УДК 007.052: 519.713
ББК 32.972
А72

Антик М.И. Теория автоматов в проектировании цифровых схем [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА – Российский технологический университет, 2020. — 1 электрон. опт. диск (CD-ROM)

В учебном пособии изложены вопросы проектирования цифровых схем различного назначения на логическом и структурном уровнях в сопровождении необходимого математического аппарата.

Предназначено для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника.

Учебное пособие издается в авторской редакции.

Авторский коллектив: Антик Михаил Ильич, Казанцева Лариса Вячеславовна.

Рецензенты:

Петров Михаил Юрьевич, к.т.н., главный научный сотрудник научно-учебного департамента ФГБУ НИИ «Восход».

Лямин Юрий Алексеевич, к.т.н., ученый секретарь НТС ФГБУ НИИ «Восход».

Системные требования:
Наличие операционной системы Windows, поддерживаемой производителем.
Наличие свободного места в оперативной памяти не менее 128 Мб.
Наличие свободного места в памяти постоянного хранения (на жестком диске) не менее 30 Мб.
Наличие интерфейса ввода информации.
Дополнительные программные средства: программа для чтения pdf-файлов (Adobe Reader).
Подписано к использованию по решению Редакционно-издательского совета
МИРЭА — Российский технологический университет.
Обьем: 2.64 мб
Тираж: 10

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
ЧАСТЬ I. ОСНОВНЫЕ МЕТОДЫ АНАЛИЗА И СИНТЕЗА 6
1. Основные понятия 6
1.1. Каноническая структура синхронного цифрового автомата 6
1.2. Абстрактный конечный автомат 6
1.3. Соглашения … 8
2. Комбинационные схемы 9
2.1. Булев куб. 9
2.2. Комбинационная схема. 10
3. Память синхронного цифрового автомата 16
3.1 RS-триггеры 16
3.2. DL-триггеры (защёлки) 18
3.3. Синхронные триггеры 20
4. Автономные автоматы 23
4.1. Определения 23
4.2. Регистры сдвига 24
4.3. Параллельная композиция 24
4.4. Последовательная композиция 24
4.5. Делители частоты. 25
4.6. Линейные автономные автоматы 26
4.7. Канонические автономные автоматы 28
5. Структуры не автономных автоматов 29
5.1. СЦА полностью и частично определенные 29
5.2. Автомат Мура 29
5.3. Автомат Мили 30
6. Способы описания автоматов 30
6.1. Автоматная таблица. 31
6.2. Автоматный граф (граф переходов). 32
6.3. Блок-схема 33
6.4. Блок-текст 34
7. Эквивалентные автоматы 34
7.1. Изоморфные и эквивалентные автоматы 34
7.2. Минимальные автоматы 35

7.3. Эквивалентность автоматов Мура автоматам Мили 37
8. «Прямое» проектирование 38
9. Автоматы распознавания языков 42
9.1. Слова 42
9.2. Языки 42
9.3. Автоматы 43
10. Автоматы без потери информации 48
ЧАСТЬ II. КОДИРОВАНИЕ ЧИСЕЛ И АРИФМЕТИКА 49
1. Двоичные числа 49
2. Арифметика двоичных чисел 50
2.1. Дополнительный код числа 50
2.2. Сложение в дополнительном коде 52
2.3. Сложение и сравнение чисел без знака 53
2.4. Сложение и вычитание в прямом коде 53
2.5. Смещённый код 56
2.6. Умножение в дополнительном коде 56
2.7. Умножение. Аппаратная реализация 57
2.8. Деление 60
2.9. Числа в формате с плавающей точкой 65
ЧАСТЬ III. УПРАВЛЯЮЩИЕ АВТОМАТЫ 66
1. Структура вычислительного устройства 66
2. Варианты взаимодействия операционного и управляющего автоматов 67
3. Основные способы адресации микрокоманд 69
3.1. Схема с адресным ПЗУ 71
3.2. Схема с двумя адресами в памяти 72
3.3. Схема с одним адресом в памяти 73
3.4. Схема с сокращенным тактом 74
3.5. Схема с регулярной адресацией 75
3.6. Схема с естественной адресацией 76
3.7. УА с переходами функциональным и на микроподпрограмму. 78
3.8. Управление с предвосхищением 79
ЛИТЕРАТУРА 81

ВВЕДЕНИЕ

Синхронный цифровой автомат (СЦА) объединяет комбинационную схему и элементы памяти в единую структуру с относительно простой процедурой взаимодействия. В тоже время, принципы такого взаимодействия являются базовыми для построения практически всей современной цифровой аппаратуры. Разработчик цифровой аппаратуры не может считаться достаточно грамотным, если он не владеет математическими методами анализа и синтеза синхронных автоматов. Применение таких методов гарантирует правильность и оптимальность решений по сравнению с интуитивными методами, основанными на некотором опыте, аналогиях и т.п. Даже если решение получено на интуитивном уровне, то оно должно быть проверено, уточнено и переработано на основе точных методов. Теория дает также ответ на вопрос «что сделать нельзя?», оставаясь в рамках структур синхронных цифровых автоматов.

Процесс проектирования цифрового устройства включает этапы логический и схемотехнический. В пособии исследуется логический этап проектирования. Схемотехническое проектирование (реализация в реально существующем элементном базисе, электрические параметры, нагрузки, помехи, быстродействие,...) в пособии не рассматривается.

Математический аппарат используется в той мере, в которой он необходим в инженерной практике анализа и синтеза автоматов.

ЧАСТЬ I. ОСНОВНЫЕ МЕТОДЫ АНАЛИЗА И СИНТЕЗА

1. Основные понятия

1.1. Каноническая структура синхронного цифрового автомата

Предметом изучения являются устройства, структура которых может быть представлена в виде конструкции из модулей двух типов - комбинационной схемы (КС), на рисунках обозначена – CL (**C**ombinational **L**ogic), и регистра RG (**R**e**G**ister) – памяти автомата (рис.1). Такие структуры называют синхронными цифровыми автоматами (СЦА).

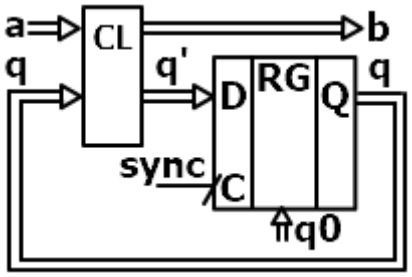


Рисунок 1.1. Структура синхронного автомата

1.2. Абстрактный конечный автомат

В качестве основной математической модели СЦА используется абстрактный конечный автомат.

Абстрактный конечный автомат - это математическая структура с тремя основными конечными множествами *A, B, Q*

A – называется множеством *входных символов*;

B – называется множеством *выходных символов*;

Q – называется множеством *(символов) состояний*,

q0 – одно из них называется *начальным состоянием*.

и двумя функциями: g: Q×A→Q, f: Q×A→B.

Такой абстрактной модели можно сопоставить структуру реального автомата с двумя комбинационными схемами (рис.2)

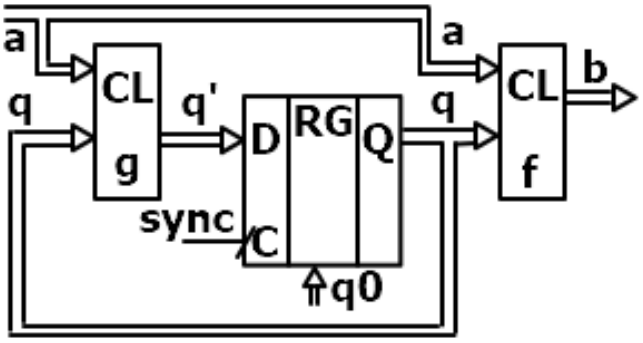


Рисунок 1.2. Синхронный цифровой автомат

Абстрактный автомат может быть *инициальным или неинициальным.* Если нет выделенного начального состояния, то автомат *неинициальный***.** Реальные автоматы всегда инициальные – начальное состояние определено. Начальное состояние может быть параметром автомата; в этом случае говорят, что определено множество начальных состояний.

Абстрактный конечный автомат функционирует в *автоматном времени* t=0,1,2,..., определяемом как упорядоченное множество целых чисел. Это означает, что автоматное время дискретно, и существенным является лишь номер момента времени; упорядоченность множества означает, что автоматное время имеет “направление”.

Если обозначить через **a**(t), **b**(t), **q**(t) переменные со значениями в соответствующих множествах в момент времени t, то работа абстрактного конечного автомата описывается уравнениями, которые называются каноническими

$$\mathbf{q}(t+1) = G(\mathbf{q}(t),\mathbf{a}(t))$$
$$\mathbf{b}(t) = f(\mathbf{q}(t),\mathbf{a}(t))$$

G - называется *оператором перехода*. Оператор перехода G, кроме вычисления значения функции g, выполняет также *сдвиг во времени* этого значения.

f - называется *функцией выхода*;

Принята также другая форма записи канонических уравнений (без указания индекса времени):

$$\mathbf{q} := g(\mathbf{q},\mathbf{a})$$
$$\mathbf{b} = f(\mathbf{q},\mathbf{a})$$

Знак двоеточие (**:**) означает операцию запоминания, а для дискретного времени – сдвиг значения на единицу времени. В этой записи g – *функция перехода*.

Вычисления, выполняемые автоматом, можно рассматривать как процесс, т.е. строгую последовательность действий (шагов) преобразования символов, поступающих на вход автомата **a** в символы, появляющиеся на его выходе **b**.

$$\mathbf{a}(1),\mathbf{a}(2),\mathbf{a}(3),...,\mathbf{a}(t),...$$
$$\mathbf{b}(1),\mathbf{b}(2),\mathbf{b}(3),...,\mathbf{b}(t),...$$

Автомат *отображает* входные последовательности символов в выходные последовательности символов. Выходная последовательность порождается с той же скоростью, что и входная последовательность, ровно один выходной символ на каждый входной. Каждый очередной выходной символ однозначно определяется последовательностью ранее поступивших входных символов, т.е. вычисление имеет свою “историю”. Эта история сохраняется в автомате в виде переменной, которая называется – ***состояние автомата***. Состояние автомата - это та минимальная информация, которая необходима для предсказания даль-

нейшего поведения автомата. Поведение автомата трактуется как его переходы из одного состояния в другое состояние в определенные моменты автоматного времени с одновременным вычислением выходного значения.

1.3. Соглашения

В реальной цифровой аппаратуре входные и выходные сигналы являются двоичными многоразрядными наборами, которые кодируют символы соответствующих множеств.

Регистр в СЦА запоминает код состояния. Состояния кодируются двоичными наборами. Количество двоичных разрядов памяти автомата, а значит и кода состояния, называется *мерностью* автомата. Комбинационные схемы, в свою очередь, реализуют, функции переходов и выхода.

Ход времени в СЦА инициируется изменением значения единственного сигнала – сигнала синхронизации sync. На каком-либо из фронтов этого сигнала изменяется содержимое RG, а значит и состояние автомата, так в схемах на рис.1.1 и 1.2 RG переключается по положительному (нарастающему) фронту сигнала sync. Переключающие фронты сигнала sync будем называть рабочими фронтами. Интервал времени между соседними рабочими фронтами называется *тактовым интервалом* или просто *такт*ом (T), рис.3.

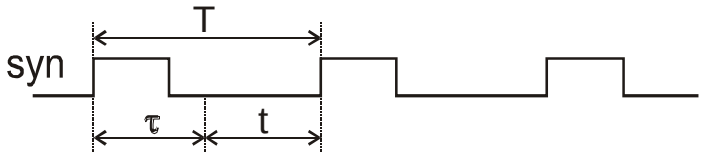


Рисунок 1.3. Структура такта СЦА

Для абстрактного автомата не имеет смысла понятие такта, как некоторого интервала “непрерывного” времени. Применительно к абстрактному автомату можно понимать термин *такт* как изменение дискретного времени на одну единицу. Для реального синхронного цифрового автомата величина этого интервала − один из важнейших технических параметров, определяющих быстродействие схемы. Величина этого интервала должна быть достаточной для того, чтобы закончились переходные процессы в КС и установившиеся значения состояния могли быть записаны в RG. Переходные процессы могут начинаться из-за изменения либо состояния RG, либо изменения входных сигналов. Изменение значений на выходах RG происходит всегда на границе такта. Для корректной работы синхронного автомата изменения входных сигналов должны происходить в интервале τ (рис.1.3) так, что бы время $t=T-\tau$ было бы достаточным для завершения переходных процессов.

Рассмотрим основные структурные составляющие СЦА.

2. Комбинационные схемы

2.1. Булев куб.

2.1.1. Определение булева куба

Множество переменных ($x_1,x_2,\ldots,x_n$), у которых значения принадлежат {0,1} образует n-мерное пространство $\{0,1\}^n$, которое можно интерпретировать как множество вершин (точек) n-мерного куба (*булева куба* − B^n). Число вершин B^n равно 2^n. Булев n-набор соответствует точке (вершине) булева куба.

Булева функция задает отображение ($\varphi:\{0,1\}^n\rightarrow\{0,1\}$), т.е. точек булева куба B^n в булев куб B^1. *Частично определённая* булева функция в каждой из 2^n точек n-мерного куба может принимать значение, равное либо 1 (1-точка), либо 0 (0-точка), либо быть неопределённой (×-точка, безразличная точка).

Точки со значение равным 1 образуют множество $\varphi^{-1}(1)$, соответственно 0-точки образуют множество $\varphi^{-1}(0)$.

Два булевых набора называются *соседними*, если они различаются только в одной компоненте (координате). У каждой вершины n-мерного куба ровно n соседних вершин.

2.1.2 Изображение булева куба.

1-й вариант изображения – графический. Булевым наборам соответствуют вершины. Вершины, соответствующие соседним наборам, соединяются отрезком.

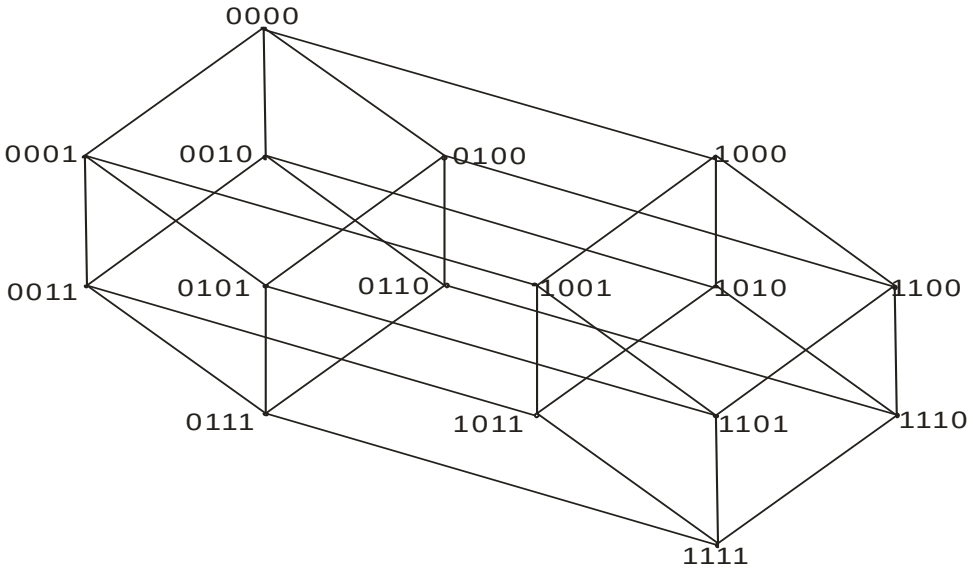


Рисунок 2.1. Булев куб (4-х мерный)

2-й вариант изображения – карта Карно. Карта Карно один из вариантов записи таблицы истинности. В булевом B^n кубе, изображённом в виде карты Карно, 2^n клеток. Булевым наборам (вершинам булева куба) соответствуют клетки карты Карно. Если число n чётное, то карта квадратная, у которой каж-

дая из сторон равна $2^{n/2}$ линий (строк, столбцов). Если число n нечётное, то одна из сторон содержит $2^{(n-1)/2}$ линий, вторая – в два раза больше $2^{(n+1)/2}$ линий. Булев набор – это координаты клетки. Идентификатор строки карты Карно – это левые значения булева набора, идентификатор столбца – оставшиеся правые. Соседние наборы имеют общую вертикальную или горизонтальную границу, соседними также являются клетки, симметрично расположенные относительно границ, делящих карту пополам.

		000	001	011	010	110	111	101	100
00		c	d	d	c	c	d	d	c
01			g	a			a	g	
11		b′	g			b″		g	b
10		c	d	d	c	c	d	d	c

Соседними являются наборы, отмеченные буквой **a;** соседние − (b, b′), (b, b″), не соседние – (b′, b″). Чтобы такое свойство соседства клеток в карте выполнялось, координаты соседних клеток (вершин, точек n-куба) должны отличаться только в одной компоненте. Поэтому строки (столбцы) карты Карно идентифицируются циклической последовательностью, соседние наборы которой различаются только в одном разряде. Такие последовательности двоичных наборов известны как коды Грея.

2.1.3 Грани булева куба.

Двоичные наборы b[1..n] с r одинаковыми значениями некоторых компонент, количество таких наборов 2^d, где d=(n-r) образуют *d-мерную грань* булева куба B^n. Вершина − 0-мерная грань. Две соседних вершины (например, отмеченные буквой ***a***) образуют 1-мерную грань. Любой из граней можно сопоставить *троичный набор* t[1..n] = (t_1,…, t_n), где $t_k=b_k$, если k–я компонента имеет значение b_k во всех наборах (точках, вершинах), принадлежащих этой грани, и t_i=(-) − в остальных случаях. Можно считать такой набор координатами грани. Вершины с буквой ***a*** 1-мерная грань с координатами (01−11); вершины, помеченные буквой ***c*** образуют 3-мерную грань (−0− −0); 3-мерную грань (−0− −1) образуют вершины, помеченные буквой ***d*;** 2-мерную грань (−101) образуют вершины, помеченные буквой ***g*.** Для любой i-ой координаты $1 \le i \le n$ ровно половина вершин булева куба, в которых эта координата равна 1, а для другой половины вершин этаже координата равна 0.

2.2. Комбинационная схема.

2.2.1. Определения

Булева функция может быть реализована в виде схемы из логических элементов, такие схемы называют *комбинационными*. Комбинационная схема, реа-

лизующая одну булеву функцию, существенно зависящую от n переменных, имеет n входов и один выход. Комбинационная схема реализует частично определённую булеву функцию, заданную на булевом кубе, если её выход равен 1 для каждого входа, соответствующего 1-точке, и равен 0 для каждого входа, соответствующего 0-точке. Выход, соответствующий неопределённой точке, при подаче на вход схемы рассматриваемой входной комбинации получит значение либо 0, либо 1, в зависимости от конкретной реализации схемы.

