечивает выполнение всех операций арифметико-логической группы.

Структура АЛУ зависит от формы представления данных (операндов), т. е. существуют:

- АЛУ с фиксированной запятой;
- АЛУ с плавающей запятой;
- десятичные АЛУ (десятичная арифметика) и др.

В универсальных ЭВМ существуют многофункциональные АЛУ.

Ограничимся рассмотрением структуры и принципов работы АЛУ для положительных и отрицательных чисел с плавающей запятой (рис. 2.61.).

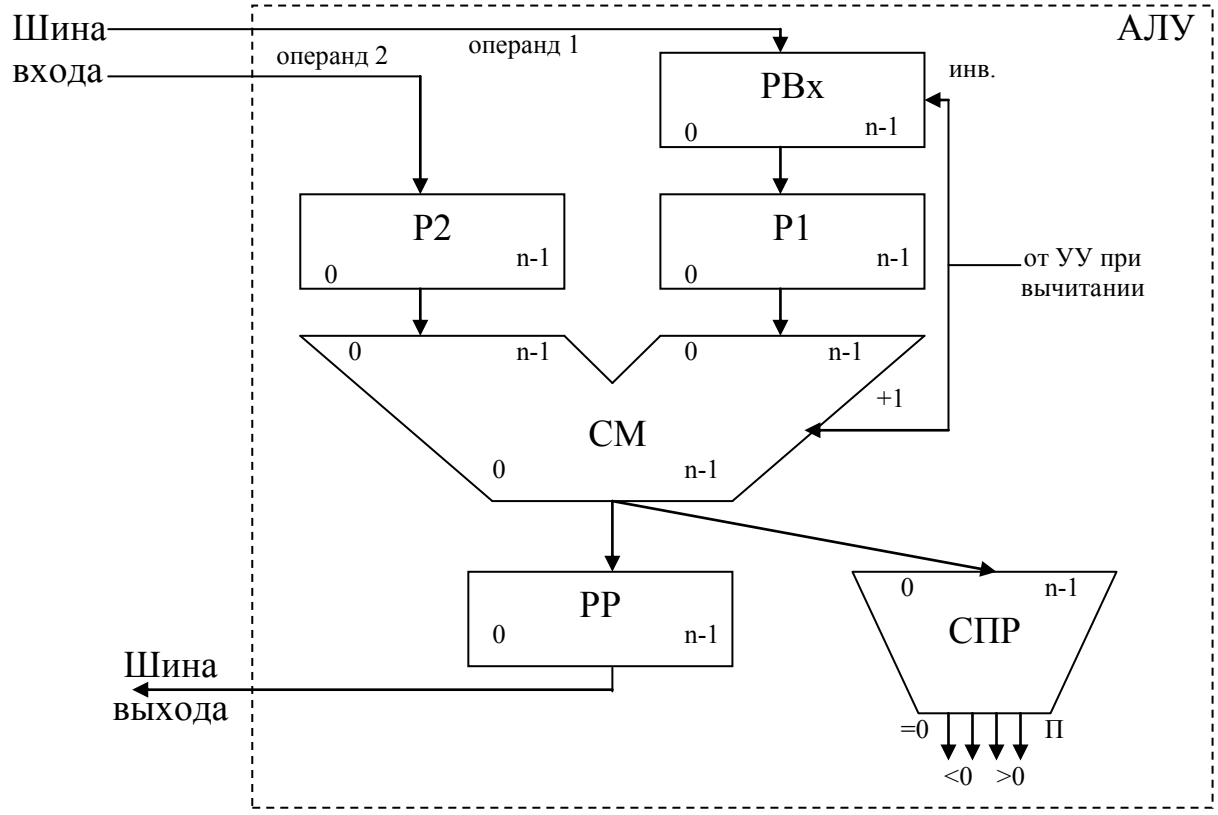


Рис. 2.61.

где: РВх – входной регистр;

Р1 и Р2 – регистры операндов;

СМ – параллельный комбинационный сумматор;

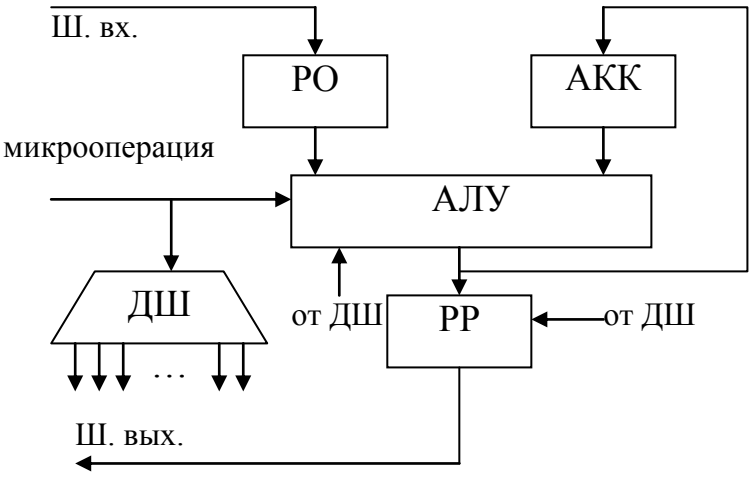
РР – регистр результата;

СПР – КС формирования признака результата:;

=0 – нулевой результат;

>0; <0 – формирование знака результата;

П – переполнение (поступает в УУ, где формируется код прерывания).



Выполнение операций умножения и деления в АЛУ возможно при реализации соответствующих микропрограмм, которые выполняются последовательностью микроопераций по схеме (см. рис. 2.62.),

Рис. 2.62.

где: РО – регистр операнда;
АКК – аккумулятор;
ДШ – дешифратор микрооперации;
РР – регистр результата.

АЛУ обладает магистральной структурой (подключается к шине входа и шине выхода). Такая структура позволяет обеспечить модульный принцип организации многофункционального АЛУ.

Например, можно дополнить структуру АЛУ блоками, реализующими основные логические операции (см. рис. 2.63.),

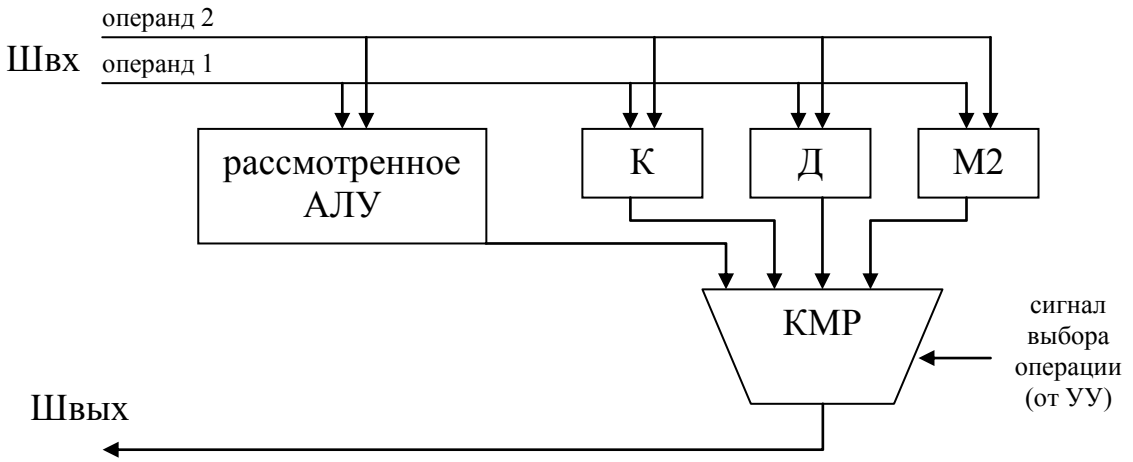


Рис. 2.63.

где: К, Д, М2 – комбинационные схемы, реализующие поразрядные операции конъюнкции, дизъюнкции и mod2 соответственно;
КМР – схема коммутатора результата.

В настоящее время АЛУ могут выполняться в виде одной или нескольких БИС. Возможен секционный принцип построения

многоразрядного АЛУ из 2-х, 4-х, 8-ми разрядных БИС АЛУ, как представлено на рис. 2.64.

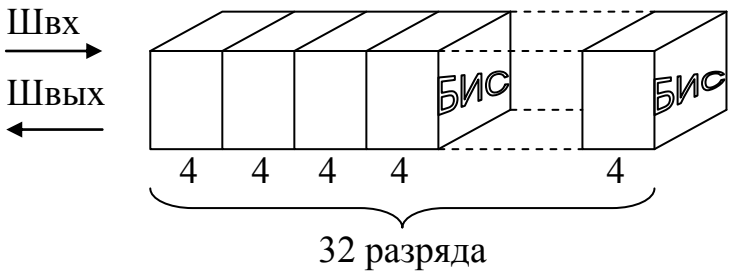


Рис. 2.64.

При таком построении АЛУ следует предусматривать организацию цепей межсекционных переносов и заемов.

2.4. Системы ввода-вывода информации.

I. Модульная организация ЭВМ и интерфейсы.

ЭВМ можно представить следующим образом (см. рис. 2.65.):

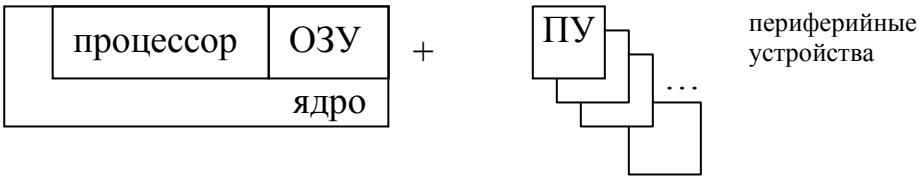


Рис. 2.65.

где ПУ – периферийные устройства. По назначению они разделяются в основном на два типа: для хранения больших объемов информации (внешние ЗУ) и для ввода-вывода информации (устройства ввода-вывода).

Современные ЭВМ имеют модульный принцип организации, т. е. ЭВМ состоит из набора блоков: Процессор, Оперативная Память, Периферийные Устройства и др., где модуль – это законченный функционально и конструктивно элемент.

Преимущества модульной организации:

- гибкость структуры (наращивание или усечение);
- оперативный ремонт (замена блоков);
- надёжность (поблочное резервирование).

Связь модулей (устройств ЭВМ) друг с другом осуществляется с помощью сопряжений, называемых в вычислительной технике интерфейсом.

Передача информации из периферийного устройства в ядро ЭВМ называется операцией ввода.

Передача информации из ядра ЭВМ в периферийное устройство называется операцией вывода.

Характеристики ввода-вывода влияют на производительность и эффективность ЭВМ.

Интерфейс – это совокупность шин, сигналов, элементов схем и алгоритмов, предназначенная для обмена информацией между устройствами. В настоящее время под интерфейсом все чаще понимают программный интерфейс, где речь идет о согласовании информационных потоков между различными программными модулями, системами программирования и т.д.

В интерфейсе, также как и в ЭВМ, вся информация передается по линиям передачи данных. Совокупность линий передач называют шиной. Их можно подразделить на:

- информационные (передача команд, адресов и данных);
- идентификации типа информации (передаваемой по информационным шинам);
- управляющие (синхронизация, инициирование и завершение передачи).

Отметим основные характеристики интерфейса:

- информационная шина (характеризуется количеством бит, передаваемых параллельно);
- скорость обмена информацией;
- максимальное расстояние передаваемой информации;
- связность:
  - односвязный И;
  - многосвязный И (обмен по нескольким независимым путям).

II. Способы организации и основные структуры систем ввода-вывода.

Обмен данными между ЭВМ и периферийными устройствами может быть организован в одном из трех режимов (см. рис. 2.66.):

- программном;
- по прерыванию;
- прямом доступе.

Для программного режима характерно то, что периферийное устройство программно опрашивается о готовности его к приемо-передаче информации. Признаком готовности ПУ является появление от него сигнала «конец преобразования» (КП). Процедура приема-передачи начинается только после полного выполнения программного цикла.

При выполнении режимов прерывания и прямого доступа в память характерным является то, что в момент появления сигнала КП в ЭВМ

выполняется фоновая программа и до тех пор пока не закончится команда, в момент выполнения которой пришел сигнал, не начнется выполняться программа обслуживания прерывания или прямого доступа.

С точки зрения оперативности обмена данных – предпочтительнее программный режим, поскольку ЭВМ постоянно опрашивает ПУ и требуется наименьшее время отклика на появление сигнала КП. Недостатком является неэффективное использование ресурсов процессора, так как за время работы внешнего устройства, а оно значительно по сравнению с тактовой частотой Пр, не выполняется никаких других операций кроме опроса сигнала – конец преобразования.

Режим обмена данными по прерыванию позволяет более эффективно использовать ресурсы ЭВМ, по сравнению с программным. Ограничения на его использование могут возникнуть в том случае, если время выполнения подпрограммы обслуживания прерывания соизмеримо со временем работы ПУ.

Прямой доступ в память по процедурам, разрешающим обмен, адекватен режиму прерывания, но имеет более высокий приоритет. Обмен осуществляется массивами данных. Характерен дорогостоящими аппаратными затратами.

Следует отметить, что несмотря на разнообразие задач, решаемых ЭВМ, процессы происходящие на системной магистрали, ограничены небольшим числом основных действий. К таким действиям относятся: запись, чтение, прерывание, прямой доступ в память (см. рис.2.66.).

Операция чтения позволяет процессору получить необходимую для выполнения программы информацию: из ОП – код очередной команды или данные, из ПУ – слово состояние ВУ или данные.

В процессе операции записи процессор передает в ОП результат вычислений, а в ВУ новые значения управляющего слова или данные.

С помощью операции прерывания ПУ оповещает процессор о своей готовности к передаче данных. Эта операция предназначена для оперативной обработки поступающих из ПУ данных (событий).

Операция прямого доступа в память служит для быстрой передачи в ОП или из нее отдельных массивов информации под управлением контролера ВУ.

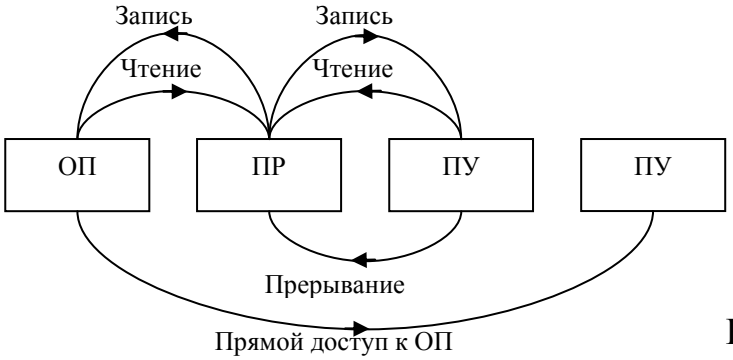


Рис. 2.66.

В процессе взаимодействия двух любых устройств ЭВМ одно из них обязательно выполняет активную, управляющую роль и является задатчиком (З), второе же оказывается управляемым, исполнителем (И). Чаще всего функцию задатчика выполняет процессор.

Другим важным принципом, заложенным в структуру магистрального (единого) интерфейса, является принцип запроса-ответа (квитирование). Каждый управляющий сигнал, посланный задатчиком, подтверждается ответным сигналом исполнителя (см. рис. 2.67.). При отсутствии ответного сигнала исполнителя в течение заданного интервала времени задатчик фиксирует ошибку обмена и прекращает операцию.

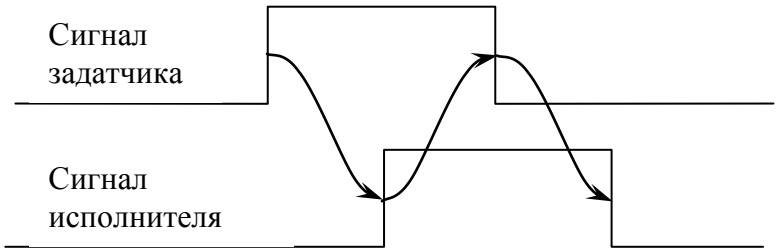


Рис. 2.67.

Основные структуры систем ввода-вывода.

Рассмотрим два типа структур: общая шина и иерархическая.

А) Структура с одним общим интерфейсом (общая шина) (см. рис. 2.68.).

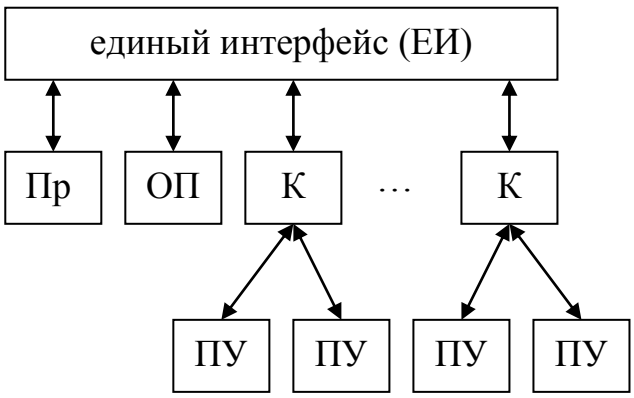


Рис. 2.68.

Основой любой ЭВМ является единый интерфейс, под которым понимают весь комплекс средств сопряжения процессора, оперативной памяти и периферийных устройств. Системный интерфейс представляет собой совокупность унифицированной магистрали для передачи

информации, электронных схем (для управления сигналами на магистрали), а также унифицированных алгоритмов (протоколов) обмена информации между отдельными устройствами ЭВМ.

Основными достоинствами магистрали «общая шина» (ОШ) является то, что все устройства подключаются идентично и имеют равный доступ к информации.

Магистральный системный интерфейс схематически можно представить в виде длинного многопроводного кабеля, к которому единообразно подключаются все устройства ЭВМ.

Устройства сопряжения с ПУ, называемые контроллером (К), служат для преобразования унифицированных сигналов магистрали в сигналы, управляющие работой периферийного оборудования. Сигналы независимо от места их возникновения (Пр, ОП, ПУ) поступают последовательно на все устройства. В задачу каждого устройства входит расшифровка этих сигналов и отбор нужных.

