ЛЕКЦИИ ПО АВМиС

Оглавление

Лекция №1

1.1. Ц.В.М.

1.1.1. Классификация ЭВМ

По принципу действия вычислительные машины делятся на:

- аналоговые (АВМ)
- цифровые (ЦВМ)

1.1.2. Поколения ЦВМ

Критерии деления ЦВМ на поколения:

- Архитектура
- Элементная база
- Уровень развитости ПО

I. Первое поколение (1950 - 1960)

Архитектура: Машина фон Неймана

Элементная база: электронная база

Уровень развитости ПО: ПО отсутствует

II. Второе поколение (1960 - 1970)

Архитектура: Машина фон Неймана

Элементная база: полупроводники

Уровень развитости ПО: БСП (библиотеки стандартных программ + первые трансляторы с первых алгоритмических языков)

III. Третье поколение (1970 - 1980)

Архитектура: Машина фон Неймана + каналы ввода-вывода

Элементная база: ИССИ (интегральные схемы средней интеграции)

Уровень развитости ПО: БСП + трансляторы с любых алгоритмических языков + первые операционные системы

IV. Четвертое поколение (1980 - 1990)

Архитектура: многопроцессорные и многомашинные комплексы

Элементная база: БИСы (большие интегральные схемы)

Уровень развитости ПО: БСП + ОС + трансляторы с любых алгоритмических языков

V. Пятое поколение (1980 – до наших дней)

Разработка и модификация персональных ЭВМ

VI. Нейрокомпьютер

Устройство переработки информации на основе принципов работы естественных нейронных систем.

Теоретические разработки.

1.1.3. Архитектура вычислительных машин

1. I и II поколения (рис. 1.1)

Рис. 1.1. Архитектура вычислительных машин I и II поколения.

Простой процессора заложен в самой архитектуре: при вводе информации в ОП и выводе ее из ОП процессор простаивает

2. III поколение (рис 1.2)

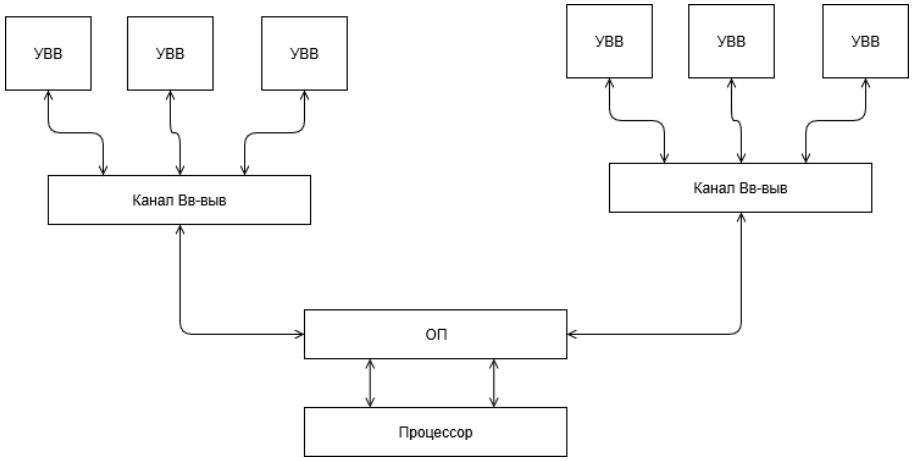


Рис. 1.2. Архитектура вычислительных машин III поколения.

Каналы ввода-вывода быстродействующие, поэтому передача информации в ОП идет гораздо быстрее, поэтому простой процессора сокращается по сравнению с I и II поколением

3. IV поколение (рис. 1.3)

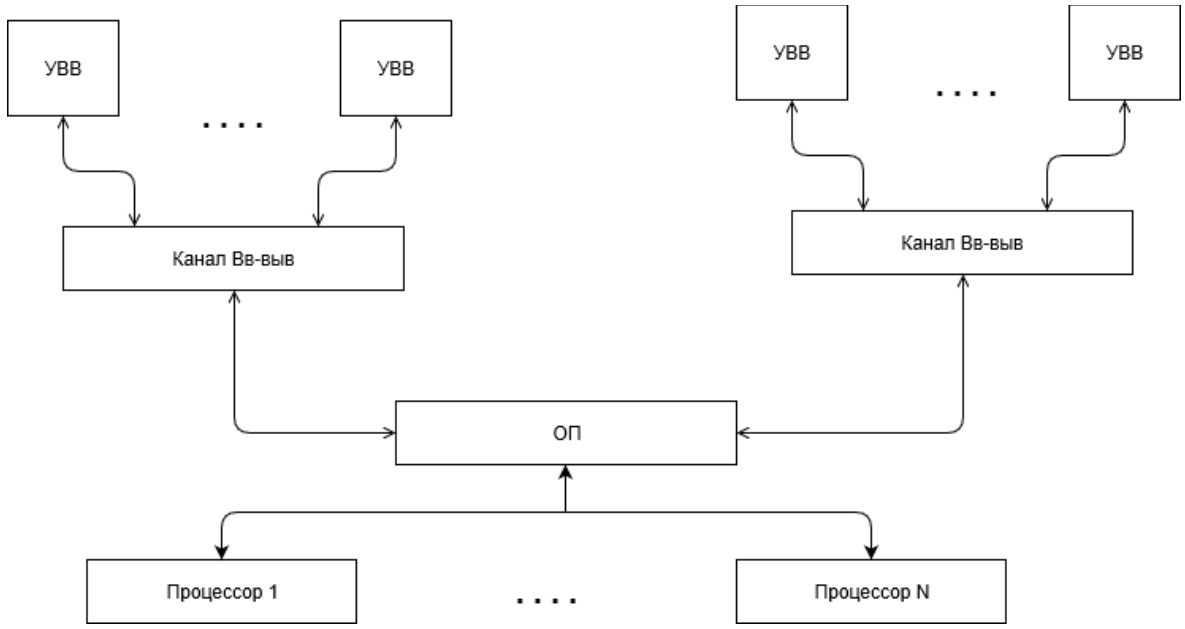


Рис. 1.3. Архитектура вычислительных машин IV поколения.

1.1.4. Типы операционных систем

1. Пакетная обработка

Предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности системы пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени. Для достижения этой цели в системах пакетной обработки используется следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличные требования к ресурсам, так чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; так, например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с

интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание. Следовательно, в таких ОС гарантировать выполнение того или иного задания в течении определенного периода времени. В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказалась от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач. Таким образом, взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя.

2. Мультипрограммный режим

Мультипрограммирование, или многозадачность, – это способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются сразу несколько программ. Эти программы совместно используют не только процессор, но и другие ресурсы компьютера: оперативную и внешнюю память, устройства ввода-вывода, данные. Мультипрограммирование призвано повысить эффективность использования вычислительной системы, однако эффективность может пониматься по-разному. Наиболее характерными критериями эффективности вычислительных систем являются: пропускная способность; удобство работы пользователей; реактивность системы.

3. Системы с разделением времени

Призваны исправить основной недостаток систем пакетной обработки - изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой. Так как в системе

разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину. Ясно, что система разделения времени обладает меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе, и, кроме того, имеются накладные расходы вычислительной мощности на более частое переключение процессора с задачи на задачу. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

1.1.5. Пример выполнения арифметической операции (сложение) на машине фон Неймана

Считается, что программа написана на алгоритмическом языке, оттранслирована в машинный язык и загружена в ОП, при этом оператор сложения пусть выглядит так: C = A + B.

После трансляции принимает вид:

Таблица 1.1.

Оттранслированная программа

0177	01	1300	1301	1302
адрес машинной команды	код операции	адрес первого числа	адрес второго числа	адрес третьего числа

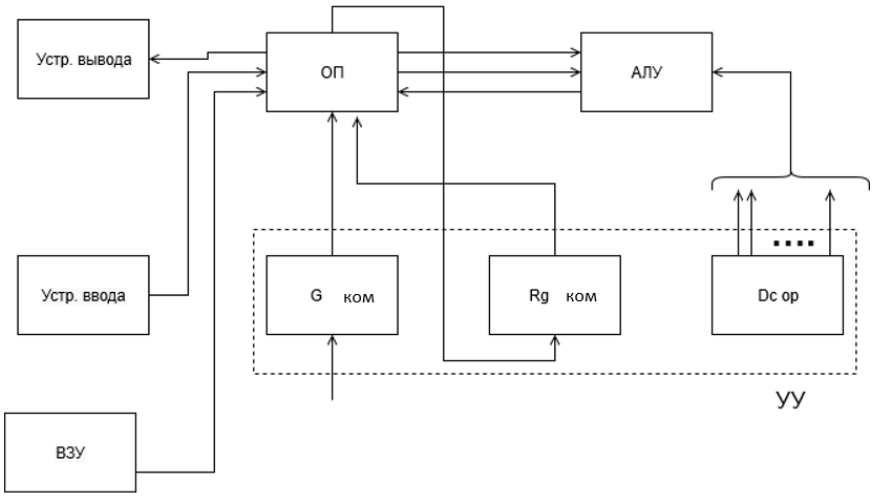


Рис. 1.4. Машина фон Неймана.

Обозначения:

УУ – устройство управления

ВЗУ – внешнее запоминающее устройство

ОП – оперативная память

АЛУ - Арифметико-логическое устройство

Ст ком - счетчик команд

Rg ком – регистр команд

Dc op – дешифратор операций

C op – код операции

Считаем, что программа и операнд уже находятся в ОП

Такт 1: На Ст ком поступает адрес выполняемой команды (0177)

Такт 2: Данный адрес (0177) из Ст ком поступает в ОП и из ячейки с этим адресом происходит выборка всей команды в УУ на Rg ком

Такт 3: адрес первого операнда поступает из Rg ком в ОП: происходит выборка операнда A на Rg1 АЛУ Rg ком

Такт 4: Адрес второго операнда поступает из Rg ком в ОП и происходит выборка числа B на Rg2 АЛУ. Одновременно с этим C op = 01 поступает из Rg com на Dc op для расшифровки.

Такт 5: расшифрованный C op поступает на SM АЛУ и происходит сложение

Такт 6: адрес результата (1302) поступает из Rg ком в ОП и происходит перезапись результата С в ОП по адресу 1302.

Такт 7: Каждая операция в машине заканчивается автоматическим прибавлением единицы т.к. команды оттранслированной программы располагаются в последовательных ячейках

Примечание.

Информация в ОП может поступать с устройства ввода, а может из ВЗУ. Результаты могут выводится на устройства вывода, а могут на ВЗУ.

Лекция №2

2.1. Булева алгебра

2.1.1. Основные определения алгебры логики

Алгебра логики (алгебра высказываний) — раздел математической логики, в котором изучаются логические операции над высказываниями.

Алгебра логики –это раздел математики, возникший в XIX веке благодаря усилиям английского математика Дж. Буля.

Законы и аппарат алгебры логики используются при проектировании различных частей компьютеров (память, процессор).

Комбинационная схема – это такая схема, которая строится только на логических элементах без использования элементов памяти.

2.1.2 Логические операции и таблицы истинности

Логическое умножение или конъюнкция:

Конъюнкция — это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложенное выражение ложно.

Обозначения: F = A & B = A ∧ B = A • B.

Таблица 2.1.

Таблица истинности для конъюнкции.

A	**B**	**F**
1	1	1
1	0	0
0	1	0
0	0	0

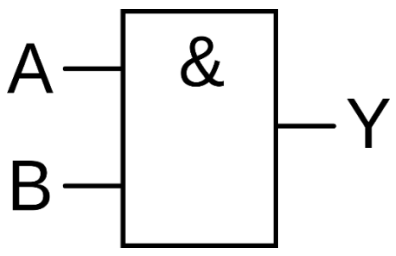


Рис. 2.1. Логическая схема конъюнкции.

Логическое сложение или дизъюнкция:

Дизъюнкция — это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выражения ложны. Обозначение: F = A v B.

Таблица 2.2.

Таблица истинности для дизъюнкции.

A	**B**	**F**
1	1	1
1	0	1
0	1	1
0	0	0

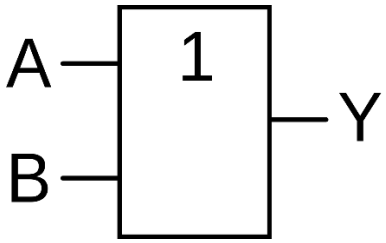


Рис. 2.2. Логическая схема дизъюнкции.

Логическое отрицание или инверсия:

Инверсия — это сложное логическое выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным. Другими простыми слова, данная операция означает, что к исходному логическому выражению добавляется частица НЕ или слова НЕВЕРНО, ЧТО.

Обозначение: F = ¬A.

Таблица 2.3.

Таблица истинности для инверсии.

A	**¬A**
1	0
0	1

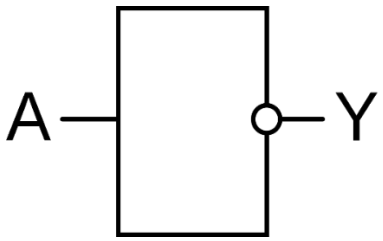


Рис. 2.3. Логическая схема инверсии.

Операция XOR (исключающие или)

«A ⊕ B» истинно тогда, когда истинно A или B, но не оба одновременно. Эту операцию также называют "сложение по модулю два".

Обозначение: F = A ⊕ B.

Таблица 2.4.

Таблица истинности для исключающего или.

A	**B**	**F**
1	1	0
1	0	1
0	1	1
0	0	0

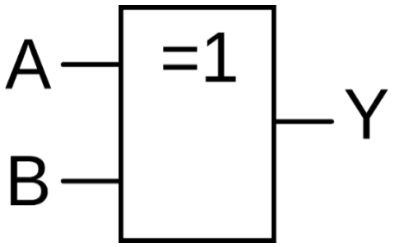


Рис. 2.4. Логическая схема исключающего или.

Функция Шеффера.

Функция Шеффера — бинарная логическая операция, булева функция над двумя переменными. Обычно обозначается | и задаётся следующей таблицей истинности:

Таблица 2.5.

Таблица истинности для функции Шеффера.

A	B	F
1	1	0
1	0	1
0	1	1
0	0	1

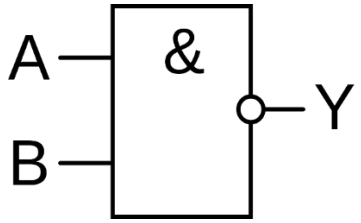


Рис. 2.5. Логическая схема функции Шеффера.

Функция Вебба.

Функция Вебба — бинарная логическая операция, булева функция над двумя переменными. Стрелка Пирса, обычно обозначаемая ↓, эквивалентна операции ИЛИ-НЕ и задаётся следующей таблицей истинности:

Таблица 2.6.

Таблица истинности для функции Вебба

A	B	F
1	1	0
1	0	0
0	1	0
0	0	1

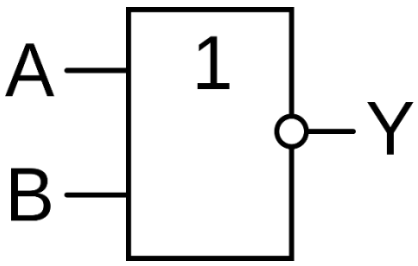


Рис. 2.6. Логическая схема функции Вебба

Примечание 1. Если комбинационная схема построена на элементах «И», «ИЛИ», «НЕ», то она построена в классическом элементном базисе

Примечание 2. Комбинационная схема, начиная с III поколения машин, строится на интегральных элементах, которые реализованы на функциях Шеффера и Вебба.

2.1.3. Законы алгебры логики

В алгебре высказываний логические законы выражаются в виде равенства эквивалентных формул. В справедливости всех законов можно убедиться, построив таблицы истинности для левой и правой частей записанного закона. После упрощения выражения с применением законов алгебры логики таблицы истинности совпадают. Справедливость части законов можно доказать, применяя инструментарий таблиц истинности.

Таблица 2.7.

Законы алгебры логики.

Закон	Для ИЛИ	Для И
Переместительный	$x \vee y = y \vee x$	$x \cdot y = y \cdot x$
Сочетательный	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Распределительный	$x \cdot (y \vee z) = x \cdot y \vee x \cdot z$	$x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z)$
Правила де Моргана	$\overline{x \vee y} = \bar{y} \cdot \bar{x}$	$\overline{x \cdot y} = \bar{y} \vee \bar{x}$
Идемпотенции	$x \vee x = x$	$x \cdot x = x$
Поглощения	$x \vee (x \cdot y) = x$	$x \cdot (x \vee y) = x$
Склеивания	$(x \cdot y) \vee (\bar{x} \cdot y) = y$	$(x \vee y) \cdot (\bar{x} \vee y) = y$
Операция переменной с ее инверсией	$x \vee \bar{x} = 1$	$x \cdot \bar{x} = 0$
Операция с константами	$x \vee 0 = x;\ x \vee 1 = 1$	$x \cdot 1 = x;\ x \cdot 0 = 0$
Двойного отрицания	$\bar{\bar{x}} = x$	

2.1.4. Основные определения булевой алгебры:

1. Конституента «1» (f^1) — это логическая функция, которая обращается в «1», только на одном наборе входных аргументов

2. Конституента «0» (f^0) — это логическая функция, которая обращается в «0», только на одном наборе входных аргументов

2.1.5. Способы построения конституент

1. Для построения функции типа f^1 надо аргументы, равные нулю взять со знаком инверсии, а аргументы равные единице, взять без знака инверсии и между ними поставить знак конъюнкции.

2. Для построения функции типа f^0 надо аргументы, равные единице взять со знаком инверсии, а аргументы равные нулю, взять без знака инверсии и между ними поставить знак дизъюнкции.

2.6. Правило построения логической функции в виде ДНФ и КНФ

Дизъюнктивная нормальная форма (ДНФ), представляющая собой дизъюнкцию нескольких элементарных конъюнкций

Конъюнктивная нормальная форма (КНФ), представляющая собой конъюнкцию элементарных дизъюнкций

Пусть логическая функция от 3х аргументов задана следующей таблицей истинности:

Таблица 2.8.

Таблица истинности.

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1

1	1	1	0

Для построения ДНФ нужно: для аргументов, обращающих функцию в 1, построить f^1 и между ними поставить дизъюнкции.

ДНФ: f (a, b, c)= $\bar{a} \bullet \bar{b} \bullet \bar{c} \vee \bar{a} \bullet \bar{b} \bullet c \vee a \bullet \bar{b} \bullet c \vee a \bullet b \bullet \bar{c}$

Для построения КНФ нужно: для аргументов, обращающих функцию в 0, построить f^0 и между ними поставить конъюнкции.

КНФ: f (a, b, c) = $(a \vee \bar{b} \vee c)\&(a \vee \bar{b} \vee \bar{c})\&(\bar{a} \vee b \vee c)\&(\bar{a} \vee \bar{b} \vee \bar{c})$

Основная теорема булевой алгебры: Любая произвольная логическая функция может быть построена на элементарных функциях «И», «ИЛИ», «НЕ».

2.1.7. Правила преобразования логических функций, построенных в классическом базисе («И», «ИЛИ», «НЕ») в элементный базис Шеффера и Вебба:

Для преобразования в эл. базис Шеффера надо взять логическую функцию, записанную в базисе ДНФ и от этой функции взять двойную инверсию: нижнюю инверсию преобразовать по формулам де Моргана, а верхнюю инверсию оставить на месте.

Пример:

$$f(a, b, c) = \overline{\overline{\bar{a} \bullet \bar{b} \bullet \bar{c} \vee \bar{a} \bullet \bar{b} \bullet c \vee a \bullet \bar{b} \bullet c \vee a \bullet b \bullet \bar{c}}} =$$

$$\overline{\overline{\bar{a} \bullet \bar{b} \bullet \bar{c}} \,\&\, \overline{\bar{a} \bullet \bar{b} \bullet c} \,\&\, \overline{a \bullet \bar{b} \bullet c} \,\&\, \overline{a \bullet b \bullet \bar{c}}}$$

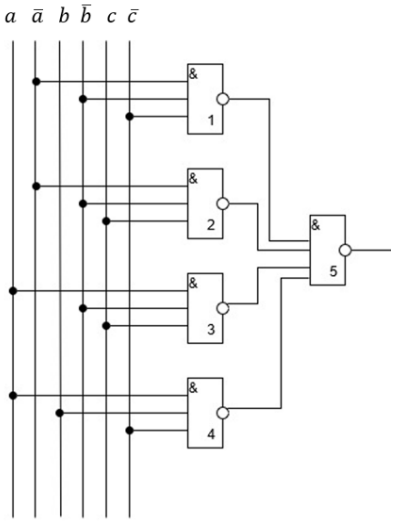


Рис. 2.7 – Схема ДНФ в базисе Шеффера.

Лекция №3

3.1. Триггеры

3.1.1. Виды триггеров

В ВТ используются следующие виды триггеров:

- RS-Tr (установочный триггер)
- T-Tr (триггер со счетным входом)
- D-Tr (триггер задержки)
- JK-Tr (универсальный триггер)
- Dv-Tr (универсальный триггер)

Все триггера имеют 2 выхода: прямой и инверсный.

Триггера бывают:

- Синхронные
- Асинхронные

Асинхронный триггер – такой триггер, который устанавливается в новое значение при подаче входных сигналов только на информационные входы.

Синхронный триггер – такой триггер, который устанавливается в новое значение при подаче входных сигналов как на информационные, так и на синхро-входы.

3.1.2. Асинхронный RS-Tr

Асинхронный RS-Tr – это такой триггер, который имеет 2 информационных входа: R и S. При подаче действующего значения на вход S триггер устанавливается в «1»; при подаче действующего значения на вход R триггер устанавливается в «0».

Построение асинхронного RS-Tr:

1. На элементах «ИЛИ-НЕ»

Действующее значение – «1».

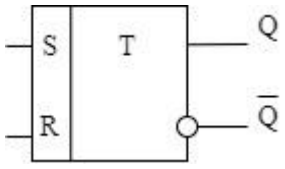


Рис. 3.1. Структурное обозначение асинхронного RS-Tr.

Таблица 3.1.

Таблица переходов (истинности).

S	R	Q_{T+1}	Примечание
0	0	Q_T	Хранение старого состояния
0	1	0	Установка «0»
1	0	1	Установка «1»
1	1	*	Запрет

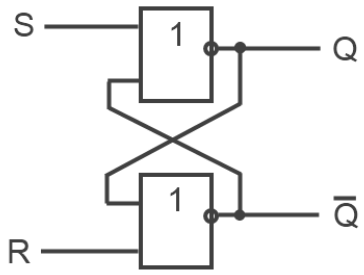


Рис. 3.2. Функциональная схема.

Правила рассмотрения схемы с элементами памяти:

1) Задать начальное состояние триггеров

2) Развести начальные состояния по всем обратным связям

3) Рассматривать работу схемы с того плеча, куда подано действующее значение для данной элементной базы (после подачи входных сигналов).

2. На элементах «И-НЕ»

Действующее значение – «0».

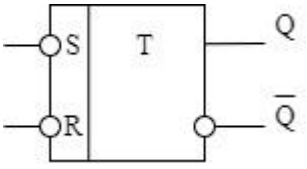


Рис. 3.3. Структурное обозначение асинхронного RS-Tr.

Знаки инверсии на входах S и R означают, что запись нового значения в триггер делается инверсным значением к общепринятому сигналу (общепринятый сигнал – «1»).

Таблица 3.2.

Таблица переходов (истинности).

S	R	Q_{T+1}	Примечание
0	0	*	Запрет
0	1	1	Установка «1»
1	0	0	Установка «0»
1	1	Q_T	Хранение старого состояния

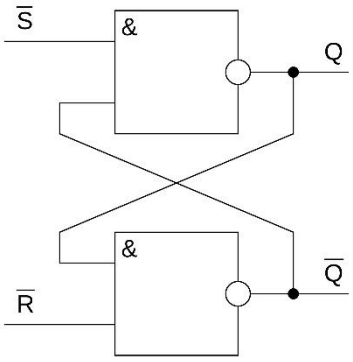


Рис. 3.4. Функциональная схема.