Реализация комбинационной схемы должна удовлетворять определённым критериям оптимальности, главные из которых быстродействие и стоимость. Каждый логический элемент схемы переключается за конечное время. Быстродействие схемы пропорционально числу элементов, через которые проходит сигнал от входа до выхода. В комбинационных схемах, реализованных в виде ДНФ или КНФ число таких элементов не более двух (не считая инверсий). Поэтому такие схемы называются двухступенчатыми. Стоимость схемы зависит от числа используемых логических элементов схемы и стоимости самих элементов. Стоимость элемента схемы зависит от числа его логических входов. Поэтому комбинационную схему можно считать оптимальной, если она реализует ДНФ с минимально возможным количеством элементарных конъюнкций, каждая из которых имеет минимально возможное число переменных, такие функции называют *кратчайшими* ДНФ (КДНФ). *Минимальными* ДНФ (МДНФ) называют реализацию схемы с минимальным в сумме количеством входов всех логических элементов схемы. Аналогично можно сформулировать требования оптимальности комбинационной схемы, реализованной в виде ККНФ (МКНФ). В дальнейшем будем использовать только сокращение МДНФ. (Минимальные варианты совпадают с кратчайшими для полностью определённых функций пяти переменных и меньше).

Если булева функция определена на n-наборах, то элементарную конъюнкцию удобно (для машинной обработки и не только) задавать наборами длины n из символов {0,1,−}. Если переменная x_k входит в конъюнкцию в форме $x_k^{\alpha_k}$, то в этом наборе на k-м месте записывается α_k, а если x_k отсутствует, то на k-м месте ставится «−». Например, для функции 4-х аргументов конъюнкции $a_1\bar{a}_3$ соответствует набор [1−0−]. Область единичных значений конъюнкции – соответствует (ещё говорят конъюнкция *покрывает*) грань с такими же координатами (1−0−), в остальных точках 4-мерного куба эта же конъюнкция равна 0. Количество переменных в элементарной конъюнкции называют *рангом* конъюнкции. Чем меньше ранг *r* конъюнкции, тем грань большей размерности d=(n-r), которую она покрывает с 2^d 1-точками.

Конъюнкция g является *импликантой* функции f, если хотя бы единственная 1-точка конъюнкции соответствует 1-точке этой функции и не одна из 1-точек конъюнкции не является 0-точкой функции f, т.е. импликанта функции покрывает только 1-точки функции и возможно неопределённые точки. Импликанта называется *простой*, если это конъюнкция, которая перестаёт быть импликантой в результате вычёркивания из неё любой из переменных. Простая импликанта это конъюнкция минимального ранга, у которой область единичных значений функции f не меньше чем у любой другой конъюнкции.

Отсюда следует, что МДНФ функции, отличной от константы 0 или 1, является дизъюнкцией некоторых простых импликант.

2.2.2. *Минимизация с использованием карт Карно.*

Простая импликанта покрывает грань максимального размера (максимальной мерности), содержащей 1-точки функции и не содержащей 0-точки функции. На карте Карно (до 6 переменных) визуальным образом выбираются грани максимального размера, содержащие 1-точки функции и не содержащей 0-точки функции. Минимальное число таких граней, содержащие все 1-точки функции, соответствует конъюнкциям МДНФ. Пример, таблица 1.9.

	00	01	11	10
00	1	0	1	0
01		0	0	
11	1	1		0
10	0	1	1	

импликанты
[0−00]
[−011]
[110−]
[1− −1]

Для полностью определённых функций, разумеется, МДНФ и МКНФ эквивалентны. Это не значит, что, найдя МДНФ просто получить МКНФ путём алгебраических (дистрибутивных) операций или наоборот. Во-первых, такое преобразование по числу операций классифицируется как экспоненциально сложное. Во-вторых, получив новое выражение могут потребоваться операции минимизации. Например, для функции трёх переменных МДНФ=ā∧e∨b∧ē, КНФ=(ā∨b)∧(ā∨ē)∧(b∨e), не очевидно, что МКНФ=(ā∨ē)∧(b∨e).

Для частично определённых функций, в общем случае МДНФ и МКНФ не эквивалентны. Например, для функции f(x,y,z), $f^{-1}(1)$={100,101,111}, $f^{-1}(0)$={000,010,110}. МДНФ=x∧ȳ∨z, МКНФ=x∧(ȳ∨z) и одно в другое не преобразуется.

2.2.3. Формализуемые методы минимизации.

1. Найти все простые импликанты функции f.
2. Из этого множества выбрать минимальное подмножество, покрывающее все 1-точки функции f.

Получение всех простых импликант последовательным склеиванием

Если конъюнкции представлены троичными наборами, то операция склеивания применима только к таким наборам, которые имеют противоположные значения только в одной из компонент.

1. Составить таблицу для всех 1-точек (это исходные максимальные импликанты) и неопределённых точек функции f, разбитых на классы (T_0,…,T_i,….,T_k) в соответствии с количеством i (столбец 1 таблицы) единичных компонент в наборах (столбец 3 таблицы). Отметить каждую 1-точку знаком + (столбец 2 таблицы).

2. Сравнить каждый элемент в T_i с каждым элементом T_{i+1} для всех i, $0 \le i \le k$. Для пар, отличающихся только в одной компоненте, образовать новые импликанты, покрывающие обе точки. Новые импликанты поместить в класс T'_i (столбец 6 таблицы). Использованные для их образования импликанты, отметить знаком √ (столбец 4 таблицы). Отметить каждую новую импликанту знаком + (столбец 5 таблицы), новый набор будет импликантой, если в его образовании участвовала импликанта.

3. Повторить шаг 2, используя T'_i и T'_{i+1}для образования T''_i и продолжать эту процедуру до тех пор, пока дальнейшие склеивания окажутся невозможными.

4. Импликанты, не участвовавшие в процедуре склеивания (определяемые по отсутствию знака √ и отмеченные знаком +), являются простыми импликантами.

1	2	3	4	5	6	7	8	9
0	+	0000	√	+	0−00			
1		0100			01−0			
				+	−100			
2	+	0011	√	+	−011		+	1− −1
		0110		+	110−		+	1− −1
	+	1100	√	+	1−01	√		
	+	1001	√	+	10−1	√		
		1010		+	101−			
3	+	1101	√	+	11−1	√		
	+	1011	√	+	1−11	√		
4		1111						

Выбор минимального подмножества простых импликант

Построим по функции f импликантную таблицу (Квайна) – таблица 1.11. Её строки соответствуют 1-точкам функции f, а столбцы простым импликантам. В пересечении строки P_i и столбца S_j ставится 1, если импликанта S_j покрывает точку P_i (иначе клетка оставляется пустой).

Каждая строка (1-точка функции) должна быть покрыта хотя бы одной импликантой, т.е. одной из единиц столбцов таблицы. Если именам строк и столбцов сопоставить логические переменные (P_i – линия покрыта, S_i – линия входит в покрытие), то на языке исчисления высказываний это соответствует выражению:

$$1= P_1\&P_2\&P_3\&P_4\&P_5\&P_6=$$

$$= S_1\&S_3\&(S_2\vee S_5)\&(S_4\vee S_5\vee S_7)\&S_7\&(S_3\vee S_6\vee S_7) \quad (2.1)$$

$$= S_1\&S_3\&S_7\&S_2\vee S_1\&S_3\&S_7\&S_5 \quad (2.2)$$

		S_1	S_2	S_3	S_4	S_5	S_6	S_7
		0−00	−100	−011	−101	110−	101−	1− −1
		+		+		+		+
P_1	0000	1						
P_2	0011			1				
P_3	1100		1			1		
P_4	1101				1	1		1
P_5	1001							1
P_6	1011			1			1	1

В результате дистрибутивных операций КНФ (2.1) преобразуется в ДНФ (2.2), в которой каждая из конъюнкций означает один из вариантов покрытия всех строк (1-точек) простыми импликантами. Надо выбрать одну самую короткую конъюнкцию в (2.2). В этом примере конъюнкции одинаковой длины поэтому в КДНФ должны быть использованы импликанты: или S_1,S_3,S_7,S_2, или S_1,S_3,S_7,S_5.

$МДНФ_1=[0−00]\vee[−011]\vee[1− −1]\vee[−100]$

$МДНФ_2=[0−00]\vee[−011]\vee[1− −1]\vee[110−]$

Решение задачи «*о покрытии матрицы*»: покрытия в 0-1-матрице одних линий (столбцов или строк) другими ортогональными линиями, методом «прямого» преобразования логического выражения из КНФ в ДНФ, в теории сложности алгоритмов оценивается асимптотически как экспоненциально сложный. Поэтому такая задача обычно решается иначе, в несколько этапов.

1. *Ядерные импликанты*: если в таблице есть строка, которая покрывается только одним столбцом (строка содержит ровно одну единицу), то такой столбец содержится в любом покрытии таблицы (в том числе минимальном). Импликанты, соответствующие такому столбцу, называются ядерными, заносятся в решение. Покрываемые таким столбцом строки и сам столбец вычёркиваются из таблицы. Это соответствует логическому поглощению на этапе представления логического выражения в виде КНФ (2.1).

2. *Доминирующая строка*: строка P_m доминирует над строкой P_k , если строка P_k содержит единицы во всех столбцах, в которых стоят единицы в строке P_m (короче, $P_m \le P_k$). Строка P_k вычёркивается из таблицы. Это соответствует логическому поглощению на этапе представления логического выражения в виде КНФ (1). Равенство $P_m = P_k$ соответствует идемпотентности.

3. *Доминирующий столбец*: столбец S_m доминирует над столбцом S_k, если столбец S_m содержит единицы во всех строках, в которых стоят единицы в столбце S_k (короче, $S_m \ge S_k$). Столбец S_k вычёркивается из таблицы. Это соответствует или поглощению, или выбору более коротких конъюнкций, на этапе представления логического выражения в виде ДНФ (2.2), т.е. выбирается решение с меньшим числом импликант.

4. Эти три правила сокращения таблиц применяются повторно до тех пор, пока дальнейшее сокращение окажется невозможным. После сокращений либо таблица пуста это значит решение получено, либо таблица не пуста. В предыдущем примере после выбора ядерных импликант (S_1,S_3,S_7), и в результате этого сокращения таблицы, остаётся одна строка. В силу пункта 3 в решение добавляется либо импликанта S_2, либо S_5.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
P_1	1						
P_2			1				
P_3		1			1		
P_4				1	1		1
P_5							1
P_6			1			1	1

	S_2	S_5
P_3	1	1

Не пустая таблица может состоять из более чем одной строки, такую таблицу называют *циклической*, в этом случае выполняется пункт 5.

5. В циклической таблице выбирается строка с наименьшим числом единиц. Для каждой из единиц этой строки выполняется следующее: импликанта, соответствующая этой единице, заносится в альтернативное решение, все строки, покрытые этой импликантой, вычеркиваются. Далее пункты 1−5.

В следующем хорошо формализуемом методе иначе выбираются простые импликанты.

Составим прямоугольную таблицу. Заголовки столбцов координаты 1-точек. Заголовки строк координаты 0-точек с инвертированными разрядами. В клетках таблицы запишем результат покомпонентной операции над заголовками строки и столбца. Если компоненты совпадают по значению, то записывается это значение, иначе ×. Например, 1100⊗1001=1x0x. В правой части равенства каждая из констант (0 или 1) определяет грань (n–1)–мерности ([1- - -], [- -0-]),

(половина точек n–мерного куба) накрывающая 1-точку с координатами заголовка столбца и не накрывающая 0-точку с координатами заголовка строки, а всё выражение трактуется как дизъюнкция этих граней.

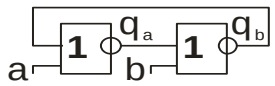
	0000	0011	1100	1101	1001	1011
1110	×××0	××1×	11×0	11××	1×××	1×1×
1101	××0×	×××1	110×	1101	1×01	1××1
1010	×0×0	×01×	1××0	1×××	10××	101×
1000	×000	×0××	1×00	1×0×	100×	10××
0001	000×	00×1	××0×	××01	×001	×0×1
0111	0×××	0×11	×1××	×1×1	×××1	××11
	0−00	−011	110−	1− −1	1− −1	1− −1
			−100			

Для каждого из столбцов находят простые импликанты как конъюнкцию всех дизъюнкций столбца. Вычисляется как покрытие 0-1-матрицы, рассмотренное выше. Второй этап такой же как в предыдущем методе: строится таблица Квайна.

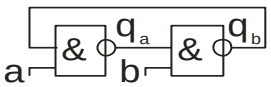
3. Память синхронного цифрового автомата

3.1 RS-триггеры

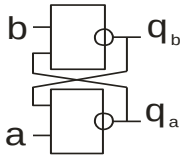
Триггер — это схема (функциональный элемент), которая способна запоминать информацию. Простейшими триггерами являются схемы, изображённые на рис.7, которые называются RS-триггерами.



а) в базисе ИЛИ-НЕ



б) в базисе И-НЕ



в) традиционное начертание схемы RS-триггера

Рисунок 3.1. RS–триггер

Таблица полных состояний:

ab	$(q_a q_b)^{v+}$	$(q_a q_b)^{\&+}$
00	$q_a q_b$	11
01	10	10
10	01	01
11	00	$q_a q_b$

Схемы RS-триггеров корректно переключаются и при одновременном переключении двух входных переменных (a,b), за исключением следующих пере-

ключений, при которых неопределённым является финальное стационарное состояние: для схемы в базисе И-НЕ (см. пример 2) – это переключение входных переменных (00)→(11); для схемы в базисе ИЛИ-НЕ – это переключение (11)→(00).

Функции, определяющие значение на входах триггера, называют *функциями возбуждения*. Соответственно таблицы, показывающие каким должно быть возбуждение при необходимом переключении, называют *таблицами возбуждениями*.

Для «*симметричных*» RS-триггеров с прямым Q и инверсным Q’ выходами, функциональные обозначения которых изображены на рис.10, таблицы возбуждения приведены на рис.11.

а) в базисе ИЛИ-НЕ б) в базисе И-НЕ

Рисунок 3.2. Функциональное изображение RS–триггера (симметричного)

$Q\ Q^+$	$(R\ S)^{\vee}$	$(R\ S)^{\&}$
0 0	X 0	X 1
0 1	0 1	1 0
1 0	1 0	0 1
1 1	0 X	1 X

Для «*несимметричных*» RS-триггеров с одним прямым выходом (рис.3.3):

а) в базисе ИЛИ-НЕ б) в базисе И-НЕ

Рисунок 3.3. RS–триггер (несимметричный). .

Таблицы возбуждения будут иными.

$Q\ Q^+$	$(R\ S)^{\vee}$	$(R\ S)^{\&}$
0 0	$*_R\ *_S$	X 1
0 1	0 1	X 0
1 0	1 X	0 1
1 1	0 X	$*_R\ *_S$

Здесь комбинация ($*_R*_S$) означает ─ любая пара значений кроме (01), или иначе ($*_R*_S$) = {X0,1X}

Многоустойчивый триггер. Можно увеличить количество элементов в схеме триггера, соединив их так, что выход каждого из элементов И-НЕ (ИЛИ-НЕ) соединён с входами всех остальных элементов схемы (рис.14). Получим некое обобщение RS-триггера ─ многоустойчивый триггер. Тот выход элемента схемы на рис14, на котором раньше других появится значение 0, запретит переключение всех остальных элементов и установит на их выходах значение 1.

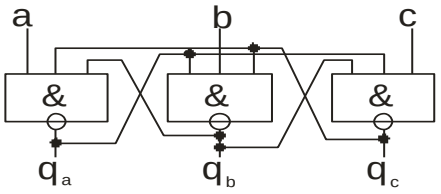


Рисунок 3.4. Многоустойчивый триггер

3.2. DL-триггеры (защёлки)

Определим DL-триггер как элемент памяти, функционирующий согласно следующей таблице переключений (рис.15):

L D	Q^+	
0 X	Q	хранение
1 0	0	установка нуля (сброс)
1 1	1	установка единицы (установка)

Такой триггер называют триггером *защёлкой* (transparent latch). При L=1 (Load) DL-триггер находится в режиме «прозрачности» (transparent), т.е. на выходе повторяет значения на D входе. При изменении значения на входе L=(1→0) и стационарном значении на входе D, триггер переходит в режим хранения «защёлкивается» (latch). Упоминание о стационарности сигнала D важно – нельзя одновременно переключать сигнал L из 1 в 0 и изменять значение сигнала D – конечное стационарное значение Q не предсказуемо при таких переключениях.

С помощью карты Карно определим минимальную реализацию (рис.3.5).

Q \ LD	00	01	11	10
0	0	0	1	0
1	1	1	1	0

На рис.3.5 видно, что схему DL-триггера можно представить в виде композиции несимметричного RS-триггера и схемы его возбуждения (рис.3.6).

$Q^{+}=L \bullet D \vee \overline{L} \bullet Q$

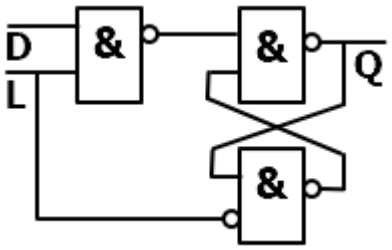


Рисунок 3.5. Несимметричный RS–триггер

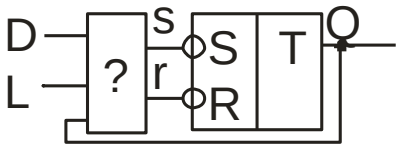


Рисунок 3.6. Структура DL-триггера

Аналогичным образом схемы симметричных DL-триггеров могут быть получены синтезом схемы возбуждения симметричных RS-триггеров. Таблица и карты Карно для синтеза такой схемы представлены на рис.20.

L D	Q	Q^{+}	$(R\ S)^{\vee}$	$(R\ S)^{\&}$
0 0	0	0	X 0	X 1
0 0	1	1	0 X	1 X
0 1	0	0	X 0	X 1
0 1	1	1	0 X	1 X
1 0	0	0	X 0	X 1
1 0	1	0	1 0	0 1
1 1	0	1	0 1	1 0
1 1	1	1	0 X	1 X

Q \ LD	00	01	11	10
0	X	X	0	X
1	0	0	0	1

$R^{\vee} = L \bullet \overline{D}$

0	0	0	1	0
1	X	X	X	0

$S^{\vee} = L \bullet D$

0	X	X	1	X
1	1	1	1	0

$R^{\&} = \overline{L} \vee D = \overline{L \bullet \overline{D}}$

0	1	1	0	1
1	X	X	X	1

$S^{\&} = \overline{L} \vee \overline{D} = \overline{L \bullet D}$

Избавляясь от инверсии D в функции возбуждения $R^{\&}$, получим $R^{\&} = \overline{L \bullet S}$

Схемы DL-триггеров защёлок с симметричными выходами в различных базисах – на рис.3.7.