Правила обмена информацией при едином интерфейсе:

1) информация передаётся словами, равными ширине интерфейса;
2) в каждый момент обмен только между одной парой устройств – источником и приёмником информации;
3) прямой обмен между ПУ и ПУ невозможен;
4) конфликт при одновременной необходимости передать данные между несколькими устройствами решается на основе приоритетов.

Как правило, более высокий приоритет устанавливается для устройств с более высоким быстродействием.

ЕИ эффективен в малых и микро ЭВМ с коротким словом (1-2 байта), небольшим количеством ПУ и умеренной производительностью.

Рассмотрим процедуры обмена данными. Для этого представим структуру ЭВМ с магистралью «общая шина» (рис. 2.68.) в более детализированном виде (см. рис. 2.69.),

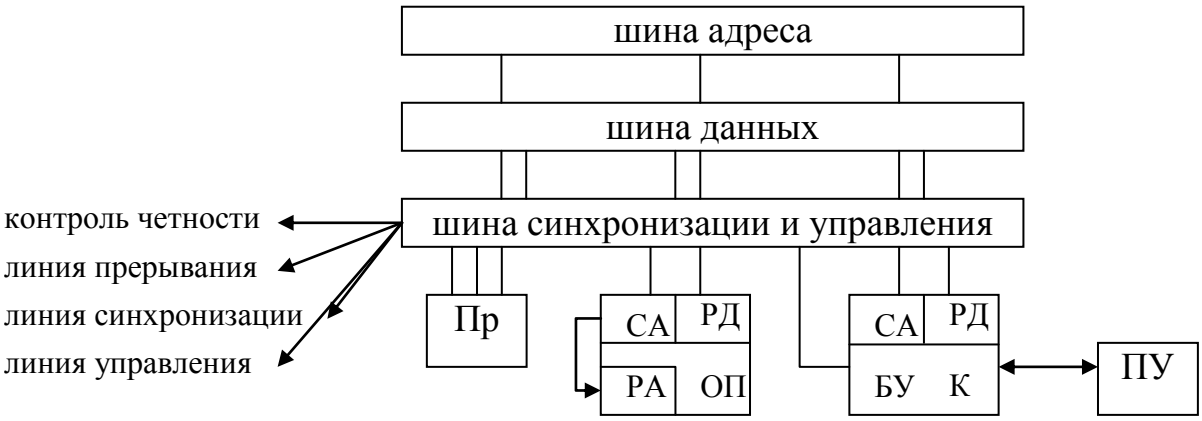


Рис. 2.69.

где: БУ – блок управления; СА – селектор адреса; РА – регистр адреса; РД – регистр данных.

В процессе обмена данными участвует лишь пара устройств: активное и пассивное (исполнитель). Инициатором процесса обмена выступает активное устройство, которое: захватывает шину интерфейса; выставляет на магистраль адрес пассивного устройства; организует процедуру квитирования; передает код операции, задающей направление обмена, и при выводе – выставляет данные на магистраль.

Пассивное устройство: считывает адрес с магистрали; проводит его идентификацию с собственным и в случае положительного решения – принимает данные, либо их выставляет на шину данных.

Процедура чтения/записи определяется относительно активного устройства.

Структура контроллера ПУ представлена на рис. 2.70.

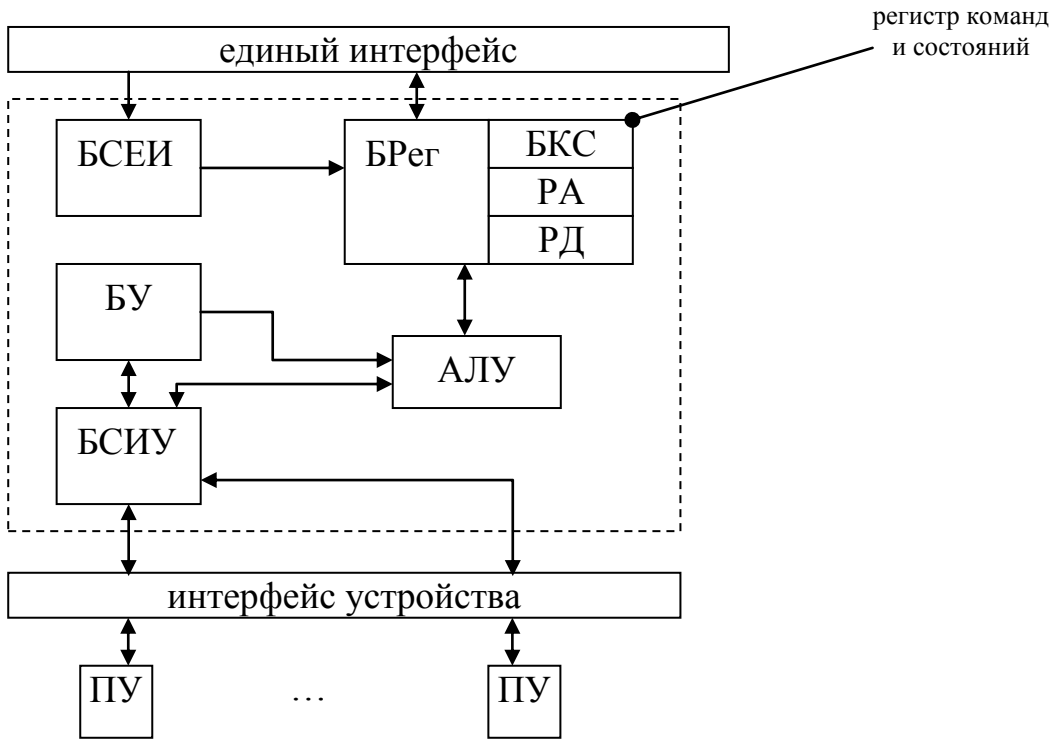


Рис. 2.70.

где: БСЕИ – блок сопряжения с единым интерфейсом; БСИУ – блок сопряжения с интерфейсом устройств. Все регистры блока регистров (Брег) доступны; БКС – блок команд и состояний.

БСЕИ включает в себя:

- селектор адреса устройства (схемы совпадения (СС)), если это адрес одного из регистров БР то СС выдаёт 1;

- дешифратор кода операции обмена;
- формирователь синхроимпульсов.

В простых контроллерах БУ и АЛУ отсутствуют.

Б. Иерархическая структура построения ЭВМ (см. рис. 2.71.).

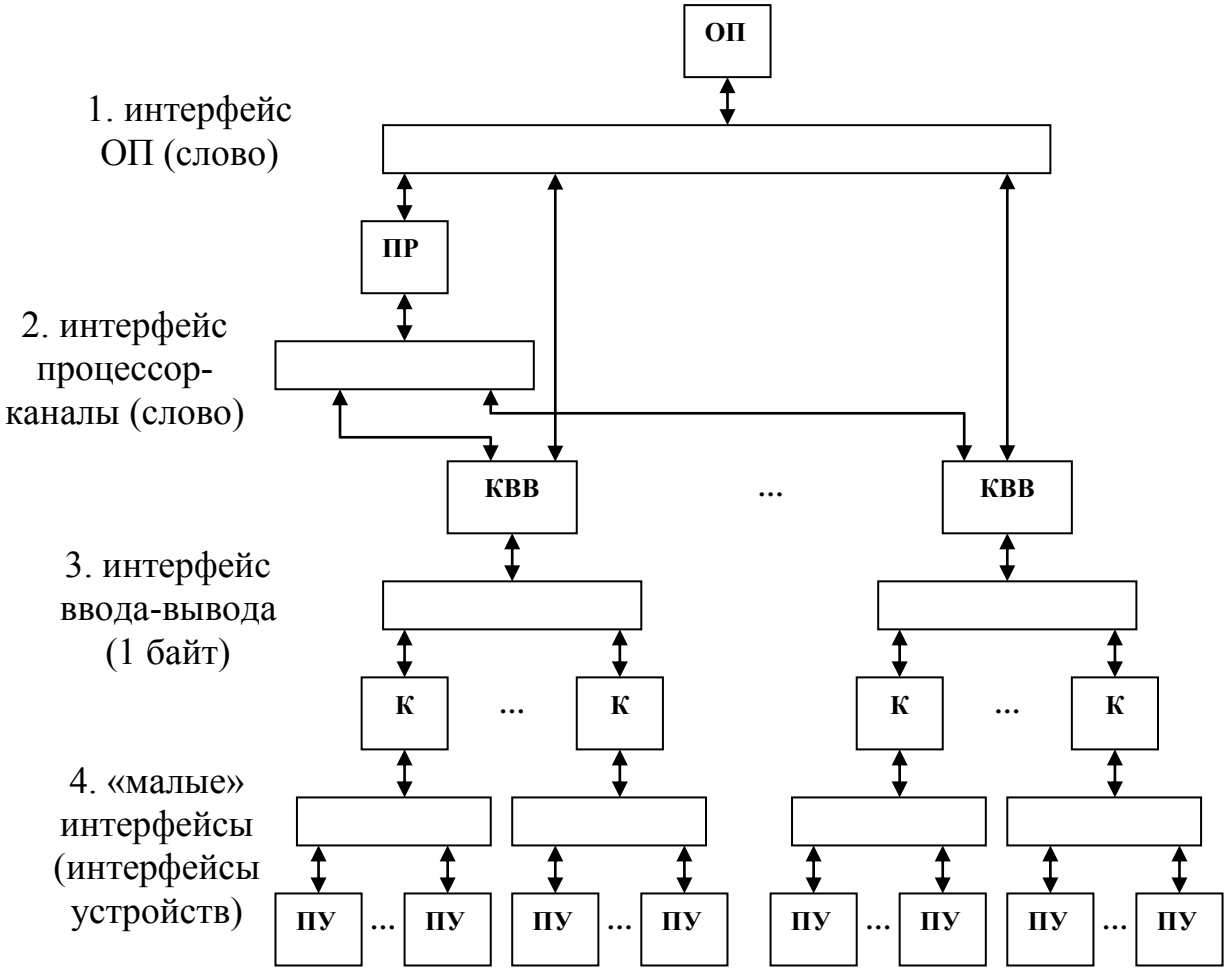


Рис. 2.71.

где: ККВ – программно-управляемый специальный процессор ввода-вывода.

Обменом данных руководит специальная программа: супервизор ввода-вывода.

Основным достоинством такого рода структуры является то, что основной процессор разгружен от управления вводом-выводом (благодаря КВВ), а недостатком – отсутствие однородности в структуре потоков и формах передаваемых данных.

Первые три уровня интерфейсов являются унифицированными, а 4-й нет, поскольку зависит от структурной организации конкретного ПУ. Обмен данными между уровнями зависит от возможностей интерфейсов. Например: первый и второй уровни поддерживаются быстродействующими интерфейсами, и обмен производится одинарными или двойными словами, а для 3-го уровня реализуется, в основном, 1 или 2-х байтный обмен (возможно и более).

Процедура обмена.

1. Процессор, получив команду ввода-вывода, передаёт её в канал (КВВ).

2. КВВ из фиксированной ячейки памяти канала выбирает начальный адрес канальной программы ввода-вывода и выполняет её. Каждая команда программы задаёт параметры одной операции обмена:

- направление обмена;
- начальный адрес;
- число слов.

3. КВВ, выполняя команды, инициирует работу ПУ и последовательно читает или записывает слова информации, обращаясь в ОП.

В. Основные типы и структуры КВВ.

В зависимости от соотношения быстродействия ОП и ПУ в КВВ реализуют два режима работы:

- монопольный – ПУ монополизирует канал на всё время передачи данных (см. рис. 2.72.а.);
- разделения времени (мультиплексирования) – каждое из ПУ связывается на короткие сеансы связи (см. рис. 2.72.б.). Передача данных по 1 или нескольким байтам.

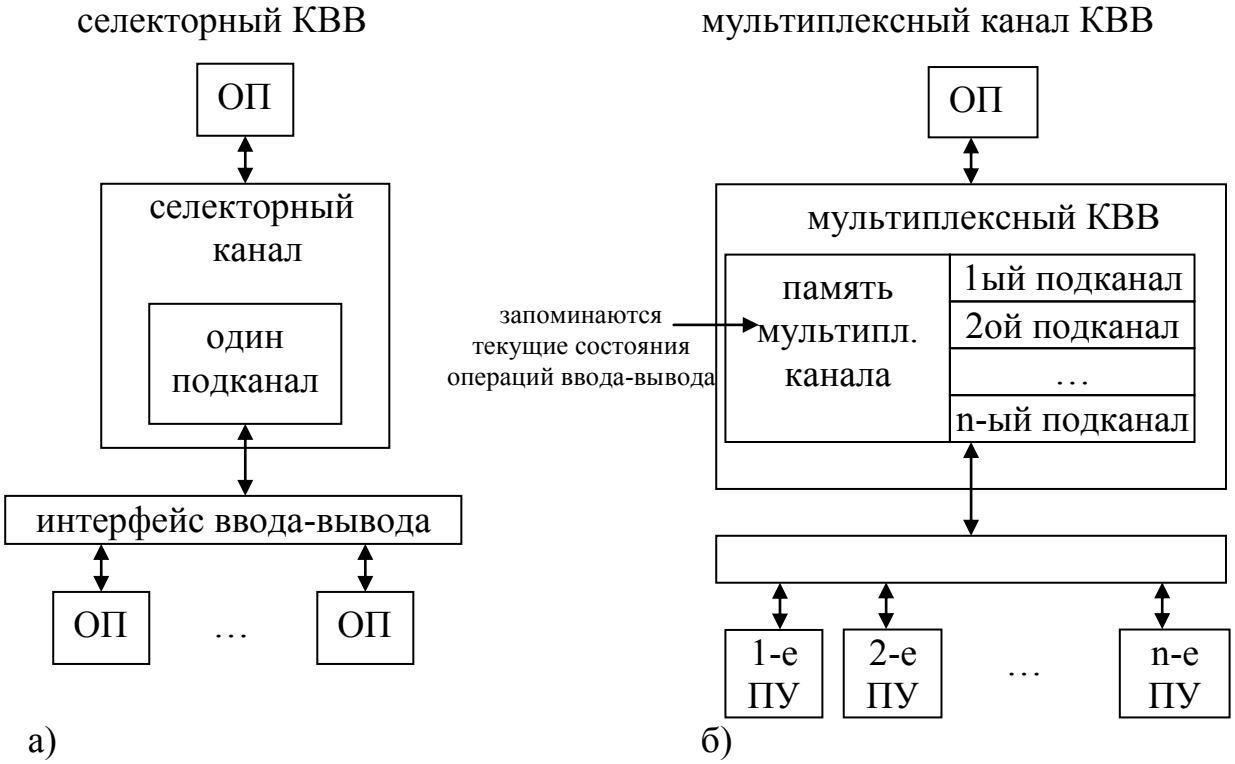


Рис. 2.72

3. Организация систем.

3.1. Принципы построения оптимальных систем.

Любую систему (аппаратную или программную) можно представить в виде системы уравнений, параметры которой описывают ее функционирование (3.1).

$$\begin{cases} F_{optBT}=f_1\{x_1, x_2,\ldots, S_1,N,\ldots\} \\ F_{optВнУстр.}=f_2\{x_1, x_2,\ldots, S_2,N,\ldots\} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ F_{opt\,n}=f_n\{ x_1, x_2,\ldots, S_n,N,\ldots\} \end{cases} \qquad (3.1)$$

F_{opt} – отдельные функции системы. Если речь идет об аппаратно-программном комплексе, то могут рассматриваться, например, функционалы, описывающие характеристики средств вычислительной техники (ВТ), внешних устройств (ВнУстр) и т.д. Структура такого рода системы представлена на рис. 3.1.

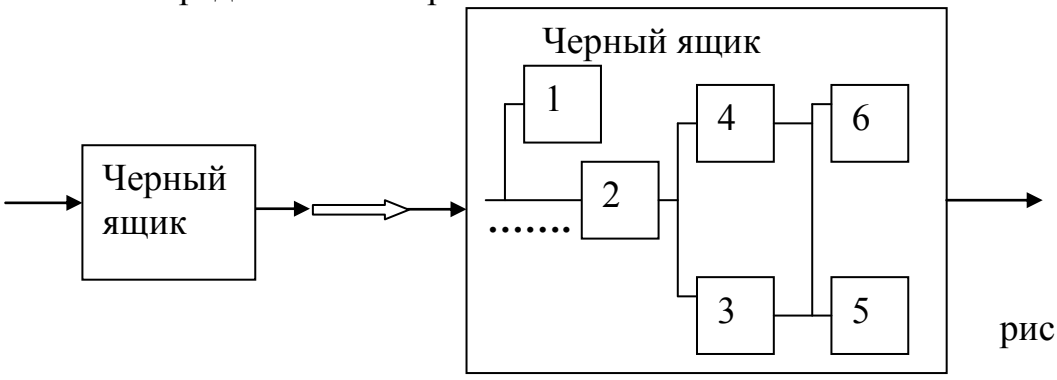


рис. 3.1.

Подсистемы 1, 2 и т.д. реализуются таким образом, чтобы число связей между элементами было минимальным, а количество выполняемых функций каждым элементом максимально.

Решение системы уравнений (3.1), в общем виде, заключается в поиске ее частных производных. В общем случае, число параметров такого рода систем стремится к бесконечности и число частных производных, соответственно, тоже. Например, для средств вычислительной техники параметры могут быть следующего типа: быстродействие (x_1), объем оперативной памяти (x_2), объем постоянного запоминающего устройства (x…),…,характер изготовления ЭВМ (промышленного или бытового типа), потребляемая электроэнергия, параметры надежности, стоимость, и т.д. Естественно, что какие-то из этих параметров носят первостепенный характер, а какие-то – второстепенный. Все зависит от конкретно решаемой задачи.