3.1.3. Синхронный RS-Tr

Синхронный RS-Tr – это такой триггер, который при подаче «1» на вход S устанавливается в «1», а при подаче «1» на вход R устанавливается в «0», и это при условии, что на синхро-вход C подана «1».

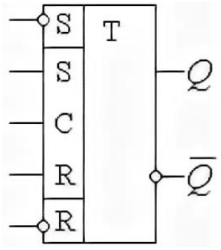


Рис. 3.5. Структурное обозначение синхронного RS-Tr.

Таблица 3.3.

Таблица переходов (C = 1).

S	R	Q_{T+1}	Примечание
0	0	Q_T	Хранение старого состояния
0	1	0	Установка «0»
1	0	1	Установка «1»

1	1	*	Запрет

3.1.4. T-Tr (триггер со счетным входом)

T-Tr – это такой триггер, который имеет один информационный вход T. При подаче «1» на вход T триггер меняет свое состояние на противоположное; при подаче «0» на вход T триггер хранит старое состояние.

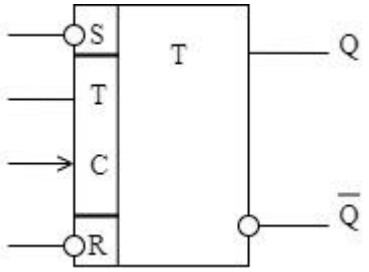


Рис. 3.6. Структурное обозначение T-Tr.

Таблица 3.4.

Таблица переходов (C = 1).

T	Q_{T+1}	Примечание
0	Q_T	Хранение старого состояния
1	$\overline{Q_T}$	Переброс

3.5. D-Tr (триггер задержки)

D-Tr – это такой триггер, который функционирует по следующей формуле:

$$Q_{T+1} = D_t$$

Таблица 3.5.

Таблица переходов (C = 1).

D_t	Q_{T+1}	Примечание
0	0	Установка «0»
1	1	Установка «1»

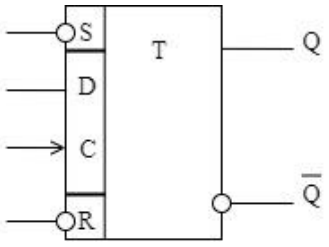


Рис. 3.7. Структурное обозначение D-Tr.

3.1.6. Универсальный JK-Tr

В машинах используются RS, T и D триггеры. На универсальном JK триггере можно построить любой из этих триггеров.

JK-Tr – это такой триггер, который имеет 2 информационных входа: J и K. При подаче J и K «1» (J=K=1) триггер работает как T-Tr, т.е. меняет свое состояние на противоположное. В остальных случаях он работает как RS-Tr, если вход J считать входом S, а вход K считать входом R.

Универсальные триггеры не имеют запрещенных комбинаций входных сигналов. Они просто переходят из одного режима триггера в другой.

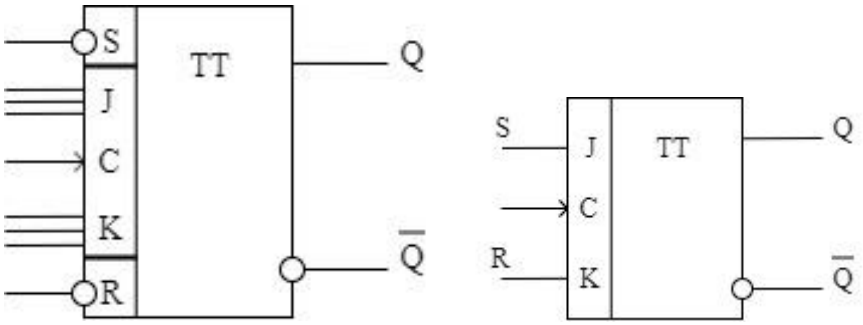


Рис. 3.8. Структурное обозначение JK-Tr.

Таблица 3.6.

Таблица переходов (C = 1).

J	K	Q_{T+1}	Примечание
0	0	Q_T	Хранение старого состояния
0	1	0	Установка «0»
1	0	1	Установка «1»
1	1	$\overline{Q_T}$	Переброс

Построение на основе JK-Tr RS-, D- и T- триггеров

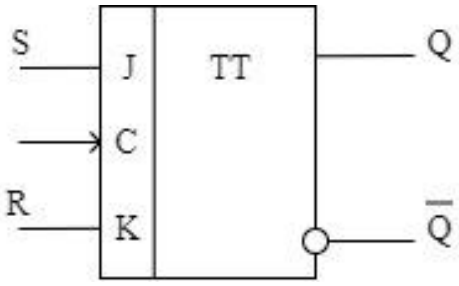


Рис. 3.9. Синхронный RS-Tr.

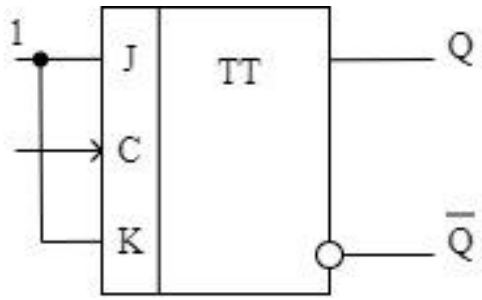


Рис. 3.10. T-Tr асинхронный.

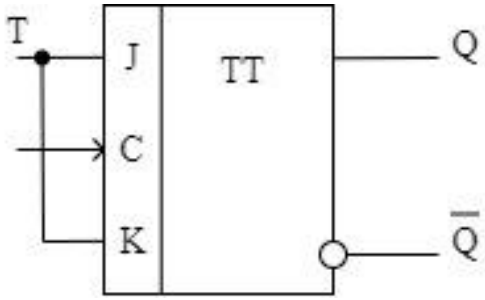


Рис. 3.11. T-Tr синхронный.

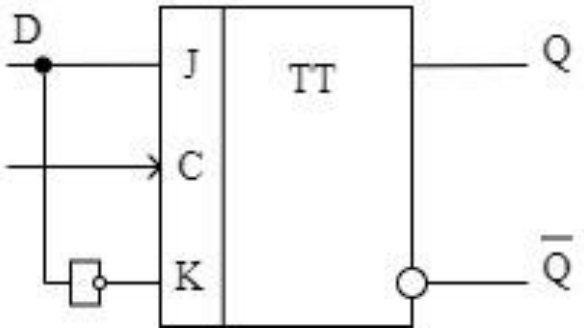


Рис. 3.11. D-Tr.

3.1.7. Универсальный Dv-Tr

Dv-Tr – это такой триггер, который имеет один информационный вход D и один управляющий вход V. При подаче «1» на вход V триггер работает как D-Tr; при подаче «0» на вход V триггер хранит старое состояние.

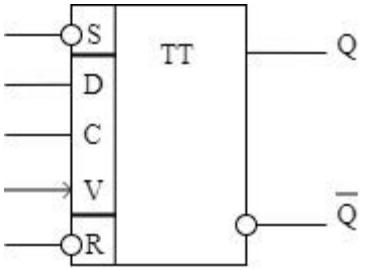


Рис. 3.12. Структурное обозначение Dv-Tr.

Таблица 3.7.

Таблица переходов (C = 1).

D	V	Q_{T+1}	Примечание
0	0	Q_T	Хранение старого состояния
0	1	0	Установка «0»
1	0	Q_T	Хранение старого состояния
1	1	1	Установка «1»

Построение на основе Dv-Tr T- и D- триггеров:

Примечание. RS-Tr на основе Dv-Tr построить нельзя, так как функцию RS-Tr берет на себя D-Tr.

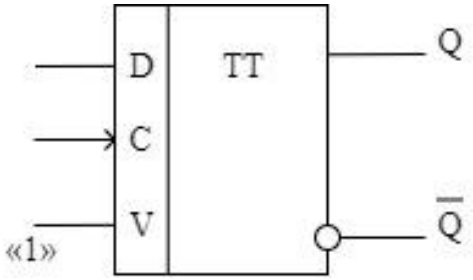


Рис. 3.13. D-Tr синхронный.

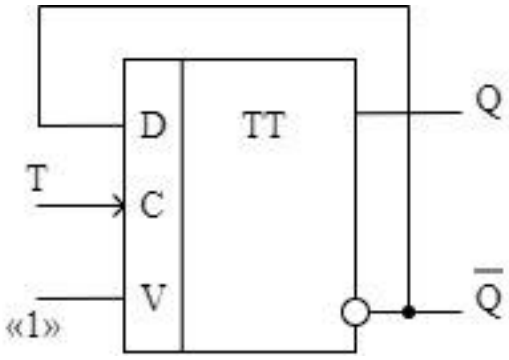


Рис. 3.14. T-Tr асинхронный.

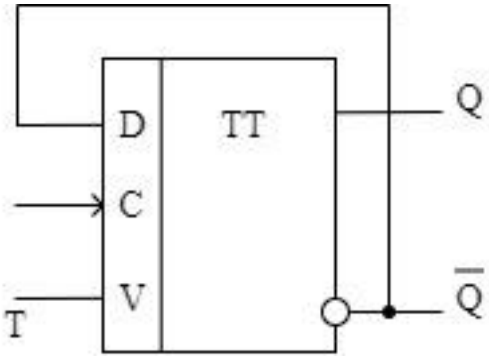


Рис. 3.15. T-Tr синхронный.

Лекция №4

4.1. Регистры

4.1.1. Виды регистров

Регистр – это узел машины, выполненный на триггерах, число которых равно числу разрядов машинного слова, и предназначенный для записи, хранения и выдачи информации.

Регистры обычно выполняются на RS и D триггерах.

Кроме того, регистры могут:

- Установка в ноль (гашение регистра)
- Преобразование последовательного кода в параллельный и наоборот
- Преобразование из прямого кода в обратный и наоборот
- Сдвиг кода вправо и влево на определенное число разрядов
- На регистрах могут выполняться следующие логические операции:

1) Дизъюнкция
2) Конъюнкция
3) Поразрядное сложение (сложение по модулю 2)

Регистры по способу записи делятся на:

1) Параллельные (информация поступает одновременно на все разряды по двум шинам S и R)
2) Последовательные (информация поступает через определенные промежутки времени разряд за разрядом по одной шине путём сдвига всего кода синхро-сигнала)
3) Параллельно-последовательные (такие регистры имеют входы как для параллельной, так и для последовательной записи)

Существует 2 вида записи информации в регистры:

1) Парафазная запись. Такая запись делается за один такт, обычно по двум шинам, – в параллельных регистрах
2) Монофазная запись. Такая запись делается за два такта:
   - В первом такте делается обнуление регистра по входам R

- Во втором такте делается собственно запись по входам S

4.1.2. Структурное обозначение регистров

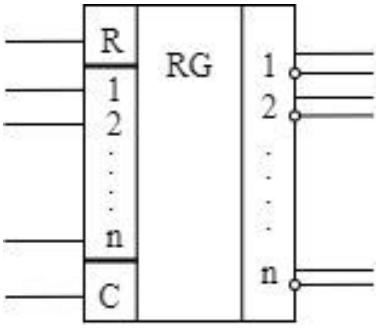


Рис. 4.1. Монофазный RG

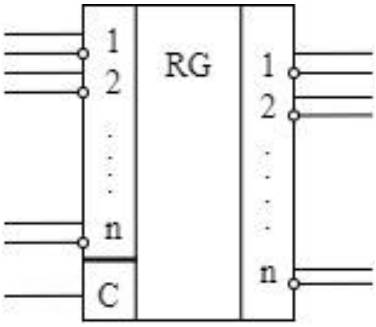


Рис. 4.2. Парафазный RG

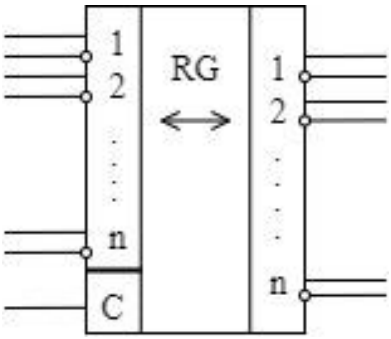


Рис. 4.3. Парафазный сдвиговый RG (реверсивный)

4.1.3. Трехразрядный парафазный регистр на RS-триггерах

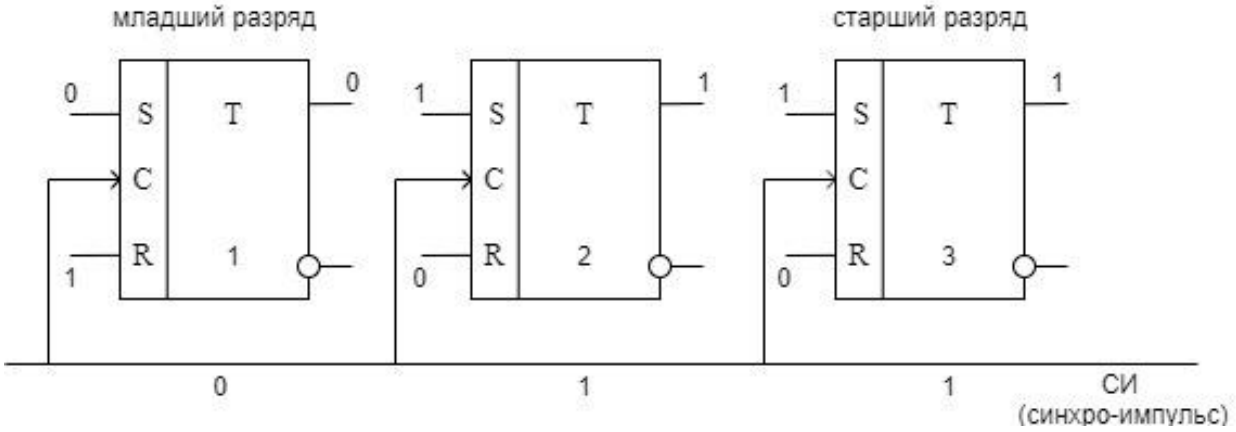


Рис. 4.4. Трехразрядный парафазный регистр на RS-триггерах.

ВАЖНО! В вычислительной технике никакая информация не должна передаваться без специального управляющего сигнала, который подается по специальным управляющим шинам через специальную управляющую логику (в некоторых схемах управляющей логики может не быть).

Схема 4.4. работает за один такт: после поступления сигнала СИ делается запись.

4.1.4. Монофазный трехразрядный регистр на RS-триггерах

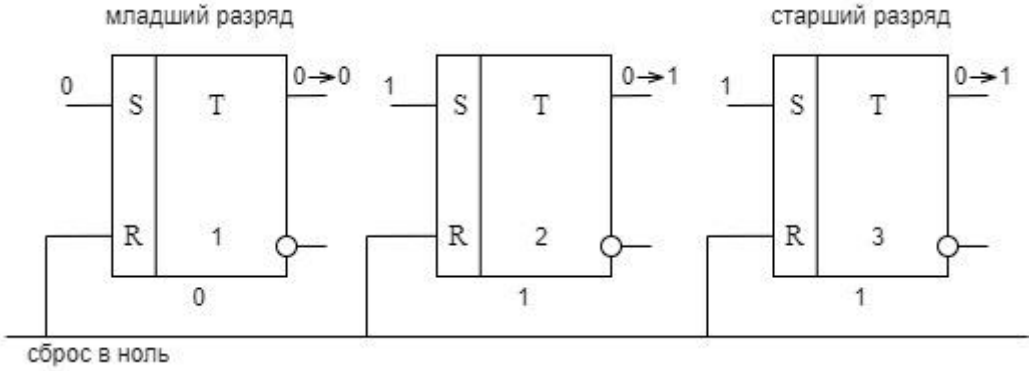


Рис. 4.5. Монофазный трехразрядный регистр на RS-триггерах.

Монофазный RG работает за два такта:

1) Сначала сброс в ноль по входам R
2) Затем запись по входам S

На схеме 4.5. производится установка числа 6.

4.1.5. Передача информации из регистра прямым парафазным кодом

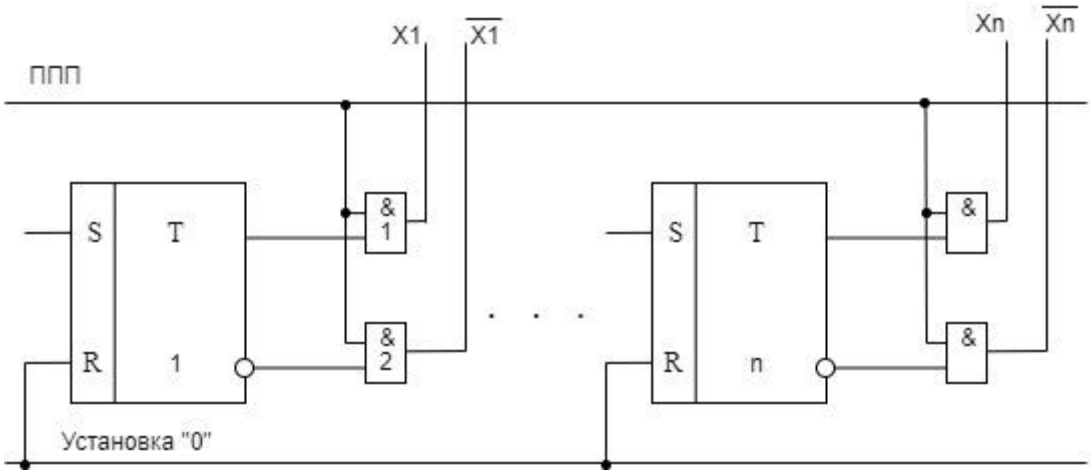


Рис. 4.6. Передача информации из регистра прямым парафазным кодом.

Прежде чем выдавать информацию из регистра эту информацию нужно в регистр записать, используя монофазную запись.

ППП – передача прямым парафазным.

В схеме 4.6. в качестве управляющей логики используется схема «И».

После того, как информация записана в регистр схема ждет управляющего сигнала «1», поступающего с управляющей шины ППП. Этот управляющий сигнал поступает на управляющую логику и происходит передача прямым парафазным кодом.

4.1.6. Передача информации между двумя параллельными регистрами

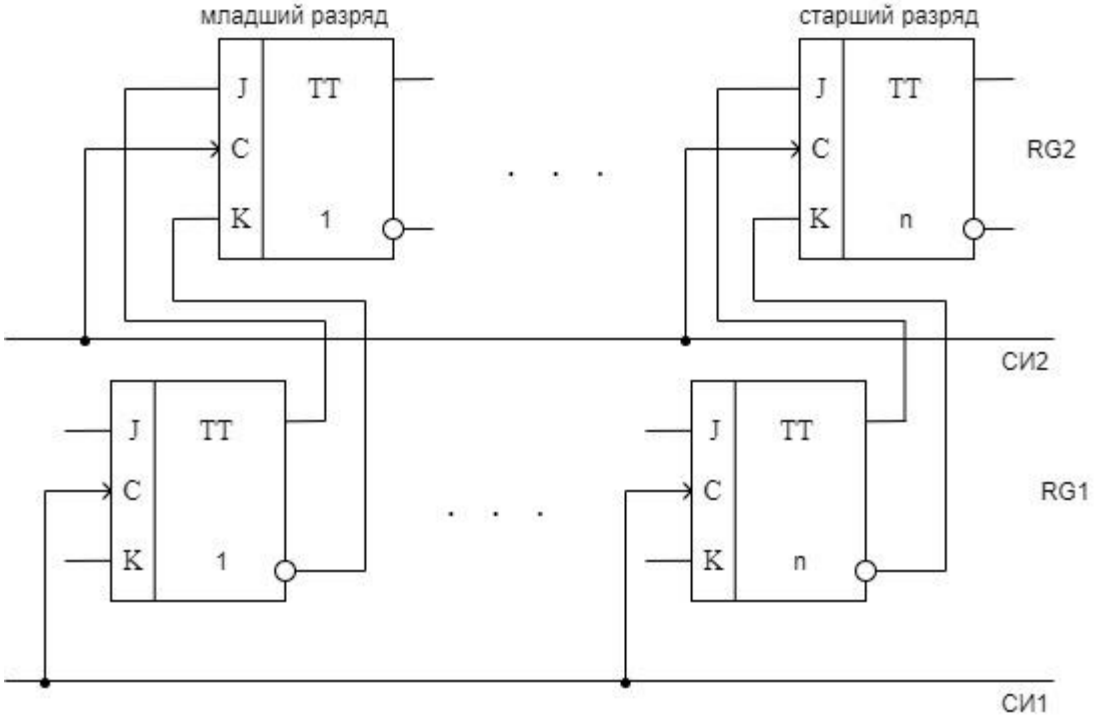


Рис. 4.7. Передача информации между двумя параллельными регистрами.

Сначала делается запись в RG1, а затем эта информация передается в RG2. Запись в оба регистра – парафазная. Регистры построены на JK-триггерах в режиме синхронного RS-триггера.

ВАЖНО! Очень часто в ВМ шина СИ используется в качестве управляющей шины «разрешение на запись»:

- СИ1 – разрешение на запись в RG1
- СИ2 – разрешение на запись в RG2

4.1.7. Сдвиговые регистры

Сдвиговые регистры по способу записи подразделяются на:

- RG прямого сдвига вправо ($\overset{1}{\rightarrow}$)

- RG прямого сдвига влево ($\overset{1}{\leftarrow}$)
- Реверсивные сдвиговые регистры ($\overset{1}{\leftrightarrow}$)
- Сдвигатели (на заданное число разрядов)
- Косая передача (рис. 4.8.)

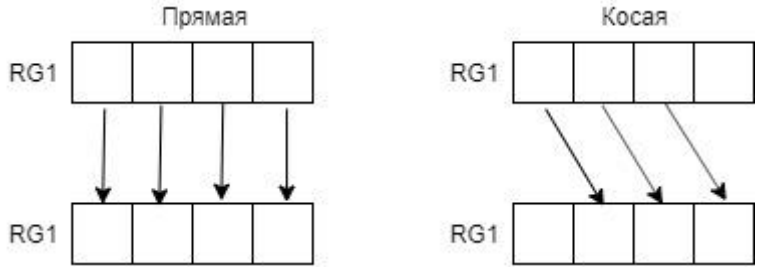


Рис. 4.8. Прямая и косая передача.

Сами регистры не являются сдвиговыми. Сдвиг делается во время передачи.

4.1.8. Сдвиговый RG на JK триггерах с парафазной передачей между разрядами

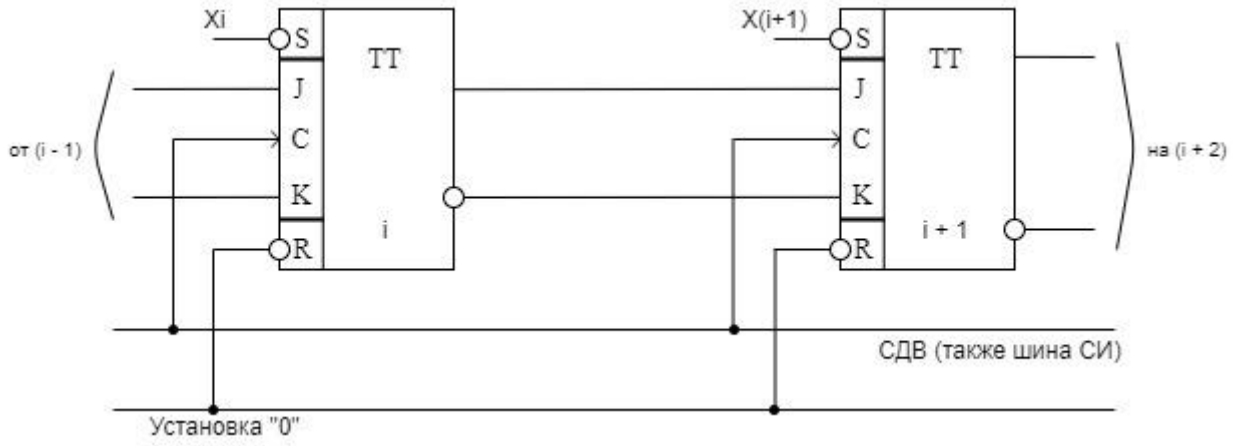


Рис. 4.9. Сдвиговый RG на JK триггерах с парафазной передачей.