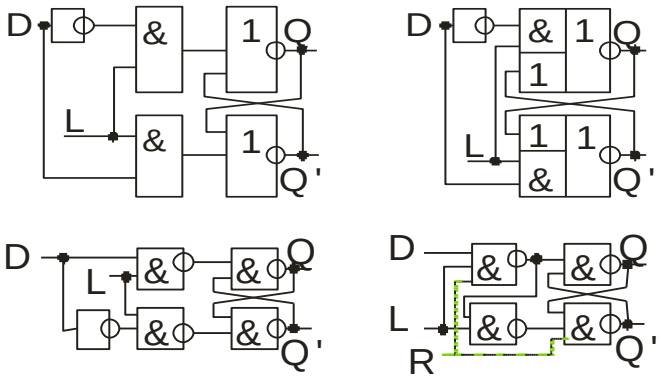


Рисунок 3.7. Схемы DL-триггеров защёлок

Особенность электроники элементов И-ИЛИ-НЕ в том, что они переключаются со скоростью одного элемента, а не как двухступенчатые схемы.

3.3. Синхронные триггеры

3.3.1. Синхронный D–триггер

Триггер типа защёлки нельзя использовать в схемах с обратными связями. (Попробуйте построить счётчик по модулю 2 с использованием D–триггера защёлки.) Причина тому – существование режима прозрачности. Триггер, у которого исключён режим прозрачности, изменяет своё состояние (выходные сигналы) только на фронте управляющего сигнала, такой сигнал называют *синхросигналом*. Триггер называют *синхронным триггером*. В функциональном обозначении синхронного триггера вход синхросигнала (C) выделяется графически (рис.3.8).

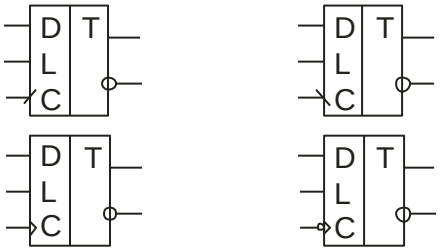


Рис.3.8. Функциональное изображение синхронных D–триггеров

Таблица переключений синхронного D–триггера с отрицательным фронтом синхронизации представлена в таблице, такие триггеры изображены справа на рис.3.8.

C	L	D	Q^+
X	0	X	Q
0	X	X	Q
1	X	X	Q
/	X	X	Q
\	1	0	0
\	1	1	1

Синхронный триггер можно построить на основе «прозрачной защелки», если изменение управляющего сигнала будет кратковременным. Работа такой схемы весьма критична к длительности τ такого кратковременного изменения ($t \le \tau < 2t$, где t время переключения одного элемента).

Исторически первыми синхронные триггеры были придуманы как *двухступенчатые* (*двухтактные*) схемы переключения триггеров защёлок (рис.3.9).

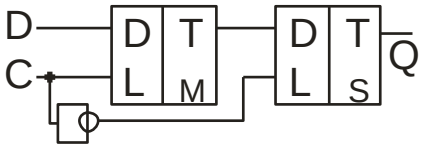


Рис.3.9. Двухтактный синхронный D–триггер

Такие схемы (рис.3.9) называли MS-триггерами (Master-Slave), FlipFlop-триггерами. Известны многочисленные варианты реализации этой идеи. Термин FlipFlop-триггер до сих пор используется в английском для обозначения синхронного триггера.

Современная реализация синхронного D–триггера выглядит иначе (рис.29). В этой схеме переключение на фронте сигнала достигается за счёт эффекта «гонок» сигнала C по цепям разной длины (длина измеряется в количестве логических элементов)

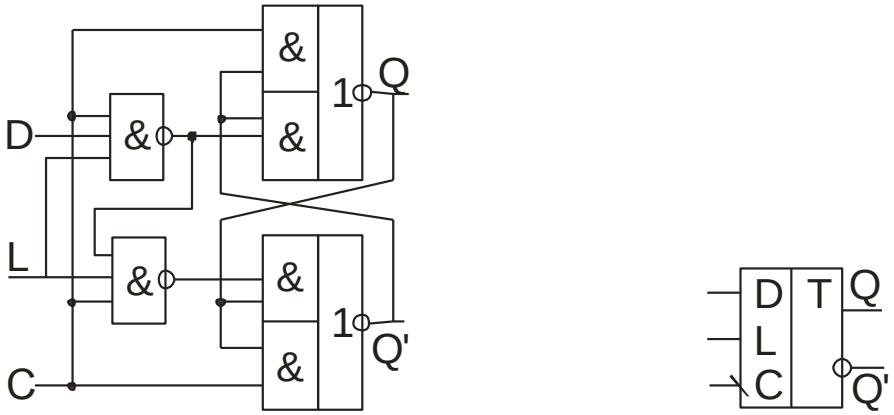


Рисунок 3.10. Схема D–триггера

3.3.2. JK–триггер

Определим ***JK-триггер***, как элемент памяти, функционирующий согласно таблице.

J K	$\mathbf{Q^+}$	
0 0	Q	хранение
1 0	1	установка единицы (установка)
0 1	0	установка нуля (сброс)
1 1	$\overline{Q}$	инвертирование состояния

Карта Карно JK-триггера

JK / Q	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$Q^{+}= \overline{Q} \bullet J \vee Q \bullet \overline{K}$$

Поскольку предусмотрен режим инвертирования состояния, то JK-триггер может быть только синхронным триггером. Синтезируя схему возбуждения RS-триггера, используя карты Карно получаем схему JK-триггера на рис.3.11.

J	**K**	**Q**	**Q^{+}**	**R**	**S**
0	0	0	0	X	1
0	0	1	1	1	X
0	1	0	0	X	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	1	X
1	1	0	1	1	0
1	1	1	0	0	1

JK / Q	00	01	11	10
0	X	X	1	1
1	1	0	0	1

$$R= \overline{Q \bullet K}$$

JK / Q	00	01	11	10
0	1	1	0	0
1	X	1	1	X

$$S= \overline{\overline{Q} \bullet J}$$

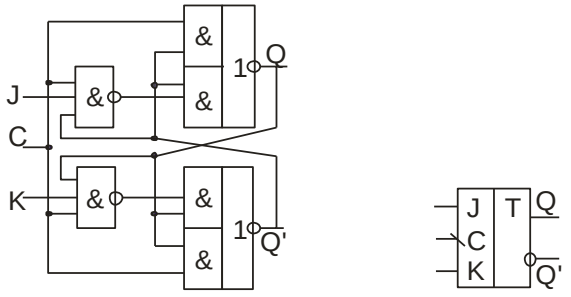


Рисунок 3.11

Если JK–триггер включён в схему с обратными связями, то схема его возбуждения не зависит от собственного выхода. Интуитивно это и так ясно, достаточно посмотреть на схему рис.3.11, на которой видно, что выходы триггера присоединены к входным элементам триггера.

Рассмотрим эту ситуацию более строго. Пусть F(Q,X) означает функцию, зависящую от переменной Q и других переменных, отличных от Q. Тогда, используя разложение Шеннона для функций возбуждения J и K по переменной Q, получим:

$$Q^{+} = \overline{Q} \bullet J(Q,X) \vee Q \bullet \overline{K(Q,X)} =$$
$$= \overline{Q} \bullet J(0,X) \vee Q \bullet \overline{K(1,X)},$$

т.е., если переменная Q присутствует в этих функциях, то её можно заменить константами.

Регистры. Триггеры одного типа объединённые конструктивно или функционально, сигналы управления которых (синхронизация, загрузка, сброс,…) объединены образуют *регистр*.

4. Автономные автоматы

4.1. Определения

СЦА, у которого единственным изменяющимся входным сигналом является сигнал синхронизации, называется *автономным автоматом*.

Автомат, имеющий ***n***-разрядную память, имеет 2^n состояний. Соответственно полный граф переходов автомата имеет 2^n вершин. Комбинационная схема автономного автомата реализует отображение $g{:}Q \to Q$ – множества состояний Q в себя. Поэтому в графе переходов автономного автомата из каждой вершины выходит ровно по одной дуге. Граф автомата может иметь более одного компонента связности. В каждом компоненте связности графа автономного автомата может быть только один цикл, к этому циклу могут быть подвешены деревья, ориентированные в его сторону.

На рис.4.1 приведён пример диаграммы автономного автомата с 2^4=16 состояниями. Разумеется, коды всех 16 состояний различны.

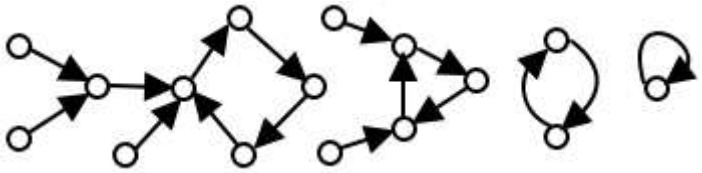


Рисунок 4.1. Диаграмма переходов автономного автомата.

Граф инициального автономного автомата имеет только один компонент связности с числом вершин $N \le 2^n$. Число состояний (вершин) в цикле инициального автономного автомата называют модулем счета.

Любой неавтономный автомат можно сделать автономным, если зафиксировать входной символ. При этом все переходы в новое состояние осуществляются при одном и том же значении входного символа.

В инженерной практике автономные автоматы используются как счетчики, делители частоты, генераторы периодических последовательностей.

4.2. Регистры сдвига

Простейшие из класса автономных автоматов – это регистры циклического сдвига (логические элементы в комбинационной схеме автомата отсутствуют). Регистр циклического сдвига реализует все периоды равные делителям числа n (число триггеров), в зависимости от начального значения, записанного в регистр.

Счётчик в коде Джонсон. Если в регистре циклического сдвига одну из связей между соседними регистрами сделать инверсной, как на рис.4.2a, то наибольший период (модуль счёта) равен 2n (n –число триггеров), при нулевом начальном значении. На рис.4.2b схема с нечётным наибольшим периодом равным 2n–1.

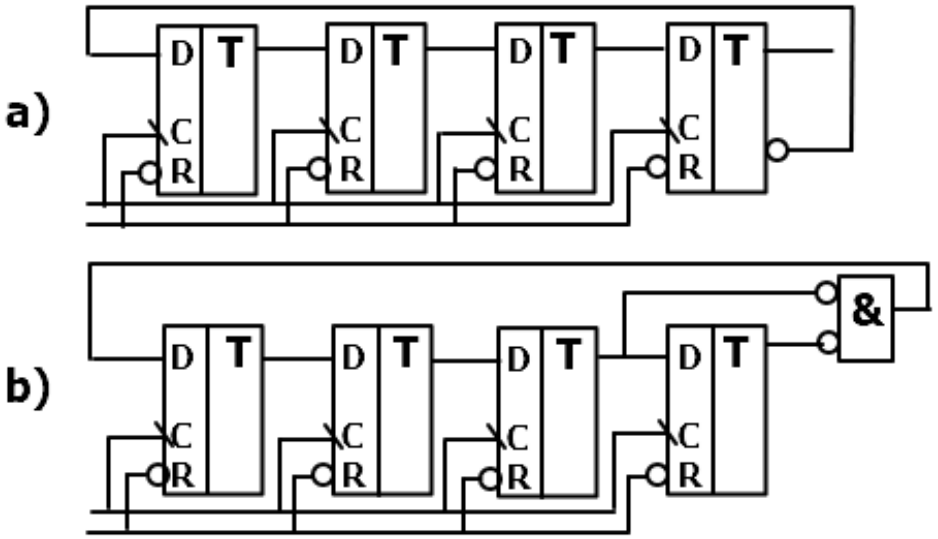


Рисунок 4.2. Счётчик в коде Джонсон

4.3. Параллельная композиция

Параллельная или синхронная композиция автономных автоматов приведена на рис.4.3. Интересны счетные возможности такой композиции, т.е. число состояний в цикле (модуль счета). Этот модуль равен наименьшему общему кратному всех модулей автоматов, входящих в синхронную композицию. Модуль будет наибольшим, если модули автоматов взаимно простые числа – тогда он будет равен их произведению.

4.4. Последовательная композиция

Последовательная или асинхронная композиция синхронных автономных автоматов приведена на рис.4.4. Для переключения автомата АА2 используется изменение значения какого-либо одноразрядного выходного сигнала автомата АА1. Счетные возможности такой композиции максимальны, если сигнал **t** выбран так, что он изменяет свое значение за цикл автомата АА1 не более двух

раз, т.е. переключающий фронт ровно один за цикл. Тогда модуль композиции равен произведению модулей автономных автоматов.

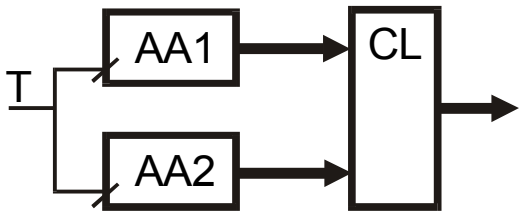


Рисунок 4.3. Параллельное включение АА

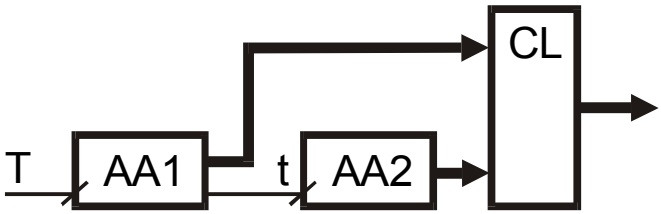


Рисунок 4.4. Последовательное включение АА

4.5. Делители частоты.

Используя автономный автомат с модулем M, можно получить двоичную периодическую последовательность, длина которой может быть равна любому из делителей числа M. Если k – чётный делитель, то можно получить меандр (длительности 0 и 1 равны) с периодом kt. Например, для M=6 делители (2,3,6) меандр (010101) с периодом 2t, меандр (000111) с периодом 6t, последовательности с периодом 3t не могут быть меандром (001001, 011011). Если всё-таки надо получить меандр с периодом kt, где k – нечётное целое, то рис.4.5 и рис.4.6 иллюстрируют как это можно реализовать на примере k=3.

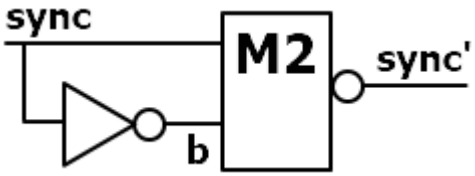


Рисунок 4.5. Преобразование sync

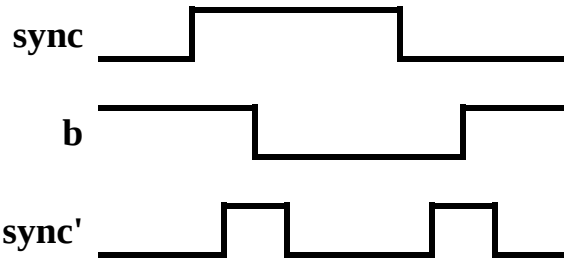


Рисунок 4.6. Временная диаграмма

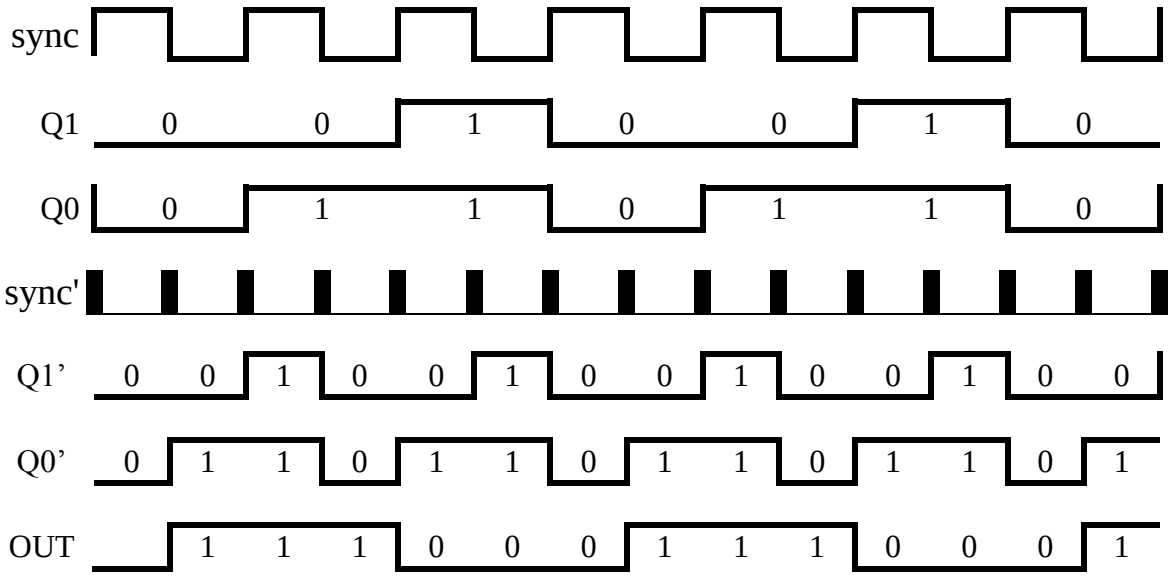


Рисунок 4.7. Временные диаграммы

4.6. Линейные автономные автоматы

Линейные автоматы (ЛА) используются в аппаратуре и программном обеспечении в системах кодирования и декодирования данных, цифровой фильтрации, устройствах обнаружения и исправления ошибок, контроля цифровых схем (сигнатурный анализ).

Синхронный цифровой автомат линеен, если комбинационная логика для вычисления функций перехода и выхода линейна. По определению функция линейна, если она удовлетворяет принципу суперпозиции. Для булевых функций это означает:

$$f(0,...,0)=0,\ f(x_1,x_2,...,x_n)=f(x_1,0,...,0)\oplus f(0,x_2,0,...,0)\oplus...\oplus f(0,...,0,x_n)$$

Комбинационная логика линейна, если используется единственная булевская операция «сложения по модулю 2». Элемент, ее реализующий (М2), будем называть сумматором.

Каждому из одноразрядных выходов КС линейного автомата можно сопоставить один многовходовой сумматор по модулю 2 ($\sum$). Тогда значения состояний могут быть заданы уравнениями:

$$q_i := g_{i1}q_1\oplus g_{i2}q_2\oplus\ldots\oplus g_{in}=\sum_{1\leqslant j\leqslant n} g_{ij}\,q_i\ ,\ i=[1..n]$$

где коэффициенты $g_{ij}\in\{0,1\}$ и означают отсутствие или присутствие булевской переменной на входе сумматора.

Это же уравнение можно рассматривать не как булевское уравнение, а как линейное уравнение. Коэффициенты и переменные имеют только двоичные значения {0,1}. Сложение и вычитание выполняются по модулю 2 (mod2). Поскольку (–1=1)mod2, то операция вычитания по mod2 может быть заменена операцией сложения по mod2, которую и обозначаем знаком +.

$$q_i := g_{i1}\,q_1+\,g_{i2}\,q_2+\ldots+\,g_{in},\ i=[1..n]$$

Уравнения (1) можно переписать в матричной форме: $\mathbf{q}:=\mathbf{G}\,\mathbf{q}$

где $\mathbf{q}=\|q_i\|_{n\times 1}$ – вектор, состояний, $\mathbf{G}=\|g_{ij}\|_{n\times n}$ – матрица, n–количество двоичных элементов памяти.

Матрица $\mathbf{G}=\|g_{ij}\|_{n\times n}$ однозначно определяет схему линейного автомата.