С этой точки зрения, основной задачей разработчиков и пользователей при построении и использовании оптимальной системы является задача поиска квазиоптимальной системы. То есть необходимо минимизировать число функций ($F_i \leq P$) и параметров ($x_1 \leq n$) системы уравнений (3.1), что приведет к практически возможному решению данной системы. При этом необходимо осознавать, что любые вводимые ограничения приводят к тем или иным искажениям конечного результата.

В зависимости от конкретно решаемой задачи можно использовать универсальные или специализированные средства. Универсальные средства, по сравнению со специализированными, уменьшают стоимость системы, позволяют решать более широкий круг задач, облегчают перенастройку, увеличивают время решаемой задачи и т.д. Все вышесказанное в равной мере относится, как к аппаратным, так и программным комплексам. Остановимся более подробно на аппаратных системах.

Они могут быть реализованы следующим образом:

1. С встроенными устройствами управления.

Пример: сверлильный станок с числовым программным управлением. Его функции ограничены типоразмерами обрабатываемой детали, набором сменного оборудования, качественными и количественными характеристиками обрабатываемого изделия и т.д.

То есть, такая система хороша для специализированных задач, но если задача меняется, то все приходится менять.

2. С внешними устройствами управления.

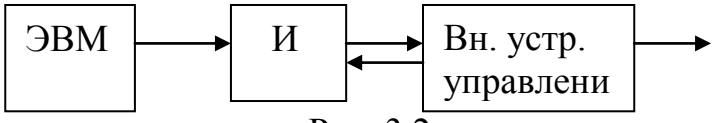


Рис. 3.2.

На рисунке 3.2. представлена система, которая позволяет осуществлять взаимодействие одной ЭВМ, через интерфейс (И) и внешнее устройство управление, с одним внешним объектом (О). Если в ЭВМ присутствует более одного свободного порта, то число подключаемых объектов может быть увеличено.

Такая система позволяет использовать специализированные и универсальные средства, но ресурсы машины, при этом, полностью не реализуются. Объясняется это тем, что быстродействие ЭВМ, как правило, во много раз превосходит возможности канала ввода/вывода информации (КВВИ), а компьютер в оперативном режиме решает задачи внешнего устройства, то есть неэффективно тратит свои ресурсы.

С точки зрения быстродействия такая система близка к системе 1-го типа.

3. На базе промышленных контроллеров (ПК).

Простейшая система такого типа представлена ниже на рисунке 3.3. В ее состав входят ЭВМ, ПК, состоящий из контроллера и магистрали крейта, и внешних устройств (ВУ), подключаемых к посадочным местам магистрали.

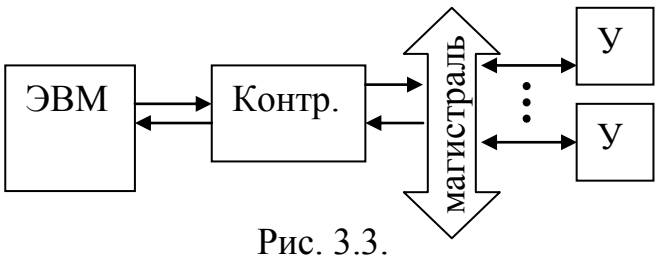


Рис. 3.3.

В такой системе, по сравнению с предыдущими, увеличивается число абонентов, но уменьшается скорость обмена данными (оперативность принятия решения). Количество внешних устройств, подключаемых к ЭВМ, определяется числом посадочных мест магистрали. Если внешний объект очень сложен и требует большого числа посадочных мест, то на базе системы рис. 3.3. проектируются многомашинные и/или многоконтроллерные системы. При этом обязательным условием работоспособности системы является обеспечение реального масштаба времени функционирования объекта.

Несмотря на внешнюю простоту структуры рис. 3.3., она является основополагающей для всех ведущих фирм мира, работающих в области автоматизации производства, диагностики, контроля качества продукции и т.п.

Самым «узким» местом систем, связанных с обменом информацией, является канал приемо-передачи информации (рис. 3.4.). Определяется это тем, что ЭВМ может обрабатывать и вырабатывать огромное количество

информации. Внешний объект или процесс, в свою очередь, может создавать или поглощать еще больший объем, а канал приемо-передачи сродни водопроводной трубе – диаметр трубы определяет возможное количество передаваемой воды.

Рис. 3.4.

Необходимо также обратить внимание на то, что при решении задач систем автоматизированного управления производством необходимо реализовывать следующие аппаратные и программные функции:

1. Сбор информации;
2. Хранение, обработка, архивирование и предоставление информации и т.д.;
3. Управление технологическими процессами.

Управление должно осуществляться в реальном масштабе времени, то есть в темпе протекания эксперимента.

Желание создать оптимальную систему привело разработчиков и пользователей к выработке основных принципов построения систем автоматического производства. Некоторые из них приведены ниже:

1. Блочно-модульный принцип;
2. Открытость архитектуры (возможность увеличения количества модулей и функций);
3. Унифицирование и агрегатирование;
4. Наличие самоконтроля на работоспособность.

Первый принцип позволяет осуществлять оперативную замену вышедшего из строя модуля и строить системы по типу игрушечного конструктора: ставится задача, имеется набор стандартных модулей и в заданном пространстве связей и объема осуществляется попытка получения конечного результата при известном входном воздействии (классическая задача «черного ящика»).

Открытость архитектуры – сродни желанию покупателя на рынке: побольше, получше и подешевле. Мы хотим иметь такую систему, чтобы любое изменение ее архитектуры и связей, по желанию пользователя, гарантировало бы ее работоспособность и обеспечивало бы желаемый результат. Реализация этого принципа на практике осуществляется путем ограничений сверху для количества модулей и функций. Например, число модулей не более 24-х или 48-и и т.д. – такое ограничение может быть введено для модулей крейта КАМАК, где в одном крейте число посадочных мест – 25.

Очень важным для практики является третий принцип. В мире существует очень большое количество фирм выпускающих однотипную аппаратуру (программное обеспечение). И если каждая фирма будет пользоваться своими стандартами на их выпуск (что происходило в нашей стране лет 10-15 назад и более), то пользователь не может безболезненно осуществлять замену отдельных элементов одной фирмы на другую. А предпосылок для этого очень много: банкротство фирм, невозможность использования отдельных элементов в конкретных климатических условиях и т.д. Этими причинами обусловлено то, что подавляющее число стран мира старается придерживаться общепринятых мировых стандартов. Рассмотрим несколько примеров. В качестве одной из составляющих для принятия решения о работоспособности турбоагрегата электрической станции используется информация о его вибрационном состоянии. Для сбора первичной информации могут быть использованы датчики (Д) перемещения. Их установка на турбоагрегат – дело трудоемкое и дорогостоящее, поэтому типоразмеры и электрические параметры этих датчиков у различных фирм конкурентов одинаковые. Если вдруг появится какая-то фирма, выпускающая аналогичные датчики с другими характеристиками, то ей понадобиться очень много средств, чтобы убедить владельцев электрических станций оснащать турбоагрегаты именно ее продукцией. Для согласования сигналов после датчиков по уровню, виду и качеству с последующей аппаратурой используют вторичные преобразователи (ВП). Их входные и выходные сигналы унифицируют таким образом, чтобы имелась возможность их использования независимо от типа датчика (температурный, давления, перемещения,…) – лишь бы выходной сигнал Д соответствовал входному ВП. И т.д.

Современные системы (и аппаратные и программные) необходимо обеспечивать самоконтролем на работоспособность. Обусловлено это тем, что пользователь очень часто не в состоянии оценить достоверность получаемой информации. Что в свою очередь может привести к непоправимым, а иногда и катастрофическим последствиям. На рис. 3.5. приведен пример структуры с самоконтролем для датчиковой аппаратуры. В качестве сигнала самоконтроля используется постоянная составляющая

(при этом считается, что измеряется чисто переменный сигнал). Если на выходе Д отсутствует постоянная составляющая, то переменная составляющая является шумом и канал не работоспособен.

Система с самоконтролем.

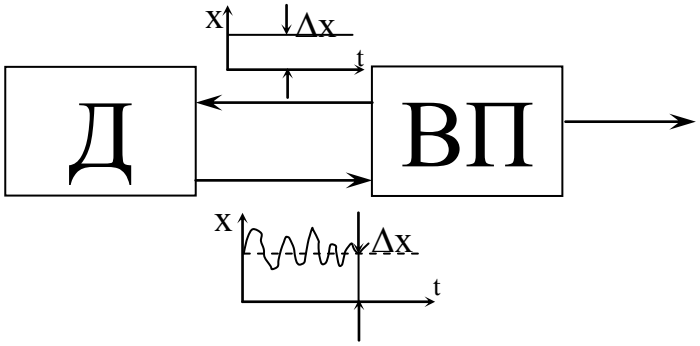


Рис. 3.5.

3.2. Топология информационно-вычислительных систем.

Техническая реализация различного рода систем с 60-х годов прошлого века идеологически не притерпела кардинальных изменений. Типовые структуры:

- общая шина;
- звезда;
- кольцо;
- иерархическая (дерево);
- полносвязанная;
- смешанная.

Можно говорить лишь об исползовании отдельных элементах (современных на сегодняшний день), например: цифровое перо (использовалось в ЭВМ – МИР в середине 70-х годов прошлого века); страничная организация памяти (ранее постраничная – сегодня прямая); регистры общего назначения; средства и устройства приема-передачи данных (Wi-Vi) и т.д. Все это относится к современным технологиям и изготовлению различного рода продукции на их базе (как аппаратной, так и программной). При этом следует помнить, что базовые понятия остаются неизменными.

3.3. Реализация технических систем на базе контроллеров.

Контроллеры – это устройства, позволяющие увеличить число подключаемых абонентов к ЭВМ и имеющие возможности предварительной обработки данных. Контроллеры можно разбить на два больших класса: для научных исследований (КНИ) и промышленные контроллеры (ПК). Их названия отражают качество изготовления, а различия такие же как для бытовой и промышленной аппаратуры.
Все контроллеры изготавливаются в соответствии со стандартами на них разработанными. Для КНИ наибольшее распространение получила аппаратура в стандарте CAMAC, а для ПК – Евростандарт.

Для всех контроллеров общим является следующее:

1. Они имеют три функциональные части: крейт (где размещаются функциональные модули); блок питания (может использоваться один или несколько контроллеров); интерфейс, осуществляющий связь с ЭВМ (называемый контроллером крейта – КК);
2. Возможность строить многоконтроллерные системы;
3. Основными функциями модулей является: сбор; хранение; диагностика; частичная обработка информации; выдача управляющих сигналов и т.д.;
4. Возможность организации межмодульных связей.

В крейте имеется определенное число (в зависимости от стандарта) посадочных мест, в которых размещаются модули. Модули могут занимать одно, два, три (как правило, не более) посадочных мест. Для интерфейса отводятся жестко закрепленные места. В зависимости от типа контроллера крейта они могут соединяться с ЭВМ той или иной архитектуры. Размеры модуля имеют стандартные габариты, и любой из модулей может быть установлен в любое посадочное место. Промышленные контроллеры используются там, где требования по влияющим факторам особые (повышенная влажность, вибрация, радиоактивность и т.д.).

На рис. 3.6. представлена структурная схема использования контроллеров в системах автоматизации производства. Где ШУ – это штатные устройства ЭВМ, такие как: процессор, память, дисплей с интерфейсом и т.д. В зависимости от типов контроллеров схема будет видоизменяться.

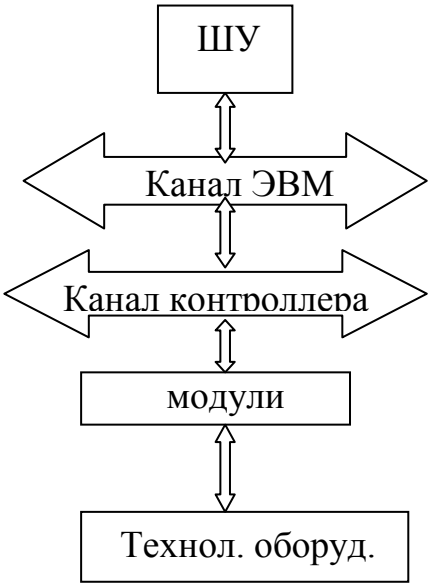


Рис. 3.6.

При построении многоконтроллерных систем возможно использование различных топологических решений. Например, на рис. 3.7. представлена группа ПК, объединенных последовательно магистралью данных (общая шина).

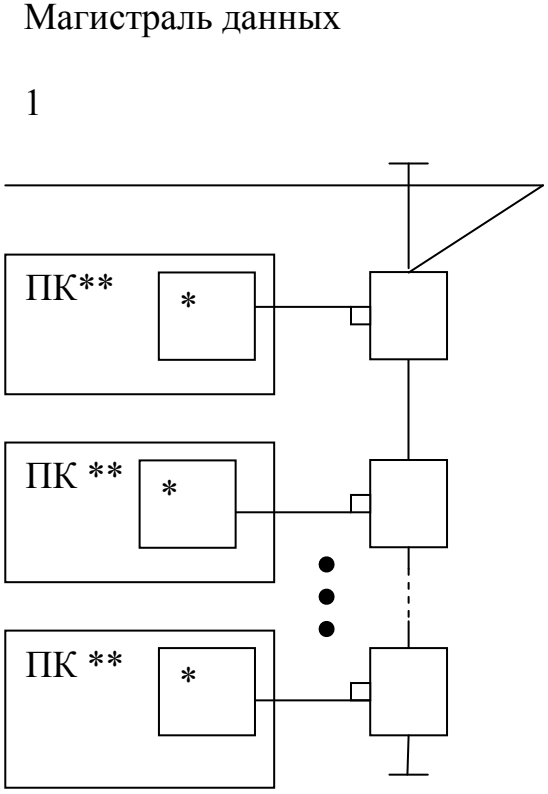


Рис. 3.7.

Структуру рис. 3.7. можно модифицировать (см. рис. 3.8.) в группу ПК, объединенную 4-мя магистралями данных:

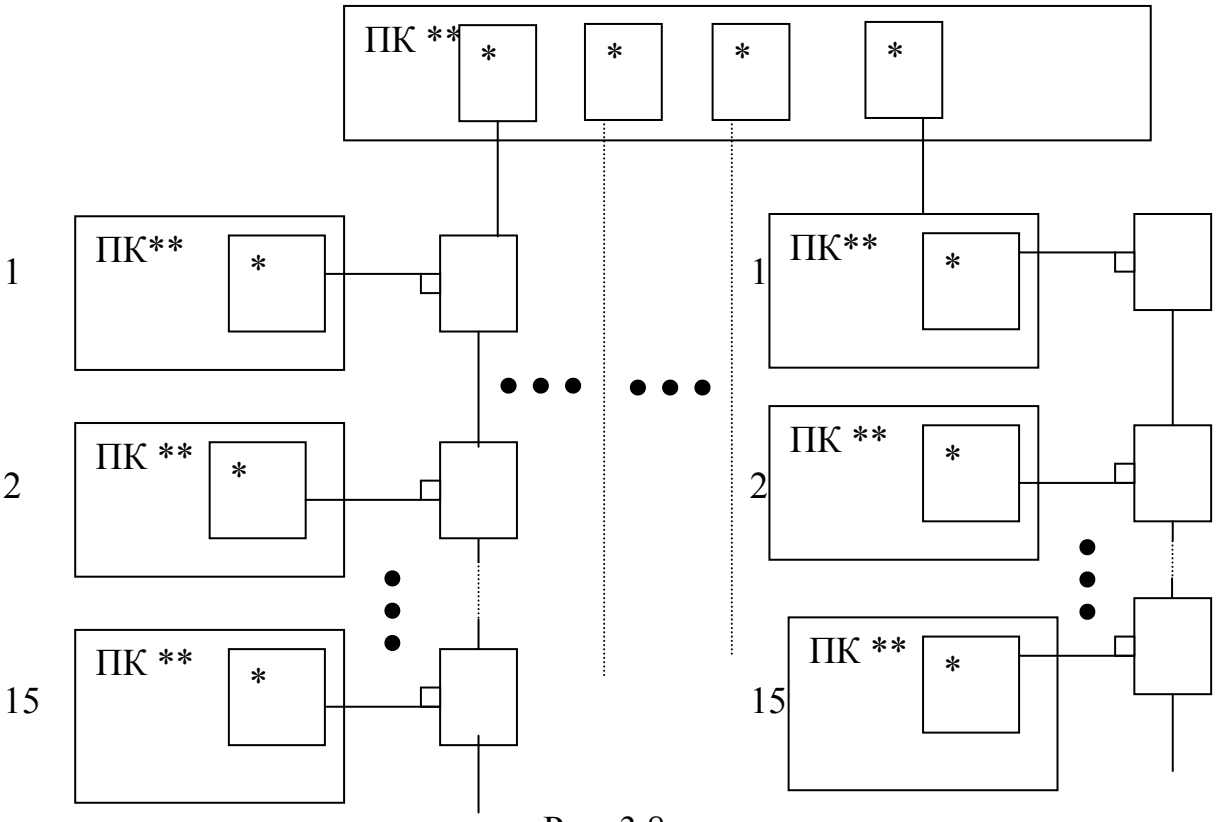


Рис. 3.8.

Отечественным аналогом Евростандарта являются контроллеры Микродат, которые позволяет объединять ПК в группы, строить иерархические системы, локальные сети (рис 3.6. и рис. 3.7.). Где:

*-модуль последовательного ввода/вывода;

**-ПК128 или ПК248.

Объединения ПК, изображённые на рис. 6 и рис. 7 могут охватывать территорию до 8 км. Если существует необходимость охватить большую территорию, то используются устройство связи объектом (УСО), схема связи которых представлена далее на рисунке 3.9.

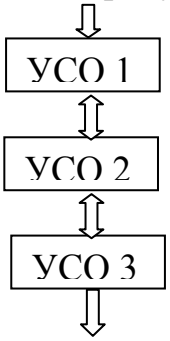


Рис. 3.9.