Прежде чем сдвигать информацию, надо ее сначала в RG записать. Запись делается монофазная по асинхронным RS входам.

Шина СИ выполняет функцию управляющей шины «разрешение на СДВ». Перезапись из i-го разряда в i+1 произойдет только после того, как на триггера поступит сигнал разрешения на сдвиг.

4.1.9. Сдвиговый RG на D триггерах

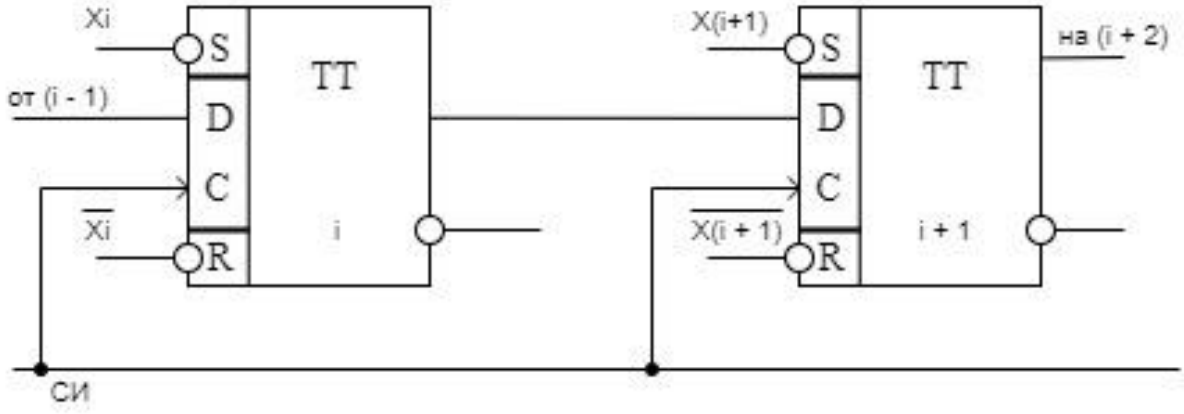


Рис. 4.10. Сдвиговый RG на D триггерах.

Запись делается парафазная по асинхронным RS входам.

Сдвиг тоже парафазный, хотя и по одной шине (в схеме 4.10. нет «Установки 0», поэтому не может быть 2 тактов).

4.1.10. Выполнение логической операции «ИЛИ» на 2-х регистрах

В этом случае на RG1 хранится операнд A, а на RG2 – операнд B. Результат тоже получается на RG2.

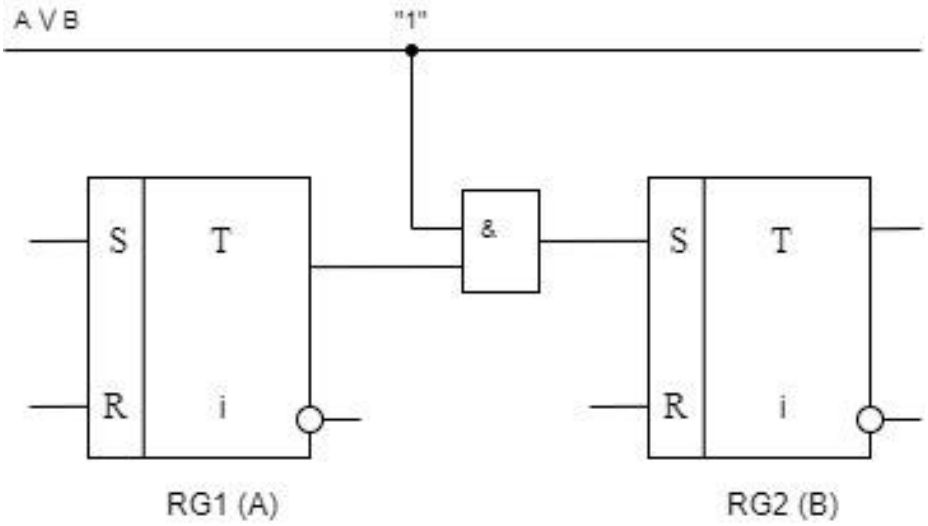


Рис. 4.11. Выполнение логической операции «ИЛИ» на 2-х регистрах.

Управляющая шина $A \vee B$ дает разрешение на управление операцией. Управляющая логика построена на элементах «И». Оба RG представлены одним i-м разрядом.

Таблица 4.1

Таблица истинности.

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Лекция №5

5.1. Дешифраторы

5.1.1. Виды дешифраторов

DC — это комбинационная схема, которая имеет n входов и 2^n выходов. При этом, при подаче сигналов определенной комбинации входных сигналов выходной сигнал возникает только на одной выходной шине.

Структурное обозначение:

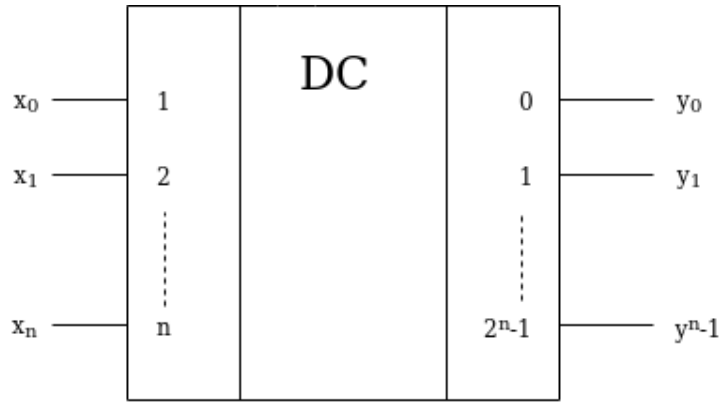


Рис. 5.1. Структурное обозначение дешифратора.

Дешифраторы используются для выработки управляющих сигналов в УУ.

По способу построения DC бывают:

- линейные
- прямоугольные
- пирамидальные

Примечание. В интегральной технике наиболее экономичными являются:

- линейные DC (для маловходовых DC).
- прямоугольные DC (для многовходовых DC).

Прямоугольные и пирамидальные DC объединяют в отдельный вид – каскадные.

Рассмотрим построение двухвходового линейного DC.

5.1.2. Линейный DC

Таблица 5.1.

Таблица истинности.

X_2	X_1	Y_0	Y_1	Y_2	Y_3

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

Считаем, что DC вырабатывает управляющие сигналы равные “1”.

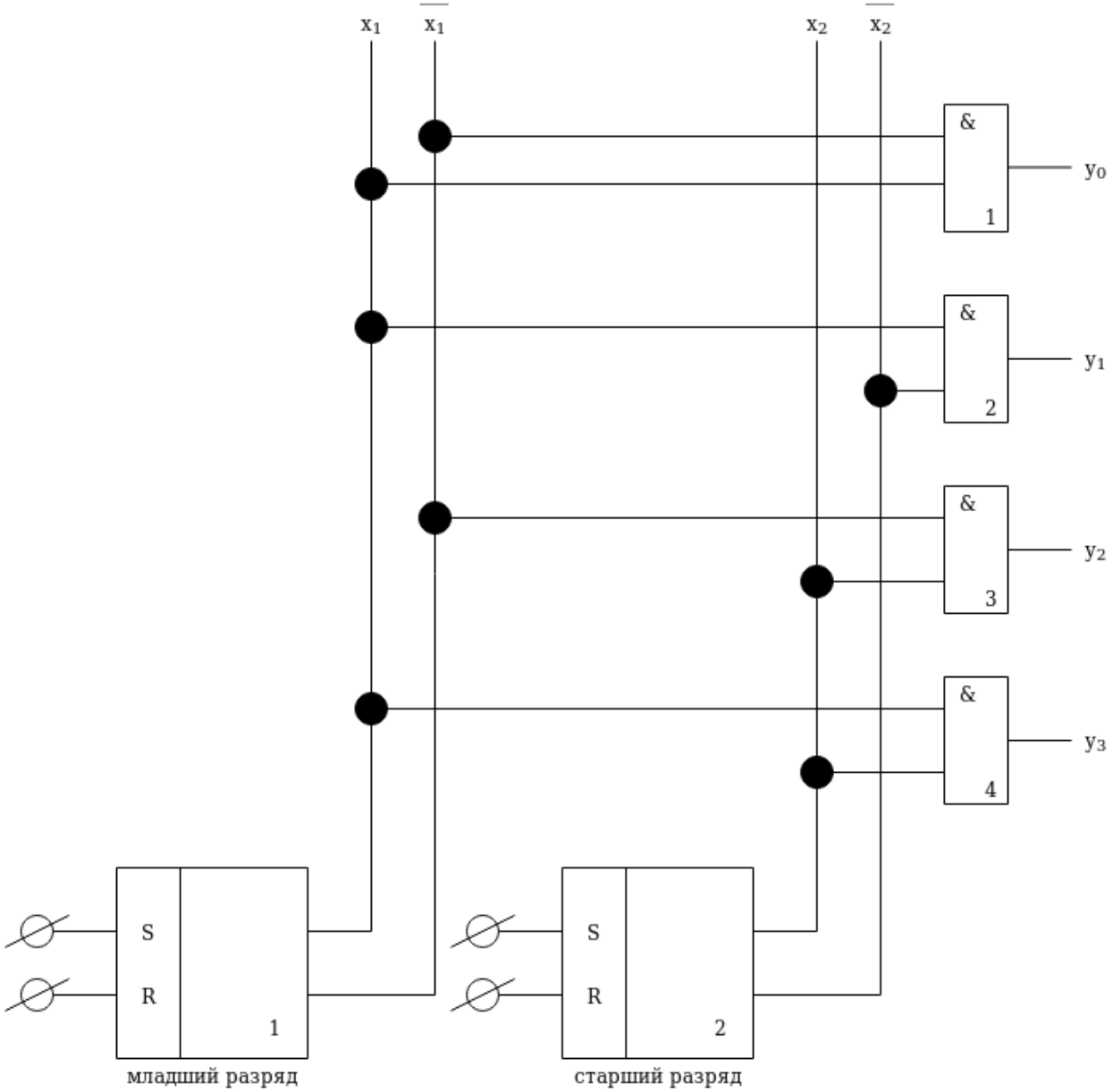


Рис. 5.2. Схема линейного дешифратора.

Чтобы DC вырабатывал управляющие сигналы равные «0», не надо менять коммутацию в DC, надо просто:

- или поставить инверторы в выходные шины;

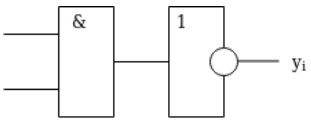


Рис. 5.3. Инвертор.

- или использовать элементы Шеффера.

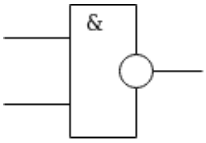


Рис. 5.4. Элемент Шеффера.

При этом в таблице истинности будет нулевая диагональ на том же месте. Остальные поля – «1».

В тех случаях, когда количество входов превышает 8, мы обращаемся к более сложным моделям дешифратора - каскадным.

5.1.3. Прямоугольный DC

Состоит из 2-х и более каскадов.

Особенность: все входные сигналы поступают только на первый каскад.

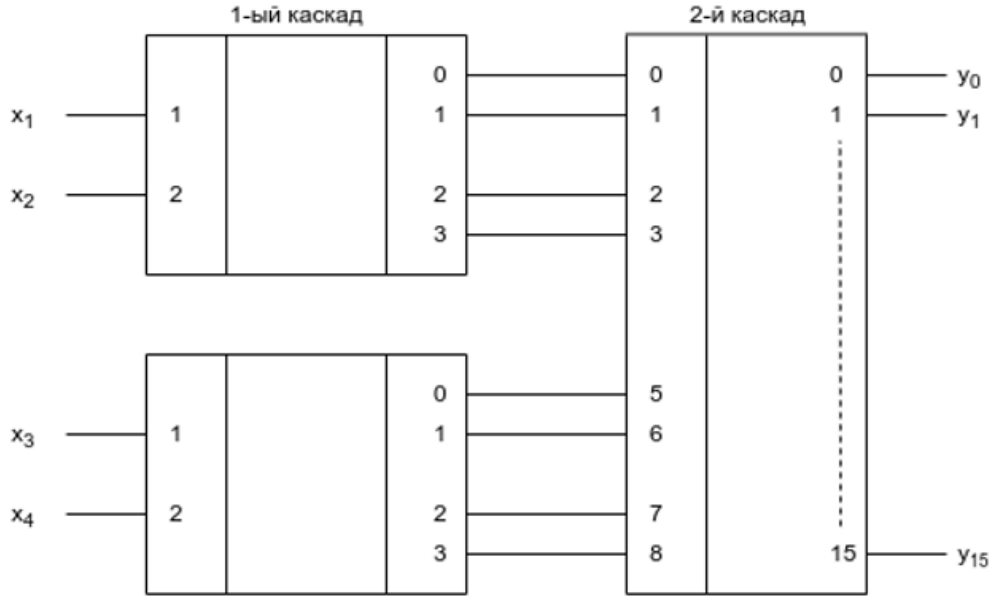


Рис. 5.5. Схема прямоугольного дешифратора.

В первом каскаде стоят 2 линейных двухвходовых дешифратора. Во втором каскаде стоит так называемая «дешифраторная матрица», у которой n входов и 2*n выходов.

5.1.4. Пирамидальный DC

Является еще одним видом каскадного DC. Количество входов и выходов совпадают с прямоугольным DC.

Особенность: на каждый каскад поступает хотя бы один входной сигнал.

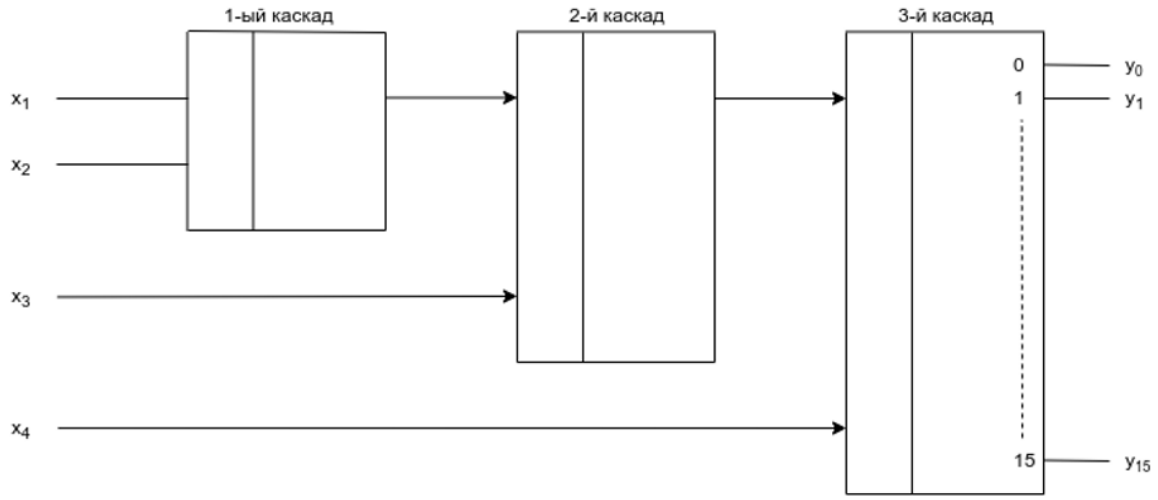


Рис. 5.6. Схема пирамидального дешифратора.

Лекция №6

6.1 Счетчики

6.1.1 Виды счетчиков

СТ — это узел машины, предназначенный для расчета входных сигналов.

Выполняется на Т-триггерах или на JK-триггерах в режиме Т-триггера.

Счетчики используются для подачи входных сигналов на DC.

СТ могут быть:

1) Двоичные с модулем (пере)счета:

- $M = 2^n$
- $M \neq 2^n$

2) Двоичные, троичные и т.д.

3) Суммирующие, вычитающие, реверсивные

4) По организации цепей переноса могут быть:

- с последовательным переносом
- со сквозным переносом
- с параллельным переносом

5) Кольцевые счетчики

Все СТ, перечисленные выше, могут быть синхронными или асинхронными (в зависимости от триггеров, на которых они строятся).

6.2. Организация цепей переноса двоичных счетчиков с модулем счета $M=2^n$.

Последовательный перенос

Возьмем для примера трехразрядный асинхронный суммирующий счетчик с последовательным переносом между разрядами на JK-триггерах.

Таблица 6.1. Таблица истинности трехразрядного асинхронного суммирующего счетчика.

Таблица 6.1

Таблица истинности

$X_{вх.}$	Q_3	Q_2	Q_1

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

Q_1, Q_2, Q_3 – выходы триггеров.

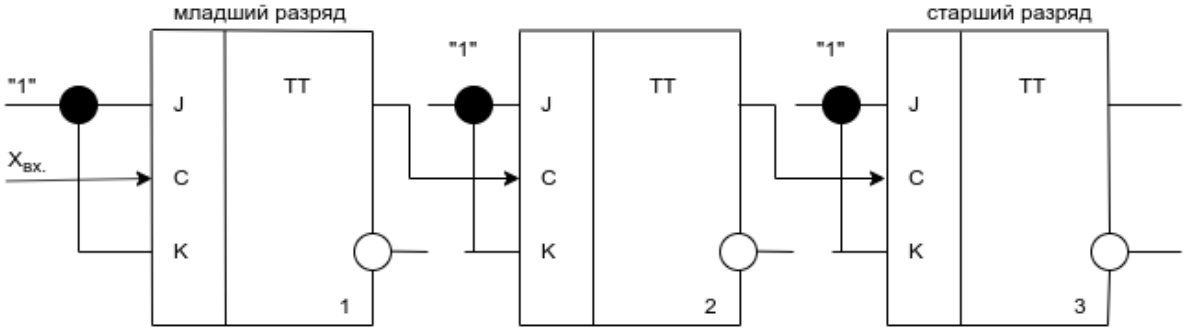


Рис. 6.1. Трехразрядный асинхронный суммирующий счетчик.

При построении счетчиков нужно ответить на 2 вопроса:

1) Куда подается входной сигнал?

2) От чего срабатывает каждый разряд счетчика?

При последовательном переносе входной сигнал поступает только на младший разряд счетчика, а каждый разряд счетчика срабатывает только от одного предыдущего разряда счетчика.

Примечание. Для построения вычитающего счетчика надо с инверсного выхода i-го разряда подавать на С-вход i+1 разряда.

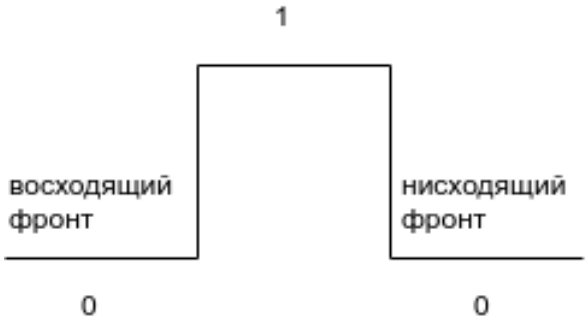


Рис. 6.2. Пояснение для различия фронтов.

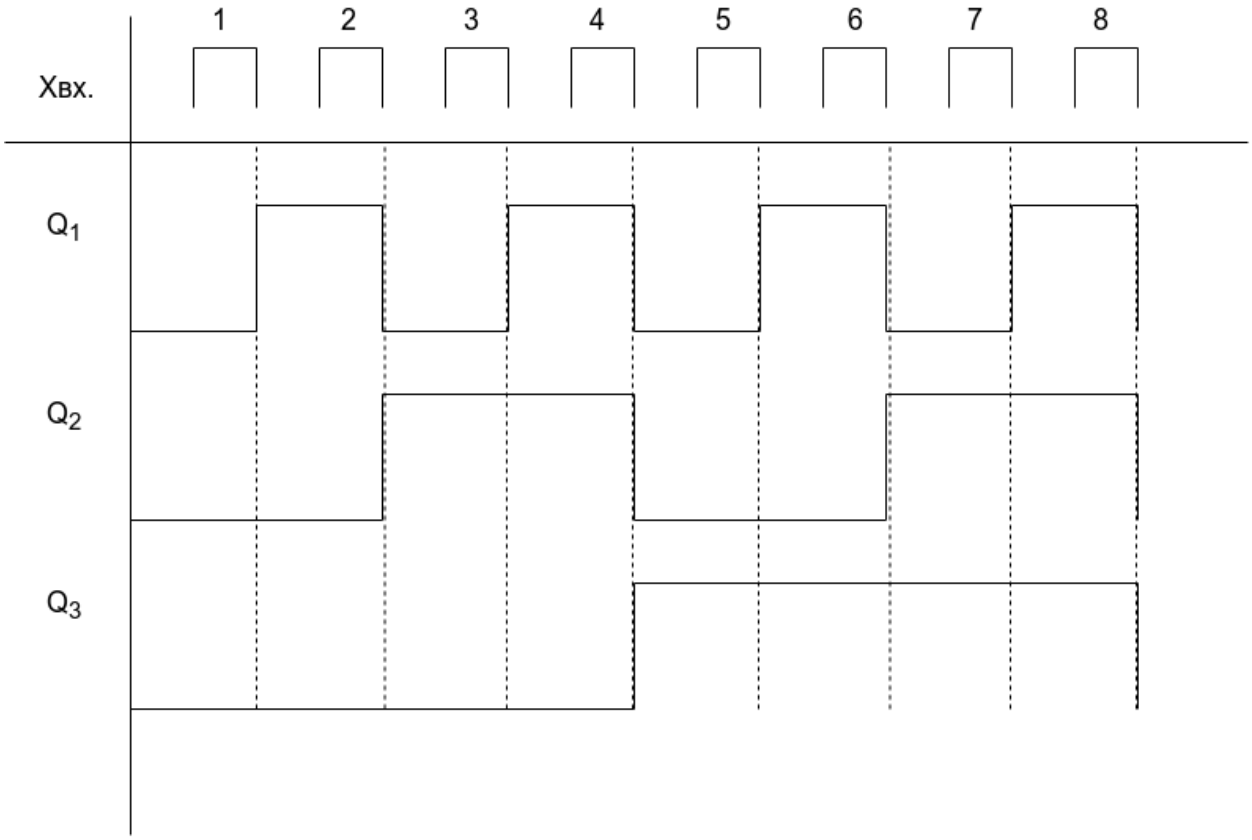


Рис. 6.3. Временная диаграмма для счетчика с последовательным переносом.

При рассмотрении работы схемы надо помнить, что JK-триггер работает по нисходящему фронту входного сигнала, а DV-триггер – по восходящему фронту.

Достоинства последовательного счетчика:

1) Простота реализации.

Недостатки:

1) Низкое быстродействие.

$t_{тр} = \tau_{тр} * n$, где n – количество разрядов счетчика.

2) Высокая вероятность появления «эффекта гонки», при котором в многоразрядном счетчике на некотором i-том такте младший разряд сработал от i-го входного сигнала, а старший разряд еще не сработал от i-1 входного сигнала, так как запаздывает.

Сквозной перенос

Идеология построения:

1) Входной сигнал поступает одновременно на все разряды.

2) Каждый разряд срабатывает от всех предыдущих разрядов.

Рассмотрим синхронный суммирующий четырехразрядный счетчик на JK-триггерах со сквозным переносом между разрядами.