Различные матрицы **G** и $\breve{\mathbf{G}}$ могут определять эквивалентные автоматы, такие матрицы называются *подобными*. Две матрицы подобны тогда и только тогда, когда у них равны характеристические полиномы (det(**M**–*x***I**)), с точностью до мультипликативной константы. Рассмотрим матрицы, которые удобны для реализации с заданным характеристическим полиномом.

Сопровождающей матрицей $\mathbf{M}_{P(x)}$ нормированного полинома

$$P(x) = p_0 + p_1x_1 + p_2x_2 + \ldots + p_{n-1}x_{n-1} + x_n$$

называется (n, n) – матрица:

$$\mathbf{M}_{P(x)} = \begin{Vmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & & 0 \\ & & \cdot & & \\ & & & \cdot & \\ 0 & 0 & 0 & \cdot & 0 \\ 0 & 0 & 0 & & 1 \\ p_0 & p_1 & p_2 & & p_{n-1} \end{Vmatrix} \qquad (4.2')$$

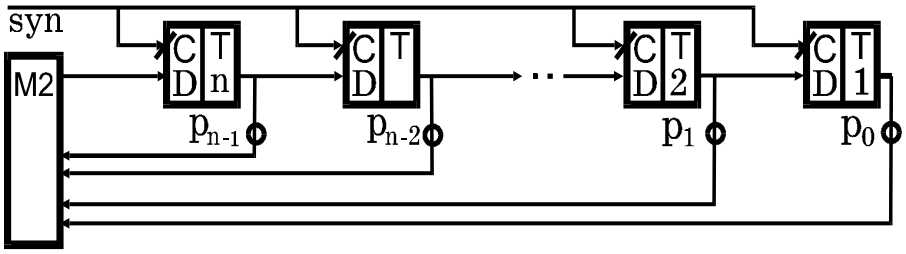
Полином P(x) является характеристическим полиномом (P(x)=det(**M**–x**I**)) этой матрицы.

Матрица $\mathbf{M}_{P(x)}$ подобна своей транспонированной матрице

$$\mathbf{M}^{TP}_{P(x)} = \begin{Vmatrix} 0 & 0 & & 0 & p_0 \\ 1 & 0 & & 0 & p_1 \\ 0 & 1 & & 0 & p_2 \\ & & \cdot & & \\ & & & \cdot & \\ & & & \cdot & \\ 0 & 0 & & 1 & p_{n-1} \end{Vmatrix} \qquad (4.2'')$$

Схема, реализующая сопровождающую матрицу, называется *односумматорным регистром сдвига*.

Схема, реализующая транспонированную сопровождающую матрицу, называется *многосумматорным регистром сдвига*.



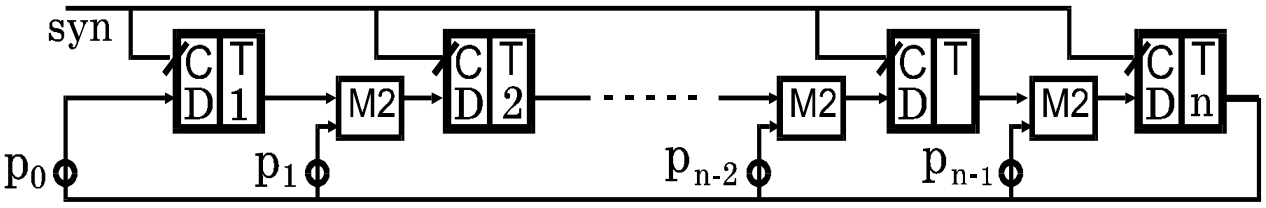


Рисунок 4.8.. Односумматорный регистр сдвига

Рисунок 4.9. Многосумматорный регистр сдвига

Полином P(x) называется *полиномом обратной связи*, поскольку однозначно определяет скалярные константы в цепях обратной связи регистров сдвига.

Если характеристический полином может быть представлен как произведение нормированных полиномов P1(x), P2(x), … , Pk(x), то матрица **M** с харак-

теристическим полиномом P(x) подобна матрице блочно - диагонального вида:

$$\mathbf{M}^* = \left\| \begin{matrix} \mathbf{M}_{P1(x)} & & & \\ & \mathbf{M}_{P2(x)} & & \\ & & \ddots & \\ & & & \mathbf{M}_{Pk(x)} \end{matrix} \right\| \qquad (4.3)$$

Подобие $\mathbf{M}^*$ и $\mathbf{M}$ не нарушается, если меняется порядок блоков или любой блок заменяется транспонированным.

Как и любой автономный автомат линейный автономный автомат порождает периодическую последовательность. При соответствующем выборе полинома её длина максимальна и равна 2^n. Такие последовательности используются в качестве псевдослучайных чисел при некоторых методах шифрования.

4.7. Канонические автономные автоматы

Канонические (типовые) функциональные элементы из класса автономных автоматов.

Счётчики инкремента: Q:=Q+1.

Счётчики декремента: Q:=Q–1.

Реверсивные счётчики: Q:=Q+\–1.

Функциональное обозначение и схема реверсивного счётчика на рис.4.10.

(S1,S0)=(00)–хранение, (01)–инкремент, (10)–декремент, (11)–параллельная загрузка.

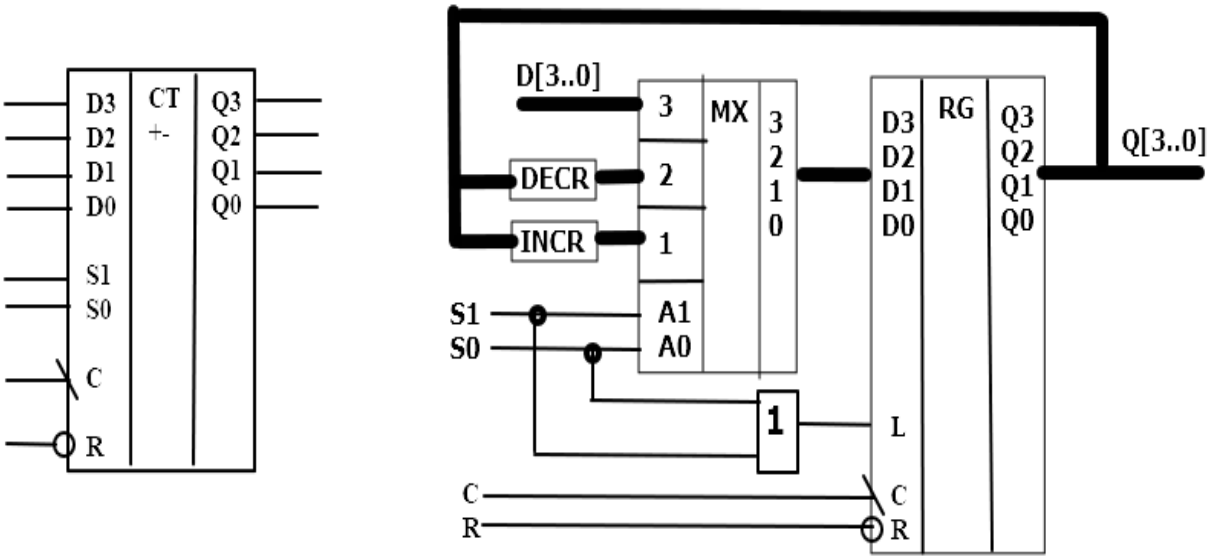


Рисунок 4.10. Канонический счётчик

Регистры сдвига (условно автономные)

Регистры сдвига в одну сторону.

Реверсивные регистры сдвига. Функциональное обозначение и схема 2-х разрядного реверсивного регистра сдвига на рис.4.11.

(S1,S0)=(00)–хранение, (01)–сдвиг к младшим разрядам, (10)–сдвиг к старшим разрядам, (11)–параллельная загрузка. DS–вход для бита, который записывается в освобождающийся разряд. Один входной сигнал DS всё-таки есть.

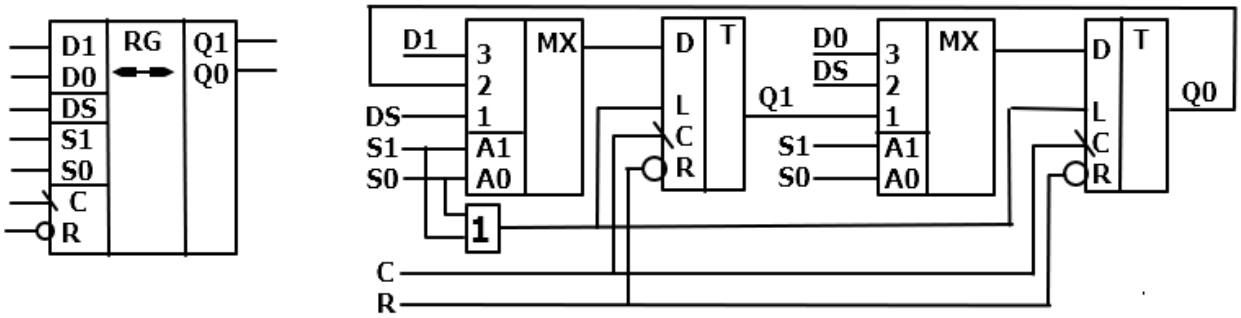


Рисунок 4.11. Регистр реверсивного сдвига

5. Структуры не автономных автоматов

5.1. СЦА полностью и частично определенные

Синхронный автомат *полностью определен*, если функции f и g определены на всех возможных парах из элементов множества *A* входных символов и множества *Q* состояний. Количество элементов этих множеств в СЦА определяется разрядностью кодирующих двоичных наборов. Реализованный СЦА всегда полностью определен.

Если функции перехода и/или выхода определены не всюду, то синхронный автомат - *частично-определенный*. При этом если в каком-либо состоянии не определен переход для входного значения, то предполагается, что это значение не может появиться во входной последовательности. Если не определено выходное значение, то в этом случае оно безразлично. Реализованный автомат всегда полностью определён.

5.2. Автомат Мура

Автомат Мура - синхронный автомат, у которого значения выхода определяются только состоянием автомата в тот же дискретный момент времени. При этом КС, вычисляющая выходное значение, не связана непосредственно с входными сигналами.

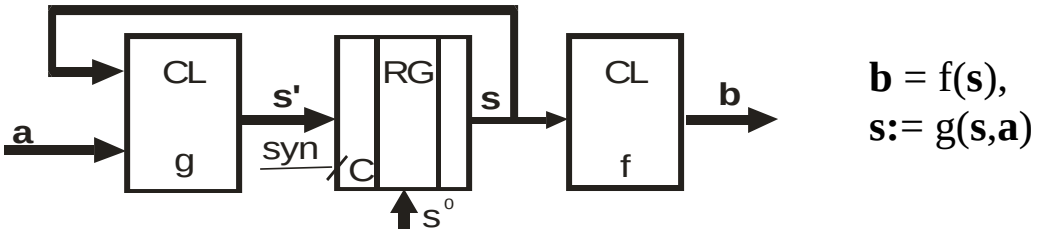


Рисунок 5.1. Автомат Мура

Поэтому момент изменения выходных сигналов зависит только от момента изменения сигнала синхронизации (рабочего фронта). О такой зависимости

между входом и выходом принято говорить, что вход **a** и выход **b** “развязаны во времени”, или иначе - выход автомата “со сдвигом” зависит от входа.

5.3. Автомат Мили

Автоматы, у которых вход и выход не развязаны во времени, т.е. хотя бы один выход зависит от текущего значения на входе, называют автоматами Мили.

В автомате Мили могут одновременно существовать выходные переменные, которые зависят от входных переменных, как со сдвигом, так и без сдвига, как, например, в схеме автомата, приведенной на рис.6.

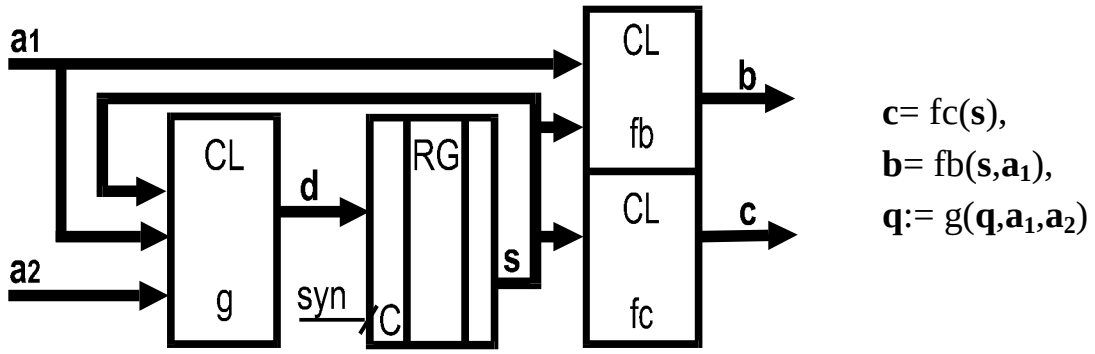


Рисунок 5.2. Автомат Мили

Переменная **c** определена только состоянием автомата, т.е. зависит «со сдвигом» от всех входных переменных. Переменная **b** зависит от переменной $\mathbf{a}_1$ и состояния автомата.

6. Способы описания автоматов

Функции f выхода и g переходов могут быть интерпретированы и заданы самым различным образом в зависимости от цели, для которой проектируется автомат. Например:

1) Функция переходов задаётся как вычисляемая функция. В этом случае СЦА называют СЦА *операционного типа*, например, канонический счётчик $Q:=(Q+1)\mathrm{mod}2^n$, где Q – код состояния; накапливающий сумматор Q:=Q+A+c, где $c\in\{0,1\}$ – входной перенос.

2) Состояние автомата интерпретируется только как идентификатор шага некоторого вычисления (алгоритма) – в этом случае СЦА будем называть СЦА *управляющего типа*.

3) Автомат – *распознаватель* отвечает на вопрос, принадлежит ли поданное на вход слово определённому множеству.

4) Автомат – *преобразователь* преобразует входные слова в выходные, например, в целях кодирования или декодирования информации.

6.1. Автоматная таблица.

Если автомат не операционного типа, то функции f и g могут быть заданы в виде таблицы истинности в любой ее форме, но применительно к конечным автоматам более других распространена форма двухвходовой таблицы, которая называется *автоматной таблицей*. Каждому состоянию автомата соответствует строка таблицы. Заголовками строк служат либо идентификаторы состояний, либо коды состояний. Каждому входному символу соответствует столбец с заголовком в виде символа или его кода. (Разумеется, можно использовать транспонированную таблицу со строками - входными символами и столбцами - состояниями.) Каждая клетка таблицы с координатами [i,j] содержит, в общем случае, два значения: ($\mathbf{b_p}$, $\mathbf{q_s}$), где $\mathbf{b_p} = f(\mathbf{q_i}, \mathbf{a_j})$ - выходное значение, $\mathbf{q_q} := g(\mathbf{q_i}, \mathbf{a_j})$ – новое (следующее) состояние автомата. Таблица автомата Мура, при такой структуре таблицы, будет содержать во всех клетках одной строки одно и тоже выходное значение. Поэтому изображение таблицы автомата Мура будет проще, если клетка содержит только одно значение - новое состояние, но есть столбец с выходными значениями для каждого состояния. Первая строка таблицы для инициальных автоматов, чаще всего, соответствует начальному состоянию автомата.

Пример 6.1.-1. Инкрементор. Автомат - инкрементор с одноразрядным входом (a) и одноразрядным выходом (b). На вход поступает последовательно многоразрядное число в дополнительном коде, начиная с младших разрядов. На выходе синхронно появляется результат - число на единицу большее, чем исходное.

Решение: Пусть автомат имеет два состояния p1 и p0.

p1 - соответствует значению переноса 1, это состояние является начальным;

p0 - соответствует значению переноса 0.

Тогда согласно правилам сложения одноразрядных двоичных кодов, получим таблицу 1. Таблица 2 получена из таблицы 1 присваиванием двоичных кодов состояниям. По таблице 2 могут быть получены таблицы истинности (например, в виде карты Карно) функций f и g.

вход / состояние	0	1
p1	1,p0	0,p1
p0	0,p0	1,p0

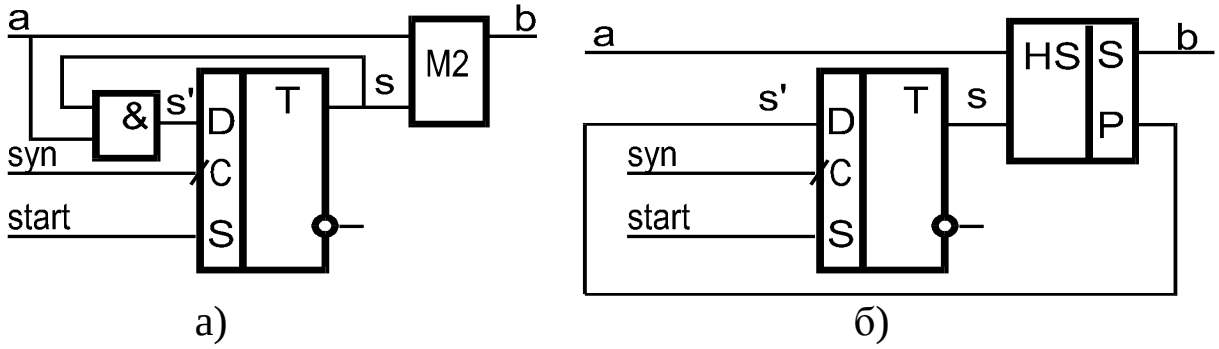
	0	1
1	1,0	0,1
0	0,0	1,0

f	0	1
1	1	0
0	0	1

g	0	1
1	0	1
0	0	0

$$b = s \oplus a \qquad\qquad s' = s \,\&\, a$$

Схема СЦА - инкрементора может быть такой, как на рис.6.1а.



а) б)

Рисунок 6.1. Схема автомата

Для правильной интерпретации работы схемы необходимо условиться о правилах (протоколе) взаимодействия схемы с внешней средой. Например: 1) начальное входное значение и начальное состояние устанавливается при start=1; 2) выходные значения читаются при значениях start=0 и syn=0. Если считать, что функция суммирования одноразрядных двоичных чисел задана (p,s)=HS(x,y), то аналогичную схему можно получить, синтезируя СЦА как автомат операционного типа. В этом случае каноническое уравнение автомата конкретизируется как

(b,s:)=HS(s,a) при s(0)=1.

Соответствующая схема приведена на рис.6.1б.

6.2. Автоматный граф (граф переходов).

При описании автоматов удобно отобразить математическую структуру абстрактного автомата на другую математическую структуру - ориентированного графа.

$$G = (S,D),$$

где S - множество вершин графа, D - множество дуг графа - упорядоченных пар вершин.

Структура графа помогает выделить два объекта математической структуры автомата: состояния − вершины графа и функцию переходов − дуги графа, остальные объекты автомата вводятся как пометки дуг или вершин. Дуги помечаются символами входного алфавита, а символами выходного алфавита помечаются дуги или вершины в зависимости от типа автомата

$G = (\{s_i\},\{d_j[a_k/b_h]\})$ для автомата Мили,

$G = (\{s_i[b_h]\},\{d_j[a_k]\})$ для автомата Мура.