Таким соединением можно охватить территорию с неограниченной площадью, т.к. происходит ретрансляция. Недостатком является то, что при неограниченном росте время выполнения будет неограниченно и при выходе одного из устройств из строя будет нарушена связь. Следует помнить, что для промышленных систем основным фактором является обеспечения функционирования системы в реальном масштабе времени.

Аппаратура для научных исследований.

В нашей стране разрабатывались и изготовлялись системы, в основном, трех стандартов: САМАС (или КАМАК), ВЕКТОР, МЭК-685. Последний стандарт ориентирован на приборостроение и мы на нем останавливаться не будем. ВЕКТОР является модификацией САМАСа и в настоящее время практически не используется. Основной причиной этого является то, что его модули не совместимы с САМАСом.

КАМАК представляет собой систему электронных модулей, предназначенных для ввода/вывода и первичной обработки информации и управляемых от ЭВМ. Первоначально он разрабатывался для управления работой синхрофазотрона, а в дальнейшем области его применения расширились. В частности, в Санкт-Петербурге на его базе был реализован диагностический комплекс для исследования технических характеристик тракторов «Кировец». Общий вид КАМАК представлен на рис. 3.10.

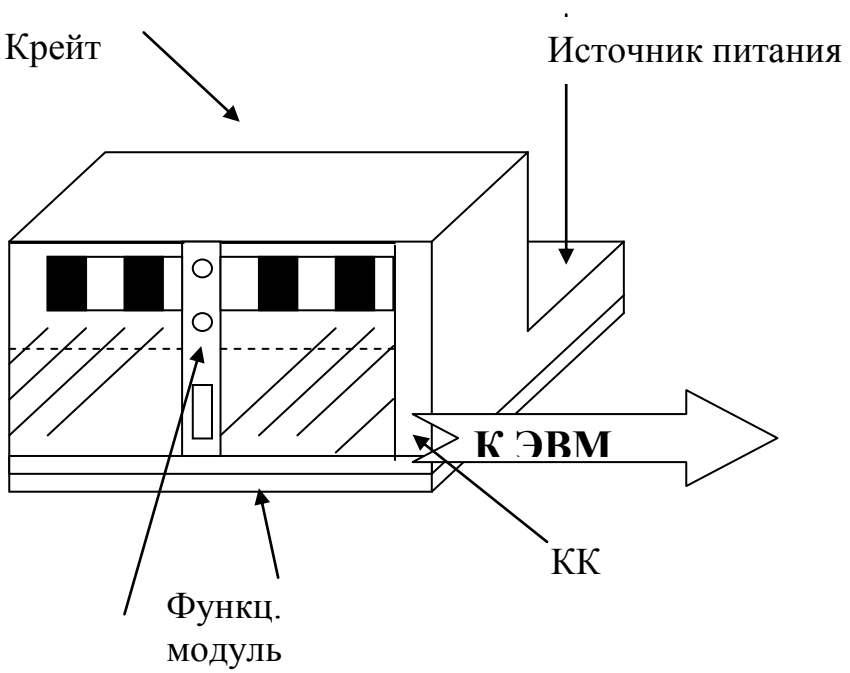


Рис. 3.10.

КАМАК удачно объединяет в себе, с одной стороны, богатый набор электронных функциональных модулей (ФМ) самого разнообразного назначения (усилители, счетчики, таймеры, аналого-цифровые преобразователи, запоминающие устройства и т.д.), а с другой стороны – средства связи всей этой аппаратуры с ЭВМ, для чего предусмотрен специальный управляющий модуль –контроллер КАМАК. Характерным для системы КАМАК является наличие унифицированного канала передачи данных (магистрали) между отдельными модулями и контроллером. И модули, и контроллер имеют выход на магистраль, по линиям которой происходит обмен рабочей и служебной информацией, а также питание модулей: контроллер кроме того связан с ЭВМ (рис 3.11.). Всеми процессами на магистрали управляет (по командам от ЭВМ) контроллер, однако если в модуле возникла ситуация, требующая вмешательства ЭВМ, модуль может послать в контроллер запрос на обслуживание и инициировать тем самым конкретную программу обработки.

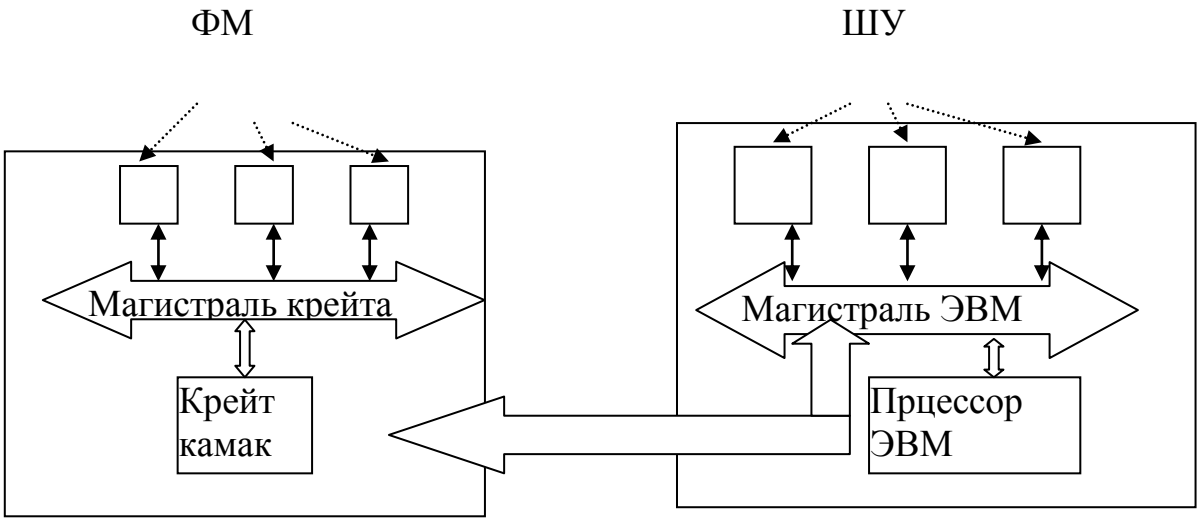


Рис. 3.11.

Стандартизация модулей по конструкции, способу подсоединения к магистрали, характеристикам электрического питания, параметрам входных и выходных сигналов позволяет быстро собирать и модернизировать экспериментальные установки, комплектуя их требуемыми модулями, а единая система команд существенно облегчает разработку алгоритмов управления системой. При этом компоновка любой системы сводится по существу к составлению программы взаимодействия ЭВМ с модулями (порядок опроса состояния модулей, записи и съема информации и т.д.), технические же вопросы согласования модулей друг с другом или с контроллером отпадают ввиду стандартизации системы.

Конструктивной основой системы КАМАК является специальный каркас – крейт, содержащий 25 станций – направляющих, по которым в крейт вдвигаются модули. В зависимости от сложности модуль может иметь единичную ширину 17,2 мм и занимать одно место в крейте либо ширину, кратную указанной. Контроллер обычно занимает два крайних правых места. Таким образом, в крейте может размещаться до 23 различных модулей. Каждый модуль имеет стандартный 86-контактный разъем для подсоединения к магистрали. Разводка линий магистрали по контактам разъемов всех станций (кроме крайней правой, принадлежащей контроллеру) выполнена единообразно, что позволяет устанавливать любые модули на любые места крейта.

На рис. 3.12. приведено схематическое изображение магистрали крейта. Большая часть линий магистрали – параллельные линии, соединяющие одноименные контакты всех разъемов; сигналы, передаваемые по этим линиям, доступны всем модулям.

Рис. 3.12.

Рабочая информация в системе КАМАК передается 24-разрядным двоичным параллельным кодом, для чего служат 24 линии чтения R (передача из модулей в контролер) и 24 линии записи W (передача данных из контролера в модули). Поскольку в каждом модуле могут размещаться несколько функциональных узлов (например, несколько счетчиков) и, кроме того, еще имеются многочисленные обслуживающие схемы, для адресации к элементам модуля служат 4 линии субадреса A, по которым номер узла в модуле или его субадрес передается также двоичным параллельным кодом. Всего, таким образом, в каждом модуле может использоваться до $2^4 = 16$ субадресов.

В процессе обращения контроллера к модулю может быть задано выполнение различных операций – чтение или запись информации, опрос состояния регистра и т.д. Для передачи кода операции предусмотрены 5 линий функций F, что дает возможность использовать до 32 различных функций. Значения функций стандартизированы, например, функция F(2) никогда не используется для записи информации в регистр, а только для чтения его содержимого с последующим его сбросом. Назначение же регистров и характер содержащейся в них информации зависят от функционального назначения и конкретной схемы модуля.

Группа параллельных линий отводится для управления и передачи служебных сигналов. Сюда относятся линии (и соответственно сигналы) Z, C, I, B, Q, S1, S2. Некоторые из этих сигналов (B, S1, S2) генерируются контроллером или модулями автоматически в процессе обмена информацией по магистрали, на них нельзя воздействовать программным образом; другие сигналы устанавливаются, снимаются либо контролируются программно, и их назначение необходимо понимать для правильного составления программ управления.

Например, сигнал C (Сброс) вызывает сброс регистров модулей крейта.

Сигнал X (Команда принята) вырабатывается модулем всякий раз при получении им «законной» команды, которую данный модуль в состоянии выполнить. Нулевое значение сигнала X = 0 указывает на наличие неисправности (например, отсутствие адресуемого модуля) или серьезной ошибки в программе обслуживания (в модуль послана команда, которую он не может выполнить).

Две группы линий (N и L) служат для установления связи контроллера с определенными модулями. В отличие от остальных линий магистрали линии N и L имеют радиальный характер; каждый модуль связан с контроллером индивидуальной парой линий N и L. Когда контроллер генерирует команду обращения к какому-то модулю, он устанавливает соответствующую функцию КАМАК на линиях F, требуемый адрес на линиях A и возбуждает линию N, соответствующую адресуемому модулю. Сигналы F и A поступают во все модули. Однако воспринимает их только тот модуль, который подсоединен к возбужденной линии N, т.е. модуль, установленный на станции с номером N.

Если в модуле создалась ситуация, требующая вмешательства ЭВМ (АЦП преобразовал входной сигнал в код, счетчик зарегистрировал заданное число импульсов и т.д.), модуль может послать в контроллер запрос на обслуживание, установив логическую 1 на линии. Обычно возбуждение линии L (L-запрос) приводит к прерыванию текущей программы и переходу на программу обработки прерывания от данного модуля. Поскольку от каждого модуля в контроллер идет индивидуальная

линия L, контроллер, получив запрос, может определить, из какого именно модуля он пришел.

Как уже отмечалось, каждая команда, с которой контроллер обращается к какому-то модулю, состоит из трех элементов: функции F, субадреса A и номера адресуемого модуля N. Управление аппаратурой КАМАК и заключается в выполнении последовательности команд NAF (команд КАМАК), соответствующей заданному алгоритму функционирования установки. Требуемая последовательность команд NAF записывается в виде машинной программы.

Как и в любой программе реального времени, последовательность выполняемых операций не является фиксированной, а определяется ходом эксперимента. Например, получив от модуля АЦП запрос на обслуживание, ЭВМ выполняет программу приема из модуля и записи в оперативную память подготовленного модулем кода входного сигнала. Если, однако, при этом выясняется, что общий объем накопленной информации достиг заданного значения, ЭВМ выполняет программу выключения модуля АЦП – блокировки его входа, запрещение генерации им L- запросов и т.д.

Система сбора данных и диагностики на базе ПК.

Рассмотрим систему сбора данных и диагностики на примере выпускаемой фирмой «Филипс», которая специализируется на изготовление и разработке блоков, модулей, устройств и систем сбора данных и диагностике, сопрягаемых между собой и применяемой в условиях промышленной эксплуатации объектов. Для примера отметим, что фирма «Брюль и Къер» производит аналогичную продукцию, но в конструктиве пригодном для использования в научных исследованиях.

Фирма «Филипс» для своих блоков, модулей и т.д. использует стандарт Евромеханика. Аналогичные стандарты существовали и у нас в стране: БУК – блок унифицированный комбинационный, БУК Б. БУК БМ (предприятия общепрома); Рябина (на предприятиях средмаша). Из контроллеров применяемых в атоматизации производства наиболее близок (по габаритам) к стандарту Евромеханика контроллер микроДАТ.

Система сбора данных и диагностики состоит из множества параллельных каналов и следующих уровней преобразования информации (каждый из которых функционально закончен и может быть использован без вышестоящего уровня): датчиков (первичных преобразователей); вторичных преобразователей; каналов ввода- вывода; ЭВМ низшего звена; ЭВМ верхнего уровня.

Вся система построена по блочно–модульному принципу, все элементы унифицированы и каждый может использоваться автономно.

На рисунке 3.13. представлена простейшая измерительная система (называемая монитором), которая состоит из датчика и вторичного преобразователя.

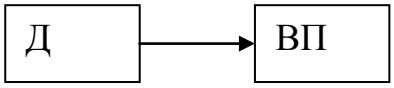


Рис. 3.13.

В конструктивном исполнении фирма «Филипс» отдает предпочтение раздельному изготовлению датчика и вторичного преобразователя (хотя работы ведутся и в противоположном направлении). Раздельное изготовление датчика и ВП позволяет уменьшить число проводов в кабеле и вероятность сбоев при передаче информации. В зависимости от уровня сигнала, вырабатываемого датчиком, возможно удаление его от вторичного преобразователя на то или иное расстояние. Для разных типов датчиков (перемещения, индуктивных, тензорезестивных, емкостных и т.д.) это расстояние колеблется от единиц до сотен метров.

Все элементы унифицированы. Датчик представляет собой законченное изделие со стандартным креплением и унифицированными сигналами. Между собой датчик (Д) и вторичный преобразователь (ВП) соединены стандартным кабелем, имеющим унифицированные характеристики. ВП имеет средства представления информации и унифицированные сигналы входа и выхода.

Вторичный преобразователь представляет, из себя законченный, конструктивный модуль со своими средствами визуального контроля текущей информации, структурная схема которого для одного канала представлена на рис. 3.14.

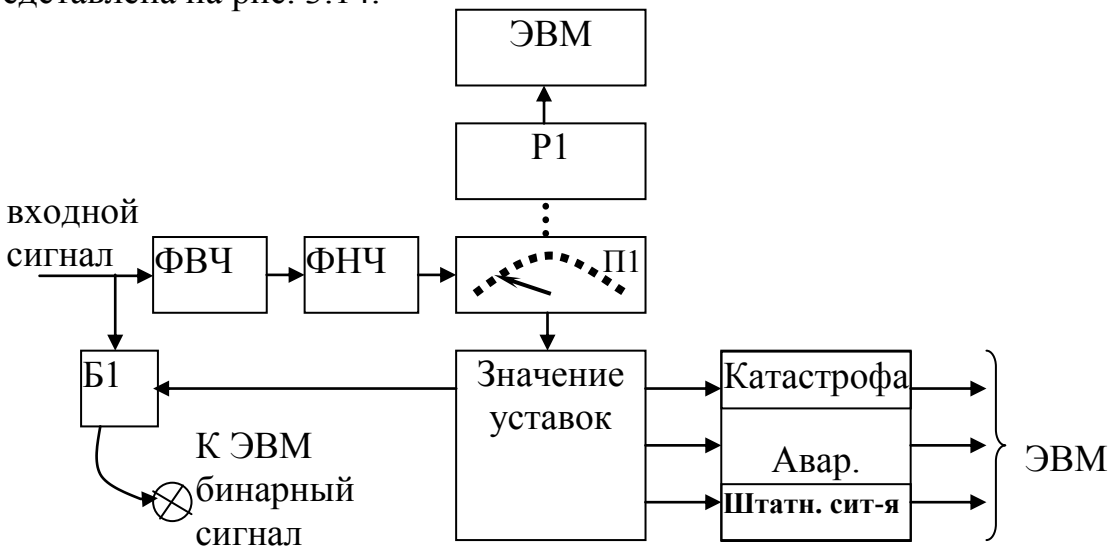


Рис. 3.14.

Основные задачи, решаемые монитором:

- фильтрация входного сигнала (ФВЧ и ФНЧ);
- сравнение входного сигнала с уставкой, выставляемой потенциометрами (диапазон выставляется переключателем – П1);
- выработка бинарного сигнала (Б1) неисправности датчика с одновременной индикацией (зажженная лампочка) на передней панели ВП;
- выработка диагностирующих сообщений.

При использовании датчика перемещений такой монитор может быть использован в системах вибродиагностики турбоагрегатов. При этом в реальных системах число такого рода унифицированных каналов достигает – 1000. Например, в аппаратуре сбора данных и диагностики PR3000.

Если используются другие типы датчиков, то вторичный преобразователь перенастраивается путем переналадки диапазона и изменения значений уставок.

При подключении, к вышеуказанной структуре, ЭВМ меняется ее назначение:

- измерение;
- масштабирование и линеаризация входных сигналов;
- контроль;
- представление информации на экране дисплея;
- хранение информации в памяти;
- создание отчетов по хранимым данным.

Если требуется создать многоконтроллерную систему (охватить значительную территорию и большое число абонентов), то фирма «Филипс» предлагает использовать для этих целей архитектуру «Кольцо» (см. рис. 3.15.). Интерфейс RS422 позволяет разносить устройства на расстояние до 500 м, а общее их число доводить до 16.

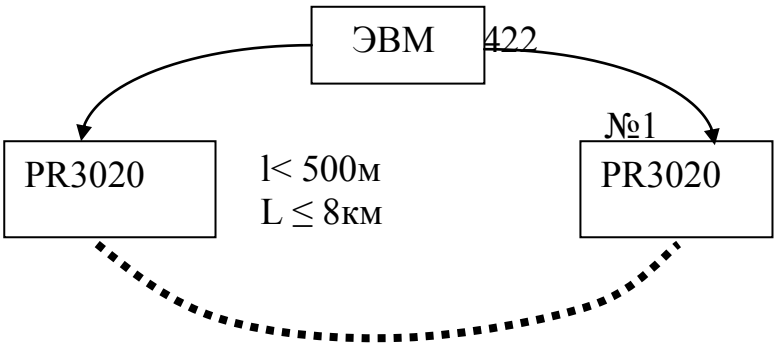


Рис. 3.15.