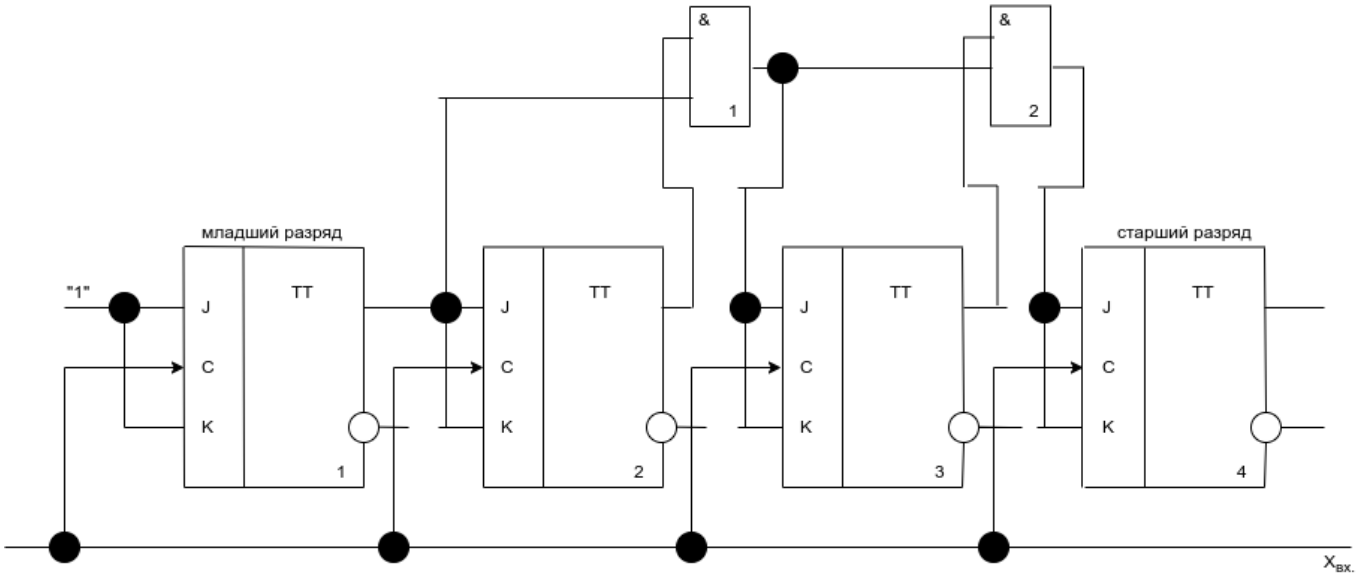


Рис. 6.4. Синхронный суммирующий четырехразрядный счетчик.

Так как счетчик синхронный, то все триггера, кроме первого должны работать в режиме синхронного Т-триггера, а первый в режиме асинхронного Т-триггера.

Примечание:

1) На шину X входа входные сигналы поступают от генератора полностью сформированными, поэтому у них всегда будет нисходящий фронт (специально заботиться о нисходящем фронте входного сигнала не надо).

2) Рассматривать работу схемы можно с любого разряда счетчика, так как входной сигнал поступает одновременно на все разряды счетчика. Лучше рассматривать начиная со старшего разряда.

3) Если рассматривать работу схемы начиная с младшего разряда, то надо помнить, что на вход каждого разряда с предыдущих разрядов поступает не новое значение, а старое.

Достоинства:

1) Более быстродействующая схема.

$t_{тр} = \tau_{тр} + \tau_{\&} * (n-2)$, где n – количество разрядов счетчика.

Недостатки:

1) Более сложная схема (добавилась цепочка сквозного переноса).

2) Остается некоторая вероятность появления “эффекта гонки” из-за задержки в цепи сквозного переноса.

Параллельный перенос

Идеология построения та же, что и у сквозного счетчика.

Отличия в реализации заключаются в том, что у сквозного переноса цепочка сквозного переноса внешняя, а у параллельного счетчика – внутренняя.

Пример: синхронный суммирующий счетчик на JK-триггерах (четырехразрядный) с параллельным переносом между разрядами.

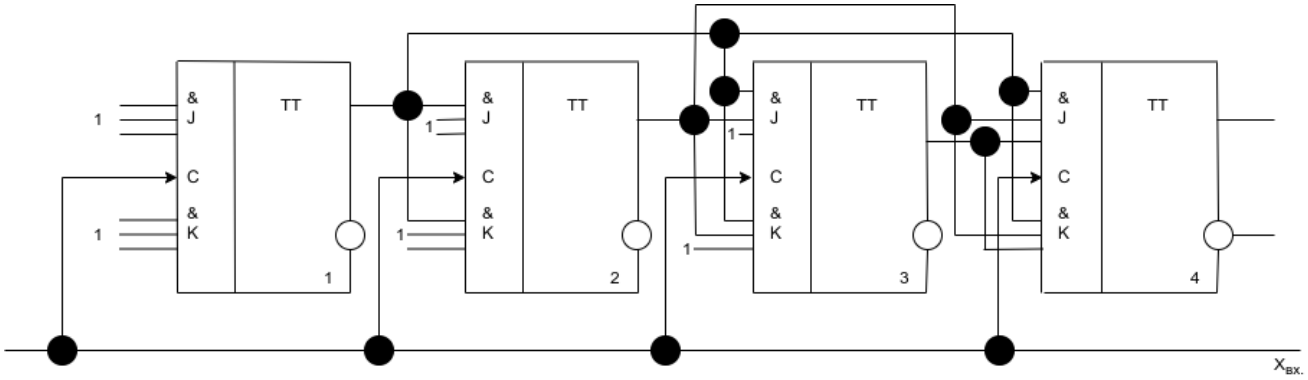


Рис. 6.5. Четырехразрядный синхронный суммирующий счетчик.

Рассматривать работу схемы желательно тоже начиная со старшего разряда. Критерий срабатывания триггеров – наличие 7 единиц, если единиц меньшее количество, то он хранит старое состояние.

Достоинства:

1) Самая быстродействующая схема.

$t_{тр} = \tau_{тр.}$

Недостатки:

1) Чисто параллельный счетчик можно построить только на 4 разряда. Если разрядов больше чем 4, то счетчик разбивается на группы по 4 разряда: внутри группы – параллельный перенос, между – последовательный или сквозной.

Счетчики с произвольным модулем счета.

Это такие счетчики, которые имеют модуль счета $M \neq 2^n$, то есть они не должны досчитывать до своего максимального значения. При этом модуль счета выбирается в следующих пределах: $2^{n-1} \leq M < 2^n$ (для того, чтобы не использовать лишние триггеры).

Для построения таких счетчиков нужно исключить лишнее состояние счетчиков, то есть использовать специальную схему сброса счетчика в ноль.

Рассмотрим двоично-десятичный счетчик на DV-триггерах (M = 10).

Идея построения:

Выбирается количество разрядов (для M=10 – 4 разряда, 4 триггера), затем в схему надо ввести мини дешифратор (на элементе Шеффера, который «увидит» состояние счетчика, равное $M - 1 = 9 = 1001_2$, выработает в обратную связь сигнал сброса, который поступает на все разряды и сбрасывает счетчик в ноль.

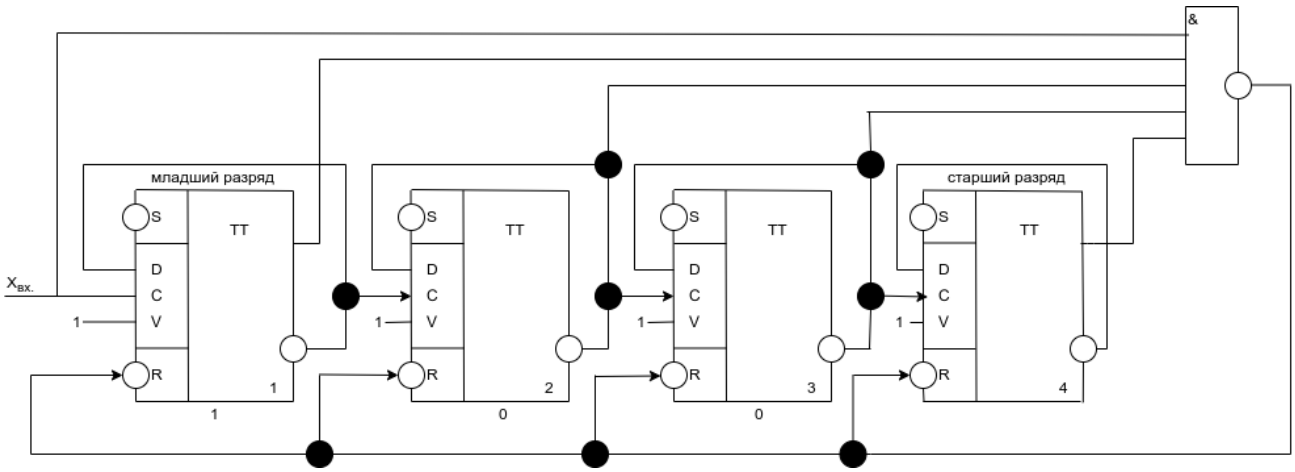


Рис. 6.6. Двоично-десятичный счетчик по модулю M = 10.

Такой счетчик строим на основе асинхронного счетчика с последовательным переносом. На элемент Шеффера подаем четыре единицы с триггеров (с 1 и 4 – с прямого выхода, 2 и 3 – с инверсных выходов).

Так как обратная связь поступает на асинхронные R входы, то сброс счетчика выполняется на 9-м такте. Чтобы это не произошло и сброс был на 10-м такте, элемент Шеффера синхронизируется входным сигналом $X_{вх}$.

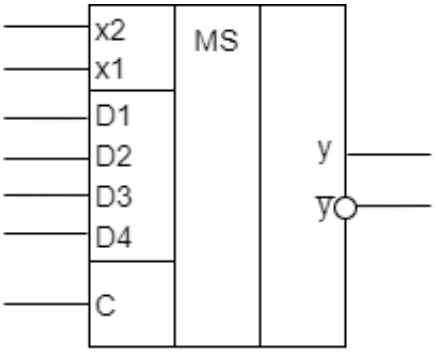
Лекция №7

7.1 Мультиплексоры (MS)

7.1.1. Определение

Мультиплексор (MS) — это коммутатор цифровых сигналов, который в зависимости от кода, поданного на управляющие входы, пропускает на выход входной сигнал только с одного цифрового входа.

7.1.2. Структурное обозначение



Обозначения:

x_1, x_2 — управляющие входы

$D_1 - D_4$ — цифровые (информационные) входы

C — синхровход

y — выход

Рис. 7.1. Мультиплексор.

7.1.3. Функциональная схема

Функциональную схему строим на основе следующей формулы (7.1):

$$y = D_1 * \overline{x_2} * \overline{x_1} \vee D_2 * \overline{x_2} * x_1 \vee D_3 * x_2 * \overline{x_1} \vee D_4 * x_2 * x_1 \qquad (7.1)$$

Схему строим в классическом базисе (рис. 7.2).

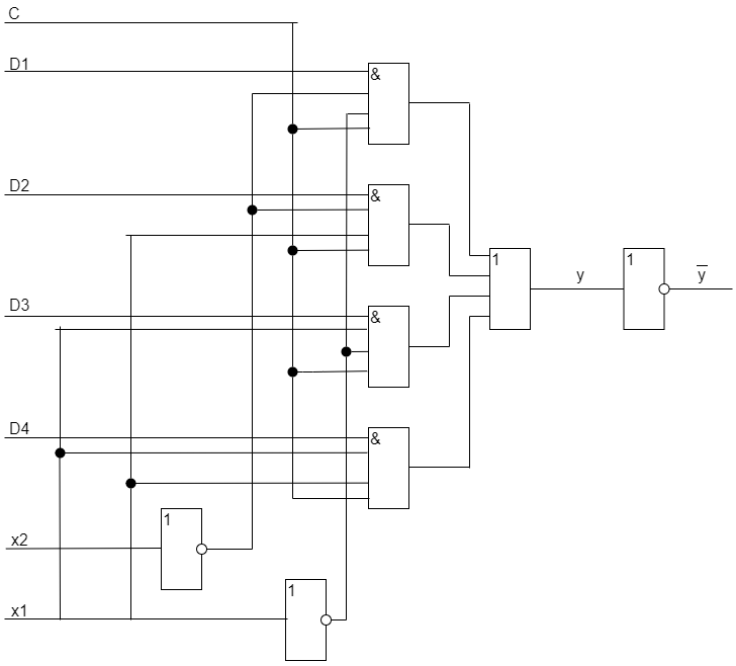


Рис. 7.2. Функциональная схема.

Мультиплексоры используются в вычислительных машинах обычно в качестве управляющей логики, т.к. мультиплексор может выполнять любую логическую операцию.

Таблица 7.1.

Логические операции для MS.

$x1\ x2$	$x1 \wedge x2$	$x1 \vee x2$	$x1 \oplus x2$
0 0	$0 - D1$	$0 - D1$	$0 - D1$
0 1	$0 - D2$	$1 - D2$	$1 - D2$
1 0	$0 - D3$	$1 - D3$	$1 - D3$
1 1	$1 - D4$	$1 - D4$	$0 - D4$

7.2 Сумматоры (SM)

7.2.1. Определение

Сумматор (SM) — это узел машины, предназначенный для выполнения арифметических операций над числами (операндами).

В зависимости от того, в какой системе счисления представлены операнды, сумматоры могут быть:

- двоичные
- десятичные
- троичные
- … и т.д.

В зависимости от числа входов сумматоры могут быть:

- полусумматоры (на 2 входа)
- полные сумматоры (на 3 входа)

По логической структуре SM могут быть:

- комбинационные
- накапливающие
- комбинационно-накапливающие

По организации цепей переноса SM могут быть:

- с последовательным переносом

- со сквозным переносом
- с параллельным переносом

Само суммирование может выполняться на:

- одноразрядных сумматорах
- многоразрядных сумматорах

Само суммирование может выполняться:

- последовательно (начиная с младшего разряда)
- параллельно (все разряды одновременно)
- параллельно-последовательно (SM разбивается на группы: внутри группы - параллельный перенос, между группами - последовательный)

7.2.1. Комбинационный одноразрядный сумматор

Это такой SM, который вырабатывает выходные сигналы суммы и переноса, которые определяются комбинациями цифр слагаемых, поданных одновременно на входы сумматора.

Такой SM не обладает памятью (для запоминания на выходе ставят RS или D-Tr), т.е. при снятии входных сигналов снимаются и выходные сигналы. Это быстродействующий сумматор, т.к. работает за один такт.

Структурные обозначения

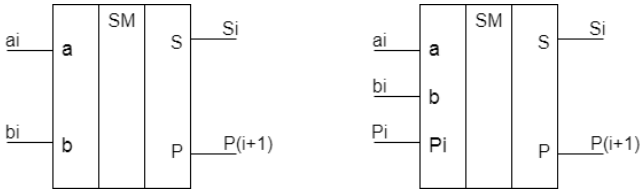


Рис. 7.3. Полусумматор и полный сумматор.

Обозначения:

a_i, b_i - разряды операндов

P_i - перенос из предыдущего разряда

P_{i+1} - перенос в следующий разряд

S_i - значение суммы в разряде

Синтез полусумматора.

Таблица 7.2.

Таблица истинности полусумматора.

A_i B_i	S_i	P_{i+1}
0 0	0	0

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

В ДНФ:

$$S_i = \bar{A}_i * B_i \vee A_i * \bar{B}_i \quad (7.2)$$

$$P_{i+1} = A_i * B_i \quad (7.3)$$

Построим комбинационную схему сумматора в классическом базисе.

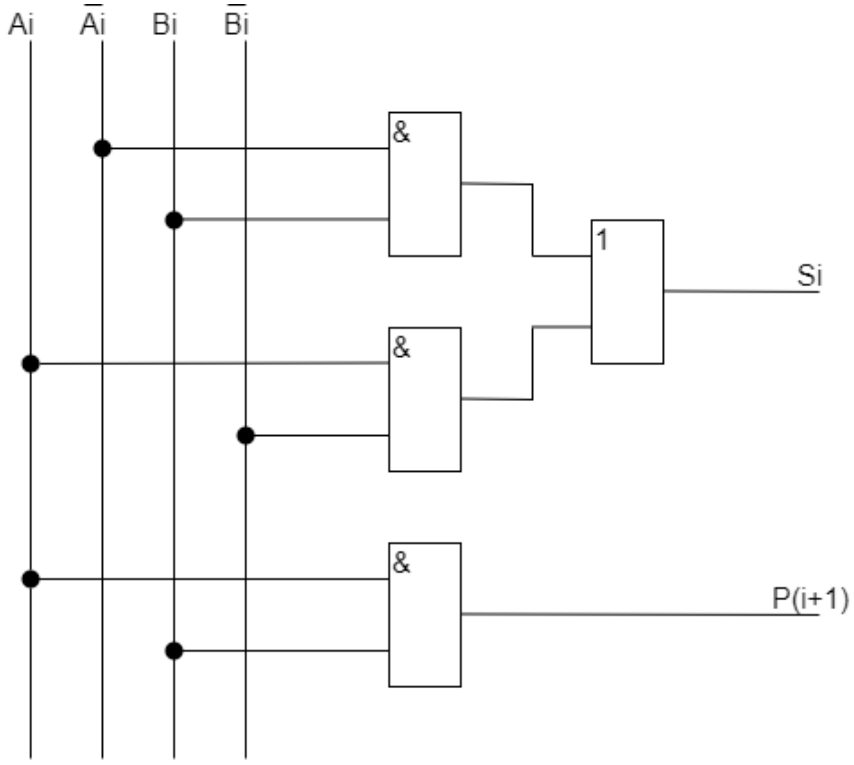


Рис. 7.4. Комбинационная схема сумматора.

Синтез полного одноразрядного сумматора.

Таблица 7.3

Таблица истинности полного одноразрядного сумматора

$A_i\ B_i\ P_i$	S_i	P_{i+1}
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

Формула:

$$S_i(\text{ДНФ}) = \overline{A_i} * \overline{B_i} * P_i \vee \overline{A_i} * B_i * \overline{P_i} \vee A_i * \overline{B_i} * \overline{P_i} \vee A_i * B_i * P_i \quad (7.2)$$

$$P_{i+1}(\text{ДНФ}) = \overline{A_i} * B_i * P_i \vee A_i * \overline{B_i} * P_i \vee A_i * B_i * \overline{P_i} \vee A_i * B_i * P_i \quad (7.3)$$

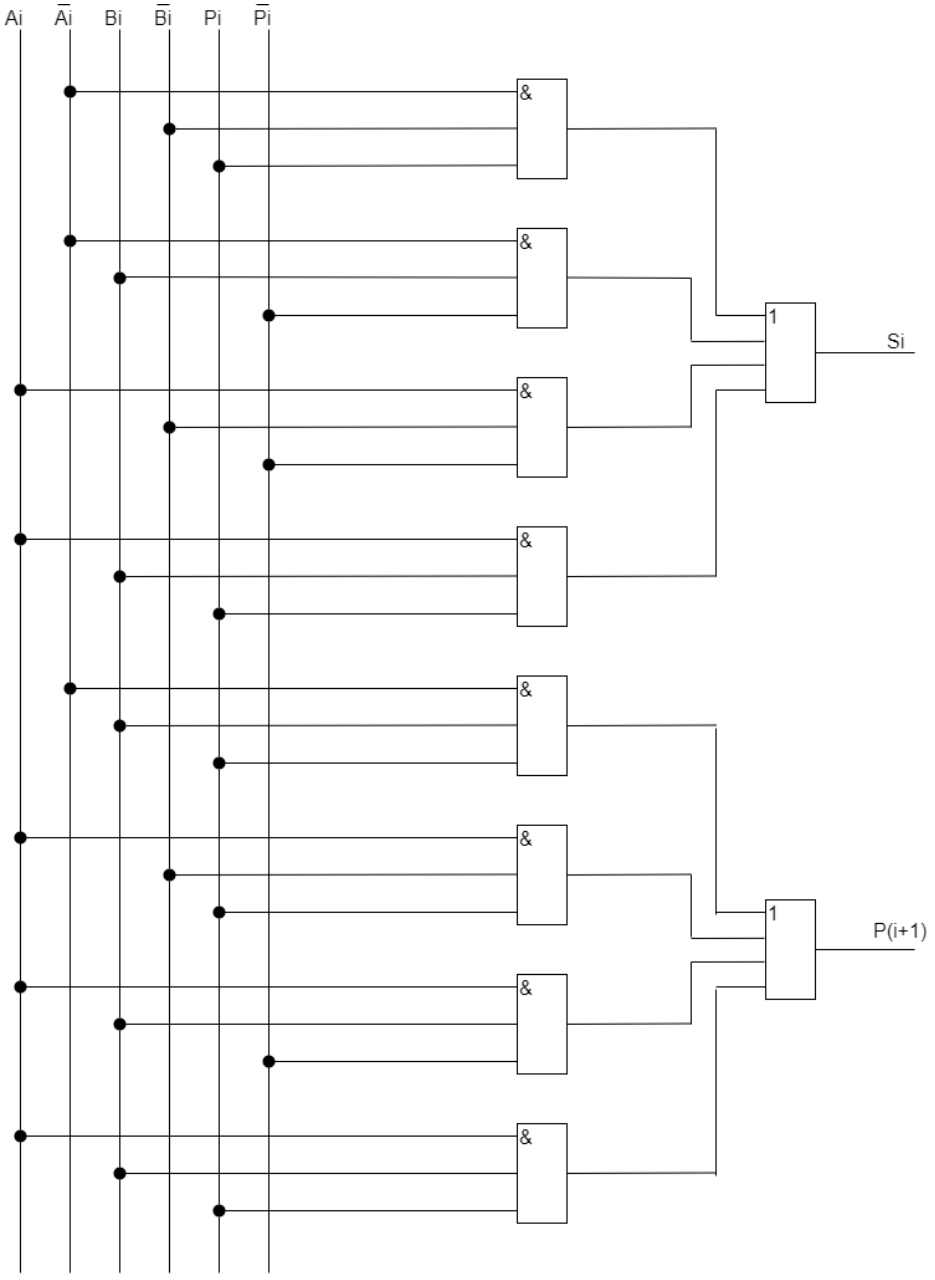


Рис. 7.5. Комбинационная схема полного одноразрядного сумматора.

7.2.2. Накапливающий сумматор

Это такой сумматор, который вырабатывает выходные сигналы суммы и переноса и хранит их значение после снятия входных сигналов. При этом коды слагаемых поступают на вход SM последовательно, один за другим.

Примечание №1.

Раз сумматор хранит значение суммы, то он должен строиться на триггерах (на T-Tr, т.к. именно T-Tr выполняет операцию сложения по модулю 2).

Примечание №2.

Быстродействие этого сумматора в 3 раза ниже, чем у комбинационных, т.к. он работает за 3 такта.

7.2.3. Многоразрядный сумматор

Их построение зависит от типа Ц.В.М.: параллельного она действия или последовательного.

Если Ц.В.М. *последовательного типа*, то сумматор можно построить на одном одноразрядном полном сумматоре, если перенос Pi+1 подавать на вход Pi с задержкой на 1 такт (рис. 7.6).