Диаграммой будем называть какое-либо геометрическое изображение графа. Диаграмма автомата - инкрементора может иметь вид рис.8.

Дуги смежные одной и той же паре вершин и одинаково направленные (параллельные дуги) могут быть объединены в одну дугу, помеченную соответствующим образом – рис.8б (a - имя входной переменной). Начальное состояние каким-либо образом помечается.

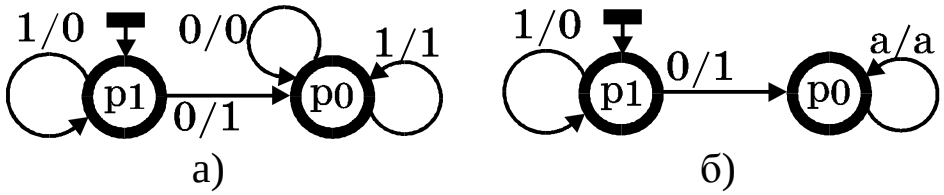


Рисунок 6.2.. Диаграмма автомата - инкрементора

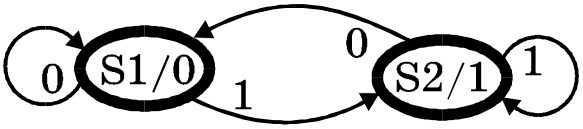


Рисунок 6.3.. Автоматная диаграмма D-триггера

Автоматный граф должен удовлетворять условию однозначности, т.е. не должно существовать двух дуг, выходящих из одной и той же вершины, с одинаковыми входными пометками. Автоматный граф полностью определенного автомата удовлетворяет условию полноты, т.е. для всякой вершины s и для всякого входного символа q имеется дуга, помеченная символом q и выходящая из s.

Примером диаграммы автомата Мура может служить автоматная диаграмма D-триггера (рис.6.3).

Отличие этой диаграммы от предыдущей в том, что выходным значением помечены не дуги, а состояния.

6.3. Блок-схема

Блок-схема автомата управляющего типа – это некоторая графическая вариация диаграммы автомата. В блок-схеме имеются вершины двух типов: 1) операторные (прямоугольные), в которых записывается выходное значение, в общем случае как функция входного значения (при этом одна операторная вершина соответствует одному состоянию автомата); 2) условные или предикатные (непрямоугольные, чаще всего ромбовидные), в которых записываются некоторые логические функции от двоичных входных переменных автомата. Выходящие из условной вершины стрелки помечаются значениями этих функций. Вершины соединяются дугами, показывающими все возможные переходы. Начальная вершина помечается.

Блок-схема должна удовлетворять условию однозначности: если любой путь, соединяющий две операторные вершины, не содержит входных переменных, которые участвует в вычислении условия более одного раза. Блок-схема автомата D-триггера приведена на рис.6.4.

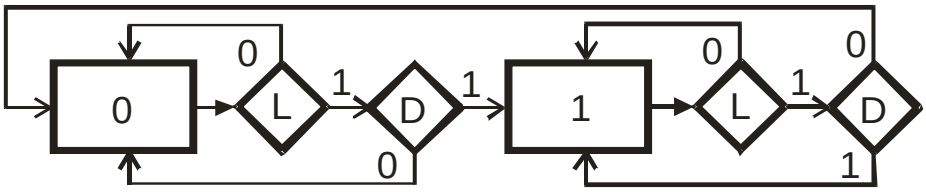


Рисунок 6.4. Блок-схема автомата – синхронного D-триггера

6.4. Блок-текст

Блок-текст − это описание блок-схемы в виде последовательного текста. В блок-тексте используются блоки двух типов операторные и блоки переходов. Операторные блоки − в скобках вида {...}, блоки перехода − в скобках вида <<...>>. И те, и другие блоки могут снабжаться метками, стоящими перед блоком. Операторный блок соответствует операторной вершине блок-схемы. В блоках перехода используется оператор GO в одной из двух форм:

GO m - безусловный переход,

GO (P; m0,m1,m2,...) - условный переход,

где, m0,m1,... - метки блоков, P - значение, интерпретируемое оператором GO как неотрицательное целое число, являющееся порядковым номером метки в списке меток оператора GO. С этой метки должно быть продолжено выполнение алгоритма. Блоки условных переходов соответствуют условным вершинам блок-схемы. Например, см. рис.6.5.

```
p1 {   b = a   }

   <<GO(a;p0,p1)>>

p0 {   b = ā   }

   <<GO p0 >>
```

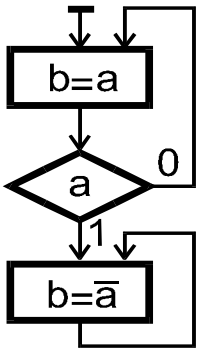


Рисунок 6.5. Блок-текст

7. Эквивалентные автоматы

7.1. Изоморфные и эквивалентные автоматы

Автоматы *изоморфны* если их описание одинаково с точностью до переобозначений.

Два инициальных автомата будем называть *эквивалентными* автоматами, если любую одну и ту же входную последовательность они перерабатывают в одну и туже выходную последовательность. Неинициальные автоматы будем называть эквивалентными, если для любого состояния, взятого в качестве начального одного из автоматов, найдется в другом автомате состояние, назна-

чив которое начальным получим эквивалентность переработки информации входной в выходную.

7.2. Минимальные автоматы

7.2.1. Автомат, эквивалентный заданному и имеющий наименьшее возможное число состояний, называется *минимальным*. Если автоматы всюду определены и эквивалентны, то они имеют изоморфные минимальные автоматы. Для частично-определенных автоматов это положение может не выполняться.

Минимизация автомата возможна, если в автомате есть состояния, которые могут быть объединены в одно состояние. Такие состояния называют *эквивалентными состояниями*. Иначе говоря, автомат минимальный, если у него нет эквивалентных состояний.

Состояния **p, q** одного и того же автомата эквивалентны, если при подаче одной и той же (любой) входной последовательности с начальными состояниями **p** или **q** образуются одинаковые выходные последовательности.

7.2.2. Достаточным условием эквивалентности состояний является совпадение соответствующих им строк в автоматной таблице. Такие состояния называют *явно эквивалентными*. Если строки одинаковы за исключением переходов в себя при одних и тех же входах, то такие состояния так же явно эквивалентны. Поэтому в автоматных таблицах следует переход в себя указывать не именем состояния, а каким-либо одинаковым символом, например таким: $. После такой замены строки могут стать одинаковыми.

7.2.3. Если состояния не явно эквивалентны, то для эквивалентности двух состояний автомата с n состояниями (n>1) достаточно, чтобы совпадали реакции (последовательности выходных значений) этих двух состояний на любые возможные входные последовательности длины, не превышающей n-1.

Определим последовательность отношений эквивалентности $E_0, E_1,\ldots,E_{n-1}$ на множестве состояний Q, которым соответствует разбиение множества Q на классы эквивалентности. В отношении E_0 состояния p и q эквивалентны $(p\equiv q)_0$, если в автомате Мура выходные значения этих состояний одинаковы, в автомате Мили $(p\equiv q)_0$, если в строках, соответствующих этим состояниям, выходные значения одинаковы для одних тех же входов. В отношении эквивалентности E_{i+1} состояния $(p\equiv q)_{i+1}$, если $(p\equiv q)_i$ & $(g(p,t) \equiv g(q,t))_i$ для $\forall t\in A$.

Таким образом при увеличении индекса i класс эквивалентности не может увеличиться, а измельчиться может.

7.2.4. Алгоритм:

1) Явно эквивалентные состояния заменяются одним состоянием.

2) Вычисление $\mathbf{E_0}$ тривиально. Для вычисления следующих отношений эквивалентности в клетки таблицы записываются имена классов эквивалентности каждого из состояний.

3) Вычисление $\mathbf{E_i}$. Состояния из одного класса эквивалентности в отношении $\mathbf{E_{i-1}}$, в строках которых одинаковые комбинации имён классов эквивалентности образуют новый класс эквивалентности или остаются в старом.

4) Если количество различных классов эквивалентности не увеличилось или равно числу состояний, то вычисление закончено, иначе пункт (3).

5) За не более чем n-1 итерацию классы эквивалентности вычислены. Каждому классу эквивалентности последнего разбиения сопоставляется одно состояние.

Пример 7.2-1. Автомат Мура.

A / Q	F	a	b	c	E0
1	0	2	3	1	A
2	0	4	5	2	A
3	0	4	6	3	A
4	0	4	4	6	A
5	1	2	3	6	B
6	1	3	2	5	B

a	b	c	E1
A/2	A/3	A/1	A
A/4	B/5	A/2	A1
A/4	B/6	A/3	A1
A/4	A/4	B/6	A2
A/2	A/3	B/6	B
A/3	A/2	B/5	B

a	b	c	E2
			A
A2/4	B/5	A1/2	A1
A2/4	B/5	A1/2	A1
			A2
A1/2	A1/3	B/6	B
A1/2	A1/3	B/6	B

Объединив эквивалентные состояния, получим таблицу автомата Мура.

A / Q	F	a	b	c
A	0	A1	A1	A
A1	0	A2	B	A1
A2	0	A2	A2	B
B	1	A1	A1	B

Пример 7.2-2. Автомат Мили.

A / Q	a	b	c	E0
1	1,2	0,2	0,4	A
2	0,1	1,3	1,3	B
3	0,1	1,2	1,2	B
4	1,5	0,3	0,1	A
5	0,7	1,8	1,5	B
6	1,5	0,2	0,7	A
7	1,3	0,3	0,6	A
8	0,6	1,8	1,6	B

a	b	c	E1
B/2	B/2	A/4	A
A/1	B/3	B/3	B
A/1	B/2	B/2	B
B/5	B/3	A/3	A
A/7	B/8	B/5	B
B/5	B/2	A/7	A
B/3	B/3	A/6	A
A/6	B/8	A/6	B1

a	b	c	E2
			A
			B
			B
			A
A/7	B1/8	B/5	B2
			A
			A
			B1

	a	b	c	E3
1				A
2				B
3				B
4	B2/5	B/3	A/3	A1
5				A
6	B2/5	B/2	A/7	A1
7				A
8				B1

a	b	c	E4
			A
			B
			B
			A1
			A
			A1
B/3	B/3	A1/6	A2
			B1

a	b	c	E5
			A
			B
			B
			A1
			A
B2/5	B/2	A1/7	A3
			A2
			B1

Объединив эквивалентные состояния, получим таблицу автомата Мили.

A / Q	a	b	c
A	1,B	0,B	0,A1
B	0,A	1,B	1,B
A1	1,B2	0,B	0,A
B2	0,A2	1,B1	1,B2
A3	1,B2	0,B	0,A2
A2	1,B	0,B	0,A3
B1	0,A3	1,B1	1,A3

7.3. Эквивалентность автоматов Мура автоматам Мили

7.3.1. На практике бывает удобно перейти к эквивалентному автомату, имеющему, быть может, большее число состояний, чем исходный автомат, но зато обладающему некоторыми другими “полезными” качествами.

Так обстоит дело при переходе от произвольно заданного автомата Мили к эквивалентному ему автомату Мура, который обладает “полезным” свойством “развязывать” вход и выход. В таких случаях вместо склеивания состояний применяется, в некотором смысле, обратная операция “расщепления состояний”.

Автомат Мура, полученный как эквивалентный автомату Мили, выдает такие же выходные последовательности, но со сдвигом на такт.

Преобразование автомата Мили в эквивалентный ему автомат Мура покажем на следующем примере.

Пример 7.3.-1.

	α	β
A	0,A	1,B
B	1,A	0,C
C	0,B	1,C

		α	β
A0	0	A0	B1
A1	1	A0	B1
B0	0	A1	C0
B1	1	A1	C0
C0	0	B0	C1
C1	1	B0	C1

		α	β
Aα	0	Aα	Aβ
Aβ	1	Bα	Bβ
Bα	1	Aα	Aβ
Bβ	0	Cα	Cβ
Cα	0	Bα	Bβ
Cβ	1	Cα	Cβ

7.3.2. При замене автомата Мура равносильным ему автоматом Мили, сопоставляем каждой дуге $d_i[a_k]$, направленной к вершине $s_z[b_h]$ в автомате Мура, дугу $d_i[a_k/b_h]$, направленную к вершине s_z в автомате Мили. Затем минимизируем состояния автомата.

8. «Прямое» проектирование

Эффективным методом решения любых задач является метод декомпозиции. Рассмотрим некоторые примеры проектирования автоматов, используя, по возможности, декомпозицию.

Пример 8.1.-1. Синтезировать автомат с одноразрядным входом и одноразрядным выходом, на выходе которого фиксируется по модулю 2 количество единичных блоков с нечетным числом единиц.

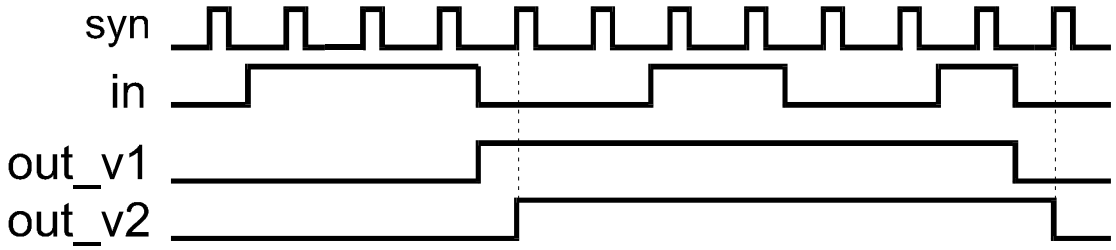


Рисунок 8.1. Временные диаграммы автомата

На временной диаграмме (рис.8.1) приведены два возможных варианта выхода: out_v1 - с “привязкой” изменения к входному сигналу и out_v2 - с “привязкой” изменения к сигналу синхронизации.

Решение 1. В этой задаче нужно считать по модулю 2 как количество блоков, так и количество единиц в блоке. Поэтому возможна следующая композиции автоматов - рис.8.2.

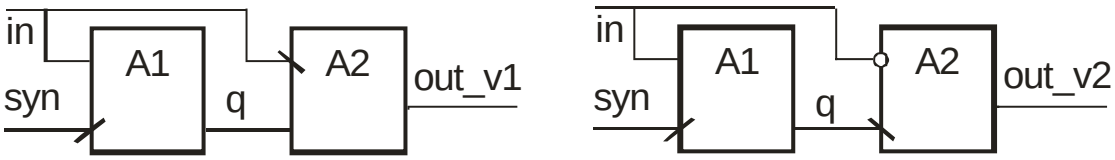


Рисунок 8.2. Декомпозиция автомата

Автомат A1 считает по модулю 2 количество единиц в блоке и сбрасывается в ноль, если нет единиц на входе. Автомат A2 − счетчик по модулю 2 с параметром. В первом варианте A2 синхронизируется отрицательным фронтом входного сигнала in, переключается при единице на q; во втором варианте A2 - синхронизируется отрицательным фронтом сигнала q, переключается при нулевом входном сигнале in. Временные диаграммы (рис.8.3), автоматные таблицы и схема (рис.23) иллюстрируют решение. В схеме использованы JK-триггеры для минимизации комбинационной схемы автомата.

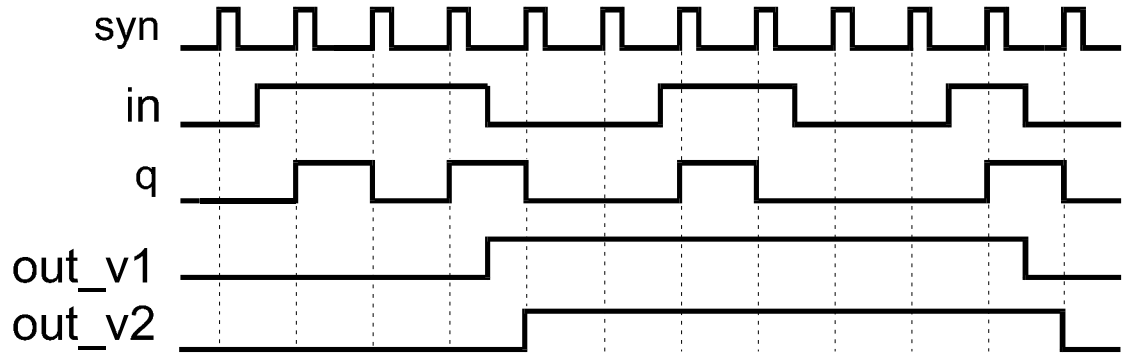


Рисунок 8.3. Временные диаграммы автоматов

Автоматные таблицы и таблицы возбуждения триггеров

A1

s \ in	0	1
0	0	1
1	0	0

A2_V1

s \ q	0	1
0	0	1
1	1	0

A2_V2

s \ in	0	1
0	1	0
1	0	1

J

s \ in	0	1	
0	0	1	J=**in**
1	X	X	

s \ q	0	1	
0	0	1	J=**q**
1	X	X	

s \ in	0	1	
0	1	0	J= $\overline{\textbf{in}}$
1	X	X	

K

s \ in	0	1	
0	X	X	K=1
1	1	1	

s \ q	0	1	
0	X	X	K=**q**
1	0	1	

s \ in	0	1	
0	X	X	K= $\overline{\textbf{in}}$
1	1	0	

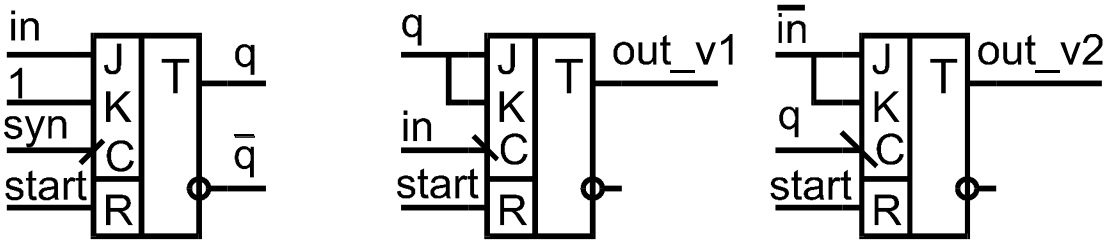


Рисунок 8.4. Схемы автоматов

Решение 2 - без использования декомпозиции. Пусть искомый автомат имеет две группы состояний:

xP - состояния, в которых автомат находится, если в текущей входной последовательности есть нечетное количество блоков с нечетным числом единиц;

xR - все тоже самое, но с четным количеством блоков.