Отметим, что кроме аппаратного обеспечения необходимо использовать также программное, решающее различные задачи (например, слежение за динамикой сигнала и т.д.).

3.3. Сети Петри

Для того, чтобы проектировать любой объект управления необходимо учитывать не только структуры связей, но и возможные изменения в их функционировании из-за отказов элементов, а также деградации их параметров. Отсюда следует, что для достижения требуемых параметров эффективности проектирование объекта управления (ОУ) должно проводиться с использованием функционального подхода (ФП). ФП базируется на критерии эффективности функционирования ОУ, включающем эффективность структуры связей и узлов, эффективность функционирования средств передачи информации (СПИ) при возможных отказах элементов связи, каналов связи и т.д. Решение такого рода задачи предполагает учитывать как структуру ОУ, так и функцинирование ее с сохранением основных параметров в заданных пределах. Для описания и анализа объекта управления, к настоящему времени, предложено несколько десятков моделей. Наиболее общими из них являются сети Дейвиса, схемы Родригеса, схемы Адамса, потоковые схемы Корла-Миллера и др.

Наибольшее распространение среди моделей такого рода в начале 1970-х получили биологические графы, которые впоследствии были представлены сетями Петри. Это самый распространенный формализм, описывающий структуру и взаимодействие параллельных систем и процессов.

Теория сетей Петри развивается в нескольких направлениях: математическая теория сетей; структурная теория сетей; приложения к задачам параллельного программирования.

Сеть Петри формально представляется как набор вида:

N = (B, D, Ф, H), где:

B – конечное множество символов, называемых вершинами, $B \neq 0$;

D – переходы (дуги), $D \neq 0$, $B \cap D = 0$;

Ф: $B \cdot D \rightarrow \{0,1\}$ – прямая функция инцедентности;

H: $D \cdot B \rightarrow \{1,0\}$ – обратная функция инцедентности.

Такое задание удобно для формального рассмотрения, но не обладает наглядностью, поэтому сети Петри обычно представляют двудольным графом. Из вершины b_i в переход d_j ведет дуга, если и только если $Ф(b_i\ d_j) = 1$.

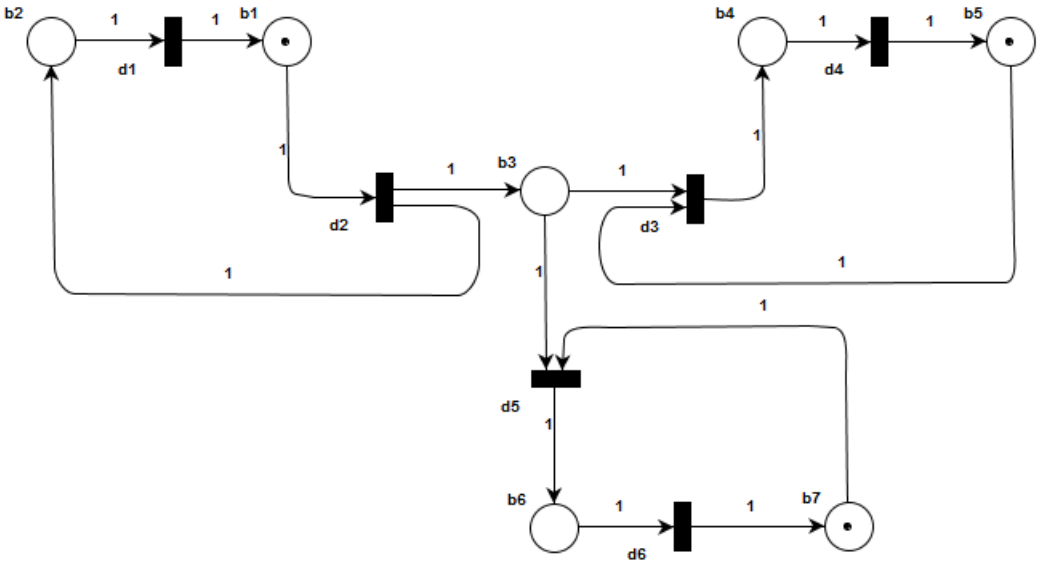
Для каждого перехода $d_j \in D$ можно определить множество b_i и выходных позиций перехода D следующим образом

$Ф(b_i) = \{di \in D/Ф(bi\ dj) = 1\}$

$H(b_i) = \{di \in D/H(di\ bj) = 1\}$

Рассмотрим пример:

Пусть задан граф следующего вида



$B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$,

$D = \{d_1, d_2, d_3, d_4, d_5\} + d_6$,

$\Phi(d_1) = \{b_2\}$ $H(d_1) = \{b_1\}$

$\Phi(d_2) = \{b_1\}$ $H(d_2) = \{b_2, b_3\}$

$\Phi(d_3) = \{b_3, b_5\}$ $H(d_3) = \{b_1\}$

$\Phi(d_4) = \{b_4\}$ $H(d_4) = \{b_5\}$

$\Phi(d_5) = \{b_3, b_7\}$ $H(d_5) = \{b_6\}$

$\Phi(d_6) = \{b_6\}$ $H(d_6) = \{b_7\}$

Наибольшее применение сети Петри нашли для моделирования систем с дискретными событиями, которые могут происходить одновременно и параллельно. При моделировании отражаются два аспекта систем: события (B) и условия (D). Функции инцендентности отражают связи между условиями и событиями в системе.

Приведенные выше формальное определение сети Петри – отражает только статику моделируемой системы, но не отражает динамику функционирования. Для представления динамических свойств вводится функция маркирования (разметки) сети

$M : B \rightarrow \{0,1,2\}$.

При графическом изображении сети Петри разметка (маркировка) отображается помещением внутри вершины соответствующего числа точек, называемых метками.

Сеть Петри функционирует, переходя от разметки к разметке. Начальная разметка обозначается M_0. Смена разметок происходит в результате срабатывания переходов $d_j \in D$. Срабатывание перехода d_j изменяет разметку сети M(b) на M'(b) по следующему правилу:

$M'(b) = M(b) - \Phi(d_j) + H(d_j)$,

то есть переход d_j извлекает по одной метке из каждой входной позиции и добавляет по одной метке в каждую из выходных позиций.

Сеть Петри, в которой используется функция разметки, называется размеченной сетью Петри и задается набором:

$N_M = (B, D, Ф, H, M_0)$, где: M_0 – начальная маркировка сети.

Рассмотрим пример.

Пусть начальная маркировка сети следующая:

$M_0 = \{1, 0, 0, 0, 1, 0, 1\}$.

При такой начальной маркировке единственным готовым к срабатыванию переходом является переход d_2. Его срабатывание ведет к смене разметки M_0 на M_1, где $M_1 = \{0, 1, 1, 0, 1, 0, 1\}$. При разметке M_1 возможно срабатывание трех переходов – d_1, d3, d5. В зависимости от того, какой переход (один из трех) сработал первый, получается одна из трех возможных маркировок.

Функционирование сети Петри продолжается до тех пор, пока существует хотя бы один возможный переход.

Размеченные сети Петри пригодны для описания событийпроизвольной длительности. Модель в этом случае отражает только наступление событий в рассматриваемой системе. Тот факт, что обычные сети Петри не отражают временных параметров моделируемой системы, является существенным недостатком при использовании их в качестве моделей вычислительных систем. Эти ограничения можно обойти, используя ряд расширений аппарата сетей Петри.

Подклассы и расширеня сетей Петри.

Использование обычных сетей Петри из-за высокой трудоемкости анализа – затруднительно. Это явилось причиной разработки подклассов сетей Петри, в которых вводятся определенные ограничения на структуру сети, что позволяет использовать более простые алгоритмы для ее анализа.

Подклассы сети Петри:

Подкласс автоматные графы – сети Петри с одной входной и одной выходной дугой у каждого перехода.

Ограничение на число входных и выходных переходов позиций сети Петри – признак подкласса маркированных графов.

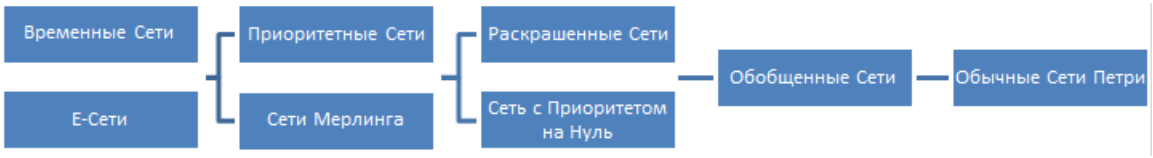
Условием выделения подкласса чистых сетей Петри является то, что переход не может иметь позицию b_i в качестве входной и выходной.

Если на сеть Петри накладываются ограничения, что каждая выходная дуго от вершины является ее единственным выходом, либо единственным входом перехода, то такую сеть относят к подклассу сетей свободного выбора.

Признаком выделения подкласса простых сетей Петри является наличие у каждого перехода не более чем одной входной позиции, имеющей больше одного выхода.

В теории сетей Петри предложено также несколько расширений, ориентированных на увеличение моделирующих возможностей данного аппарата.

Расширение аппарата сетей Петри



Граф обобщенной сети Петри имеет несколько дуг между позициями и переходами и использует веса дуг: $w_ф$ и $w_н$.

Раскрашенные сети Петри по своим возможностям эквивалентны обычным сетям. Введением в данной сети еще одного множества – раскрасок меток позволяет уменьшить размерность графа при моделировании сложных систем.

Приоритетные сети и сети с проверкой на нуль позволяют учитывать в модели приоритетность сообщений.

Временные сети и сети Мерлинга позволяют отражать в модели временные параметры системы. Задается временная сеть: $N_s = (B, D, Ф, H, M_0, u, v)$, где: $u = (t_1, t_2, \ldots t_i, \ldots)$ – возрастающая последовательность действительных чисел, называемая временной базой, а $v = B \cdot u \rightarrow u$ – функция временных задержек. Фактор времени учитывается путем введения пассивного состояния метки в позицию. При поступлении метки в позицию b_i она остается пассивной на время $v(b_i, t_s) – t_s$ и только после этого переходят в активное состояние.

Сети Мерлинга используют множество временных min и max задержек для переходов, то есть срабатывание перехода d_i может наступить через время не менее t_1 после его возбуждения и не более t_2 после указанного события.

Временные сети и сети Мерлинга не позволяют моделировать операции над данными, выполняемыми в системе, не отражают эквивалентностей процессов обработки от типа задачи.

Учет этих факторов возможен в наиболее широком расширение сетей Петри – класс E – сетей.

E – сеть задается совокупностью следующих множеств:

$N_s = (B, B_p, B_r, D, M_0)$, где:

B - конечное множество позиций, $B \neq 0$;

B_p – множество периферийных позиций;

B_r – множество решающих позиций;

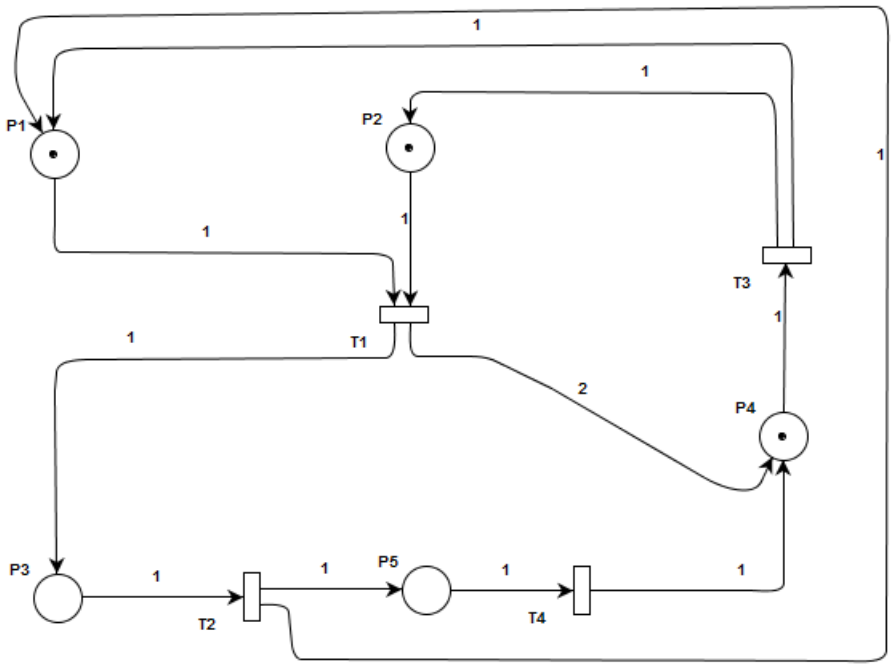
D – конечное множество описаний периодов $d_i = (s, H(d_i), p)$, а:

S - тип перехода;

$t(d_i)$ – время перехода;

p – процедура перехода.

Пусть задано:



$N = (B, D, \Phi, H, M_0)$

$B = (p_1, p_2, p_3, p_4, p_5)$

$D = (t_1, t_2, t_3, t_4)$

$M_0 = (1, 1, 0, 0, 0)$

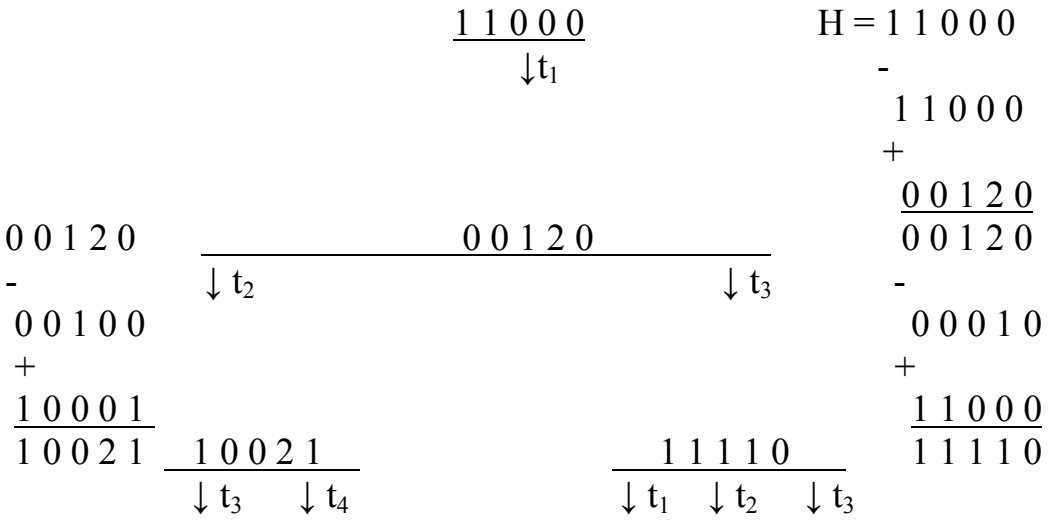
$\Phi =$

p_1	1	0	0	0
p_2	1	0	0	0
p_3	0	1	0	0
p_4	0	0	1	0
p_5	0	0	0	1
	t_1	t_2	t_3	t_4

$H =$

t_1	0	0	1	2	0
t_2	1	0	0	0	1
t_3	1	1	0	0	0
t_4	0	0	0	1	0
	p_1	p_2	p_3	p_4	p_5

Фрагмент графа заместимости для такой сети будет выглядеть следующим образом:



Принимаем – отказ события заключается в том, что разметка попадает в вершину p_5, то есть маркировка M' = (., ., ., .,1) – является не разрешенной. После отказа (попадания в p_5), система начинает работу заново. Примем за одну реализацию – прохождение через переход t_1.

Допустим, что переход t_1 срабатывает 10 раз. При этом разметка дважды попала в вершину p_5. Отсюда следует, что вероятность отказа будет равна Q = 2/10 = 0,2, а P = 1 – 0,2 = 0,8.

Далее можно рассчитать вероятности переходов.

Приложение № 1

Общие требования к тематике, выполнению и оформлению работ

Отчет о проделанной работе должен включать в себя, как минимум, следующие разделы: титульный лист (с подписью исполнителя); аннотацию/реферат (зависит от требований к выполняемой работе); полный текст задания (с подписью руководителя работы и ответственного данного направления обучения); содержание/оглавление с указанием страниц; введение; основная часть; выводы и результаты (заключение); список литературы; приложения.

Содержание и объем, выше указанных разделов, формируются преподавателем при выдаче задания. Общие требования к ним следующие.

Титульный лист должен соответствовать установленному образцу для СПбПУ и требованиям к выполняемой работе. Автор(ы), сдаваемой работы, обязаны расписаться на титульном листе и зафиксировать число, месяц и год сдаваемой работы.

Аннотация (при необходимости) раскрывает основное содержание работы и возможные направления развития для данного класса задач.

Текст задания определяет перечень проблем и задач, решаемых студентом.

Содержание является путеводителем, который определяет алгоритм и последовательность решаемой задачи.

Введение включает в себя общие вопросы, позволяющие оценить проблему со стороны (решения данного класса задач другими авторами) и изнутри (попытка представить реализацию проблемы со своей точки зрения). При этом, автор должен минимизировать общий объем своих исследований и предложений.

Основная часть состоит из разделов и подразделов, в которых рассматривается существо проблемы, аналитически решается рассматриваемая задача. Показывается возможность решения проблемы в виде: математического представления; графического; табличного и т.д. В рамках использования средств вычислительной техники для достижения желаемого результата, приводятся блок-схемы алгоритмов, листинги программ, краткое описание системы программирования. Желательно отметить использование конкретной ЭВМ и системы программирования для решения данной конкретной задачи. Проводится сравнение исходных данных к решению проблемы и полученных в результате эксперимента.