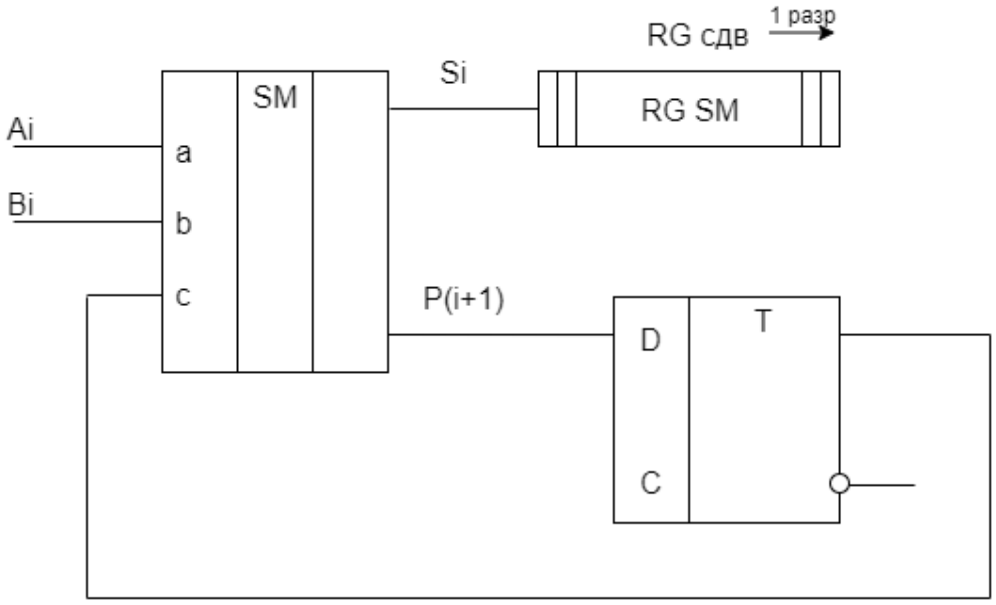


Рис. 7.6. Многоразрядный сумматор на одноразрядном полном сумматоре

На таком SM за n тактов можно просуммировать 2 n-разрядных числа.

Если Ц.В.М *параллельного действия*, то:

1) Сумматор комбинационного типа (рис. 7.7)

Его структура:

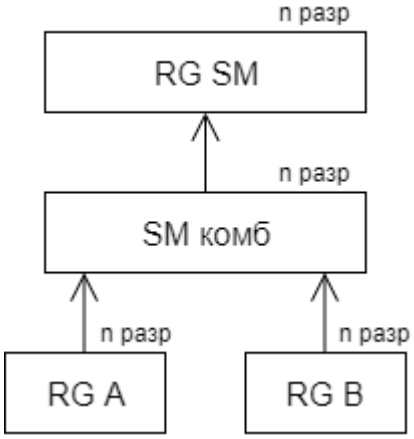


Рис. 7.7. Сумматор комбинационного типа

2) Сумматор накапливающего типа (рис. 7.8)

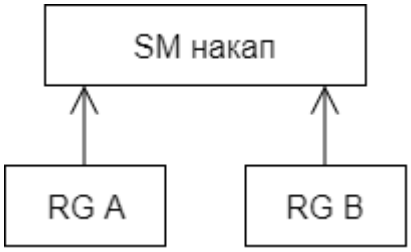


Рис. 7.8. Сумматор накапливающего типа

В таких сумматорах самым сложным является вопрос о распространении переноса.

Переносы могут быть:

- последовательный
- сквозной (у накапливающего SM)
- параллельный

Примечание.

Самое большое количество оборудования требуется для организации параллельного переноса.

УСТРОЙСТВА Ц.В.М.

Лекция №8

8.1 Арифметико-логическое устройство (АЛУ)

8.1.1 Определение

АЛУ — это устройство, предназначенное для выполнения арифметических и логических операций.

АЛУ бывают:

- универсального типа (состоит из одного блока, который выполняет набор арифметических и логических операций)

- специализированного типа (состоит из нескольких блоков, каждый из которых выполняет часть набора операций. Это позволяет параллельно выполнять несколько арифметико-логических операций).

8.1.2 Сложение в Ц.В.М.

1) Сложение с фиксированной запятой.

Выполняется на одно- или многоразрядном SM.

Структурная схема АЛУ для выполнения алгебраического сложения (B±A) (рис. 8.1):

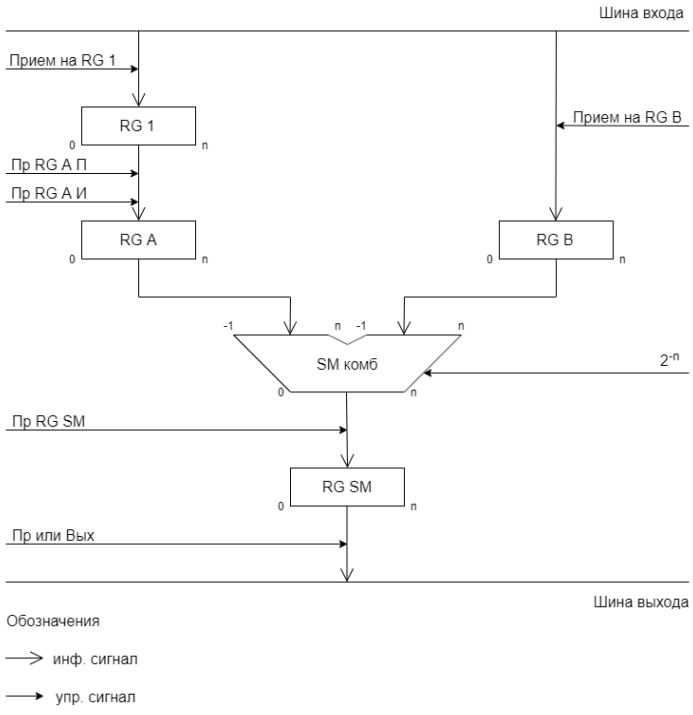


Рис. 8.1. Структурная схема АЛУ для сложения

Слагаемые поступают в АЛУ последовательно, но разным управляющим сигналом, т.к. ОП выдает слагаемые на шину входа последовательно.

В этой схеме слагаемые поступают с шины входа в прямом обычном коде (с первым знаковым разрядом), а сама операция выполняется в модифицированном дополнительном, поэтому при передаче слагаемых на сумматор делается размножение знакового разряда (рис. 8.2).

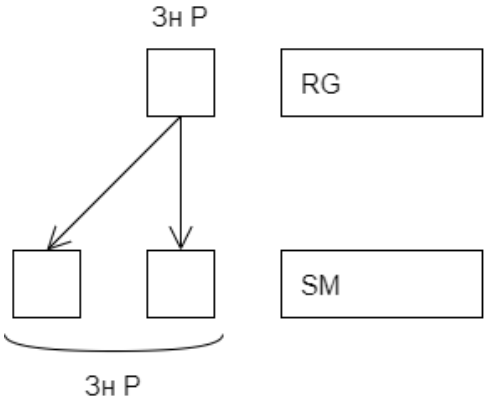


Рис. 8.2. Размножение знакового разряда

Если слагаемое А - положительное, то оно с RG 1 подается прямым кодом на RG A и далее на левое плечо сумматора с размножением знакового разряда.

Если слагаемое А - отрицательное, то оно должно быть на SM в модифицированном дополнительном коде. Для этого оно передается с RG 1 на RG A инверсным кодом (обратным), а единица, которая прибавляется на SM. С выхода SM на RG SM результат передается обычным кодом.

Работу структурной схемы обычно описывают на специальном языке, который называется ЯМП (язык микропрограммирования) — это алголо-подобный язык (ALGOL).

НАЧАЛО

ПР RG B: RG B := ШИ ВХ <прием операнда B>

ПР RG B: RG 1 := ШИ ВХ <прием операнда A>

ПР SM: ЕСЛИ сложение ТО

ПР RG A: RG A := RG 1 ИНАЧЕ

ПР RG A: RG A := $\overline{\text{RG 1}}$

СУММА: ЕСЛИ сложение ТО

ПР RG SM: RG SM := RG A + RG B

ИНАЧЕ RG SM := RG A + RG B + 2^{-n}

ПР ИЛИ ВХ: ПР ИЛИ ВХ: RG SM <вывод результата>

КОНЕЦ

2) Сложение чисел с плавающей запятой (8.1).

$$\begin{array}{r} 350 = 0{,}35 * 10^3 \\ +\ 35 = 0{,}35 * 10^2 \\ \hline 385 = 0{,}385 * 10^3 \end{array} \qquad (8.1)$$

Алгоритм:

- выравнивание порядков в сторону большего
- сложение мантисс по правилу сложения чисел с фиксированной запятой
- присваивание общего порядка
- нормализация результата
- округление результата

8.1.3. Правила машинного округления

1) Для результата, полученного на сумматоре:

К SM добавляется 1 или несколько дополнительных разрядов и к старшему прибавляем единицу. После этого дополнительные разряды отбрасываются. Если в старшем дополнительном разряде была 1, то это округление с избытком, если 0, то округление с недостатком.

2) Для результата, полученного на RG, например, при делении:

В младший разряд RG принудительно ставится единица. При этом, если в младшем разряде была 1, то округление с недостатком, если 0 - то с избытком.

Лекция №9

9.1. Умножение в ЦВМ

9.1.1. Способы умножения в ЦВМ

		1	0	1	
x		0	1	1	
		1	0	1	
	1	0	1		
0	0	0			
0	1	1	1	1	

	1	0	1		
x	0	1	1		
	0	0	0		
		1	0	1	
			1	0	1
0	1	1	1	1	

Первый способ делится на 2 способа:

1. SM
2. Множ.

Второй способ делится на 2 способа

1. SM
2. Множ.

Из этого примера видно, что существует 4 способа умножения.

9.1.2 Способ №1 умножения в ЦВМ

Младший разряд вперёд со схемой RG множителя и RG множимого (рис. 9.1)

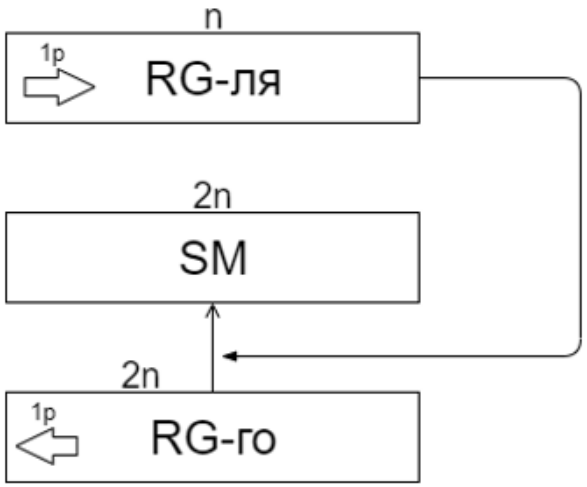


Рис. 9.1. Структура первого способа умножения в ЦВМ.

9.1.3. Способ №2 умножения в ЦВМ.

Младший разряд вперёд со сдвигом SM и RG множителя (рис 9.2).

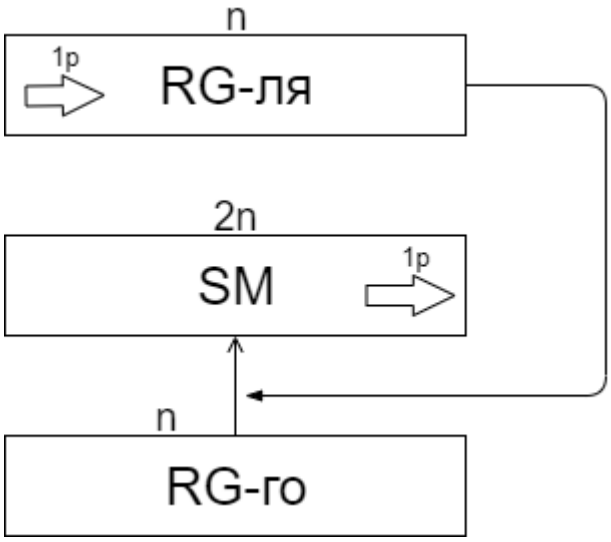


Рис.9.2. Структура второго способа умножения в ЦВМ.

9.1.4. Способ №3 умножения в ЦВМ

Старший разряд вперёд со сдвигом RG множимого и RG множителя (рис. 9.3).

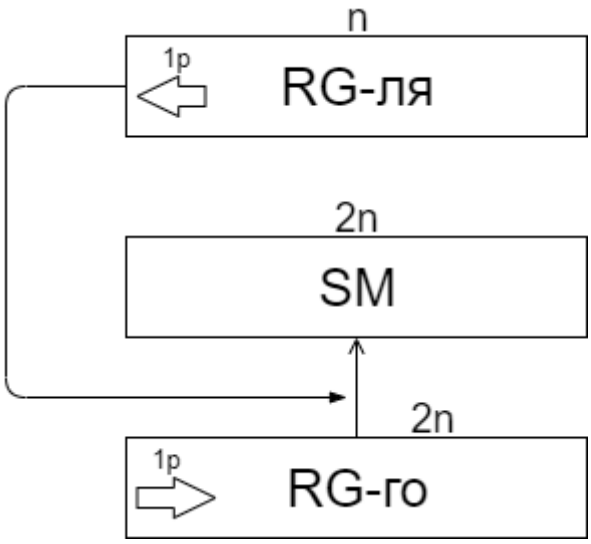


Рис. 9.3. Структура третьего способа умножения в ЦВМ.

9.1.5. Способ №4 умножения в ЦВМ.

Старший разряд вперёд со сдвигом SM и RG множителя (рис. 9.4).

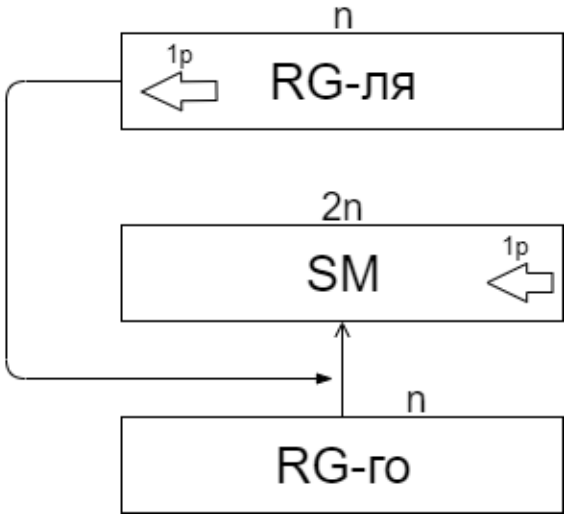


Рис. 9.4. Структура четвёртого способа умножения в ЦВМ.

Первый и третий способы позволяют выполнить операции с большим быстродействием (проигрывают в затратах на оборудование).

Второй и четвёртый способы экономят оборудование (проигрывают в быстродействии).

При выборе метода надо учитывать 2 критерия:

— экономия оборудования

— повышение быстродействия

Допустим, выбирают критерий экономии оборудования, т.е. выбираем 2-й или 4-й способ.

Примечание:

Способы «Старший разряд вперёд» в машинах не используют, т.к. при сбое потерянными оказываются старшие разряды множителя, т.е. операцию требуется повторить заново, в то время как при умножении младшими разрядами вперёд операцию можно продолжить, хотя и с потерей точности, т.к. теряются незначащие младшие разряды множителя.

Правила умножения по способу №2.

Вся операция умножения разбивается на циклы, число которых равно числу разрядов множителя.

Каждый цикл состоит из двух тактов:

1. УУУ (устройство управления умножением) анализирует младший разряд множителя: если он равен 1, то множимое передаётся на SM, где складывается с частичным произведением. Если младший разряд равен 0, то в SM передаётся множимое, равное 0 (фиктивное сложение).

2. SM и RG-ля сдвигаются вправо на 1 разряд.

Примечание:

После получения результата делается дополнительный цикл округления по правилу 1.

Структурная схема АЛУ для умножения чисел с фиксированной запятой (рис. 9.5).

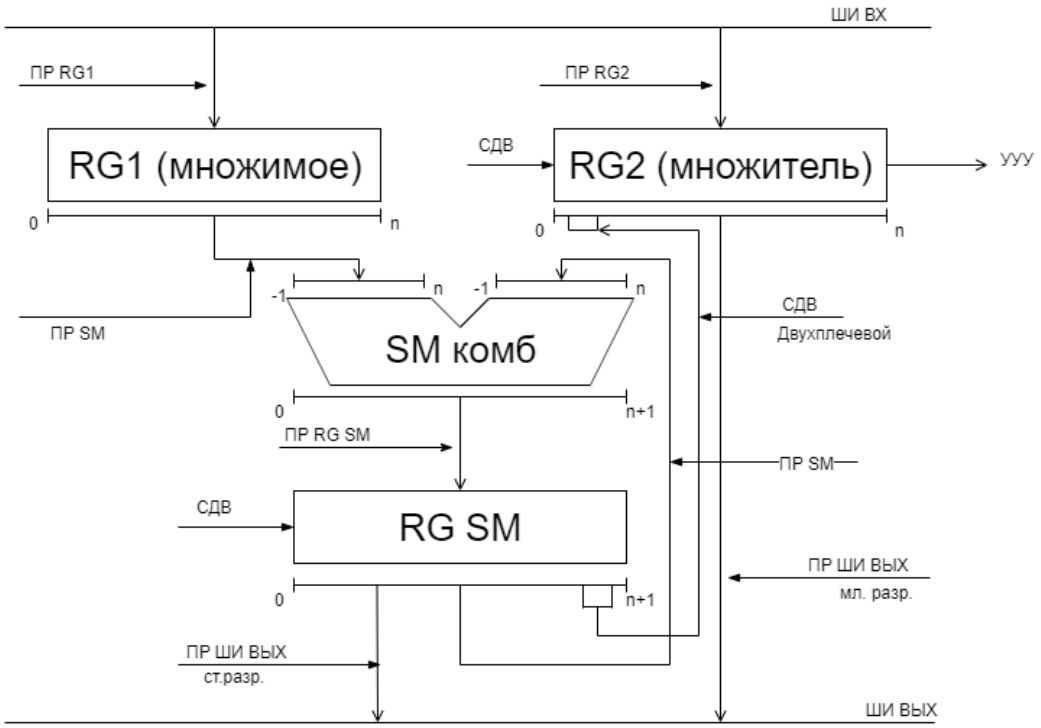


Рис. 9.5. Структурная схема АЛУ для умножения чисел с фиксированной запятой

Т.к. эта схема работает в модифицированном коде, то SM должен иметь 2 знаковых разряда. Поэтому, при переходе на плечи SM множимого и частичного произведение делается разложение знаковых разрядов (см. АЛУ для сложения).

Примечание:

Эта схема АЛУ ещё более экономна, чем классический способ, т.к. в классическом способе SM имеет 2n разрядов, а в данной схеме – n+1, т.е. SM

имеет столько разрядов, сколько нужно для собственно сложения (n разрядов). Ещё n разрядов нужно для сдвига. В качестве таких разрядов можно использовать старшие освободившиеся RG-ля. Поэтому:

— если операция выполняется с одинарной точностью, то результат на шине выхода снимается только с RG SM.

— если операция выполняется с двойной точностью, то старшие разряды результата снимаются с RG SM, а младшие – с RG-ля.

Процедура округления на схеме не показана.

Таблица 9.1

Пример машинного умножения 0,01010 x 0,1101

№y	№t	Арифметически действия	Примечание	RG
1	1	0, 0 0 0 0 0 0 0 0	(SM) := 0	1 1 0 1
		+ 0, 1 0 1 0 0 0 0 0		
		0, 1 0 1 0 0 0 0 0	1-я ∑	
	2	0, 0 1 0 1 0 0 0 0	1-ый сдвиг	
2	1	+ 0, 0 0 0 0 0 0 0 0		
		0, 0 1 0 1 0 0 0 0	2-я ∑	1 0 1
	2	0, 0 0 1 0 1 0 0 0	2-ой сдвиг	→1p
3	1	+ 0, 1 0 1 0 0 0 0 0		
		0, 1 1 0 0 1 0 0 0	3-я ∑	0 1
	2	0, 0 1 1 0 0 1 0 0	3-ий сдвиг	→1p
4	1	+ 0, 1 0 1 0 0 0 0 0		
		1, 1 0 0 0 0 1 0 0	4-я ∑	1
	2		4-ый сдвиг	→1p
5		0, 1 0 0 0 0 0 1 0	Округление	
		+ 1	с	
		0, 1 0 0 0 1 0 1 0	недостатком	

Считаем, что машины выполняют операцию умножения с одинарной точностью, поэтому 5, 6, 7, 8 разряды считают дополнительными (1-4 – основные), и округление делается прибавлением 1-цу к 5-му разряду.

Примечание:

Если операция выполняется с двойной точностью, то к SM надо реально добавлять 9-ый разряд.

9.1.6. Умножение чисел с плавающей запятой

Алгоритм:

1. Определить знак произведения (знак мантиссы)
2. Перемножить модули мантис сомножителей по первому умножению числом с фиксированной запятой
3. Определить порядок результата. (сложить разряды в дополнительном или обратном коде)
4. Нормализация результата
5. Округление результата

9.1.7. Способы ускорения умножения

Статистика показывает, что машины I-го поколения операцию умножения на классе вычислительных задач занимала ~70% машинного времени. Поэтому вопрос ускорения операции умножения – очень актуальный.

Существует 2 метода ускорения умножения:

1. Аппаратно-логический (убирается фиктивное сложение это ускоряет операцию на 40%, теоретически, этим способом можно ускорить на 50%, а оставшиеся 10% ускоряется … методами: одновременным умножением на 2, 4, 8 разрядов).
2. Аппаратный метод. Этот метод требует огромных затрат на аппаратуру, строится специальное дерево сумматоров. Такой метод позволяет ускорить умножение до 2-х, а иногда до 1 такта.

Лекция №10

10.1. Деление в ЦВМ

10.1.1. Введение

Деление несколько более сложная операция, чем умножение, но базируется на тех же принципах. Если умножение выполняется путем многократных сдвигов и сложений, то деление, будучи операцией обратной умножению, — путем многократных сдвигов и вычитаний. Основу составляет общепринятый способ деления с помощью операций вычитания (сложения) и сдвига.

Задача деления числа A на число B сводится к вычислению частного Q и остатка R:

$$\frac{A}{B} = Q + \frac{R}{B}$$

$$A = QB + R; \quad R < B$$

Частное от деления 2n-разрядного числа на n-разрядное может содержать более чем n разрядов. В этом случае возникает переполнение, поэтому перед выполнением деления необходима проверка. Переполнения не будет, если число, содержащееся в старших n разрядах делимого, меньше делителя (для целых чисел) или делимое меньше делителя (для дробных чисел). Помимо этого требования, перед началом операции необходимо исключить возможность ситуации деления на 0.

Существуют 2 способа деления:

— с восстановлением остатка

— без восстановления остатка

10.1.2. Деление с восстановлением остатка (способ – плохой)

Этот метод требует в каждом цикле деления то два такта (вычитание, сдвиг), то три такта (вычитание, сдвиг, восстановление остатка), т.е. этот метод проигрышный по быстродействию (дополнительный такт восстановление остатка) и по затратам оборудования (усложняется устройство управления

делением для обслуживания третьего такта). Поэтому ни в одной машине этот способ не используется.

Алгоритм очень похож на общепринятый способ деления столбиком. Данный алгоритм может быть описан следующим образом.

1. Исходное значение частичного остатка (ЧО) полагается равным старшим разрядам делимого.
2. Из ЧО вычитается делитель и анализируется знак остатка.
3. Если остаток положительный, то деление невозможно, формируется признак переполнения и процесс завершается, в противном случае ЧО восстанавливается путем прибавления делителя и деление продолжается.
4. Частичный остаток сдвигается на один разряд влево, а в освобождающийся при сдвиге младший разряд ЧО заносится очередная цифра делимого.
5. Из сдвинутого ЧО вычитается делитель и анализируется знак результата вычитания.
6. Очередная цифра модуля частного равна единице, когда результат вычитания положителен, и нулю, если отрицателен. В последнем случае ЧО восстанавливается до того значения, которое было до вычитания.
7. Пункты 4–6 последовательно выполняются для получения всех цифр модуля частного.