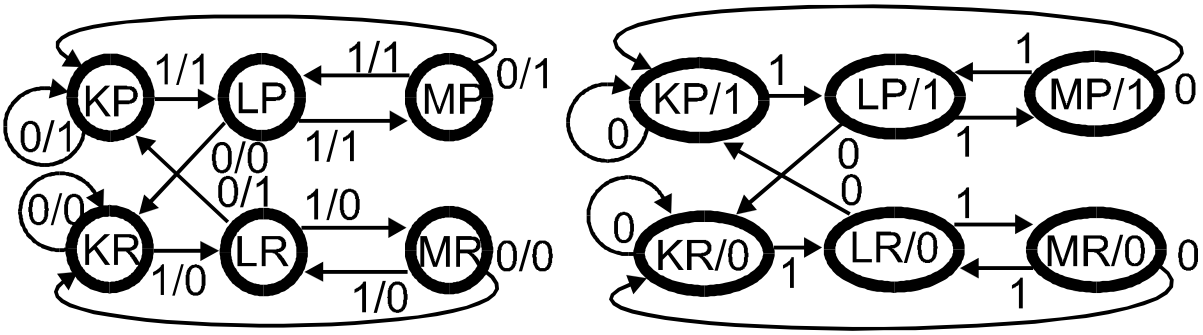
В каждую из групп входят состояния:

Kx - закончился или не начался блок единиц;

Lx - текущее количество единиц в блоке нечетно;

Mx - текущее количество единиц в блоке четно;

Чтобы получить временную диаграмму по варианту 1 (out_v1), надо синтезировать автомат Мили, а, если – по варианту 2 (out_v2), то надо синтезировать автомат Мура.



ВАРИАНТ-1 ВАРИАНТ-2

Рисунок 24. Диаграммы переходов автоматов без декомпозиции

Состояния (KP,MP) и (KR,MR) явно эквивалентны, поэтому в графах останется по четыре состояния.

Если коды состояний для автомата Мура (вариант-2) выбрать так, чтобы начальное состояние (KR) имело код [00], а выходное значение совпадало со значением одного из разрядов кода, то способов кодирования состояний, приводящих к различным схемам, всего 2.

	KR	LR	KP	LP
код1	00	10	01	11
код2	00	10	11	01

После минимизации комбинационной части автомата получаем: при реализации памяти на D-триггерах − переменные di, а для JK-триггеров – переменные (Ji, Ki).

s2,s1 / a	00	01	11	10
0	00	01	00	01
1	10	11	01	00

код1

(d1=a · s2 · $\bar{s}$1 **v** 2 · s1 **v** a · s1) (J1=K1= $\bar{a}$ · s2)

(d2= $\bar{a}$ · s2) (J2= a, K2=1)

s2,s1 / **a**	00	01	11	10
0	00	00	11	11
1	10	11	01	00

код2

(d1= $\bar{a}$·s2 **v** a·s1) (J1= $\bar{a}$ · s2, K1= $\bar{a}$ · $\bar{s}$2)

(d2= a **+** s2) (J2=K2= a)

s2,s1 / **a**	00	01	11	10
0	0	0	1	1
1	0	1	1	0

автомат Мили (ВАРИАНТ-1)

код2

out_v1=d1= $\bar{a}$ · s2 **v** a · s1

Пример 8.1.-2. Синтезировать устройство, которое вычисляет максимальное из нескольких чисел. Каждое из чисел поступает по одноразрядной шине, начиная со старших разрядов. Одноименные разряды всех чисел поступают синхронно. На одноразрядном выходе синхронно появляется результат - значения разрядов максимального из чисел.

Для каждого отдельного числа спроектируем локальный автомат со следующими состояниями:

M - число максимальное - значение на выходе равно входному;

N - число немаксимальное - на выходе автомата 0, т.е. наименьшее значение.

Теперь можно сравнить значения на выходе всех автоматов и выдать максимальное, т.е. выполнить дизъюнкцию, см. рис.8.5.

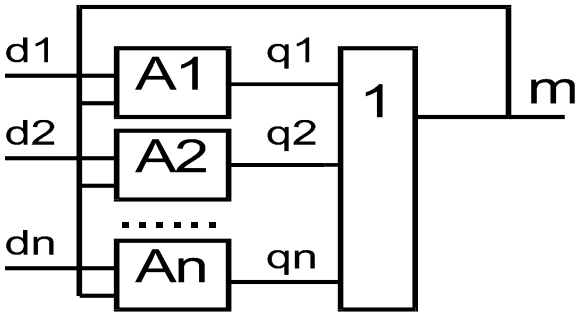


Рисунок 8.5. Структура автомата поиска максимального

Локальный автомат (Ai) переходит из состояния M в состояние N тогда, когда число перестает быть максимальным, т.е. когда значение входного разряда числа меньше значения на выходе дизъюнкции. Выходное значение в состоянии M – равно входному, в состояние N – минимально возможному, т.е. 0.

Выход m, а значит и выход qi, должен зависеть без сдвига от переменной di. В то же время выход автомата qi должен зависеть со сдвигом от переменной m, поскольку это переменная обратной связи. Поэтому структура локального автомата должна быть такой, как на рис.26.

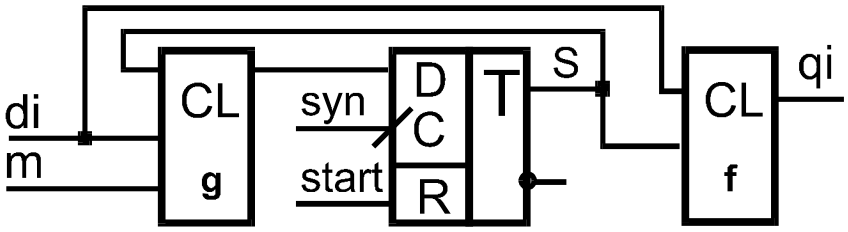


Рисунок 8.6. Схема локального автомата

Работа автомата определяется следующими таблицами:

s \ d,m	00	01	11	10
M	0,M	0,N	1,M	X,X
N	0,N	0,N	0,N	0,N

s \ d..	0	1
0	0,	1,
1	0,	0,

s \ d,m	00	01	11	10
0	,0	,1	,0	,X
1	,1	,1	,1	,1

9. Автоматы распознавания языков

9.1. Слова

9.1.1. Определения. Алфавитом называется конечное непустое множество $A=\{a_1,\ldots,a_n\}$. Его элементы a_t называются *символами*.

9.1.2. Определение: *словом* (*цепочкой*, *строкой, кортежем*) в алфавите A называется конечная последовательность $\tilde{a}_t=(a_{t1},\ldots,a_{tm})$ элементов A.

9.1.3. Определения. Длина слова $\tilde{a}_t=(a_{t1},\ldots,a_{tm})$ обозначаемая $|\tilde{a}_t|=m$, есть число символов в $\tilde{a}_t$. Слово, не содержащее ни одного символа (т.е. последовательность длины 0), называется *пустым словом* и обозначается ε.

9.1.4. Определения. Множество всех слов в алфавите A обозначается A*.
Множество всех непустых слов в алфавите A обозначается A^+.

9.1.5. Определение: если $\tilde{a}$ и $\tilde{y}$ − слова в алфавите A, то слово $\tilde{a}\tilde{y}$ (результат приписывания слова $\tilde{y}$ в конец слова $\tilde{a}$) называется *катенацией* (*конкатенацией*, *сцеплением*) слов $\tilde{a}$ и $\tilde{y}$. Если $\tilde{a}$ − слово и $n \in \mathbb{N}$, то через a^n обозначается слово, состоящее из n символов a. $a^0 \stackrel{\text{def}}{=} \varepsilon$ (знак $\stackrel{\text{def}}{=}$ читается «равно по определению»). Пример: $ba^3 = baaa$, $(ba)^3 = bababa$.

9.1.6. Определение: через $|\tilde{y}|_a$ обозначается количество вхождений символа a в слово $\tilde{y}$. Пример: Если A = {a, b, c}, то $|baaa|_a = 3$, $|baaa|_b = 1$, $|baaa|_c = 0$.

9.1.7. Определения. Слово $\tilde{a}$ − *префикс* (*начало*) слова $\tilde{y}$ (обозначение $\tilde{a} \sqsubset \tilde{y}$, $\tilde{a} = Pref(\tilde{y})$, если $\tilde{y} = \tilde{a}\tilde{u}$. Пример: $\varepsilon \sqsubset baa$, $b \sqsubset baa$, $ba \sqsubset baa$, $baa \sqsubset baa$.
Слово $\tilde{a}$ − *суфикс* (*конец*) слова $\tilde{y}$ (обозначение$\tilde{y} \sqsupset \tilde{a}$, $\tilde{a} = Suf(\tilde{y})$, если $\tilde{y} = \tilde{u}\tilde{a}$.
Слово $\tilde{a}$ – подслово слова $\tilde{y}$, если $\tilde{y} = \tilde{u}\tilde{a}\tilde{v}$.

9.1.8. Определение: *обращением* или *зеркальным образом* слова $\tilde{a}$ (обозначается $\tilde{a}^R$) называется слово, составленное из символов слова $\tilde{a}$ в обратном порядке. Пример: Если $\tilde{u} = baaca$, то $\tilde{u}^R = acaab$.

9.2. Языки

9.2.1. Определение: если $L \subset A^*$, то L называется ***языком*** (*формальным языком*) над алфавитом A.

9.2.2. Определение: пусть $L_1, L_2 \subset A^*$. Тогда $L_1L_2 = \{\tilde{u}\tilde{y} \mid \tilde{u} \in L_1, \tilde{y} \in L_2\}$. Язык L_1L_2 называется *катенацией* языков L_1 и L_2. Пример: Если L_1 = {a, abb} и L_2 = {bbc, c}, то L_1L_2 = {ac, abbc, abbbbc}.

9.2.3. Определение: Пусть $L \subset A^*$, Тогда $L^n = L...L_{n\ раз}$, $L^0 = \{\varepsilon\}$

9.2.4. Определение: *итерацией* языка L (обозначение L^*) называется язык $\bigcup_{n \in N} L^n$.

9.2.5. Определение: пусть $L \subset A^*$. Тогда $L^R = \{\tilde{a}^R \mid \tilde{a} \in L\}$.

9.3. Автоматы

9.3.1. Автомат можно рассматривать как устройство, распознающее некоторый язык над входным алфавитом. Напомним, в инициальном автомате $D=(A,B,Q,g,f,q_0)$ функция $g: Q\times A\to Q$ определяет конечное множество переходов $(p,a,q)\in g$ из p в q, символ a − метка этого перехода, где $p,q\in Q$, $a\in A$. Автомату сопоставлен ориентированный размеченный граф, в котором дуга $(p,a,q)\in g$ соединяет вершину p (начало) с вершиной q (конец), с меткой в виде символа $a\in A$.

9.3.2. Определения. *Путь* конечного автомата − это последовательность $(e_1,e_2,\ldots,e_{n-1})$, где $n \geqslant 0$ и $e_i=(q_{i-1},a_i,qi)\in g$ для каждого i. При этом q_0 − *начало пути*, q_n − *конец пути*, n − *длина пути*, слово $\tilde{a}=a_1\ldots a_n$ − *метка пути*. (Любому конечному пути мы приписываем одно слово, читая символы вдоль этого пути.)

9.3.3. Пусть язык L_y представим в инициальном автомате выходным символом $y\in B$. В автоматном графе инициального автомата языку L_y соответствуют все те слова, которые переводят автомат из начального состояния q_0 к дугам (в автомате Мили) или вершинам (в автомате Мура), помеченным выходным символом y. Слова языка L_y называют также *событиями*.

Путь, меткой которого является слово языка L_y называется *успешным*. Слово $\tilde{a}$ *допускается* конечным автоматом *D*, если оно является меткой некоторого успешного пути.

9.3.4. Определение: *язык, распознаваемый конечным автоматом D*, − это язык *L*(*D*), состоящий из меток всех успешных путей (то есть из всех допускаемых данным автоматом слов). Будем также говорить, что автомат *D распознаёт* язык *L*(*D*).

Если орграф конечного автомата содержит цикл, то количество слов в таком языке бесконечно.

9.3.5. Определение: Язык *L* называется *автоматным*, если существует конечный автомат, распознающий этот язык. Каждый конечный язык является автоматным.

9.3.6. Удобно в качестве распознающего автомата рассматривать разновидность автомата Мура – автомат без выхода $(A,Q,g,q_0,F\subseteq Q)$ с некоторым подмножеством состояний F, которые называются представляющими или финальными состояниями. Язык L представим в автомате без выхода, если L состоит из тех и только тех слов, которые переводят автомат из начального состояния в состояния из множества F. Если начальное состояние принадлежит множеству финальных состояний F, то язык будет включать слово нулевой длины – пустое

слово ε. Пустое слово не следует путать с пустым (невозможным) событием (т.е. с пустым множеством слов). Автомат представляет пустое событие, если ни одно из его финальных состояний не достижимо из начального состояния.

Автомат представляет *систему* (*семейство*) *непустых языков* $\{L_i\}$, если каждый из языков представлен отдельным (своим) подмножеством финальных состояний $\{F_i\}$ или различающимися финальными выходными символами – индикаторами $\{y_i\}$. Пустое слово может принадлежать не более чем одному языку.

9.3.7. Не все языки автоматные. Любое конечное множество слов конечной длины представимо в автомате. В тоже время существуют бесконечные языки, не представимые в автомате. В некоторых случаях по описанию языка это свойство может быть установлено. Например, язык, состоящий из всех слов, в которых число нулей равно числу единиц, не автоматный. Интуитивные соображения состоят в том, что автомат способен “считать” только до числа своих состояний, а поскольку число состояний автомата конечно, то он и не может распознать все слова из этого языка (разница между количеством нулей и единиц может быть неопределённо большой). Но, в общем случае, задача установления представимости языка алгоритмически не разрешима.

В силу сказанного следует, что надо, по возможности, пользоваться средствами спецификации, обеспечивающими представимость языка. Перечислим некоторые из таких способов.

9.3.8. Дефинитный (определенный) язык может быть представлен как катенация $A^*\Omega$, где A – входной алфавит, Ω – конечное множество слов ограниченной длины. Этот язык является бесконечным языком из слов, заканчивающихся словами $\tilde{\omega}\in\Omega$.

Проектируя автомат, распознающий дефинитный язык, удобно сопоставить состояниям автомата все различные ***префиксы*** (***начала***) ***распознаваемых слов***. При поступлении нового символа автомат переходит в состояние, которое соответствует одному из префиксов, совпадающему с суффиксом нового слова, если таких совпадений более одного, то выбирается самое длинное. Одно из состояний должно соответствовать пустому началу. Будем его обозначать символом Λ.

Пример 9.3.-1. Спроектировать автомат, который устанавливает на выходе 1, если последние 4 такта это последовательность 0110.

Автомат должен распознать в последовательности символов следующее слово: 0110. Этому слову соответствуют следующие различные префиксы: Λ, 0, 01, 011, 0110. Тогда автомат Мура задается следующей автоматной таблицей:

№	сост.	вых.	0	1
0	Λ	0	0	Λ
1	0	0	$	01
2	01	0	10	011
3	011	0	0110	Λ
4	0110	1	0	01

Диаграмма автомата Мура на рис.9.1

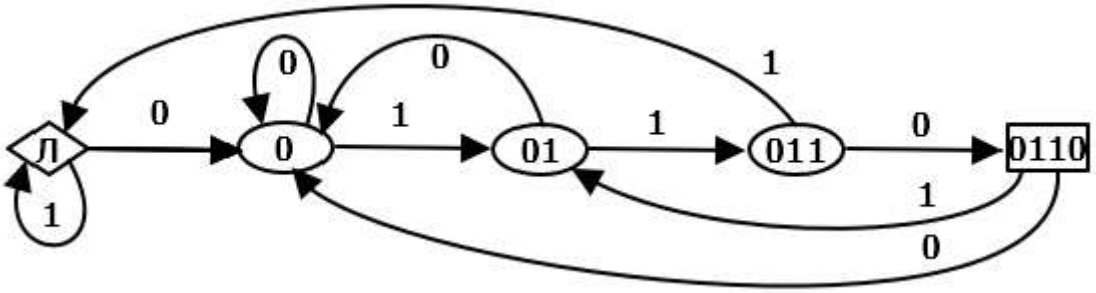


Рисунок 9.1. Автоматная диаграмма Мура

Распознаём тоже самое событие, но автоматом Мили.

№	сост.	0	1
0	Λ	0/0	Λ/0
1	0	$/0	01/0
2	01	10/0	011/0
3	011	0/**1**	Λ/0

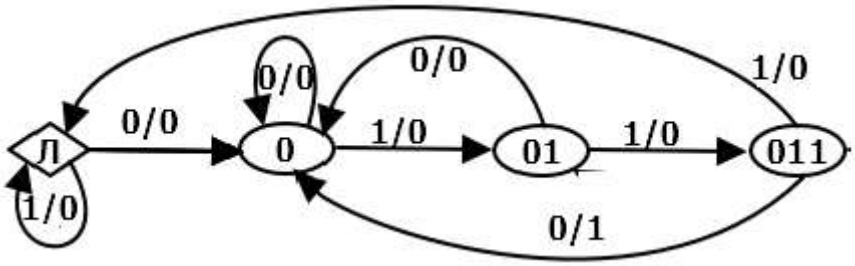


Рисунок 9.2. Автоматная диаграмма Мили

Следующий пример несколько сложнее.

Пример 9.3.-2. Спроектировать автомат, который устанавливает на выходе 1, если последние 4-е такта это последовательность 0110 или 1001.

Автомат должен распознать в последовательности символов следующие слова: 0110, 1001. Этим словам соответствуют следующие различные префиксы: Λ, 0, 1, 01, 10, 011, 100, 0110, 1001. Тогда автомат Мура задается следующей автоматной таблицей:

№	сост.	вых.	0	1
0	Λ	0	0	1
1	0	0	$	01
2	1	0	10	$
3	01	0	10	011
4	10	0	100	01
5	011	0	0110	1
6	100	0	0	1001
7	0110	1	100	01
8	1001	1	10	011

Вариант для автомата Мили.

№	сост.	0	1
0	Λ	0/0	1/0
1	0	$/0	01/0
2	1	10/0	$/0
3	01	10/0	011/0
4	10	100/0	01/0
5	011	0/**1**	1/0
6	100	0/0	1/**1**

9.3.9. Асинхронный язык. В асинхронный язык входят слова, в которых повторение любого из символов может быть произвольным. Точнее, если слово ã принадлежит асинхронному языку, то в языке также содержатся все слова, полученные из ã повторениями любых букв из ã либо вычеркиванием из ã некоторых повторений отдельных букв.

Функция переходов автомата, определяющего асинхронный язык, имеет следующую особенность: для любого состояния q выполняется: если q:=g(z,a), то q:=g(q,a). Автомат может перейти в другое состояние только при изменении входного символа.

Назовем *ядром* асинхронного языка язык, в котором нет повторений символов. В частности, если ядро – дефинитный язык, то при синтезе автомата, распознающего асинхронный язык с таким ядром, можно воспользоваться приемом из пункта 9.3.8. – сопоставить состояниям автомата префиксы распознаваемых слов ядра.