Заключение должно содержать количественные и качественные оценки полученных результатов и включает в себя обобщенную информацию полученную в ходе выполнения работы.

Список использованной литературы содержит перечень источников, использованных при выполнении задания. Ссылки на источники необходимо указывать в тексте оформленного материала.

Приложения включают в себя вспомогательную информацию (ауди/видео, листинги, аббревиатура и пр.).

Тематика курсовых и практических/лабораторных работ соответствует основным разделам программ «Системное моделирование», «Архитектура ЭВМ и систем» и «Дискретная оптимизация» и является одним из основных направлений учебно-исследовательской и научно-технической деятельности Высшей школы Киберфизических систем и управления. Теоретическая часть, для данного рода занятий и упражнений, базируется на выше указанных программах курсов и существенно зависит от индивидуальных знаний студентов по курсу «Теория вероятности».

Выбор тематики работ основан на необходимости поднятия общего уровня студентов, как в теоретическом, так и в практическом плане, для решения задач частного и глобального масштаба, которые и должны решать специалисты – «СИСТЕМНЫЕ АНАЛЕТИКИ».

Выдаваемые задания, ориентированы на их решения с помощью средств вычислительной техники. С точки зрения получения результата – автор (кто хочет и может) имеет право найти альтернативу стандартно предлагаемым вариантам, но при этом, готовым защитить свою точку зрения.

Поощряется поиск студентами альтернативных заданий, связанных с научной деятельностью кафедры и привлечением организаций ориентированных на совместные проекты. Желаемая цель – индивидуальная, целевая и интенсивная подготовка специалиста.

Приложение № 2

Курсовой проект
«Оптимизация схемотехнических решений»

В рамках курсового проекта должны быть выполнены следующие действия:

- оптимизация с помощью Карт Карно;
- построение схемы в заданном базисе;
- моделирование работы счетчика на ЭВМ в тактовом и циклическом режиме (с ограниченным/устанавливаемом числе циклов).

1. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

2. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

3. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

4. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

5. Синтезировать схему КА, который при подаче на вход счетных импульсов формирует на выходах двоично-двенадцатиричный код. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

6. Синтезировать схему КА, который при подаче на вход счетных импульсов формирует на выходах двоично-двенадцатиричный код. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

7. Синтезировать схему КА, который при подаче на вход счетных импульсов формирует на выходах двоично-двенадцатиричный код. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

8. Синтезировать схему КА, который при подаче на вход счетных импульсов формирует на выходах двоично-двенадцатиричный код. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

9. Синтезировать схему КА в соответствии с заданием 78. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

10. Синтезировать схему КА в соответствии с заданием 78. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

11. Синтезировать схему КА в соответствии с заданием 78. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

12. Синтезировать схему КА в соответствии с заданием 78. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

13. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11, 16, 24, 25, 29. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

14. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11, 16, 24, 25, 29. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

15. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11, 16, 24, 25, 29. В качестве ЭА

выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

16. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11, 16, 24, 25, 29. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

17. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 3, 7, 8, 13. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

18. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 3, 7, 8, 13. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

19. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 3, 7, 8, 13. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

20. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 2, 3, 7, 8, 13. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

21. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 4, 5, 6, 9, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

22. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 4, 5, 6, 9, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

23. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды,

соответствующие числам 1, 4, 5, 6, 9, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

24. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 4, 5, 6, 9, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

25. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 8, 9, 10, 11, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

26. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 8, 9, 10, 11, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

27. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 8, 9, 10, 11, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

28. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 8, 9, 10, 11, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

29. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 3, 4, 5, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

30. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 3, 4, 5, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

31. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 3, 4, 5, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

32. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 3, 4, 5, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

33. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 2, 4, 6, 8, 10, 12, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

34. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 2, 4, 6, 8, 10, 12, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

35. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 2, 4, 6, 8, 10, 12, 14. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

36. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 2, 4, 6, 8, 10, 12, 14. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

37. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 4, 8, 10, 12. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

38. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 4, 8, 10, 12. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

39. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 4, 8, 10, 12. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

40. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 2, 4, 8, 10, 12. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

41. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 2, 4, 10, 12, 14, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

42. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 2, 4, 10, 12, 14, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

43. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 2, 4, 10, 12, 14, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

44. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 1, 2, 4, 10, 12, 14, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

45. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 9, 12, 14, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

46. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 9, 12, 14, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

47. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 9, 12, 13, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

48. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды,

соответствующие числам 0, 3, 6, 9, 12, 13, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

49. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 7, 8, 9, 13. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

50. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 7, 8, 9, 13. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

51. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 7, 8, 9, 13. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

52. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 3, 6, 7, 8, 9, 13. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

53. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 5, 6, 7, 8, 9, 10, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

54. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 5, 6, 7, 8, 9, 10, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2ИЛИ-НЕ». Провести тестирование.

55. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 5, 6, 7, 8, 9, 10, 15. В качестве ЭА выбрать RS – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

56. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 5, 6, 7, 8, 9, 10, 15. В качестве ЭА выбрать JK – триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование.

Приложение № 3

Пример выполнения курсового проекта

13. Синтезировать схему КА, который на входные счетные импульсы последовательно формирует на выходах двоичные коды, соответствующие числам 0, 1, 4, 5, 6, 9, 11, 16, 24, 25, 29. В качестве ЭА выбрать JK-триггеры. КС реализуется в базисе «2И-НЕ». Провести тестирование

Выходные сигналы

	Выход				
	5	4	3	2	1
0	0	0	0	0	0
1	0	0	0	0	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
9	0	1	0	0	1
11	0	1	0	1	1
16	1	0	0	0	0
24	1	1	0	0	0
25	1	1	0	0	1
29	1	1	1	0	1

Таблица переходов JK-триггера

	JK-триггер							
q0|K	0	0	0	0	1	1	1	1
q1|J	0	0	1	1	0	0	1	1
Q(t)	0	1	0	1	0	1	0	1
Q(t+1)	0	1	1	1	0	0	1	0

Обозначения:

- q0, qk – вход триггера Kill
- q1, qj – вход триггера Jump

Таблица истинности Блока Возбуждения Элементарных Автоматов

ТИ БВЭА

	текущее				Следующее				комб. Схема							
	Q3	Q2	Q1	Q0	Q3	Q2	Q1	Q0	q3'	q3	q2'	q2	q1'	q1	q0'	q0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	1
2	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	1
3	0	0	1	1	0	1	0	0	0	0	1	0	0	1	1	1
4	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	1
5	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1
6	0	1	1	0	0	1	1	1	0	0	0	0	0	0	1	1
7	0	1	1	1	1	0	0	0	1	0	0	1	0	1	1	1
8	1	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1
9	1	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1

Обозначения:

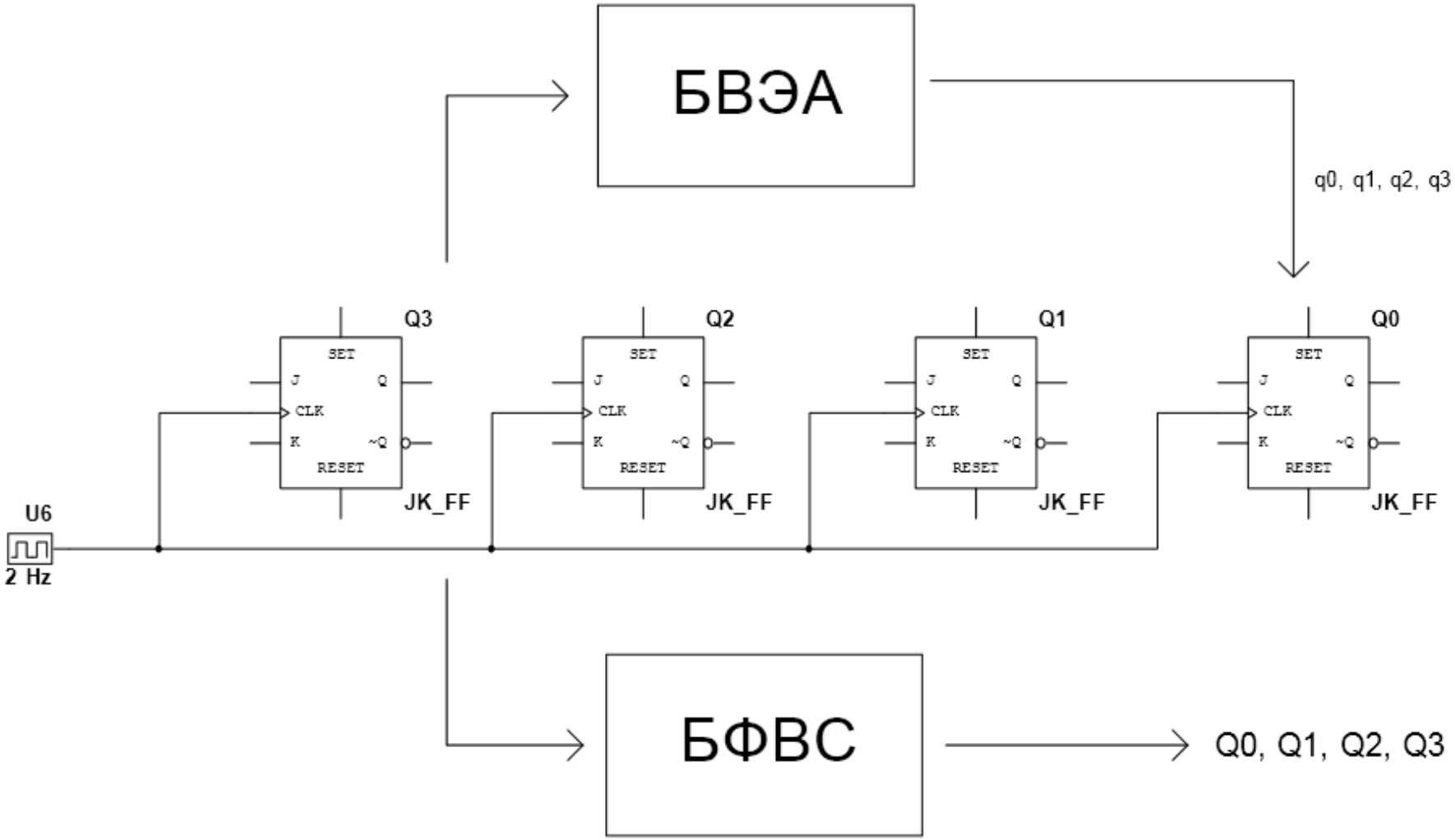
- q, q0 – вход триггера Kill
- q', q1 – вход триггера Jump

Таблица истинности Блока Формирования Выходных Сигналов

ТИ БФВС

Q3	Q2	Q1	Q0	O4	O3	O2	O1	O0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	1	0	0
0	0	1	1	0	0	1	0	1
0	1	0	0	0	0	1	1	0
0	1	0	1	0	1	0	0	1
0	1	1	0	0	1	0	1	1
0	1	1	1	1	0	0	0	0
1	0	0	0	1	1	0	0	0
1	0	0	1	1	1	0	0	1
1	0	1	0	1	1	1	0	1

Условная схема конечного автомата


БВЭА
q0, q1, q2, q3
Q3
Q2
Q1
Q0
SET
J
Q
CLK
K
~Q
RESET
JK_FF
U6
2 Hz
БФВС
Q0, Q1, Q2, Q3

Синтез комбинационной схемы возбуждения ЭА

q0

q0 = 1
q0' = 1

q1

q1

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	-	-	-	-
10	0	0	-	-

q1 = Q1 *
Q0

q1'

Q3Q2 \ Q1Q0	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	-	-	-	-
10	0	1	-	-

q1' = !Q1 * Q0

q2

q2

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	-	-	-	-
10	0		-	-

q2 = Q2 * Q1 * Q0 = Q2
* q1

q2'

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	-	-	-	-
10	0	0	-	-

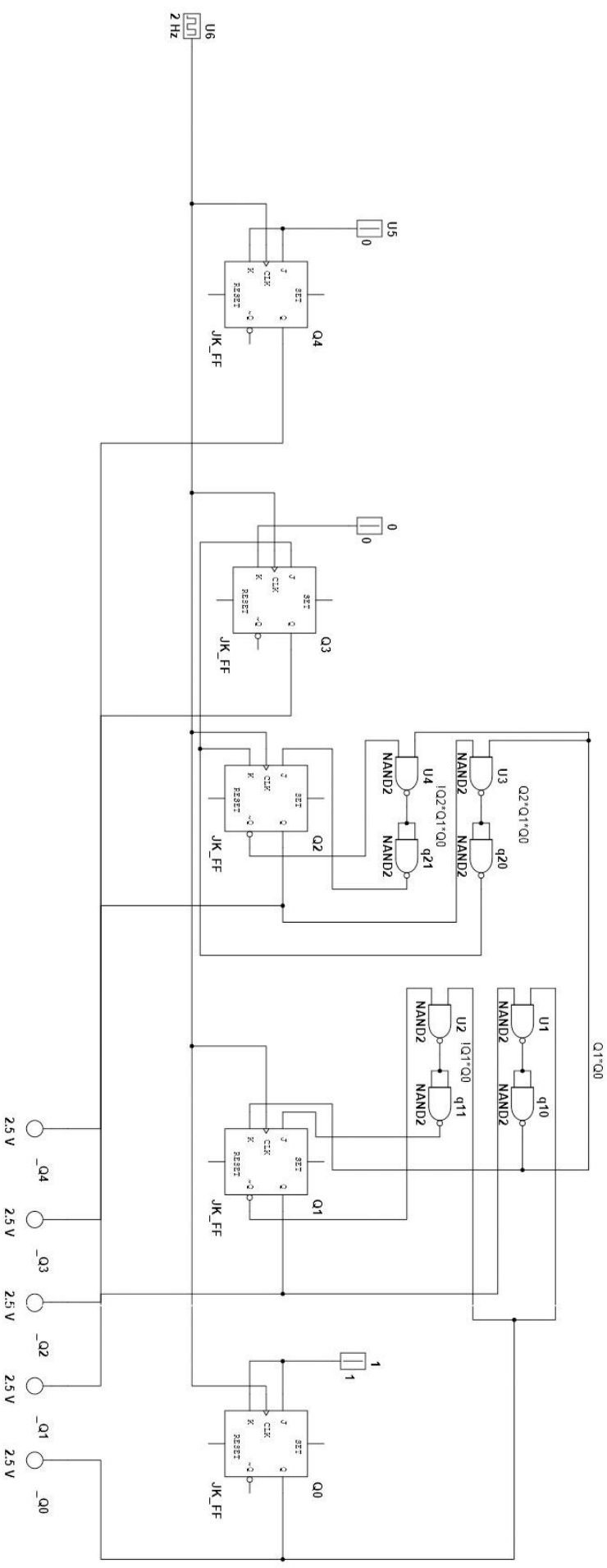
q2' = !Q2 * Q1 * Q0 = !Q2 * q1

q3

q3 = 0

q3' = q2

Комбинационная схема блока возбуждения конечных автоматов


U6
2 Hz
U5
0
JK_FF
Q4
0
0
JK_FF
Q3
JK_FF
Q2
U4
!Q2*Q1*Q0
NAND2
NAND2
q21
U3
Q2*Q1*Q0
NAND2
NAND2
q20
U2
!Q1*Q0
NAND2
NAND2
q11
U1
NAND2
NAND2
q10
Q1*Q0
JK_FF
Q1
1
1
JK_FF
Q0
2.5 V
_Q4
2.5 V
_Q3
2.5 V
_Q2
2.5 V
_Q1
2.5 V
_Q0

Синтез комбинационной схемы блока формирования выходных сигналов

O0

Q3Q2 \ Q1Q0	00	01	11	10
00	0	1	1	0
01	0	1	0	1
11	-	-	-	-
10	0	1	-	1

0	1	1	0
0	1	0	1
-	-	-	-
0	1	-	1

0	1	1	0
0	1	0	1
-	-	-	-
0	1	-	1

0	1	1	0
0	1	0	1
-	-	-	-
0	1	-	1

O0 = !Q2 * Q0 + !Q1 * Q0 + Q3 * Q1 + Q2 * Q1 *
!Q0 =
Q0 * !(Q2 * Q1) + Q1 * (Q3 + Q2 * !Q0)

$$\overline{Q0 \cdot \overline{Q2 \cdot \overline{Q1}} \cdot Q1 \cdot \overline{Q3} \cdot Q2 \cdot \overline{Q0}}$$

O1

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	-	-	-	-
10	0	0	-	0

O1 = Q2 * !Q0

O2

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	1	1
01	1	0	0	0
11	-	-	-	-
10	0	0	-	1