10.1.3. Без восстановления остатка

Таких методов много, все они:

— искусственные (т.е. не универсальные)

— «привязаны» к определённому коду

Рассмотрим простейший из таких методов (привязанный к обычному дополнительному коду)

Алгоритм:

Чтобы определить очередную цифру нечётного, надо:

1. Сдвинуть текущий остаток влево на 1 разряд

2. Прибавить к нему (остатку) делитель, со знаком, противоположному знаку текущего остатка.

3. Если остаток положительный, то в разряд нечётных записывается единица, а если остаток отрицательный, то в разряд нечётного ставится ноль.

Примечание 1:

Сдвиг и сложение делаются столько раз, сколько нужно получить разрядов у нечётного.

Примечание 2:

Первый раз модуль делимого считается нулевым остатком, уже сдвигается на 1 разряд влево.

Примечание 3:

Округление делается по правилу 2.

Пример:

x/y (x<y)

x=0,101 – делимое

y=0,110 – делитель

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

	1,	1	1	0	
+	0,	1	1	0	
~~1~~	0,	1	0	0	

	1,	0	0	0	
+	0,	1	1	0	
	1,	1	1	0	

$$
\begin{array}{rr}
 & 1,\ 1\ 0\ 0 \\
+ & \\
 & 0,\ 1\ 1\ 0 \\
\hline
\cancel{1} & 0,\ 0\ 1\ 0
\end{array}
$$

Если бы делилось округление, то оно было бы с недостатком.

10.1.4. Алгоритм деления чисел со знаком

Как и в случае умножения, деление чисел со знаком может быть выполнено путем перехода к абсолютным значениям делимого и делителя, с последующим присвоением частному знака «плюс» при совпадающих знаках делимого и делителя либо «минус» — в противном случае.

Деление чисел, представленных в дополнительном коде, можно осуществлять, не переходя к модулям. Рассмотрим необходимые для этого изменения в алгоритме без восстановления остатка. Так как делимое и делитель не обязательно имеют одинаковые знаки, то действия с частичным остатком (прибавление или вычитание B) зависят от знаков частичного остатка и делителя. Если эти знаки совпадают, то в очередной итерации деления производится вычитание делителя, а очередной цифрой частного будет 1. При разных знаках ЧО и делителя последний прибавляется к ЧО, и очередная цифра частного — 0.

10.1.5. Коррекция результатов деления

По завершении деления может понадобиться коррекция результата, заключающаяся в увеличении частного на единицу. Такая коррекция производится в следующих случаях:

$A > 0$ и $B < 0$;

$A < 0$, $B > 0$ и $R \neq 0$;

$A < 0$, $B < 0$ и $R = 0$.

При стандартном определении операции деления частное Q и остаток R отвечают отношению $A = Q \times B + R$, где остаток от деления $0 \leq |R| < |B|$. При таком определении возможны два вида остатка. Например, результат деления −42 на −5 может быть представлен как $-42 = 9 \times (-5) + 3$ или $-42 = 8 \times (-5) + (-2)$, то есть остаток может быть 3 или –2. Для определенности в ВМ принято, что

остаток всегда приводится к положительному числу, то есть если по завершении деления он отрицателен, к нему следует прибавить модуль делителя.

10.1.6. Структурная схема АЛУ для деления чисел с фиксированной запятой

Специально АЛУ для деления не делают, а используют АЛУ для умножения, при этом оно несколько усложняется: сдвиговые RG делают реверсивными, т.к. при умножении требуется сдвиг вправо, а при делении – влево.

10.1.7. Методы ускорения целочисленного деления

Следует отметить, что операция деления предоставляет не слишком много путей для своей оптимизации по времени. Тем не менее определенные возможности для убыстрения деления существуют. В ряде случаев эффект достигается применением алгоритмов, сокращающих число итераций в процедуре деления. Ускорение деления может быть достигнуто и за счет более совершенной аппаратной реализации операции. Наилучшие результаты обычно дает сочетание обоих этих подходов.

Наиболее распространенные методы ускорения операции деления основаны на применении алгоритмов, где частное представляется в избыточной системе счисления. Очередная цифра частного в избыточной системе счисления, в зависимости от базы этой системы, соответствует двум или более цифрам в двоичном представлении частного, и для вычисления нужного количества двоичных цифр частного и остатка требуется меньше итераций. В то же время реализация такого подхода ведет к усложнению аппаратуры делителя, в частности надстраивается логика определения операции, выполняемой в очередной итерации. Для этой цели в состав устройства деления включается специальная память, хранящая таблицу для определения необходимых действий (в зависимости от текущей комбинации цифр в частичном остатке и делителе). Тем не менее выигрыш в быстродействии оказывается решающим моментом. Так, в микропроцессорах Pentium при делении мантисс чисел с плавающей запятой используется алгоритм SRT с базой 4 (Radix-4), то есть частное сначала

вычисляется с использованием цифр –2, –1, 0, 1, 2 с последующим преобразованием результата к стандартному двоичному представлению. В этом варианте выбор очередной цифры частного производится с помощью таблицы, состоящей из отдельных секций. Конкретную секцию определяют четыре старшие цифры делителя (после его нормализации). Входом в секцию служат шесть старших цифр частичного остатка. ЧО в каждой итерации сдвигается не на один, а на два разряда, то есть число итераций сокращается вдвое. В микропроцессорах Nehalem и Penryn фирмы Intel деление как целых чисел, так и мантисс чисел с плавающей запятой реализуется блоком, получившим название Fast Radix-16 Divider. Поскольку используется система счисления с основанием 16, блок способен обрабатывать сразу 4 бита за такт, благодаря чему новые микропроцессоры выполняют операции деления целых и вещественных чисел примерно в два раза быстрее, чем в рассмотренном выше варианте, в которых использовалось основание 4 (Radix-4) и за один такт обрабатывались два бита.

10.1.8. Деление чисел с плавающей запятой

Алгоритм:

— Определение знака частного (сложение знакового разряда по mod2)

— Деление мантиссы делимого и делителя по правилу деления чисел с фиксированной запятой

— округление порядка результата. (сложение в дополнительном или обратном коде)

— Нормализация результата

— Округление

10.2. Некоторые методы контроля работы АЛУ

10.2.1. Методы контроля

Виды неисправностей в ЦВМ

1. Отказ (элементы чисел из строя, например из-за короткого замыкания)

2. Систематический сбой (Возникает из-за отклонения параметром элемента от номиналов из-за нестабильности питания сети)

3. Случайный сбой (длится очень короткое время, возникает из-за сигналов в электрических цепях самого элемента)

Существует два метода контроля:

— программный

— аппаратный

10.2.2. Программный метод

Существует 2 способа этого метода:

1. Тестовый контроль (делается периодически, обрабатывает отказ и систематический сбой)

2. Двойной-тройной просчёт (программа или часть её просчитывается первый раз, затем все поля памяти суммируется с циклическими переносом и получается первая контрольная сумма k∑1. Далее восстанавливается программа к первоначальному виду, просчитывается второй раз и получается k∑2. Если две контрольные суммы не совпали, то делается третий просчёт)

Считается универсальным, т.к. по совокупности обнаруживает все три вида неисправности.

Лекция №11

11.1. Аппаратный контроль

11.1.1 Общие сведения

Контролировать в машинах нужно все действия, включая хранение информации. Особенно:

- передачу информации с RG на RG (с устройства на устройство)
- выполнение арифметических операций

Контроль передачи информации.

Существует 2 способа контроля:

- контроль на четность/нечетность
- контроль с помощью кода Хэмминга

11.1.2. Контроль на четность/нечетность

Этот вид контроля обнаруживает только одиночную ошибку или нечетное число ошибок, но не исправляет ее. Статистика показывает, что в 70% возникших неисправностей (сбоев) имеет место быть одиночная ошибка.

К RG добавляется один дополнительный разряд (контрольный). Его значение выбирается таким образом, чтобы во всем коде, включая контрольный разряд, было четное число единиц (для контроля на четность).

Синтез элемента контроля.

Таблица 11.1.

Таблица истинности элемента контроля.

a_i f_{i-1}	f_i	$\overline{f_i}$
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

f_i – функция четности с предыдущим разрядом.

$$f_i = \overline{\overline{a_i} * f_{i-1} \vee a_i * \overline{f_{i-1}}} = \overline{\overline{\overline{a_i} * f_{i-1}} \,\&\, \overline{a_i * \overline{f_{i-1}}}} \qquad (11.1)$$

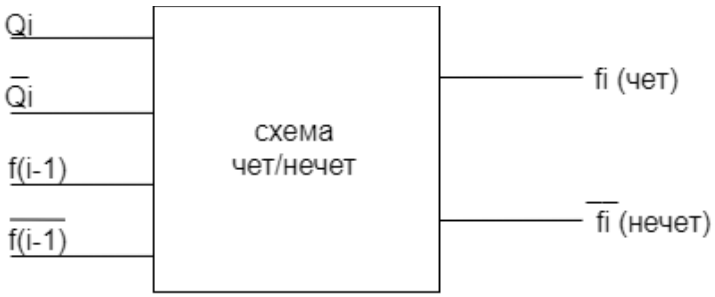
Т.к. значение разряда поступает с триггеров RG, то

$$f_i = \overline{\overline{Q_i} * f_{i-1}} \,\&\, \overline{Q_i * \overline{f_{i-1}}} \tag{11.2}$$

Тогда

$$\overline{f_i} = \overline{\overline{Q_i} * \overline{f_{i-1}}} \,\&\, \overline{Q_i * f_{i-1}} \tag{11.3}$$

То есть схема свертки имеет вид (рис. 11.1):



сх. свертки

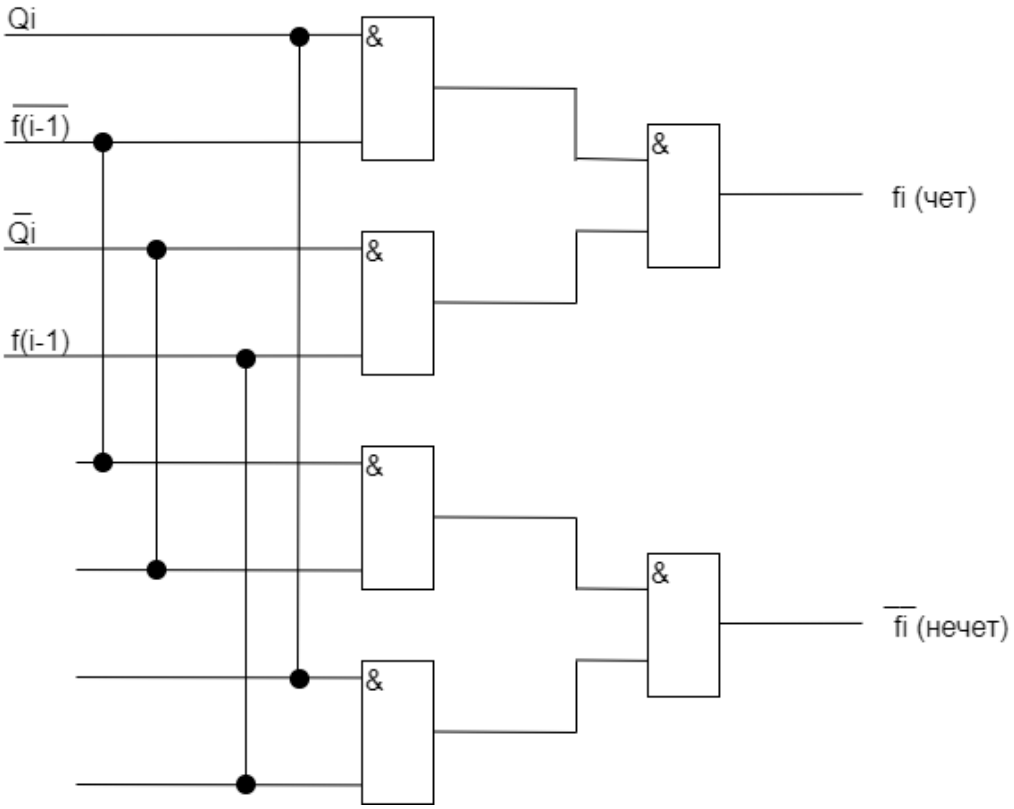


Рис. 11.1. Схема свертки.

В многоразрядных регистрах используются:

А) последовательная свертка

Б) пирамидальная свертка

Рассмотрим *последовательную свертку* для четырехразрядного регистра на RS-Tr (рис. 11.2).

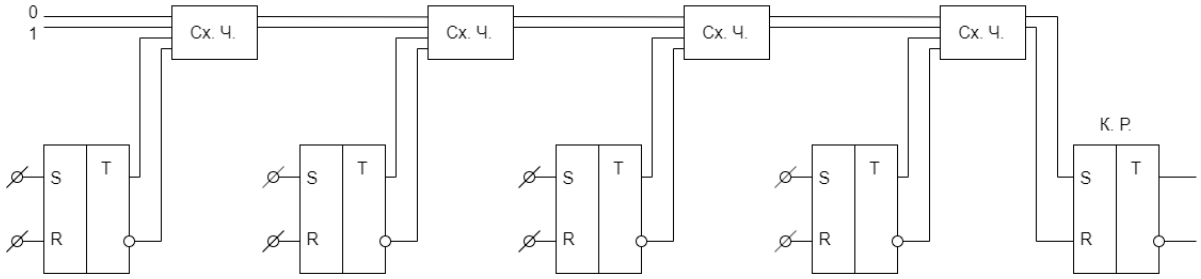


Рис. 11.2. Последовательная свертка.

Такая схема захватывает контрольный разряд, что нехорошо, так как контрольный разряд тоже бывает неисправен.

Пирамидальная свертка может охватывать контрольный разряд, а может не охватывать. Если охватывается контрольный разряд, то схем свертки на одну больше, чем в схеме без охвата.

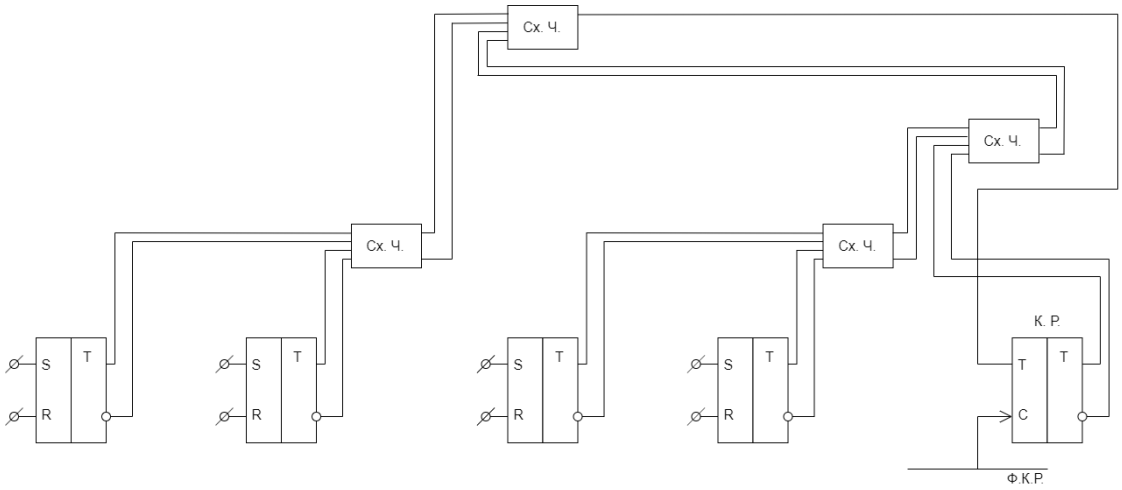


Рис. 11.3. Пирамидальная свертка.

В этой схеме (рис. 11.3) контроль делается на четность, с охватом контрольного разряда, при этом значение контрольного разряда от старого значения RG. После подачи нового значения срабатывает схема свертки и устанавливает контрольный разряд.

Ф.К.Р. – шина разрешения формирования контрольного разряда

Примечание.

В качестве контрольного разряда ставится T-Tr, а не RS-Tr, как в случае пирамидальной свертки без охвата контрольного разряда.

11.1.3. Контроль по коду Хэмминга

Код Хэмминга обнаруживает и исправляет одиночную ошибку и только обнаруживает двойную ошибку. Для построения кода Хэмминга надо последовательно ответить на следующие вопросы:

1) Сколько нужно ввести контрольных разрядов?
2) На какие места в общем коде их поставить?
3) Как выбирается значение каждого контрольного разряда?
4) Как обнаруживается и как исправляется одиночная ошибка?
5) Как обнаруживается двойная ошибка?

Отвечаем на эти вопросы:

1) Количество контрольных разрядов определяется из неравенства (11.4)

$$2^{m-1} \leq n + 1 < 2^m \qquad (11.4)$$

$$n = k + m$$

$$\begin{cases} k - \text{количество информационных разрядов} \\ m - \text{количество контрольных разрядов} \end{cases}$$

Пусть количество информационных разрядов: k = 4, тогда m = 3 (в результате решения неравенства (11.4)). То есть длина общего кода составляет n = 7 разрядов.

2) Пронумеруем эти разряды в двоичной системе и составим из них группы (число групп = число столбцов, в группы входят только те номера разрядов, в которых есть «1»).

1 – 0 0 1	В первую группу входят те номера разрядов, в которых стоит
2 – 0 1 0	«1», то есть:
3 – 0 1 1	1 группа: 1, 3, 5, 7
4 – 1 0 0	2 группа: 2, 3, 6, 7
5 – 1 0 1	3 группа: 4, 5, 6, 7
6 – 1 1 0	
7 – 1 1 1	

Заметим, что первый, второй и четвертый разряды входят только в одну группу, поэтому их можно использовать в качестве контрольных, т.е. номер контрольного разряда определяется как 2^n, где n = 0, 1, 2, 4...

3) С этого момента требуется знать информационный код, пусть он = 1010. Строим код Хэмминга для этого информационного кода.

		1		0	1	0

Значение контрольного разряда есть значение четности в своей группе (11.5).

$$1\text{гр: } x_1 \oplus 1 \oplus 0 \oplus 0 = 0 \rightarrow x_1 = 1 \qquad (11.5)$$
$$2\text{гр: } x_2 \oplus 1 \oplus 1 \oplus 0 = 0 \rightarrow x_2 = 0$$
$$3\text{гр: } x_4 \oplus 0 \oplus 1 \oplus 0 = 0 \rightarrow x_4 = 1$$

В итоге получаем код Хэмминга:

1	0	1	1	0	1	0

4) Передаем этот код с ошибкой в третьем разряде (рис. 11.4).

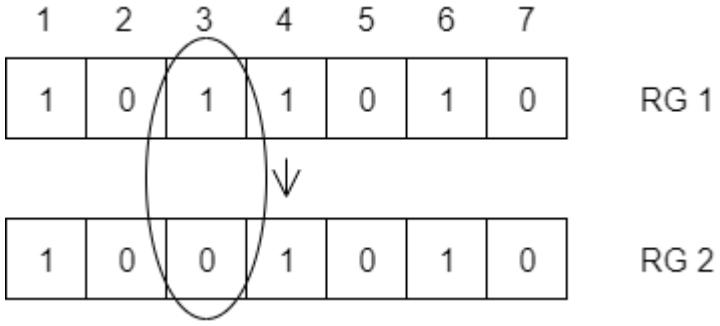


Рис. 11.4. Передача кода с ошибкой.

Для обнаружения ошибки в коде строим группы по mod 2 и определяем значение четности в контрольных разрядах (11.6). Номер неисправного разряда читается снизу-вверх.

$$1\text{гр: } 1 \oplus 0 \oplus 0 \oplus 0 = 1 \qquad (11.6)$$
$$2\text{гр: } 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$3\text{гр: } 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

011 – это значит, что ошибка в третьем разряде. Для исправления ошибки: в третий разряд записываем его инверсное значение согласно схеме на рисунке 11.5.

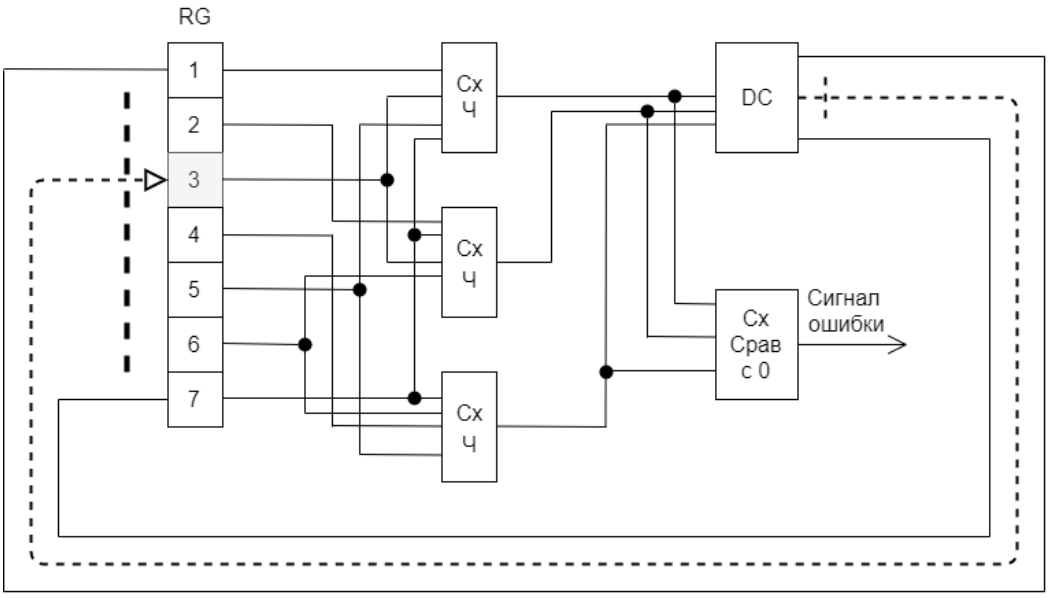


Рисунок 11.5. Структурная схема для к. Хэмминга.

5) Для обнаружения дойной ошибки надо ввести еще один контрольный разряд с номером, равным 0, в который записывается значение четности всего кода.

Если происходит одиночная ошибка, то групповой контроль ее обнаруживает, и общая четность ее тоже обнаруживает. Если происходит двойная ошибка, то общая четность ее не обнаруживает, а групповой контроль – обнаруживает, следовательно несоответствие результатов этих двух видов контроля и является признаком обнаружения двойной ошибки. Но при этом номер неисправного разряда, который показывает групповой контроль, является неверным.

Лекция №12

12.1. Устройства управления (УУ)

12.1.1. Определение

Это устройство предназначено для выбора очередной машинной команды и для выработки последовательности управляющих сигналов.

Существует 2 метода построения УУ:

— микропрограммное управление

— управление с жёсткой логикой (схемой управления)

12.1.2. Микропрограммное управление

Идея метода:

Для каждой машинной команды составляется своя микропрограмма, которая обычно состоит из стольких микрокоманд, сколько нужно вырабатывать управляющих сигналов для АЛУ.

Все микропрограммы хранятся в специальной области ОП, которая называется ПМК (пакет микрокоманд).