Пример 9.3.-3. Спроектировать автомат с двухразрядным входом (in1,in2) и одноразрядным выходом, который индицирует следующее событие: после нулей на обеих линиях по каждой из линий in1 и in2 прошло ровно по одному стробу любой длительности, строб на линии in2 не начинался раньше чем строб на линии in1.

Обозначим возможные символы (сочетания значений сигналов) на входах автомата следующим образом:

$\frac{in1}{in2}$ $\frac{0}{0}$ = a $\frac{1}{0}$ = b $\frac{0}{1}$ = q $\frac{1}{1}$ = c

Имеем дело с асинхронным языком. Если удалить все повторения символов, то получим дефинитный язык со следующими минимальными последовательностями, которые должен распознать автомат:

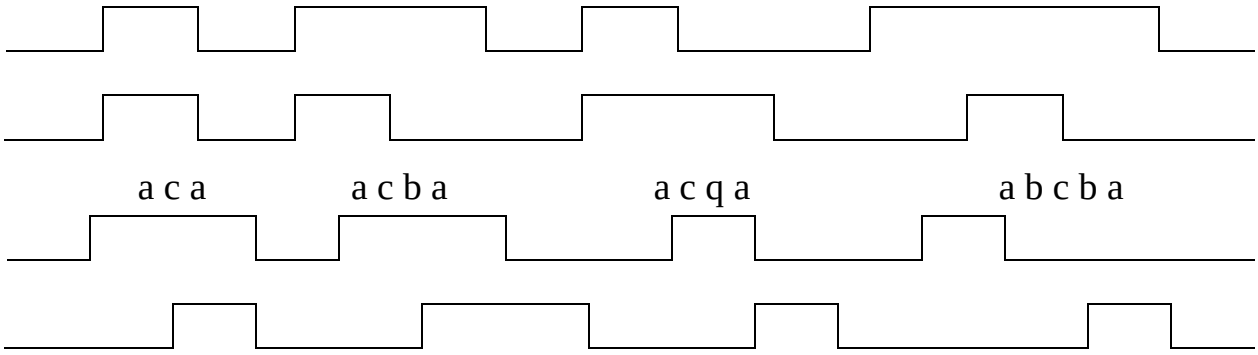


Рисунок 9.3. Временные диаграммы ядерных вариантов слов

Требует уточнения в постановке задачи последний вариант (a b a q a) – может рассматриваться или не рассматриваться. Слегка упростим задачу – пусть не будет этого варианта.

№	Q	Fl	a	b	c	q	эквивалентности
1	Λ		a	Λ	Λ	Λ	
2	a		$	ab	ac	Λ	
3	ac		aca	acb	$	acq	
4	ab		a	$	abc	abq	
5	aca	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
6	acb		acba	$	Λ	Λ	6≡12
7	acq		acqa	Λ	Λ	$	7≡9≡14
8	abc		abca	abcb	Λ	abcq	
9	abq		abqa	Λ	Λ	$	7≡9≡14
10	acba	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
11	acqa	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
12	abcb		abcba	$	Λ	Λ	6≡12
13	abca	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
14	abcq		abcqa	Λ	Λ	$	7≡9≡14
15	abqa	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
16	abcba	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17
17	abcqa	1	$	ab	ac	Λ	5≡10≡11≡13≡15≡16≡17

Из явной эквивалентности всех финальных состояний следует эквивалентность состояний 6≡12 и 7≡9≡14.

Объединив эквивалентные состояния, получим таблицу автомата Мура.

Q	Fl	a	b	c	q
1		2	1	1	1
2		$	4	3	1
3		F	5	$	6
4		2	$	7	6
5		F	$	1	1
6		F	1	1	$
7		F	5	1	6
F	1	$	4	4	1

Вариант для автомата Мили.

№	Q	a	b	c	q	эквивалентности
1	Λ	a/0	Λ/0	Λ/0	Λ/0	
2	a	$/0	ab/0	ac/0	Λ/0	
3	ac	a/**1**	acb/0	$/0	acq/0	
4	ab	a/0	$/0	abc/0	abq/0	
5	acb	a/**1**	$/0	Λ/0	Λ/0	5≡9
6	acq	a/**1**	Λ/0	Λ/0	$/0	6≡8≡10
7	abc	a/**1**	abcb/0	Λ/0	abcq/0	
8	abq	a/**1**	Λ/0	Λ/0	$/0	6≡8≡10
9	abcb	a/**1**	$/0	Λ/0	Λ/0	5≡9
10	abcq	a/**1**	Λ/0	Λ/0	$/0	6≡8≡10

Q	a	b	c	q
1	2/0	1/0	1/0	1/0
2	$/0	4/0	3/0	1/0
3	2/**1**	5/0	$/0	6/0
4	2/0	$/0	7/0	6/0
5	2/**1**	$/0	1/0	1/0
6	2/**1**	1/0	1/0	$/0
7	2/**1**	5/0	1/0	6/0

10. Автоматы без потери информации

Автомат можно рассматривать как устройство, которое осуществляет кодирование входных последовательностей в выходные. Если после такого кодирования можно путем декодирования восстановить исходную информацию, то такие автоматы называют автоматами *без потери информации, IL-автоматами* (Information Lossless).

Рассмотрим только такие IL-автоматы, для которых можно построить *инверсный* автомат, т.е. автомат, восстанавливающий исходную информацию.

Автомат является IL-автоматом тогда и только тогда, когда для каждого состояния выходные значения различны для различных входных значений.

Необходимость этого условия следует из следующего рассуждения. Если в некотором состоянии при различных входных значениях выходное значение одно и тоже, то в этом случае входное значение нельзя восстановить однозначно.

Достаточность условия следует из возможности однозначного определения входного значения по отличающимся друг от друга выходным значениям в каждом состоянии автомата. Построение инверсного автомата очевидно.

Автомат Мура вычисляет выходные значения со сдвигом на один такт. Поэтому автомат Мура является IL-автоматом тогда и только тогда, когда для различных входных значений для каждого состояния следующие состояния имеют различные выходные значения.

ЧАСТЬ II. КОДИРОВАНИЕ ЧИСЕЛ И АРИФМЕТИКА

1. Двоичные числа

Запись числа в однородной позиционной системе счисления с основанием равным q определяется следующим правилом:

$(d_n\, d_{n-1} \ldots\, d_1\, d_0 \,.\, d_{-1}\, d_{-2} \ldots d_{-k+1} d_k)_q =$

$= d_n q^n + d_{n-1} q^{n-1} + \ldots + d_2 q^2 + d_1 q^1 + d_0 + d_{-1} q^{-1} + d_{-2} q^{-2} + \ldots + d_{-k+1} q^{-k+1} + d_{-k} q^{-k}$

где $d_i \in \{0,1,\ldots,q-1\}$. Точка, стоящая между d_0 и d_{-1}, разделяет целую и дробную части числа.

Аналогично в двоичной системе счисления q=2, $b_i \in \{0,1\}$ – бит (bit (BInary digiT)) занимающий разряд с индексом **i** и с весом 2^i.

Например, $(110.101)_2 = 2^2 + 2^1 + 0 + 2^{-1} + 0 + 2^{-3} = (6\ {}^5/_8)_{10}$.

Такое представление числа для краткости будем называть *двоичным позиционным*.

Существует простая связь между записью чисел по основаниям 2 и 2^k:

$$(\ldots b_3\, b_2\, b_1\, b_0 \,.\, b_{-1}\, b_{-2} \ldots)_2 = (\ldots d_3\, d_2\, d_1\, d_0 \,.\, d_{-1}\, d_{-2} \ldots)_{2^k}$$

где $d_j \in \{0,1,\ldots,2^k-1\}$ цифра разряда **j** с весом 2^{kj}

$$d_j = (b_{kj+k-1} \ldots\, b_{kj+1}\, b_{kj})_2$$

В технической документации и текстах программ могут использоваться различные варианты обозначений таких чисел. Для выше приведённого примера, двоичное, четверичное (k=2), восьмеричное (k=3), шестнадцатеричное (k=4) представления могут иметь вид:

$$110.101b = 12.22f = 6.5o = 6.Ah,$$

соответственно. Для шестнадцатеричного представления в языках программирования часто используется запись, начинающаяся с префикса 0x:

$$6.Ah = 0x6.A = 110.101b.$$

При машинных вычислениях длина слов, представляющих числа, фиксирована. Кроме того, форматы с разделительной точкой внутри слова не поддерживаются. Поддерживаются следующие форматы чисел различной, но фиксированной длины:

- целые без знака (только положительные и ноль),

дробные без знака (только положительные меньше единицы и ноль). В стандарте IEEE 754 – представления чисел в экспоненциальном формате (см. ниже) дробные числа с одним разрядом в целой части.

- целые со знаком, в дополнительном коде,
- целые со знаком, в смещённом коде,
- дробные со знаком, в дополнительном коде,
- в формате с плавающей точкой,
- двоично-десятичные числа.

2. Арифметика двоичных чисел

2.1. Дополнительный код числа

2.1.1. Номер двоичного набора. Любому двоичному набору можно сопоставить номер, определяемый функцией: $[\![b_k,\ldots,b_1,b_0]\!] = b_k2^k+\ldots+b_12^1+b_02^0$, т.е. двоичный набор трактуется как целое число без знака.

Канонический n–разрядный сумматор складывает два n–разрядных двоичных набора, получая при этом на выходе (n+1)–разрядную сумму номеров этих наборов. Старший разряд этой суммы называют выходным переносом и обозначают буквой P.

Дополнительным кодом кодируются целые и дробные числа со знаком. При этом, если $\boldsymbol{\alpha, \beta, \sigma}$ – числа со знаком, $A_д$, $B_д$, $C_д$ – их дополнительные коды, $\boldsymbol{\alpha+\beta=\sigma}$, $|\boldsymbol{\sigma}| <2^{n-1}$ то, суммируя дополнительные коды на каноническом n–разрядном сумматоре, получим $A_д+B_д=C_д$. Поэтому такой сумматор называют ещё сумматором дополнительных кодов.

2.1.2. Дополнительный код целых чисел. Пусть $\boldsymbol{\alpha}$ – целое число со знаком такое, что $|\boldsymbol{\alpha}| <2^{n-1}$ ($|\boldsymbol{\alpha}|$ – модуль числа), тогда число $\boldsymbol{\alpha}$ кодируется n–разрядным дополнительным кодом $A_д$, следующим образом:

если $\boldsymbol{\alpha} \geq 0$, то $[\![A_д]\!]= |\boldsymbol{\alpha}| = \boldsymbol{\alpha} <2^{n-1}$, старший разряд равен 0

если $\boldsymbol{\alpha} < 0$, то $[\![A_д]\!]= 2^n-|\boldsymbol{\alpha}|=(2^{n-1} + 2^{n-1} - |\boldsymbol{\alpha}|)>2^{n-1}$,старший разряд равен 1, номер набора без старшего разряда $[\![a_д]\!]= 2^{n-1} - |\boldsymbol{\alpha}|$.

Например, n = 5

число 6 кодируется 00110,

число −6 кодируется 11010.

Старший разряд является индикатором знака. При этом 0 означает число не отрицательное, 1 – число отрицательное. Если придать старшему разряду отрицательный вес (-2^{n-1}), то всегда $\boldsymbol{\alpha} = -b_{n-1}2^{n-1} + [\![a_Д]\!]$. Такая интерпретация полезна при умножении чисел в дополнительном коде (смотри ниже).

2.1.3. Изменение знака. Пусть $\boldsymbol{\alpha} \leftrightarrow A_Д$, $(\boldsymbol{-\alpha}) \leftrightarrow \text{Ä}_Д$, n–разрядные дополнительные коды чисел с противоположными знаками, можно записать:

$$A_Д + \text{Ä}_Д = 2^n, \qquad A_Д + \overline{A_Д} = 2^n - 1$$

где $\overline{A_Д}$ – означает инверсию всех разрядов набора $A_Д$.

Отсюда: $\text{Ä}_Д = \overline{A_Д} + 1$

Теперь операция вычитания сводится к операции сложения следующим образом: $\boldsymbol{\alpha} \leftrightarrow A_Д$, $\boldsymbol{\beta} \leftrightarrow B_Д$, $\boldsymbol{\sigma} \leftrightarrow C_Д$, $\boldsymbol{\alpha - \beta = \sigma}$, $|\boldsymbol{\sigma}| < 2^{n-1}$

$$A_Д + \overline{B_Д} + 1 = C_Д$$

В каноническом сумматоре есть одноразрядный вход, через который эта единица прибавляется. Этот вход называют входным переносом (carry), обозначают чаще всего буквой C.

2.1.4. Дополнительный код дробных чисел. Знаковый разряд дроби имеет вес $-2^0 = -1$. Соответственно для двоичной n–разрядной дроби (со знаком та же дробь на один разряд больше) с весами разрядов

$$(p_0 \bullet p_{-1}\, p_{-2} \ldots p_{-n})_2 = -p_0 + p_{-1}2^{-1} + p_{-2}2^{-2} + \ldots + p_{-n}2^{-n}.$$

Для дополнительных кодов дробей выполняется равенство $A_Д + \text{Ä}_Д = 2$. Поэтому дополнительный код двоичных дробей называют «дополнением до двух» (twos complement). С дробями, в значительной мере, можно обращаться как с целыми числами. Двоичная n–разрядная дробь $0.p_{-1}p_{-2}\ldots p_{-j}\ldots p_{-n}$ может быть представлена как отношение целых чисел ($B/2^n$), где $B = b_{n-1}b_{n-2}\ldots b_{n-j}\ldots b_0$, $(b_{n-j} = p_{-j})$, или как несокращаемая дробь:

$0.p_{-1}p_{-2}\ldots p_{-(k-1)}10\ldots0 = (b_{k-1}b_{k-2}\ldots b_1 1)/2^k$, $(b_{k-j} = p_{-j})$,

где последняя значащая 1 в дроби стоит на (–k)-ом месте, тогда числитель дроби – нечётное число, простые делители знаменателя – только двойки. Например,

$$0.010010000 = 9/32$$
$$1.101110000 = -9/32.$$

2.2. Сложение в дополнительном коде

Канонический сумматор, складывая n–разрядные двоичные наборы, фактически складывает номера этих наборов. Рассмотрим подробнее, что происходит при сложении, если наборы интерпретируются как дополнительные коды целых чисел.

$\boldsymbol{\alpha} \leftrightarrow A_Д$, $\boldsymbol{\beta} \leftrightarrow B_Д$, $\boldsymbol{\sigma} \leftrightarrow C_Д$, $\boldsymbol{\alpha}+\boldsymbol{\beta}=\boldsymbol{\sigma}$,

2.2.1. $\boldsymbol{\alpha} \geq 0$, $\boldsymbol{\beta} \geq 0$, $\boldsymbol{\delta} < 2^{n-1}$,

тогда $A_Д+B_Д = |A_Д| + |B_Д| = \boldsymbol{\alpha}+\boldsymbol{\beta}=\boldsymbol{\delta}=|C_Д|$.

Например,

0010	+2
0101	+5
0111	+7

2.2.2. $\boldsymbol{\alpha} > 0$, $\boldsymbol{\beta} > 0$, $\boldsymbol{\sigma} \geq 2^{n-1}$ (разумеется, $2^n > \boldsymbol{\delta}$)

тогда $A_Д+B_Д = |A_Д| + |B_Д| = \boldsymbol{\alpha}+\boldsymbol{\beta} = \boldsymbol{\sigma} = 2^{n-1}+(\boldsymbol{\sigma} - 2^{n-1})$ – набор с 1 в старшем разряде. Складывая положительные числа, получили отрицательное число. Это - индикатор *переполнения* (overflow).

Например,

0011	+3
0101	+5
1000	–8

2.2.3. $\boldsymbol{\alpha} < 0$, $\boldsymbol{\beta} < 0$, $|\boldsymbol{\sigma}| \leq 2^{n-1}$,

тогда $A_Д+B_Д = |A_Д|+|B_Д| = 2^{n-1}+(2^{n-1}-|\boldsymbol{\alpha}|)+2^{n-1}+(2^{n-1}-|\boldsymbol{\beta}|) =$

$= 2^n+2^{n-1}+(2^{n-1}-|\boldsymbol{\sigma}|) = 2^n+|C_Д|$,

Например,

1110	−2		1101	−3
1011	−5		1011	−5
11001	−7		11000	−8

2.2.4. $\boldsymbol{\alpha} < 0$, $\boldsymbol{\beta} < 0$, $|\boldsymbol{\sigma}| > 2^{n-1}$, (разумеется, $2^n \geq |\boldsymbol{\sigma}|$)

тогда $A_Д+B_Д = |A_Д|+|B_Д| = 2^n - |\boldsymbol{\alpha}| + 2^n - |\boldsymbol{\beta}| = 2^n + 2^n - |\boldsymbol{\sigma}|$

$2^{n-1} > (2^n - |\boldsymbol{\sigma}|) \geq 0$, т.е. получили набор с 0 в старшем разряде. Складывая отрицательные числа, получили положительный результат. Это является индикатором переполнения.

Например,

1101	−3
1010	−6
10111	+7

2.2.5. $\boldsymbol{\alpha} \geq 0$, $\boldsymbol{\beta} \leq 0$, $\boldsymbol{\sigma} = \boldsymbol{\alpha}+\boldsymbol{\beta} = |\boldsymbol{\alpha}|-|\boldsymbol{\beta}|$

тогда $A_Д+B_Д = |A_Д|+|B_Д| = |\boldsymbol{\alpha}|+2^n-|\boldsymbol{\beta}|) = 2^n + \boldsymbol{\sigma}$

если $\boldsymbol{\sigma} \geq 0$ (разумеется, $\boldsymbol{\sigma} < 2^{n-1}$), то $A_Д+B_Д = 2^n + \boldsymbol{\sigma} = 2^n + |C_Д|$.

Например,

0111	+7
1010	−6
10001	+1

если **σ** <0, ($|\sigma| \le 2^{n-1}$), то $A_Д+B_Д=2^n-|\sigma|=2^{n-1}+(2^{n-1}-|\sigma|)=|C_Д|$.

Например,

0110	+6
1001	−7
1111	−1

2.2.6. Изменение знака числа выполняется как операция вычитания и также требует проверки на переполнение.

$$\dot{A}_Д = 0_Д + \overline{A_Д} + 1$$

Например, (0000) – (1000) = (0000) + $\overline{(1000)}$ + 1

	0000
	0111
	1
1000	переполнение

Это единственный вариант переполнения при изменении знака.

2.2.5. Механизм сложения дробей в дополнительном коде аналогичен, только интерпретация весов разрядов другая. Знаковый разряд правильной (меньшей единицы) дроби в дополнительном коде имеет вес минус единица (–1). См. п.2.1.4.

2.3. Сложение и сравнение чисел без знака

При сложении чисел без знака признаком переполнения является единица в самом старшем разряде суммы – выходном переносе.

При сравнении n–разрядных чисел без знака на n–разрядном сумматоре выполняется операция вычитания в дополнительных кодах без использования знаковых разрядов (их значения предопределены). Значение выходного переноса определяет знак результата.