0	0	1	1
1	0	0	0
-	-	-	-
0	0	-	1

O2 = !Q2 * Q1 + Q2 * !Q1 *
!Q0

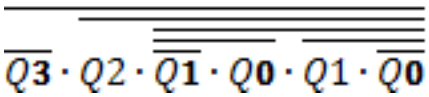
O3

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	-	-	-	-
10	1	1	-	1

0	0	0	0
0	1	0	1
-	-	-	-
1	1	-	1

0	0	0	0
0	1	0	1
-	-	-	-
1	1	-	1

O3 = Q3 + Q2 * !Q1 * Q0 + Q2 * Q1 * !Q0



O4

		Q1Q0			
		00	01	11	10
Q3Q2	00	0	0	0	0
	01	0	0	1	0
	11	-	-	-	-
	10	1	1	-	1

0	0	0	0
0	0	1	0
-	-	-	-
1	1	-	1

O4 = Q3 + Q2 * Q1 * Q0

O0 = Q0 * !(Q2 * Q1) + Q3 * Q1 + A

O1 = Q2 * !Q0

O2 = !Q2 * Q1 +B * !Q0

O3 = Q3 + B * Q0 +
A

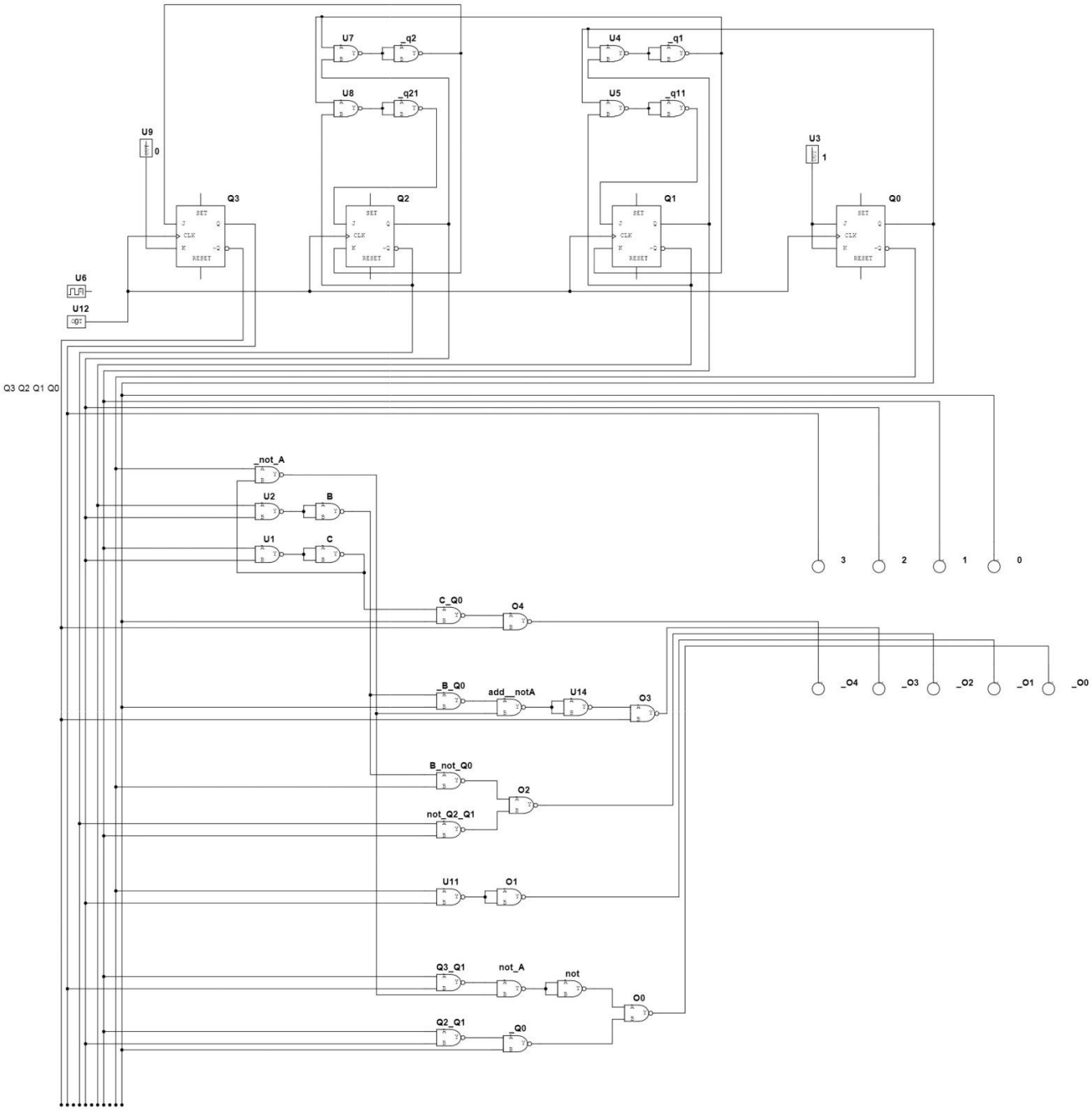
O4 = Q3 + C * Q0

A = C * !Q0

B = Q2 * !Q1

C = Q2 * Q1

Схема конечного автомата



Тестирование и заключение

Тестирование проведено с помощью симулятора электронных схем «Multisim 12.0». Работа синтезированного конечного автомата соответствует поставленным условиям.

Приложение № 4

Курсовой проект

«Оптимизация информационных потоков»

Если закон распределения случайных величин не указан, то он равномерный.

Задание №1. Вычислительная сеть состоит из четырех ЭВМ и сервера, который контролирует общую оперативную память. Задания, поступающие на обработку через интервалы времени 5+-2 мин. (закон распределения – экспоненциальный), занимают объем оперативной памяти размером в страницу. После трансляции первым процессором в течении 5+-1 мин. их объем увеличивается до двух страниц и они поступают в оперативную память. Редактирование во втором процессоре занимает 2,5+-0,5 мин. на страницу, объем возрастает до трех страниц. Аналогичные процедуры происходят в третьем процессоре. Отредактированные задания через оперативную память поступают на четвертую ЭВМ для решения, требующего 1,5+-0,6 мин. на страницу, и покидают сеть, минуя сервер.

Смоделировать работу вычислительной системы в течение 100 часов. Определить характеристики функционирования общей оперативной памяти по всем видам заданий.

Задание №2. На вычислительный центр поступают пакеты информации (идентичные) в среднем через 50 минут. Первичная обработка пакетов производится на двух ЭВМ. Если ЭВМ заняты, то пакеты отправляются в накопитель. Первая ЭВМ обрабатывает пакет в среднем 30 минут и выявляет до 4% брака, вторая – соответственно 70 минут и 8% брака. При повторной обработке, все выбракованные пакеты возвращаются на вторую ЭВМ. Пакеты, выбракованные дважды, считаются отходами. Вторичную обработку проводят также еще две ЭВМ в среднем 90 минут каждая. Причем первая ЭВМ обрабатывает имеющиеся в накопители пакеты, а вторая подключается при образовании в накопителе задела более трех пакетов. Все интервалы времени распределены по экспоненциальному закону.

Смоделировать обработку на вычислительном центре 600 пакетов. Определить загрузку второй ЭВМ на вторичной обработке и вероятность появления отходов. Определить возможность снижения задела в накопителе и повышения загрузки второй ЭВМ при вторичной обработке.

Задание №3. На центральную ЭВМ через интервалы времени, распределенные экспоненциально со средним значением 10 минут, поступают файлы, каждый из которых состоит из трех подпрограмм.

Половина всех поступающих подпрограмм должна пройти предварительную обработку в течение 6 минут. На компоновку подаются обработанная и необработанная подпрограммы. Процесс компоновки занимает 6 минут. Затем проводится тестирование, продолжающееся в среднем 8 минут (время выполнения его распределено экспоненциально). В результате компоновки возможно выявление 5% ошибок (брака). В этом случае процесс тестирования не проводится, а подпрограммы отправляются на предварительную обработку.

Смоделировать работу ЭВМ в течение 20 часов. Определить возможные места появления очередей и их вероятностно-временные характеристики. Выявить причины их возникновения, предложить меры по их устранению и смоделировать скорректированную схему.

Задание №4. Система обработки информации содержит мультиплексный канал и три ЭВМ. Сигналы от датчиков поступают на вход канала через интервалы времени 10+-5 мкс. В канале они буферизуются и предварительно обрабатываются в течение 10+-4 мкс (время выполнения распределено экспоненциально). Затем они поступают на обработку в ту ЭВМ, где имеется наименьшая по длине входная очередь. Емкости входных накопителей во всех ЭВМ рассчитаны на хранение величин 10 сигналов. Время обработки сигнала в любой ЭВМ равно 35 мкс.

Смоделировать процесс обработки 600 сигналов, поступающих с датчиков. Определить средние времена задержки сигналов в канале и ЭВМ и вероятности переполнения входных накопителей. Обеспечить ускорение обработки сигналов в ЭВМ до 30 мкс при достижении суммарной очереди сигналов значения 30 единиц.

Задание №5. Система передачи данных обеспечивает передачу пакетов данных из пункта А в пункт С через транзитный буфер пункта В.В пункт А пакеты поступают через 10+-5 мс (закон распределения экспоненциальный). Здесь они буферизуются в накопителе емкостью 25 пакетов и передаются по любой из двух линий АВ1 – за время 19 мс или АВ2 – за время 20+-5 мс. В пункте В они снова буферизуются в накопителе емкостью 22 пакета и далее передаются по линиям ВС1 (за 25+-3 мс) и ВС2 (за 22 мс). Причем пакеты из АВ1 поступают в ВС1, а из АВ2 – в ВС2. Чтобы не было переполнения накопителя, в пункте В вводится пороговое значение его емкости – 20 пакетов. При достижении очередью порогового значения происходит подключение резервной аппаратуры и время передачи снижается для линий ВС1 и ВС2 до 15 мс.

Смоделировать прохождение через систему передачи данных 600 пакетов. Определить вероятность подключения резервной аппаратуры и

характеристики очереди пакетов в пункте В. В случае возможности его переполнения определить необходимое для нормальной работы пороговое значение емкости накопителя.

Задание №6. На сервер через случайные интервалы времени по два пакета данных в среднем через каждые 30 минут. Первичная обработка для двух пакетов одновременно занимает около 30 минут. Если в момент прихода пакетов предыдущие не были обработаны, то они не принимаются. Пакеты после первичной обработки, получившие отказ, поступают в промежуточный накопитель. Из накопителя пакеты, прошедшие первичную обработку, поступают попарно на вторичную, которая выполняется в среднем за 30 минут, а не прошедшие первичную поступают на полную, которая занимает 100 минут для одного пакета. Все величины заданные средними значениями, распределены экспоненциально.

Смоделировать работу сервера в течение 150 часов. Определить вероятность отказа в первичной обработке и загрузку накопителя пакетами, нуждающимися в полной обработке. Определить параметры и ввести в систему накопитель, обеспечивающий безотказное обслуживание поступающих пакетов.

Задание №7. Магистраль передачи данных состоит из двух каналов (основного и резервного) и общего накопителя. При нормальной работе сообщения передаются по основному каналу за 8+-2 секунды (закон распределения экспоненциальный). В основном канале происходят сбои через интервалы времени 250+-40 секунд. Если сбой происходит во время передачи, то за 2 секунды запускается запасной канал, который передает прерванное сообщение с самого начала. Восстановление основного канала занимает 31+-4 секунды. После восстановления резервный канал выключается и основной канал продолжает работу с очередного сообщения. Сообщения поступают через 8+-5 секунд и остаются в накопителе до окончания передачи. В случае сбоя передаваемое сообщение передается повторно по запасному каналу.

Смоделировать работу магистрали передачи данных в течение 2 часов. Определить загрузку запасного канала, частоту отказов канала и число прерванных сообщений. Определить функцию распределения времени передачи сообщений по магистрали.

Задание №8. На участке термической обработки выполняются цементация и закаливание шестерен, поступающих через 11+-5 минут (закон распределения экспоненциальный). Цементация занимает 11+-6 минут, а закаливание – 11+-5 минут. Качество определяется суммарным

временем обработки. Шестерни с временем обработки больше 25 минут покидают участок, с временем обработки от 20 до 25 минут передаются на повторную закалку и при времени обработки менее 20 минут должны пройти повторную полную обработку. Детали с суммарным временем обработки меньше 20 минут считаются вторым сортом.

Смоделировать процесс обработки на участке 450 шестерен. Определить функцию распределения времени обработки и вероятности повторения полной и частичной обработки. При выходе продукции без повторной обработки менее 90% обеспечить на участке мероприятия, дающие гарантированный выход продукции первого сорта 90%.

Задание №9. В студенческом машинном зале расположены две ЭВМ и одно устройство подготовки данных (УПД). Студенты приходят с интервалом в 9+-3 минуты (закон распределения экспоненциальный) и треть из них хочет использовать УПД и ЭВМ, а остальные только ЭВМ. Допустимая очередь в машинном зале составляет четыре человека, включая работающего на КПД. Работа на УПД занимает 7+-1 минуту, а на ЭВМ – 17 минут. Кроме того, 20% работавших на ЭВМ возвращается для повторного использования УПД и ЭВМ.

Смоделировать работу машинного зала в течение 60 часов. Определить загрузку УПД, ЭВМ и вероятности отказа в обслуживании вследствие переполнения очереди. Определить соотношение желающих работать на ЭВМ и на УПД в очереди.

Задание №10. На вычислительном центре принимают три класса заданий A, B и C. Исходя из наличия оперативной памяти ЭВМ задания классов A и B могут решаться одновременно, а задания класса C монополизирует ЭВМ. Задания класса A поступают через 19+-4 минуты (закон распределения экспоненциальный), класса B – через 20+-9 минут и класса C – через 30+-10 минут и требуют для выполнения: класс A – 20+-5 минут, класс B – 22+-3 минуты, класс C 29+-4 минуты. Задачи класса C загружаются в ЭВМ, если она полностью свободна. Задачи классов A и B могут дозагружаться к решаемой задаче.

Смоделировать работу ЭВМ за 90 часов. Определить ее загрузку.

Задание №11. В системе передачи данных осуществляется обмен пакетами данных между пунктами A и B по дуплексному каналу связи. Пакеты поступают в пункты системы от абонентов с интервалами времени между ними 11+-3 мс (закон распределения экспоненциальный). Передача пакета занимает 10 мс. В пунктах имеются буферные регистры, которые могут хранить два пакета (включая передаваемый). В случае прихода пакета в момент занятости регистров пунктам системы предоставляется

выход на спутниковую полудуплексную линию связи, которая осуществляет передачу пакетов данных за 10+-5 мс. При занятости спутниковой линии пакет получает отказ.

Смоделировать обмен информацией в системе передачи данных в течение 2 минут. Определить частоту вызовов спутниковой линии и ее загрузку. В случае возможности отказов определить необходимый для безотказной работы системы объем буферных регистров.

Задание №12. Специализированная вычислительная система состоит из трех процессоров и общей оперативной памяти. Задания, поступающие на обработку через интервалы времени 6+-2 минуты (закон распределения экспоненциальный), занимают объем оперативной памяти размером в страницу. После трансляции первым процессором в течение 5+-1 минуты их объем увеличивается до двух страниц и они поступают в оперативную память. Затем после редактирования во втором процессоре, которое занимает 2,5+-0,5 минут на страницу, объем возрастает до трех страниц. Отредактированные задания через оперативную память поступают в третий процессор на решение, требующее 1,5+-0,4 минуты на страницу, и покидают систему, минуя оперативную память.

Смоделировать работу вычислительной системы в течение 60 часов. Определить характеристики занятия оперативной памяти по всем трем видам заданий.

Задание №13. В ЭВМ проводится компоновка подпрограмм двух типов. Каждые 6+-1 минуту (закон распределения экспоненциальный) поступает по 5 подпрограмм первого типа и каждые 19+-6 минут поступает по 20 подпрограмм второго типа. Перед компоновщиком установлены буферные зоны, вмещающие в себя по 10 подпрограмм каждого типа. Компоновка начинается только при наличии подпрограмм обоих типов в требуемом количестве и длится 10 минут. При нехватке подпрограмм буфер остается пустой.

Смоделировать работу компоновщика в течение 9 часов. Определить вероятность пропуска буфера, средние и максимальные очереди по каждой из подпрограмм. Определить экономическую целесообразность перехода на буфера по 20 подпрограмм с временем компоновки 20 минут.

Задание №14. Транспортный цех объединения обслуживает три филиала A, B, и C. Грузовики перевозят изделия из A в B и из B в C, возвращаясь затем в A без груза. Погрузка в A занимает 20 минут, переезд из A в B длится 30 минут, разгрузка и погрузка в B – 40 минут, переезд в C – 30 минут, разгрузка в C – 20 минут и переезд в A – 20 минут. Если к

моменту погрузки в А и В отсутствуют изделия, грузовики уходят дальше по маршруту. Изделия в А выпускаются партиями по 1000 штук, через 20+-4 минуты (закон распределения экспоненциальный), в В такими же партиями через 19+-5 минут. На линии работает 8 грузовиков, каждый перевозит 1000 изделий. В начальный момент все грузовики находятся в А.

Смоделировать работу транспортного цеха объединения в течение 1500 часов. Определить частоту пустых перегонов грузовиков между А и В, В и С и сравнить с характеристиками, полученными при равномерном начальном распределении грузовиков между филиалами и операциями.

Задание №16. Система автоматизации проектирования состоит из ЭВМ и трех терминалов. Каждый проектировщик формирует задание на расчет в интерактивном режиме. Набор строки задания занимает 11+-5 секунд (закон распределения экспоненциальный). Получение ответа на строку требует 3 секунды работы ЭВМ и 5 секунд работы терминала. После набора 10 строк задание считается сформированным и поступает на решение, при этом в течение 10+-2 секунд ЭВМ прекращает выработку ответов на вводимые строки. Анализ результата занимает у проектировщика 29 секунд, после чего цикл повторяется.

Смоделировать работу системы в течение 7 часов. Определить вероятность простоя проектировщика из-за занятости ЭВМ и коэффициент загрузки ЭВМ.

Задание №17. К ЭВМ подключено четыре терминала, с которых осуществляется решение задач. По команде с терминала выполняются операции редактирования, транслирования, компоновки и решения. Причем, если хоть один терминал выполняет компоновку, остальные вынуждены простаивать из-за нехватки оперативной памяти. Если два терминала выдают требования на решение, то оставшиеся два простаивают, и если работают три терминала, выдающих задания на трансляцию, то оставшийся терминал блокируется. Интенсивности поступления задач различных типов равны. Задачи одного типа от одного терминала поступают через экспоненциально распределенные интервалы времени со средним значением 155 секунд. Выполнение любой операции длится 10 секунд.

Смоделировать работу ЭВМ в течение 5 часов. Определить загрузку процессора, вероятности простоя терминалов и частоту одновременного выполнения трансляции с трех терминалов.