Структурная схема:

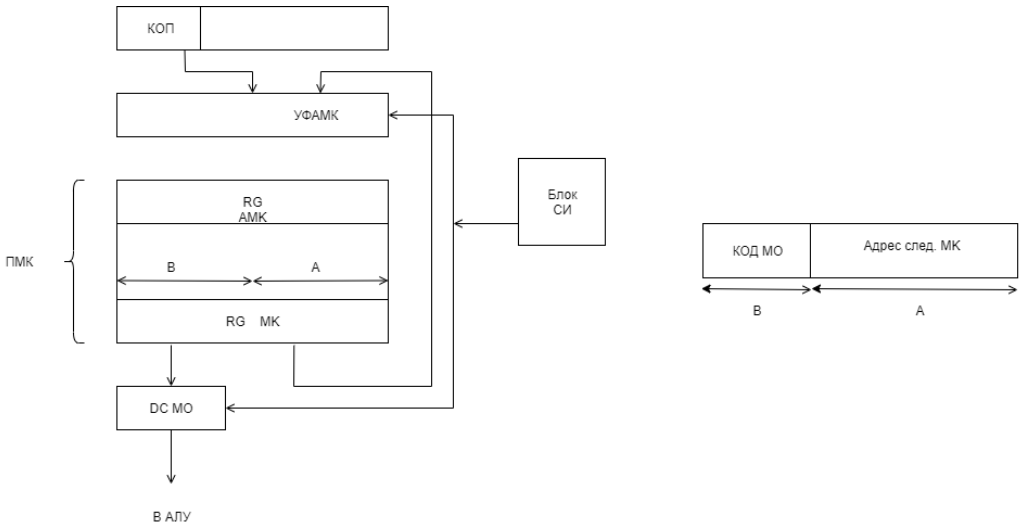


Рис. 12.1. Структурная схема микропрограммного управления.

Условные обозначения:

КОП – код операций машинной команды

УФАМК – узел формирования адреса микрокоманды (состоит из таблицы соответствия: каждому КОЛу машинной команды соответствует адрес первой

микрокоманды, соответствующий микрооперациями. Адрес следующей микрокоманд из микрооперации находится в поле А МК)

RG АМК – регистр адреса микрокоманд

RG МК – регистр микрокоманд

ПМК – память микрокоманд

DC МО – дешифратор микроопераций

Достоинства:

Просто добавляется новая машинная команда (для этого пишется новая микропрограмма (ещё одна) и корректируются таблица соответствия в УФАМК).

Недостатки:

Это устройство имеет сравнительно низкое быстродействие.

12.1.3. Устройство управления с жёсткой логикой

Это есть логические схемы, вырабатывающие распределённое во времени управляющие сигналы.

Структурная схема:

Рис. 12.2. Структурная схема устройства управления с жёсткой логикой.

Счётчик тактов выбирается по самой «длинной» (по числу тактов) команде. Сами управляющие сигналы вырабатывают блок логических схем, в

котором этот сигнал, пройдя большое количество логических элементов, имеет заваленное фронты. Т.к. требование к фронтам управляющие сигналов высокие, то для их выправления используются специальные формирователи.

Достоинства:

Самое высокое быстродействие.

Недостатки:

1. Т.к. Ст. тактов выбирается по самой длинной команде, то это заметно снижает быстродействие.

Поэтому делают так: все машинные команды разбивают на группы примерно одной длинны, и для каждой группы ставится свой счётчик тактов, который выбираются по самой длинной команде в своей группе.

2. Чтобы добавить новую команду нужно полностью переделать блок логических схем: добавить новую команду нельзя.

Лекция №13

13.1. Процессор

13.1.1. Структура процессора

Процессор – это устройство, которое осуществляет процессы обработки данных и программное управление этими процессами.

Основные функции процессора:

- Дешифровать и выполнять машинные команды и программы
- Организовывать обращение к оперативной памяти
- Инициировать работу каналов ввода/вывода и периферийных устройств
- Воспринимать и обрабатывать запросы от устройств машины и внешней среды (запросы на прерывание)

Рис. 13.1. Структурная схема процессора.

В состав процессора могут также входить:

1) Местная память высокого быстродействия
2) Блоки организации вычислительного процесса:
    - Блок прерываний (обслуживает запросы на прерывание)
    - Блок защиты памяти (в старых машинах делалась защита от записи в чужое поле памяти; в новых машинах делается и защита от записи в

чужое поле памяти и защита от несанкционированного чтения из чужого поля памяти)

- Блок контроля за правильностью работы процессора
- Блок диагностики процессора

Примечание. Блок контроля локализует неисправность с точностью до устройства, а блок диагностики локализует ошибку в неисправном устройстве с точностью до ТЭЗа (типовый элемент замены).

13.1.2. Способы адесации информации

Различают:

- Адресный код – это тот адрес, который записан в машинной команде
- Исполнительный адрес – это тот адрес, по которому происходит фактическое обращение к единице информации (операнду).

Адресный код и исполнительный адрес далеко не всегда совпадают.

Существует 7 методов адресации (формирования исполнительного адреса):

1) Непосредственная адресация (в этом случае команда содержит не адрес операнда, а сам операнд)
2) Прямая адресация (адресный код является исполнительным адресом)
3) Относительная адресация (базирование)

$$A_{исп} = A_{базы} + A_i,$$

где A_i – адресный код.

Такая адресация уменьшает разрядность адресной части команды, так как в адресную часть записывается не весь адрес, а только его смещение (A_i), которое определяет положение операнда относительно начала поля памяти, которое задается базовым адресом ($A_{базы}$). Базовые адреса хранятся в специально выделяемых ячейках памяти, которые называются базовыми регистрами.

4) Индексная адресация

$$A_{исп} = A_i + A_и,$$

где A_i – адресный код команды, $A_и$ – содержимое индексного регистра.

Такая адресация позволяет обращаться к элементам массива, начало которого задается A_i, а смещение – $A_и$.

5) Косвенная адресация

Адресный код в команде указывает не адрес операнда, а тот адрес ячейки памяти, в которой находится адрес операнда. Такая адресация может быть многоступенчатой. Используется в мини- и в микро-ЭВМ, т.е. там, где используются короткие команды.

6) Укороченная адресация

В команде задаются только младшие разряды адреса, а старшие разряды считаются равными нулю. Это позволяет обращаться только к части оперативной памяти, поэтому такая адресация используется только совместно с другими видами адресации, например, с базированием.

7) Адресация слов переменной длины

Используется в машинах с переменной длиной слов. В этом случае в команде указывается:

- Адрес начала слова
- Длина слова

Примечание. В каждой конкретной машине используются от 3-х до 5 методов адресации.

13.1.3. Форматы команд

Существуют следующие форматы команд:

1. Одноадресные команды

A1 – адресная часть, которая указывает адрес одного операнда.

Пример выполнения операции сложения:

ЗАС1	A1	«a» – RG1 АЛУ
ЗАС2	A2	«b» – RG2 АЛУ
СМ	A3	«a + b» – в ОП по A3

2. Двухадресные команды

Код операции	A1	A2

A1 – адрес первого операнда.

A2 – адрес второго операнда.

Результат ставится или в A1 или в A2.

Примечание. При этом теряются операнды из A1 (или из A2), поэтому существуют команды, которые оставляют результат на сумматоре.

Пример 1 (операция сложения)

СМ A1 A2 «a + b» → A1 (A2)

Пример 2

СМ* A1 A2 «a + b» → SM АЛУ

3. Трехадресные команды

Код операции	A1	A2	A3

Используются в машине Фон-Неймана.

4. Четырехадресные команды

Код операции	A1	A2	A3	A4

A4 – адрес следующей команды

Примечание. В 1, 2 и 3 формате все команды программы располагаются в последовательных ячейках памяти. В 4-м формате команды могут находится в совершенно разных ячейках памяти, не смежных друг с другом. Такой формат обычно используют в спецмашинах.

13.2. Запоминающее устройство

Память имеет иерархическую структуру (рис. 13.2.).

Рис. 13.2. Структура памяти.

СОЗУ – сверхоперативное запоминающее устройство.

ЗУ с произвольным доступом – позволяет адресоваться к отдельной единице информации и реализуется на магнитных дисках (МД).

ЗУ с последовательным доступом – позволяет адресоваться только к целому массиву информации (к файлу), поэтому, для того, чтобы обратиться к отдельной единице информации, надо сначала весь массив считать в ОП, и только в ОП уже обратиться к отдельной единице информации. Такое ЗУ реализуется на магнитных лентах (МЛ).

Лекция №14

14.1. Система организации ОП

14.1.1. Общие сведения

Существует 3 системы организации ОП:

— Система со структурой 2D

— Система со структурой 3D

— Система со структурой 2,5D

Все три системы используют принцип совпадения токов, при котором входной ток может складироваться из нескольких слагаемых.

Покажем принцип совпадения токов на структуре 2D (рис. 14.1):

$$I_p = \frac{1}{2} I_{max}$$

$$I_{\text{вх}} = I_{px} + I_{py}$$

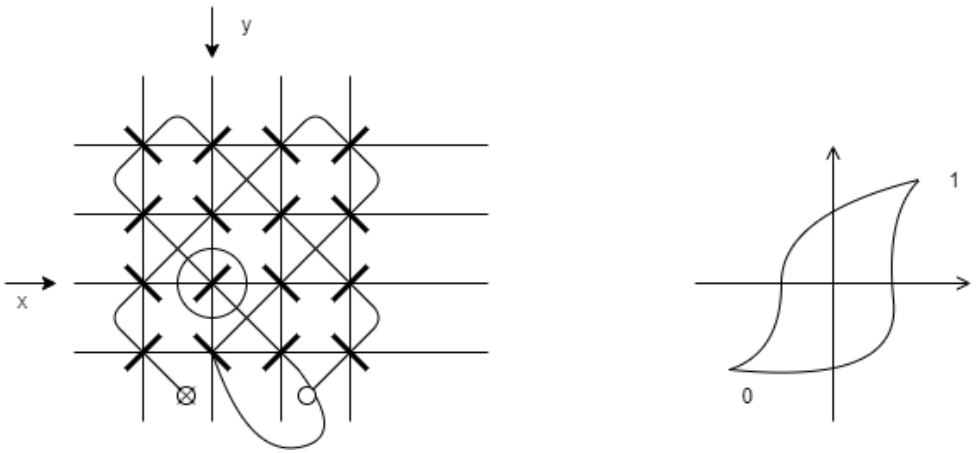


Рис. 14.1. 2D структура.

Кроме координатных обмоток x и y существует ещё обмотка считывания.

При подаче половины тока на координаты x_i и y_i только один сердечник (на пересечении координат) получает полный ток нужной полярности и перемешивается. Остальные сердечники на линейные x_i и y_i получают половинный ток. Основная проблема такой памяти – это помехи от полувозбуждённых сердечников.

Первый способ борьбы с помехами – использовать структуру 3D, где $I_p = \frac{1}{3} I_{max}$.

Позднее была разработана структура 2,5D, в которой на выбранном сердечнике токи складываются, на другую «полувозбуждённых» – вычитание.

14.1.2. Оперативная память на ферритовых сердечниках со структурой 2D

В этой структуре имеются 2 оси:

Ось x – адресная шина (на неё подаётся адрес ячейки памяти)

Ось y – разрядная шина (на неё подаётся записываемое слово, на него подаётся счётное слово).

Кроме них имеются ещё области считывания.

14.1.3. Режим записи

На RG адреса подаётся адрес записываемого слова, а на RG слова – само записываемое слово (рис 14.2).

Запись делается в 2 этапа:

— обнуление ячейки памяти (фиктивное чтение)

— собственно запись

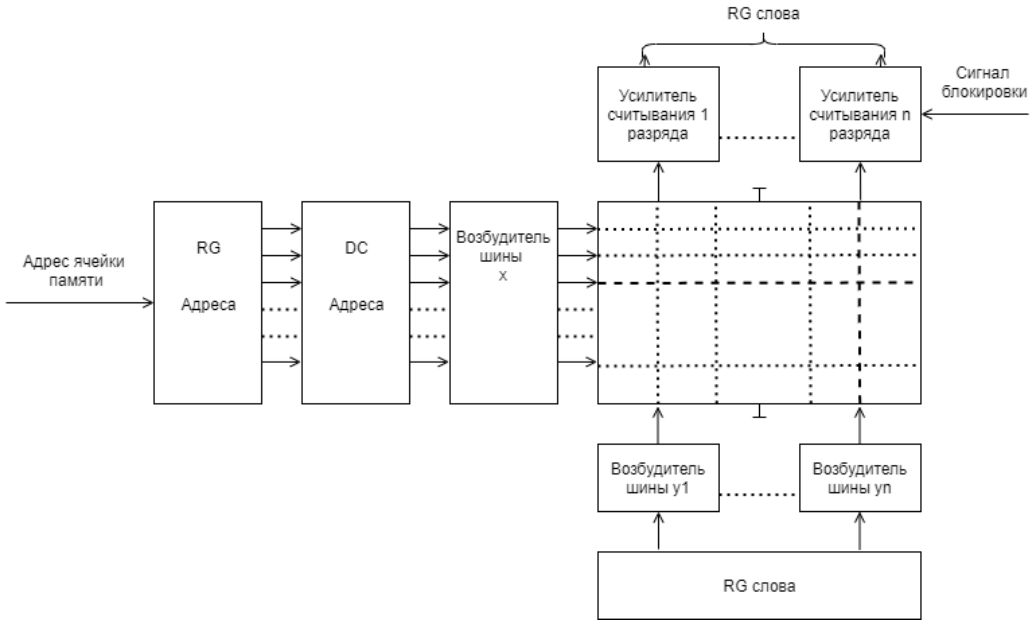


Рис. 14.2. Режим записи в ОП на ферритовых сердечниках со структурой 2D.

Этап 1 – фиктивное чтение.

Подаётся адрес, дешифруется, выбирается одна горизонтальная линейка, на которую возбудитель шины x подаёт полный ток отрицательной полярность. При этом считывается находящееся там слово, которое поступает на усилители считывания и хочет «испортить» записывающее слово, стоящее на RG слова. Чтобы это не произошло на усилители считывания подают специальный блокирующий сигнал, который это слово на RG слова не пропускает.

Этап 2 – собственно запись

Снова подаётся адрес на RG адреса, снова дешифруется, снова выбирается та же горизонтальная шина, на которую возбудитель шины x подаёт половинный ток положительной полярности. Второй половинный ток поступает с возбуждённой шины y только с тех разрядов RG слова, которое находится в 1.

14.1.4. Режим чтения

Состоит из двух этапов:

— собственно чтение

— регенерация

Этап 1 (см. фиктивное чтение), но без блокируемого сигнала.

Адрес ячейки памяти подаётся на RG адреса, дешифруется, выбирается одна горизонтальная полный ток отрицательной полярности. Т.е. делается то же самое, что и при фиктивном чтении, но блокирующий сигнал на усилитель считывания не подаётся. В результате считанное слово поступает на RG слова, а ячейка оказывается обнулённой.

Этап 2 – регенерация

Для регенерации нужно только считанное слово вновь записать. Для этого снова подаётся адрес, дешифруется, выбирается одна линейка, на которой возбудитель шины x подаёт половинный ток положительной полярности. Второй половинный ток поступает с возбудителя шины y, со всех разрядов, которые в RG слова равны 1.

14.2. Полупроводниковое ЗУ

14.2.1. Общие сведения

Строится на БИСах. Существуют:

- статические (строится на триггерах: ТТЛ-Tr, ТЛЭС-Tr)
- динамические ЗУ (строится на МОП-Tr, в которых электрический заряд хранится на так называемых затворах транзисторов).

14.2.2. Структурная схема статического ЗУ на 16 одноразрядных слов со структурой 2D

Емкость любой памяти определяется как 2^n, где n – число адресных шин.

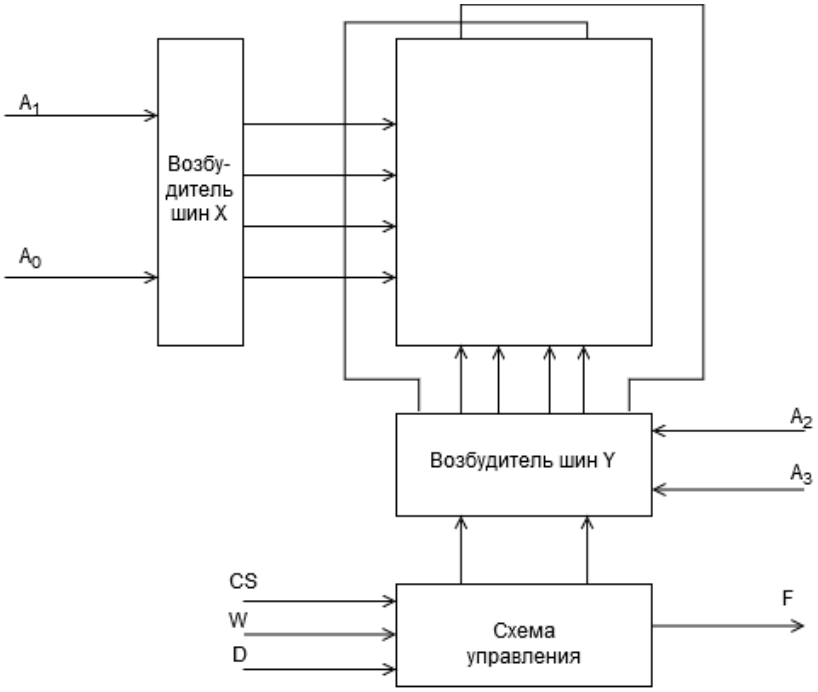


Рис. 14.3. Структурная схема статического ЗУ.

Возбудители шин X, шин У – включают DC.

Обозначения:

CS – сигнал выбора модуля

W – сигнал записи

D – одноразрядный информационный вход

F – одноразрядный информационный выход

14.2.3. Выбор триггера

По младшим разрядам адреса выбирается одна горизонтальная линейка. По старшим разрядам адреса выбирается одна вертикальная линейка. На пересечении стоит требуемый триггер.

14.2.4. Назначение разрядных шин

Если со входа D в выбранный триггер записывается «0», то он поступает в этот триггер по разрядной шине «0». Если записывается «1» со входа D, то она поступает по разрядной шине «1».

Аналогично считанный «0» или «1» поступают на выход F либо по разрядной шине «0», либо по разрядной шине «1» соответственно.

14.2.5. Режим записи

Для этого подаются следующие сигналы:

- CS (выбор модуля)
- W (сигнал записи)
- D (на него подается либо «0», либо «1»)
- адресная выборка (адрес)

14.2.6. Режим чтения

Подаются следующие сигналы:

- CS
- адресная выборка (адрес)

14.2.7. Режим хранения информации

Никакие сигналы не подаются.

Примечание. Для многоразрядной (n разрядов) памяти нужно параллельно поставить n таких БИС.

14.2.8. Динамическое ЗУ

Строится на МОП-транзисторах, у которых информация хранится (1) на так называемых затворах.

При этом «0» хранится долго, а «1» хранится ограниченное время, поэтому периодически делается процедура восстановления «1», которая называется Refresh (рефреш).

Процесс считывания заключается в определении есть заряд на затворе, или его там нет.

Лекция №15

15.1. Каналы ввода-вывода (КВВ)

КВВ и интерфейсы обеспечивают взаимодействие центральных устройств машины и периферийных устройств.

КВВ и интерфейсы выполняют следующие функции:

1) Позволяют иметь машины с переменным составом ПУ, в зависимости от требований потребителя.

2) Обеспечивают параллельную работу ПУ как между собой, так и по отношению к процессору.

3) Обеспечивают автоматическое распознавание и реакцию процессора машины на различные ситуации, возникающие в ПУ.

КВВ — это самостоятельные в логическом отношении устройства, работающие под управлением собственных программ, хранящихся в памяти.

В современных машинах КВВ называются периферийными процессорами или процессорами ввода/вывода.

Основные типы КВВ:

- мультиплексный канал (работает с медленнодействующими ПУ. При этом, подключившись к одному устройству, передает одно машинное слово, и после этого переключается на другое устройство).

- селекторный канал (работает с быстродействующими ПУ, поэтому подключившись к одному устройству, передает всю пачку информации, и только после этого переключаются на другое устройство).

- блок-мультиплексный (подключившись к одному устройству, передает часть пачки информации и только после этого переключается на другое устройство).

15.2. Интерфейсы ЦВМ

Все устройства машины связываются друг с другом через сопряжение – интерфейсы.

Интерфейсы – это совокупность унифицированных шин для передачи информации, унифицированных сигналов, электрических схем и алгоритмов, управляющих процессом обмена информации.

От характеристик интерфейсов во многом зависит производительность вычислительной системы в целом и её надежность.

Типы интерфейсов:

- тип A (интерфейс ОП - процессор)

- тип B (интерфейс процессор- КВВ)

- тип C (интерфейс ПУ-в)

- тип D (интерфейс периферийных аппаратов)

Структурная схема

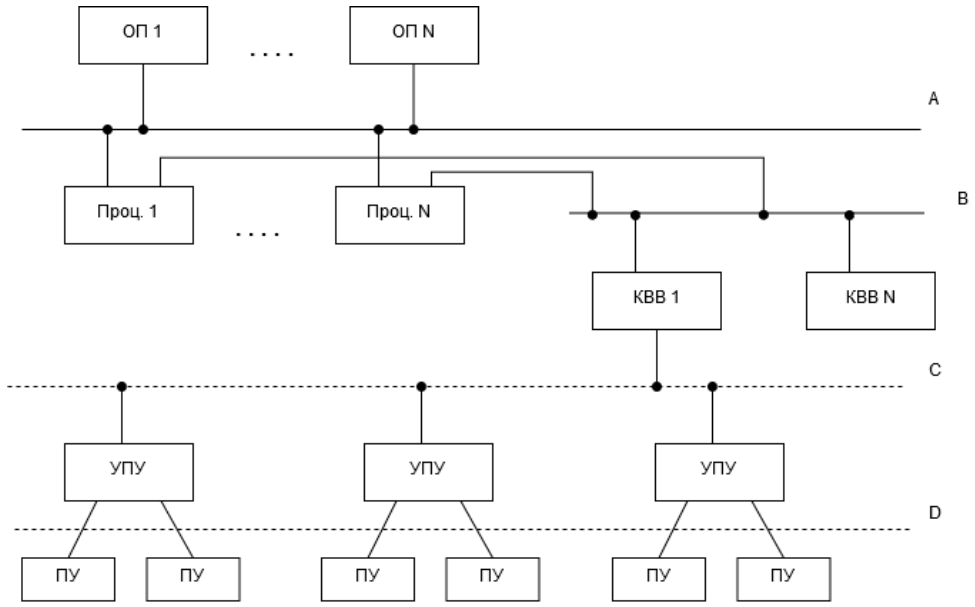


Рис. 15.1. Структурная схема интерфейсов.

УПУ – управление периферийным устройством.

Самыми быстродействующими являются интерфейсы A и B. По ним информация передается:

- одиночными словами

- двойными словами.

Через интерфейсы типа C информация передается байтами (обычно), (в маломощных машинах - битами).

П интерфейсу типа D информация обычно передается битами (в мощных машинах - байтами).

Интерфейсы бывают:

- односвязными (все устройства машины в этом случае связываются друг с другом по одной линии, которая работает в режиме разделения времени).

- многосвязными (каждая пара устройств в этом случае связывается друг с другом по своим независимым системам шин).

Примечание

При односвязном интерфейсе живучесть вычислительной системы – низкая.

15.3. Аналоговые вычислительные машины (АВМ)

Построение АВМ основано на принципе моделирования.

Моделирование – это метод исследования физических процессов, проводимых не на натуре, а на модели.

Моделирование бывает:

- физическое (натура и модель имеют одинаковую физическую природу и идентичные физические проекции)

- математическое (натура и модель имеют разную физическую природу, но математическое описание физических процессов в них - аналогичное).

Именно математическое моделирование лежит в основе построения АВМ.

Математическое моделирование бывает двух типов:

- групповые устройства (такие устройства нельзя подразделить на отдельные компоненты, т.к. их структуры полностью зависит от решаемого дифференциального уравнения)

- структурные модели (такие устройства состоят из различных блоках, каждый из которых непрерывно отрабатывает одну математическую операцию, как-то: интегрирование, дифференцирование, суммирование массива чисел и т.д.).

Именно структурные модели и лежат в основе АВМ.

Порядок работы на АВМ.

- аналитический расчёт коэффициентов

- разработка структурной схемы

- набор структурной схемы и коэффициентов на наборном поле АВМ

- подача входных сигналов и практически мгновенное получение результата на выходе, при этом выходной сигнал непрерывно изменяется в соответствии с изменением входного сигнала.