Задание №18. В системе передачи цифровой информации передается речь в цифровом виде. Речевые пакеты передаются по двум транзитным каналам, концентрируясь в накопителях перед каждым

каналом. Время передачи пакета по каналу составляет 6 мс. Пакеты поступают через 6+-3 мс (закон распределения экспоненциальный). Пакеты, передававшиеся более 10 мс, на выходе системы уничтожаются, так как их появление в декодере значительно снизит качество передаваемой речи. Уничтожение более 30% пакетов недопустимо. При достижении такого уровня система за счет ресурсов ускоряет передачу до 4 мс на канал. При снижении уровня до приемлемого происходит отключение ресурсов.

Смоделировать 10 секунд работы системы. Определить частоту уничтожения пакетов и частоту подключения ресурсов.

Задание №18. ЭВМ обслуживает три терминала по круговому циклическому алгоритму, предоставляя каждому терминалу 35 секунд. Если в течение этого времени задание обрабатывается, то обслуживание завершается; если нет, то остаток задачи становится в специальную очередь, которая использует свободные циклы терминалов, то есть задача обслуживается, если на каком-либо терминале нет заявок. Заявки на терминалы поступают через 30+-5 секунд и имеют длину 300+-50 знаков (закон распределения экспоненциальный). Скорость обработки заданий ЭВМ равна 10 знаков в секунду.

Смоделировать 5 часов работы ЭВМ. Определить загрузку ЭВМ, параметры очереди неоконченных заданий. Определить величину цикла терминала, при которой все заявки будут обслужены без специальной очереди.

Задание №19. В узел коммутации сообщений, состоящий из входного буфера, процессора, двух исходящих буферов и двух выходных линий, поступают сообщения с двух направлений. Сообщения с одного направления поступают во входной буфер, обрабатываются в процессоре, буферизуются в выходном буфере первой линии. Применяемый метод контроля потоков требует одновременного присутствия в системе не более трех сообщений на каждом направлении. Сообщения поступают через интервалы 15+-6 мс (закон распределения экспоненциальный). Время обработки в процессоре равно 7 мс на сообщение, время передачи по выходной линии равно 15+-5 мс. Если сообщение поступает при наличии трех сообщений в данном направлении, то оно получает отказ.

Смоделировать работу узла коммутации в течение 10 мс. Определить загрузки устройств и вероятность отказа в обслуживании из-за переполнения буфера направления. Определить изменения в функции распределения времени передачи при снятии ограничений, вносимых методом контроля потоков.

Задание №20. Распределенный банк данных системы сбора информации организован на базе ЭВМ, соединенных дуплексным каналом связи. Поступающий запрос обрабатывается на первой ЭВМ и с вероятностью 50% необходимая информация обнаруживается на месте. В противном случае необходима посылка запроса на вторую ЭВМ. Запросы поступают через 10+-3 секунды (закон распределения экспоненциальный), первичная обработка занимает 2 секунды, выдача ответа требует 17+-3 секунды, передача по каналу связи занимает 3 секунды. Временные характеристики второй ЭВМ аналогичны первой.

Смоделировать прохождение 500 запросов. Определить необходимую емкость накопителей перед ЭВМ, обеспечивающую безотказную работу системы, и функцию распределения времени обслуживания заявки.

Задание №21. Из литейного цеха на участок обработки и сборки поступают заготовки через 21+-6 минут (закон распределения экспоненциальный). Треть из них обрабатывается в течение 65 минут и поступает на комплектацию. Две трети заготовок обрабатывается за 31 минуту перед комплектацией, которая требует наличия одной детали первого типа и двух деталей второго. После этого все три детали подаются на сборку, которая занимает 60+-3 минуты для первой детали и 60+-7 минут для двух других, причем они участвуют в сборке одновременно. При наличии на выходе одновременно всех трех деталей изделие покидает участок.

Смоделировать работу участка в течение 150 часов. Определить места образования и характеристики возможных очередей.

Задание №22. В машинный зал с интервалом времени 10+-4 минуты (закон распределения экспоненциальный) заходят пользователи, желающие произвести расчеты на ЭВМ. В зале имеется одна ЭВМ, работающая в однопрограммном режиме. Время, необходимое для решения задач, включая вывод результатов на печать, характеризуется интервалом 15+-4 минуты. Третья часть пользователей после решения своей задачи производит вывод текста программы на внешний носитель (продолжительность – 3+-1,5 минут). В машинном зале не допускается, чтобы более семи пользователей ожидали своей очереди на доступ к ЭВМ.

Смоделировать процесс обслуживания 200 пользователей. Подсчитать число пользователей, не нашедших свободного места в очереди. Определить среднее число пользователей в очереди, а также коэффициенты загрузки ЭВМ и внешнего носителя.

Задание №23. Вычислительная система включает в себя три ЭВМ. В систему в среднем через 29 секунд поступают задания, которые попадают в очередь на обработку к первой ЭВМ, где они обрабатываются около т30 секунд (закон распределения экспоненциальный). После этого задание поступает одновременно во вторую и третью ЭВМ. Вторая ЭВМ может обработать задание за 14,6+-4 секунды, а третья – за 15+-1 секунду. Окончание обработки задания на любой ЭВМ означает снятие ее решения с той и другой машины. В свободное время вторая и третья ЭВМ заняты обработкой фоновых задач.

Смоделировать 4 часов работы системы. Определить необходимую емкость накопителей перед всеми ЭВМ, коэффициенты загрузки ЭВМ и функцию распределения времени обслуживания заданий. Определить производительность второй и третьей ЭВМ на решении фоновых задач при условии, что одна фоновая задача решается 3 минуты.

Задание №24. Для обеспечения надежности АСУ ТП вней используются две ЭВМ. Первая ЭВМ выполняет обработку данных о технологическом процессе и выработку управляющих сигналов, а вторая находится в «горячем резерве». Данные в ЭВМ поступают через 11+-2 секунды (закон распределения экспоненциальный), Обрабатываются в течение 3 секунд, затем посылается управляющий сигнал, поддерживающий заданный темп процесса. Если к моменту посылки следующего набора данных не получен управляющий сигнал, то интенсивность интенсивность выполнения технологического процесса уменьшается вдвое и данные посылаются через 20+-4 секунды. Основная ЭВМ посылает резервной ЭВМ сигнал о работоспособности. Отсутствие сигнала означает необходимость подключения резервной ЭВМ вместо основной. Характеристики обеих ЭВМ одинаковы. Подключение резервной ЭВМ занимает 6 секунд, после чего она заменяет основную до восстановления, а процесс возвращается к нормальному темпу. Отказы ЭВМ происходят через 300+-30 секунд. Восстановление занимает 100 секунд. Резервная система абсолютно надежна.

Смоделировать 2 часа работы системы. Определить среднее время нахождения технологического процесса в заторможенном состоянии и среднее число пропущенных из-за отказов данных.

Задание №25. На вычислительный центр через 250+-50 минут (закон распределения экспоненциальный) поступают задания длиной 550+-150 байт. Скорость ввода, вывода и обработки заданий – 100 байт/мин. Задания проходят последовательно ввод, обработку вывод, буферируясь перед каждой операцией. После вывода 5% заданий оказываются выполненными неправильно вследствие сбоев и возвращаются на ввод.

Для ускорения обработки задания в очередях располагаются по возрастанию их длины, то есть короткие сообщения обслуживаются в первую очередь. Задания, выполненные неверно, вовращаются на ввод и во всех очередях обслуживаются первыми.

Смоделировать работу вычислительного центра в течение 45часов. Определить необходимую емкость буферов и функцию распределения времени обслуживания заданий.

Задание №26. Детали, необходимые для работы цеха, находятся на цеховом и центральном складах. На цеховом складе хранится 20 комплектов деталей, потребность в которых возникает через 60+-10 минут (закон распределения экспоненциальный) и составляет один комплект. В случае снижения запасов до трех комплектов формируется в течение 60 минут заявка на пополнение запасов цехового склада до полного объема в 20 комплектов, которая посылается на центральный склад, где в течение 60+-20 минут происходит комплектование и за 65+-5 минут осуществляется доставка деталей в цех.

Смоделировать работу цеха в течение 500 часов. Определить вероятность простоя цеха из-за отсутствия деталей и среднюю загрузку цехового склада. Определить момент пополнения запаса цехового склада, при котором вероятность простоя цеха будет равна 0.

Приложение № 5

Расчетное задание

Выполняются задания для методов повторяющихся членов и последовательного наращивания. Четные номера – базис «2ИЛИ-НЕ», нечетные номера – базис «2И-НЕ». Тестирование – для строчки таблицы истинности: 0 0 1 0.

1.

$F_m = \sum$ 00-1 1--0 0101 1110
$F_n = \sum$ 1110 --10 -001 0001

2

$F_m = \sum$ 1-11 1001 0101 1110
$F_n = \sum$ 1--0 --11 -011 11--

3.

$F_m = \sum$ 11-1 1110 -101 0010
$F_n = \sum$ 1000 -110 -011 0-01

4.

$F_m = \sum$ 00-1 0--0 1101 0110
$F_n = \sum$ 1010 --11 -101 0-01

5.

$F_m = \sum$ 0001 1-10 0-01 0110
$F_n = \sum$ 0110 -110 0001 0101

6.

$F_m = \sum$ 11-1 10-0 1101 1010
$F_n = \sum$ 1100 --11 -011 1101

7.

$F_m = \sum$ 00-1 1-10 1101 1010
$F_n = \sum$ 0110 0-10 1001 1101

8.

$F_m = \sum$ 0001 1110 0101 1110
$F_n = \sum$ 1110 --10 1001 1001

9.

$F_m = \sum$ 10-1 1--- 01-1 1-10
$F_n = \sum$ 1-10 1-10 -00- 0-01

10.

$F_m = \sum$ 01-1 10-0 0101 0010
$F_n = \sum$ 0110 1-10 --01 0011

11.

$F_m = \sum$ 0--1 1000 1101 1-10
$F_n = \sum$ 1-10 -110 0001 1101

12.

$F_m = \sum$ 10-1 1-00 0-01 1110
$F_n = \sum$ 1010 1-10 -0-1 0-01

13.

$F_m = \sum$ -0-1 1010 0101 1--0
$F_n = \sum$ 11-0 1-10 1001 --01

14.

$F_m = \sum$ -0-1 1110 ---1 0110
$F_n = \sum$ 1010 --11 -000 1001

15.

$F_m = \sum$ 00-1 1--0 ---1 0010
$F_n = \sum$ 111- -110 -000 0101

16.

$F_m = \sum$ 10-1 11-0 1--1 1010
$F_n = \sum$ 101- -111 -001 1101

17.

$F_m = \sum$ -0-1 1-10 -1-1 0110
$F_n = \sum$ 110- -100 -010 0101

18.

$F_m = \sum$ 01-1 0--0 1--1 1010
$F_n = \sum$ 110- -100 -001 1101

19.

$F_m = \sum$ 0011 11-0 -1-1 1010
$F_n = \sum$ 101- 0110 -001 0001

20.

$F_m = \sum$ 0101 1010 -0-1 0-10
$F_n = \sum$ 110- -11- -00- 01-1

21.

$F_m = \sum$ 00-1 1-00 -1-1 1010
$F_n = \sum$ 101- 0110 -001 1101

22.

$F_m = \sum$ 10-1 1-00 00-1 0-10
$F_n = \sum$ 001- -11- -010 0-01

23.

$F_m = \sum$ 01-1 1000 -001 0110
$F_n = \sum$ 1110 -100 -001 0111

24.

$F_m = \sum$ 0011 11-0 1--1 1010
$F_n = \sum$ 1110 -100 -001 1101

25.

$F_m = \sum$ 0011 1110 0001 0010
$F_n = \sum$ 1110 1110 1000 0101

Приложение № 6

Пример дискретной оптимизации в метрологии

Совокупные и совместные измерения

При совокупных и совместных измерениях значения измеряемых величин $A_1,....,A_m$ определяют на основании совокупности уравнений:

$$f_1(A_1,....,A_m,\ a_{11},......,a_{1n}) = 0,$$
$$f_2(A_1,....,A_m,\ a_{21},......,a_{2n}) = 0,$$
$$\text{(П.6.1)}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \ \ldots\ldots\ldots$$
$$f_k(A_1,....,A_m,\ a_{k1},......,a_{kn}) = 0,$$

где $a_{11}, a_{21},......,a_{kn}$ — величины, измеряемые прямым методом.

Совокупные измерения состоят из ряда прямых измерений однородных величин, причем при переходе от одного ряда измерений к другому меняются сочетания измеряемых величин. Совокупные измерения применяются, например, при определении действительных значений гирь из одного набора. Для одной гири определяют ее действительное значение путем сравнения с образцовой гирей. Действительные значения остальных гирь находят в результате решения системы уравнений (П.6.1). Уравнения (П.6.1) построены на основании сравнения в разных сочетаниях всех гирь, входящих в набор.

Точность результатов совокупных измерений зависит от числа уравнений: чем больше число независимых уравнений, связывающих измеряемые величины, тем точнее результаты измерений. Поэтому измерения выполняют так, чтобы число уравнений (П.6.1) превышало число измеряемых величин. Однако следует заметить, что число получаемых уравнений может быть ограничено возможным числом получения независимых уравнений (числом всевозможных сочетаний ограниченного числа измеряемых величин). Точность совокупных измерений можно повысить сглаживанием случайных погрешностей, вызываемых индивидуальными особенностями измеряемой величины. Для этого комбинации между ними составляют так, чтобы веса искомых величин были одинаковы, тогда все измеряемые величины одинаковое количество раз будут участвовать в сличениях. При прочих равных условиях предпочтение следует отдать комбинации сличений, в которой

образцовая мера подвергается меньшей нагрузке (естественно, что чем меньше используется образцовая мера, тем лучше).

Пример. Три меры A_1, A_2, A_3 сличают с образцовой мерой N. Сличение может быть выполнено при следующих комбинациях.

1. Каждая мера по два раза сличается с образцовой мерой:

$A_1 - N = X_1$, $A_2 - N = X_3$, $A_3 - N = X_5$,

$A_1 - N = X_2$, $A_2 - N = X_4$, $A_3 - N = X_6$.

2. Первая мера сличается два раза с образцовой мерой, вторая — два раза с первой и третья — два раза со второй:

$A_1 - N = X_1$, $A_2 - A_1 = X_3$, $A_3 - A_2 = X_5$,

$A_2 - N = X_2$, $A_2 - A_1 = X_4$, $A_3 - A_2 = X_6$.

3. Каждая мера сличается по одному разу с образцовой, затем они сличаются друг с другом:

$A_1 - N = X_1$, $A_3 - N = X_3$, $A_3 - A_1 = X_5$,

$A_2 - N = X_2$, $A_2 - A_1 = X_4$, $A_3 - A_2 = X_6$.

Рассмотрим, какая из трех комбинаций сличений дает наилучшие результаты. Для этого сравним веса неизвестных, определяемые путем решения уравнений каждой из комбинаций методом наименьших квадратов.

Веса первой комбинации: $p_{A1} = 2$, $p_{A2} = 2$, $p_{A3} = 2$, второй комбинации: $p_{A1} = 2$, $p_{A2} = 1$, $p_{A3} = 2/3$ и третьей комбинации: $p_{A1} = 2$, $p_{A2} = 2$, $p_{A3} = 2$.

Из расчетов следует, что вторая комбинация сличений невыгодна, так как в ней третья мера имеет вес в три раза меньший, чем первая. В первой и второй комбинациях веса неизвестных одинаковы, но третья комбинация лучше, так как в ней меньше используют образцовую меру. Значит, необходимо выбрать третью комбинацию.

Совместные измерения широко применяются для определения функциональных зависимостей между физическими величинами. Для этого сначала находят приближенный вид зависимости, например, путем построения графика на основании полученных при измерении экспериментальных данных. Затем эту экспериментальную зависимость аппроксимируют приближенным теоретическим уравнением. Это уравнение уточняют, определяя его коэффициенты, путем получения ряда прямых измерений и изменением условий измерений при переходе от одного ряда к другому. Полученную систему уравнений решают методом наименьших квадратов. Внедрение ЭВМ для обработки результатов

измерений позволяет использовать метод наименьших квадратов не только для точных измерений, но и для технических измерений на производстве.

СПИСОК ЛИТЕРАТУРЫ

1. Баранов В.Е., Сотсков Ю.В. Архитектура ЭВМ и систем: учеб. пособие / В. Е. Баранов, Ю. В. Сотсков. – СПб.: Изд-во Политехн. ун-та. 2007. – 120 с.
2. Баранов В.Е., Сотсков Ю.В., Шмаков В.Э. Системное моделирование. Часть 1. Анализ и синтез сигналов.: учеб. пособие/В. Е. Баранов, Ю. В. Сотсков. – СПб.: Изд-во Политехн. ун-та. 2004. – 87 с.
3. Баранов В.Е. : Системное моделирование. Часть 2. Моделирование систем и планирование эксперимента.: учеб. пособие / В. Е. Баранов, Ю. В. Сотсков. – СПб.: Изд-во Политехн. ун-та. 2011. – 144 с.
4. Букреев И. Н. и др. Микроэлектронные схемы цифровых устройств. Изд. 2-е и доп. М., «Сов. Радио», 1975. 368с.
5. Вуд А. Микропроцессоры в вопросах и ответах. М.: энергоатомиздат, 1985. 184с.
6. Каган Б. М. Электронные вычислительные машины и системы. М.: Энергоатомиздат, 1985. 522с.
7. Карлащук В.И. Электронная лаборатория на IBM PC. 2-е изд. – Москва, 2001. – 726 с.;ил.
8. Компьютеры: Справочное руководство. В 3-х т. Т.1. Под ред. Г. Хелмса М.: Мир, 1986. 416с.
9. Сергеев Н. П., Вашкевич Н. П. Основы вычислительной техники. М.: Высш. шк., 1988. 311с.
10. Танебаум Э. Архитектура компьютера. 4-е изд. – СПб; Питер, 2003. – 704 с.: ил.
11. Черноруцкий И.Г. Методы оптимизации. Компьютерные технологии. – СПб.: БХВ-Петербург, 2011. – 384 с.