Структурная схема АВМ.

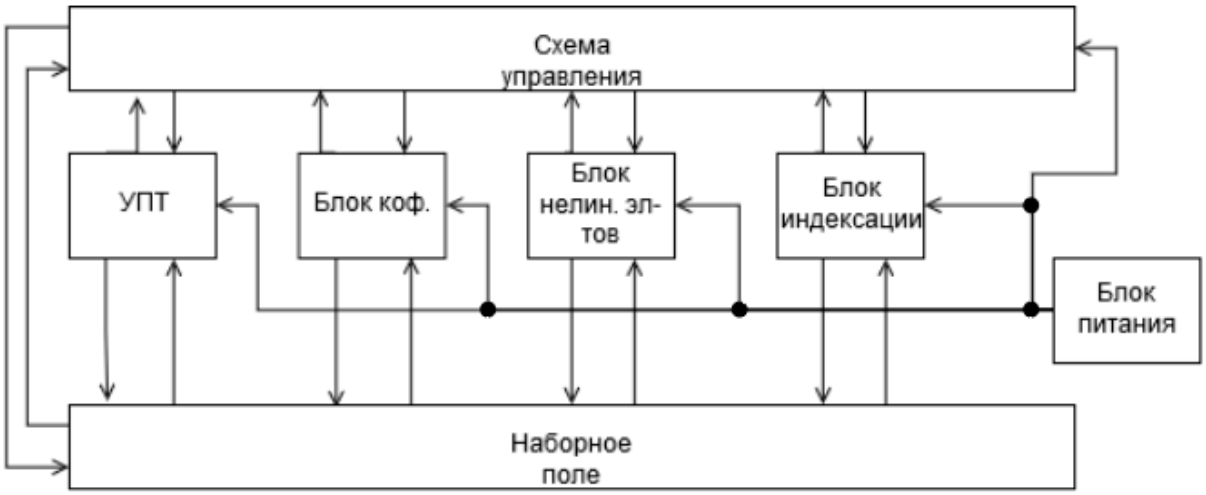


Рис. 15.2. Структурная схема АВМ.

УПТ – усилитель постоянного тока.

Различия между АВМ и УВМ

- в способе представления величин в машине

- в способе выполнения математических операций.

Примечание

В АВМ – достаточно долгий процесс – сбор структурной схемы в соответствии с решаемым уравнением.

Лекция №16

16.1. Система счисления(СС)

16.1.1. Определение системы счисления

Система счисления — символический метод записи чисел, представление чисел с помощью письменных знаков.

Существует множество систем счислений (двоичные, десятичные, восьмеричные, шестнадцатеричные и т.д.).

Таблица 16.1.

Основания систем счисления.

Название	Цифры
двоичная	0, 1
троичная	0, 1, 2
восьмеричная	0,..., 7
шестнадцатеричная	0,...,9,A, ..., F

В позиционной системе счисления любое вещественное число в развернутой форме может быть представлено в следующем виде:

$A = \pm (a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+ \dots +a_0q^0+a_{-1}q^{-1}+a_{-2}q^{-2}+ \dots +a_{-m}q^{-m})$

Здесь:

A - само число,

q - основание системы счисления,

a_i - цифры, принадлежащие алфавиту данной системы счисления,

n - число целых разрядов числа,

m - число дробных разрядов числа.

Развернутая форма записи числа - сумма произведений коэффициентов на степени основания системы счисления.

Пример. Десятичное число $A_{10} = 4718,63$ в развернутой форме запишется так:

$A_{10} = 4•10^3 + 7•10^2 + 1•10^1 + 8•10^0 + 6•10^{-1} + 3•10^{-2}$

Двоичное число $A_2 = 1001,1 = 1•2^3 + 0•2^2 + 0•2^1 + 1•2^0 + 1•2^{-1}$

Восьмеричное число $A_8 = 7764,1 = 7•8^3 + 7•8^2 + 6•8^1 + 4•8^0 + 1•8^{-1}$

Шестнадцатеричное число $A_{16} = 3AF = 3•16^2 + 10•16^1 + 15•16^0$

16.2. Алгоритмы перевода в системы счисления по разным основаниям

16.2.1. Алгоритм перевода чисел из любой системы счисления в десятичную

1. Представить число в развернутой форме. При этом основание системы счисления должно быть представлено в десятичной системе счисления.
2. Найти сумму ряда. Полученное число является значением числа в десятичной системе счисления.

16.2.2. Алгоритм перевода целых чисел из десятичной системы счисления в любую другую

1. Последовательно выполнять деление данного числа и получаемых целых частных на основание новой системы счисления до тех пор, пока не получится частное, меньше делителя.
2. Полученные остатки, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления.
3. Составить число в новой системе счисления, записывая его, начиная с последнего остатка.

16.2.3. Алгоритм перевода правильных дробей из десятичной системы счисления в любую другую

1. Последовательно умножаем данное число и получаемые дробные части произведения на основание новой системы счисления до тех пор, пока дробная часть произведения не станет равна нулю или будет достигнута требуемая точность представления числа.
2. Полученные целые части произведений, являющиеся цифрами числа в новой системе счисления, привести в соответствие с алфавитом новой системы счисления.
3. Составить дробную часть числа в новой системе счисления, начиная с целой части первого произведения.

16.2.4. Алгоритм перевода произвольных чисел из десятичной системы счисления в любую другую

Перевод произвольных чисел, т.е. содержащих целую и дробную часть, осуществляется в два этапа:

1. Отдельно переводится целая часть.
2. Отдельно переводится дробная.
3. В итоговой записи полученного числа целая часть отделяется от дробной запятой.

16.3. Сложение двоичных чисел

16.3.1 Правила сложения двоичных чисел

Сложение в двоичной системе счисления выполняется по тем же правилам, что и в десятичной. Два числа записываются в столбик с выравниванием по разделителю целой и дробной части и при необходимости дополняются справа незначащими нулями. Сложение начинается с крайнего правого разряда. Две единицы младшего разряда объединяются в единицу старшего.

Пример: **$1011{,}1_2 + 1010{,}11_2$**

```
   1←  1← 1← 1←
     1 0 1 1,1 0
+    1 0 1 0,1 1
-------------------
1 0 1 1 0,0 0
```

Рис. 16.1. Пример сложения в двоичной СС.

Интересна также ситуация, когда складываются больше двух чисел. В этом случае возможен перенос через несколько разрядов.

Пример: **$111{,}1_2 + 111_2 + 101{,}1_2$**

Рис. 16.2. Пример сложения в двоичной СС.

При сложении в разряде единиц (разряд 0) оказывается 4 единицы, которые, объединившись, дают $\mathbf{100_2}$. Поэтому из нулевого разряда в первый разряд переносится **0**, а во второй — **1**.

Аналогичная ситуация возникает во втором разряде, где с учетом двух перенесенных единиц получается число $\mathbf{5 = 101_2}$**. 1** остается во втором разряде, **0** переносится в третий и **1** переносится в четвёртый.

16.4. Вычитание двоичных чисел

16.4.1. Правила вычитания двоичных чисел

В случаях, когда занимается единица старшего разряда, она дает две единицы младшего разряда. Если занимается единица через несколько разрядов, то она дает по одной единице во всех промежуточных нулевых разрядах и две единицы в том разряде, для которого занималась.

Пример: $\mathbf{10110{,}01_2 — 1001{,}1_2}$

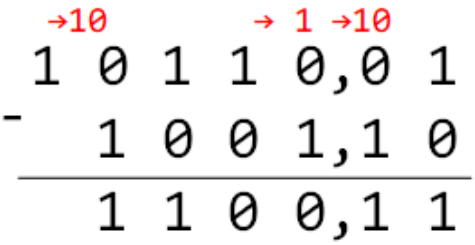


Рис. 16.3. Пример вычитания в двоичной СС.

16.5. Умножение и деление двоичных чисел

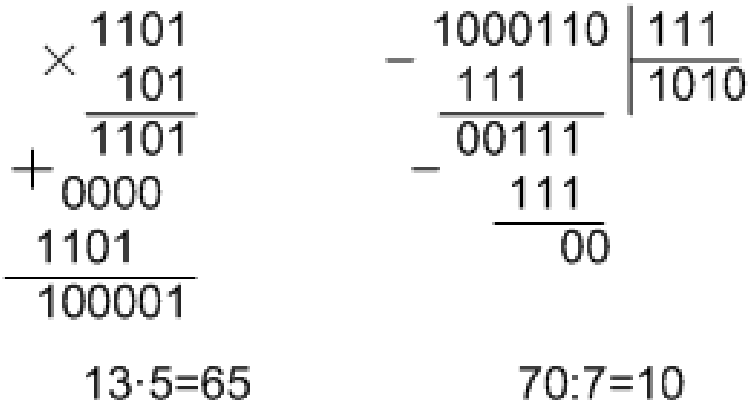


Рис. 16.4. Пример умножения и деления в двоичной СС.

Зная операции двоичной арифметики, можно переводить числа из двоичной системы счисления в любую другую.

Пример: перевести число 101111011_2 в десятичную систему счисления. Поскольку $10_{10} = 1010_2$, запишем

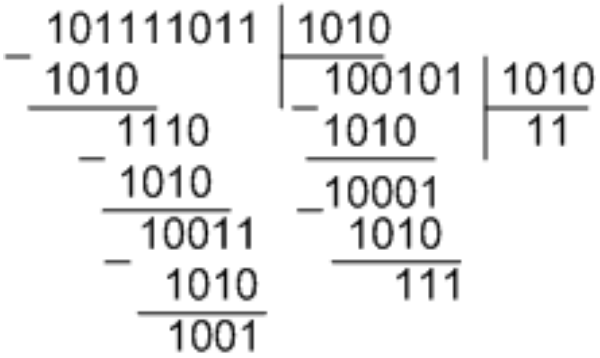


Рис. 16.5. Пример деления в двоичной СС.

Полученные остатки, $1001_2 = 9_{10}$, $111_2 = 7_{10}$, $11_2 = 3_{10}$. Искомое число $101111011_2 = 379_{10}$.

16.6. Представление чисел в ЭВМ

16.6.1. Прямой код для целых чисел

Прямой код двоичного числа совпадает по изображению с записью самого числа. Значение знакового разряда для положительных чисел равно 0, а для отрицательных чисел 1.

16.6.2. Обратный код для целых чисел

Обратный код для положительного числа совпадает с прямым кодом. Для отрицательного числа все цифры числа заменяются на противоположные (1 на 0, 0 на 1), а в знаковый разряд заносится единица.

16.6.3. Дополнительный код для целых чисел

Дополнительный код положительного числа совпадает с прямым кодом. Для отрицательного числа дополнительный код образуется путем получения обратного кода и добавлением к младшему разряду единицы.

В любом представлении старший бит определяет знак числа:

0 - положительное число;

1 - отрицательное число

Пример.

Для числа +1101:

Таблица 16.2

Число +1101 в разных представлениях

Прямой код	Обратный код	Дополнительный код
0,0001101	0,0001101	0,0001101

Таблица 16.3

Число -1101 в разных представлениях

Прямой код	Обратный код	Дополнительный код
1,0001101	1,1110010	1,1110011

16.6.4. Сложение и вычитание чисел в дополнительном коде для целых чисел

Если оба числа имеют n–разрядное представление, то алгебраическая сумма будет получена по правилам двоичного сложения (включая знаковый разряд), если отбросить возможный перенос из старшего разряда. Если числа принадлежат диапазону представимых данных и имеют разные знаки, то сумма всегда будет лежать в этом диапазоне. Переполнение может иметь место, если оба слагаемых имеют одинаковые знаки.

16.6.5. Определение вещественных числа (числа с плавающей точкой)

Все равные по абсолютному значению положительные и отрицательные числа отличаются только этим битом. В остальном числа с разным знаком полностью одинаковы. Для представления отрицательных чисел здесь не используется дополнительный код.

Поле мантиссы содержит мантиссу нормализованного числа.

Одинарная точность:

(цифры мантиссы)*$2^{(P-127)}$

Двойная точность:

(цифры мантиссы)*$2^{(P-1023)}$

Расширенная точность:

(цифры мантиссы)*$2^{(P-16383)}$

16.7. Понятие нормализации

Число называется нормализованным, если его в старшем разряде мантиссы стоит значащая цифра.

Примечание: все арифметические операции в машине заканчиваются нормализацией результата.

16.8. Переполнение разрядной сетки

При выполнении некоторых арифметических операций может возникать явление переполнения разрядной сетки. Причиной переполнения может служить суммирование двух чисел с одинаковыми знаками , которые в сумме дают величину, большую или равную 1 (при сложении правильных дробей), или величину r^n (при сложении целых чисел).

Пример: A=+0,101 $[A]_{доп}$ = 0,101

B=+0,110 $[B]_{доп}$ = 0,110

$[A+B]_{доп}$ = 1,011

В результате сложения двух положительных чисел получено отрицательное число, что является ошибкой. Результат неверен также и по величине.

Для обнаружения переполнения можно использовать следующие признаки:

- знаки слагаемых не совпадают со знаком суммы;

- есть перенос только в знаковый или только из знакового разряда.

Если при сложении чисел с фиксированной запятой возникло переполнение, то вырабатывается сигнал переполнения разрядной сетки и вычисления прекращаются.

Машинный ноль — это число, вышедшее за разрядную сетку справа.

16.9. Модифицированные коды

В отличие от обычных машинных кодов в модифицированных кодах под знак числа отводится два разряда: плюс изображается двумя нулями, а минус – двумя единицами. Это весьма удобно для выявления переполнения разрядной сетки, которое может получиться при сложении чисел с одинаковыми знаками.

16.10. Сложение чисел в модифицированном дополнительном коде

Сложение чисел в модифицированном дополнительном коде осуществляется по правилам двоичной арифметики. Единица переноса, возникающая в старшем знаковом разряде суммы, отбрасывается. Знаковым разрядом числа является второй слева от запятой разряд; первый разряд служит для анализа переполнения разрядной сетки.

Лабораторные работы

17.1 Лабораторная работа №1

1. Перевести из 10→2 число 23,5.

2. Перевести из 10→8→2 число 23,5.

3. Перевести из 10→16→2 число 23,5.

4. Представить отрицательное число X= –0,0110 в обратном и обратном модифицированном коде:

[X]обр. = ?

[X]обр. мод. = ?

5. Представить отрицательное число X= –0,1100 в дополнительном и дополнительном модифицированном коде:

[X]доп. = ?

[X]доп. мод. = ?

6. Заменить операцию вычитания в прямом коде на операцию сложения в обратном и обратном модифицированном коде:

X= +0,1100	[X]обр. = ?	[X]обр. мод. = ?
Y= +0,0110	[Y]обр. = ?	[Y]обр. мод. = ?
X – Y = ?	[X]обр. + [Y]обр. = ?	[X]обр. мод. + [Y]обр. мод. = ?

7. Заменить операцию вычитания в прямом коде на операцию сложения в дополнительном и дополнительном модифицированном коде:

X= +0,1100	[X]доп. = ?	[X]доп. мод. = ?
Y= +0,0110	[Y]доп. = ?	[Y]доп. мод. = ?
X – Y = ?	[X]доп. + [Y]доп. = ?	[X]доп. мод. + [Y]доп. мод. = ?

8. Построить конституенту «1» и конституенту «0» для следующих аргументов: A = 0; B = 0; C = 1; D = 1.

9. По заданной таблице истинности записать логическую функцию в виде ДНФ и КНФ:

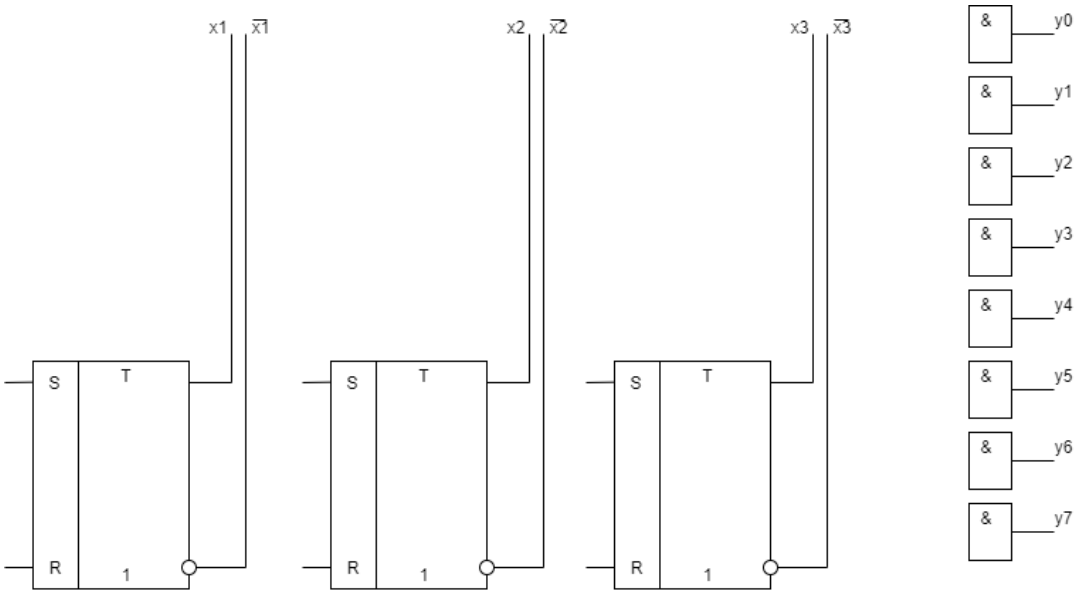
a b c	f	ДНФ
0 0 0	1	f(a,b,c) = ?

0 0 1	0	
0 1 0	1	КНФ
0 1 1	1	f(a,b,c) = ?
1 0 0	0	
1 0 1	0	
1 1 0	1	
1 1 1	0	

10. Построить комбинационную схему на элементах «И-НЕ».
11. Построить комбинационную схему на элементах «ИЛИ-НЕ».
12. Устный ответ.

17.2 Лабораторная работа №2

1)

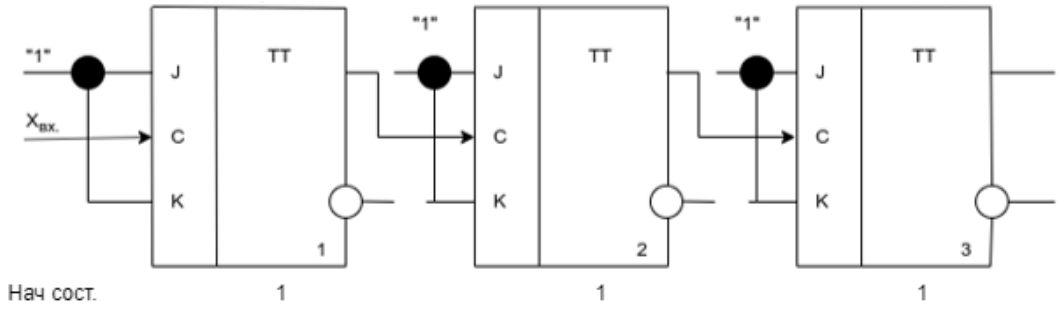


2)

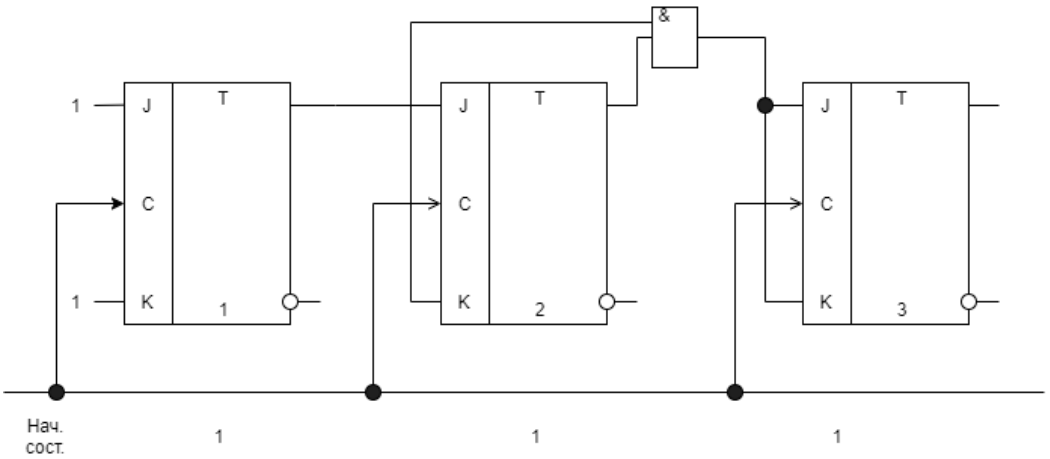
x_3	x_2	x_1	y_0	y_0	y_0	y_0	y_0	y_0	y_0	y_0
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								

1	0	1								
1	1	0								
1	1	1								

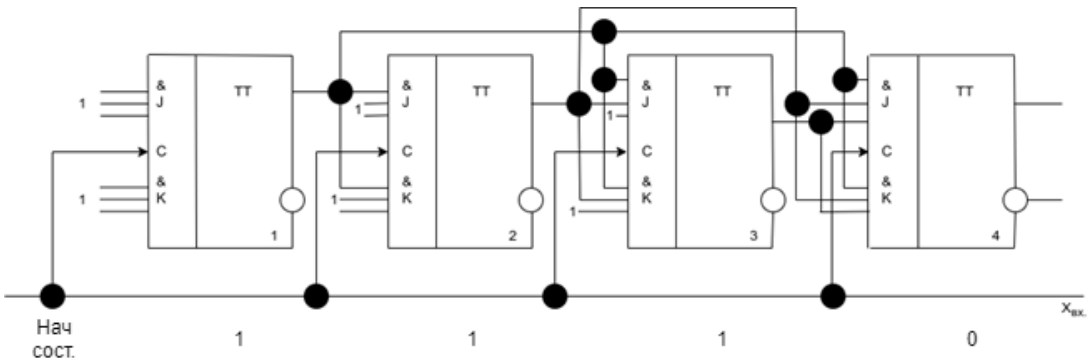
3)



4)



5)



6) Построить временную диаграмму для пункта 3.

17.3 Лабораторная работа №3

АЛУ для сложения.

Заменить следующее вычитание на сложение в дополнительном модифицированном коде:

A = 0,1100

B = 0,0110

A – B = ?

Построить структурную схему АЛУ для сложения и написать, что находится на:

RG 1 := ?

RG A := ?

RG B := ?

Пр SM := ?

Лев SM := ?

SM := ?

Вых SM := ?

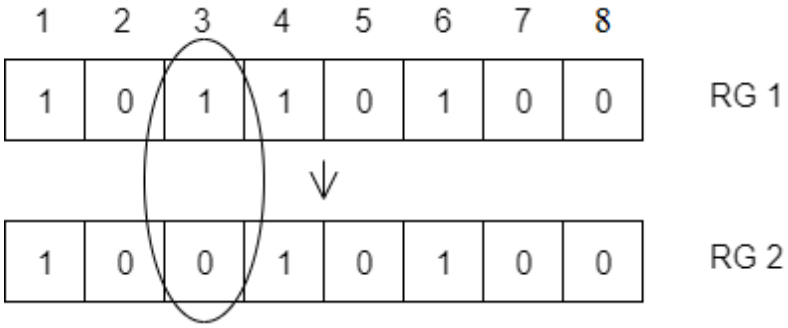
RG SM := ?

ШИ ВЫХ := ?

17.4 Лабораторная работа №4

Контроль передачи информации с RG 1 на RG 2.

1) Контроль на <u>четность</u>/нечетность (ошибка в третьем разряде)



- последовательная свертка

- пирамидальная свертка

2) Контроль по коду Хэмминга (информационный код 1010)

- одиночная ошибка (в третьем разряде)
- двойная ошибка (в третьем и пятом разрядах)