$X + \overline{Y} + 1$. Например:

1101	D	1001	9	1101	D
0110	9	0010	D	0010	D
1		1		1	
10100	+4	**0**1100	–4	**10000**	+0

2.4. Сложение и вычитание в прямом коде

Прямой код (direct code), прежде всего, используется в формате с плавающей точкой для кодирования мантиссы. К основным разрядам, которые являются двоичным позиционным представлением модуля (абсолютного значения)

числа, добавляется ещё один двоичный разряд, который кодирует алгебраический знак числа; ему не присваивается никакого определённого веса, поэтому и расположение его может быть произвольным. Возможна такая трактовка бита знака **s**, если |A| модуль, то число $A=(-1)^S \bullet |A|$. При арифметических манипуляциях с прямыми кодами возможно появление как «минус нуля», так и «плюс нуля».

Результат сложения или вычитания исходных чисел $X \pm Y$ в прямом коде должен быть представлен в прямом коде. В тоже время операндами сумматора должны быть дополнительные коды, выход сумматора – дополнительный код. Поэтому следует минимизировать количество преобразований кодов.

Независимо от знаков чисел и знака операции число X участвует в сложении всегда как |X|. Код второго слагаемого Y зависит от знаков чисел и знака операции и равен либо |Y| либо $(-|Y|)_Д$. Код результата необходимо преобразовать, если сумма отрицательна.

$$X + \overline{Y} + 1 = S, \text{ если } S<0, \text{ то } \overline{S} + 1 = |S|$$

Проще сделать следующим образом:

$$X + \overline{Y} = S, \text{ если } S<0, \text{ то } \overline{S} = |S|$$

Это следует из следующих соображений:

обозначим $X + \overline{Y} = W$, тогда $X + \overline{Y} + 1 = W + 1$

$$\overline{W+1} + W+1 = 2^n - 1$$

$$\overline{W} + W = 2^n - 1$$

отсюда $\overline{W+1} + 1 = \overline{W}$

Если $S \geq 0$, то $X + \overline{Y} + 1 = |S|$

Сложение на сумматоре можно выполнять без знаковых разрядов, поскольку все знаки предопределены. Выходной перенос в одном случае, когда оба операнда сумматора неотрицательные, будет индикатором переполнения, в другом – индикатором знака. Требуется также вычислить знак результата.

Сведём эти соображения в следующие таблицы и алгоритм, где K – знак операции.

Z_X – знак числа X.

Z_Y – знак числа Y.

iY – признак преобразования кода числа, функция от K, Z_X, Z_Y.

P – выходной перенос сумматора.

Z_R – знак результата. Это значение проще вычислять отдельно при разных значениях признака iY (разных ветвях алгоритма).

K	Z_X	Z_Y	iY	P	Z_R
+	+	+	0		+
+	–	–	0		–
–	+	–	0		+
–	–	+	0		–
+	+	–	1	0	–
+	+	–	1	1	+
+	–	+	1	0	+
+	–	+	1	1	–
–	+	+	1	0	–
–	+	+	1	1	+
–	–	–	1	0	+
–	–	–	1	1	–

Если кодировать знак плюс – 0, знак минус – 1, тогда карты Карно:

iY \ Z_XZ_Y / K	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$iY = K \oplus Z_X \oplus Z_Y$

Z_R \ Z_XZ_Y / K	00	01	11	10
0	0	x	x	**1**
1	x	0	x	**1**

При iY=0
$Z_R = Z_X$

Z_R \ Z_XZ_Y / KP	00	01	11	10
00	x	**1**	x	0
01	x	0	x	**1**
11	0	x	**1**	x
10	**1**	x	0	x

При iY=1
$Z_R = \overline{Z_X \oplus P}$

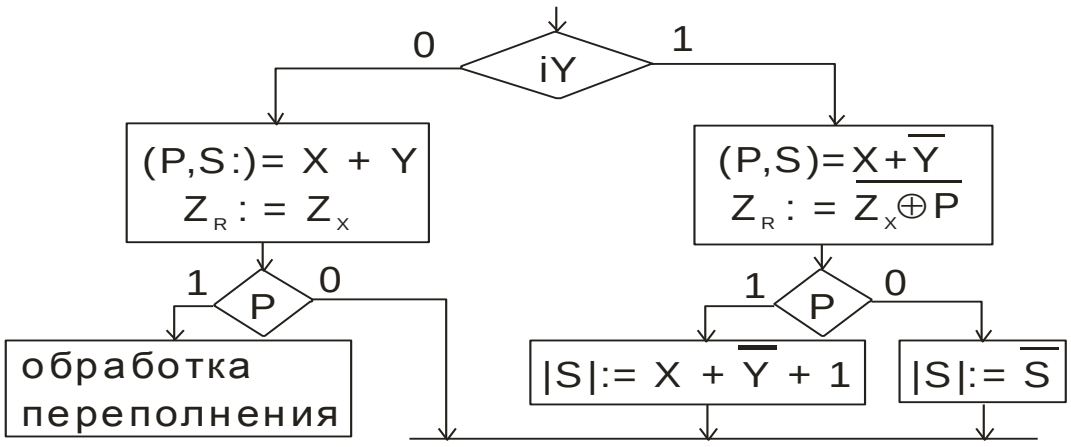


Рисунок 2.1. Алгоритм сложения

2.5. Смещённый код

Смещённый (bias) код, прежде всего, используется для кодирования целых чисел со знаком, определяющих порядок числа в формате с плавающей точкой.

Номера n-разрядных наборов, кодирующих целые числа со знаком **α** в смещенном коде A_C, определяются следующим образом:

$$|A_C| = C + \boldsymbol{\alpha}, \text{ где константа (смещение) } C>0,\ 2^n > (C+\boldsymbol{\alpha}) \geq 0$$

Если для n–разрядного кода выбрать $C=2^{n-1}$, то для $(-2^{n-1} \leq \boldsymbol{\alpha} < 2^{n-1})$

$$|A_C| = 2^{n-1} + \boldsymbol{\alpha},\ ,\qquad \text{если } \boldsymbol{\alpha} \geq 0, \text{ то } |A_C| = 2^{n-1} + |\boldsymbol{\alpha}|$$

$$\text{если } \boldsymbol{\alpha} < 0, \text{ то } |A_C| = 2^{n-1} - |\boldsymbol{\alpha}|.$$

Смещённый код удобен для сравнения чисел, если **α** < **β**, то $|A_C| < |B_C|$. Если $C=2^{n-1}$, то смещённый код отличается от дополнительного кода того же числа только старшим (знаковым) разрядом. Это означает, что смещённый код столь же «арифметичен», как и дополнительный код. Складывая смещённые коды, результат получаем в дополнительном коде. Изменение знака аналогично изменению знака дополнительного кода:

$$(-A)_C = \overline{A_C} + 1$$

Индикатор переполнения для смещённых кодов (смещение=2^{n-1}) такой же (с учётом знаков) как и для дополнительных кодов. Если смещённый код используется как код порядка для чисел в формате с плавающей точкой (см. п.2.9), то код 00…00 обычно резервируется для других целей (означает «машинный ноль»), поэтому получение такого кода после арифметических операций со смещёнными кодами порядков также является переполнением.

2.6. Умножение в дополнительном коде

2.6.1. Умножение целых.

				1	0	1	0	$A_Д \leftrightarrow \boldsymbol{\alpha} = -6$
				1	0	1	0	$B_Д \leftrightarrow \boldsymbol{\alpha} = -6$
				0	0	0	0	0 – частичное произведение
			0	0	0	0	0	$A_Д \bullet b_0$, 1 – частичное произведение
		+	1	0	1	0		$A_Д \bullet b_1 \bullet 2$
		1	1	0	1	0	0	2 – частичное произведение
	+	0	0	0	0			$A_Д \bullet b_2 \bullet 2^2$
	1	1	1	0	1	0	0	3 – частичное произведение
+	**0**	**1**	**1**	**0**				$A_Д \bullet (-b_3 \bullet 2^3)$ умножение на бит знака
0	0	1	0	0	1	0	0	
0	0	1	0	0	1	0	0	2n – разрядный результат = +36

При умножении двух n-разрядных чисел получаем 2n-разрядный результат. Приведём пример умножения «столбиком» двух отрицательных целых чисел в дополнительном коде. Нули младших разрядов множителя (до младшей единицы) порождают только сдвиг нулевого частичного произведения, увеличивая его разрядность. Следует обратить внимание на цифры, выделенные жирным шрифтом, старший разряд суммы всегда – знаковый разряд множимого, не исключая последнего сложения, и даже при умножении на 0 (после младшей единицы множителя).

2.6.2. Умножение дробей в дополнительном коде. Буквальное повторение примера п.2.3.1 с теми же двоичными наборами дает правильный результат, но с двумя знаковыми разрядами.

				1.	0	1	0	$A_Д \leftrightarrow \boldsymbol{\alpha}$= –6/8 = –3/4
				1.	0	1	0	$B_Д \leftrightarrow \boldsymbol{\alpha}$= –6/8 = –3/4
0	**0.**	1	0	0	1	0	0	+9/16

Разумеется, таким будет формат результата при умножении любых наборов, интерпретируемых как двоичные дроби в дополнительном коде. При умножении n–разрядных дробей точный результат, в общем случае, – это 2n-разрядная дробь. Умножение вместе со знаковыми разрядами дает результат на два разряда больше. Обычно при умножении дробей оставляют n старших разрядов.

2.7. Умножение. Аппаратная реализация

Умножение «столбиком» выполняется как итерационный процесс, когда текущее значение частичного произведения G_{i+1} получено из предыдущего значения G_i с учётом множимого A и очередного разряда множителя: $G_{i+1}=G_i+A\bullet b_i\bullet 2^i$, $i=0\div(n-1)$, $G_0=0$, G_n – результат (см. п.2.1.2.). Изменяющийся множитель 2^i делает эту итерацию не удобной для машинной реализации. Для более удобной реализации преобразуем это выражение: $G_{i+1}/2^i=G_i/2^i+A\bullet b_i$, пусть $H_i=G_i/2^i$, тогда $2\bullet H_{i+1}=H_i+A\bullet b_i$, или иначе

$$(H_{i+1},q_i)=(H_i+A\bullet b_i)/2.$$

Справа от равенства в скобках сумма n-разрядных слагаемых с n-разрядным результатом (без n+1 разряда). Делением на 2 (сдвигом вправо) и присоединением старшего разряда равного старшему (знаковому) разряду множимого получаем n старших разрядов частичного произведения и остаток – один бит (q_i), который является i–битом результата (младших разрядов результата) с весом 2^i. $H_0=0$, H_n – старшие разряды результата, $Q=(q_{n-1}\ \ldots\ q_1\ q_0)$ – младшие разряды результата.

A	B	
1010	1010	
H	Q.B	
0000	.1010	
0000	0.101	сдвиг множителя
1010		умножение на 1
1010		сумма
1101	00.10	косая передача и сдвиг множителя
0000		умножение на 0
1101		сумма
1110	100.1	косая передача и сдвиг множителя
0110		умножение на знаковый разряд (–1)
0100		сумма
0010	0100.	косая передача и сдвиг мл. разрядов

Машинная реализация этой итерационной идеи может быть такой:

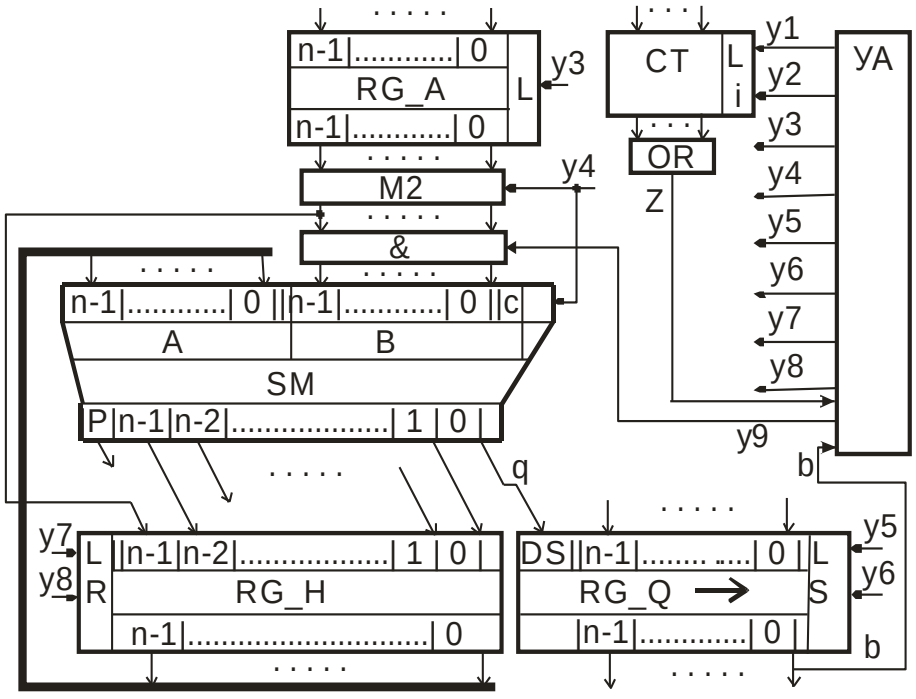


Рисунок 2.2. Схема умножения

Процедурное описание умножения целых чисел со знаком в реализации рис.2 будет следующим:

RG_A:= A -- множимое

RG_Q:= B -- множитель (младшие разряды произведения)

RG_H:= 0 -- старшие разряды произведения

-- В дальнейшем вместо RG_X будем писать X

if (Q = 0) then GoTo end

CT:= n-2

```
while (b=0) do {Q[n-1]:=0; Q:=SR(Q); CT:= CT–1}
while (z =1) do {                          -- (z =1, если CT ≠ 0)
        (H: , q) = (A[n-1],(H + A•b)/2)     -- (b = Q[0])
        Q[n-1]:=q; Q: = SR(Q)
         CT:= CT–1}

    if (b = 1) then {(H: , q) = ( A[n-1] , (H – A)/2)
                        Q[n-1]:=q; Q:=SR(Q)}
               else {(H: , q)=( A[n-1] , (H + 0)/2)
                        Q[n-1]:=q; Q:=SR(Q)}}
  end:;
```

Операция «/2»-деление на 2 обозначает косую передачу вправо. Операция «SR» означает сдвиг вправо (Shift Right).

Числа со знаком и числа без знака умножаются по-разному. Для чисел со знаком при сдвиге (косой передаче) частичного произведения в старший разряд записывается значение знакового разряда множимого. Для чисел без знака при сдвиге (косой передаче) частичного произведения в старший разряд записывается значение выходного переноса сумматора. При умножении без знака, умножение на старший разряд множителя не отличается от умножения на остальные разряды.

числа без знака -		A	B
		1010	1010
	P	H	Q.B
умножение на 0		0000 0000	.1010
сумма	0	0000	
косая передача и сдвиг множителя		0000	0.101
умножение на 1		1010	
сумма	0	1010	
косая передача и сдвиг множителя		0101	00.10
умножение на 0		0000	
сумма	0	0101	
косая передача и сдвиг множителя		0010	100.1
умножение на 0		1010	
сумма	0	1100	
косая передача и сдвиг мл. разрядов		0110	0100.

Особенности умножения дробей, см. п.2.3.2.

2.8. Деление

2.8.1. Деление целых без знака. Рассмотрим алгоритм деления близкий (похожий) на школьный алгоритм деления уголком. Начнём с примера 14/3.

1110b/0011b = 0100b – частное, 0010b – остаток.

14/3=4 – частное, 2 – остаток, при этом 14=3•4+2

Всегда при целочисленном делении A=B•E+R,

где A – делимое, B – делитель, E – частное, R – остаток.

Полное название алгоритма: ***деление с восстановлением и сдвигом остатка***.

Цифры частного получаются последовательно, начиная со старшего разряда: на первом шаге путём вычитания делителя из делимого удвоенной длины, а затем делителя из полученного остатка. Если получен неотрицательный остаток, то цифра частного равна единице, если остаток отрицательный, то цифра частного равна нулю, при этом восстанавливается предыдущий неотрицательный остаток.

	делимое		делитель	
	0000	1110	0011	
			1101	Отрицательный делитель без знакового разряда
частное	остаток			
	0001	110x	сдвиг остатка (делимого)	
	1101		вычитание делителя	
0	1110		отрицательный остаток	
	0001	110x	восстановление положител. остатка	
	0011	10xx	сдвиг остатка	
	1101		вычитание делителя	
1	0000		положительный остаток	
	0001	0xxx	сдвиг остатка	
	1101		вычитание делителя	
0	1110		отрицательный остаток	
	0001	0xxx	восстановление положител. остатка	
	0010	xxxx	сдвиг остатка	
	1101		вычитание делителя	
0	1111		отрицательный остаток	
	0010	восстановленный положительный остаток (результат)		

Формальное описание итерационного процесса:

$$(e_{n-(i+1)}, A_{i+1}) = 2A^{+}_{i} - G,$$

где $G=B\bullet 2^n$, $i=0\div(n-1)$, $A^+_0=A$ – 2n-разрядное делимое, e_k – бит частного с весом 2^k, A^+_n – остаток-результат, A^+_i – неотрицательный восстановленный остаток

$A^+_i = A_i$, если $A_i \geq 0$

$A^+_i = A_i + G$, если $A_i < 0$,

последняя операция восстанавливает неотрицательный остаток. Как вариант можно не запоминать отрицательный остаток, а только положительный.

Деление без восстановления остатка. Итак, в выше рассмотренном алгоритме, если на очередной итерации получен остаток $A_i<0$, то $A_{i+1} = 2A^+_i - G = 2(A_i + G) - G = 2A_i + G$, т.е. вместо восстановления неотрицательного остатка на этом шаге надо прибавить делитель вместо вычитания.

$$(e_{n-(i+1)}, A_{i+1}) = 2A_i \begin{array}{l} - G, \text{ если } A_i \geq 0 \\ + G, \text{ если } A_i < 0 \end{array}$$

Деление более «капризная» операция, чем умножение. Если используется алгоритм «деление без восстановления остатка», то надо добавить старший разряд к обоим операндам (можно считать его знаковым), иначе операции с делителем большим 2^{n-1} будут выполняться неверно. (См. также деление чисел со знаком в дополнительном коде.)

	делимое		делитель	
	00000	01110	00011	
			11101	Отрицательный делитель
частное	остаток			
	00000	1110x	сдвиг остатка (делимого)	
	11101		вычитание делителя	
0	11101		отрицательный остаток	
	11011	110xx	сдвиг остатка	
	00011		сложение с делителем	
0	11110		отрицательный остаток	
	11101	10xxx	сдвиг остатка	
	00011		сложение с делителем	
1	00000		положительный остаток	
	00001	0xxxx	сдвиг остатка	
	11101		вычитание делителя	
0	11110		отрицательный остаток	
	11100	xxxxx	сдвиг остатка	
	00011		сложение с делителем	
0	11111		отрицательный остаток	
+	00011	восстановление положительного остатка		
	00010	восстановленный положительный остаток		

Приведём пример возможной аппаратной реализации:

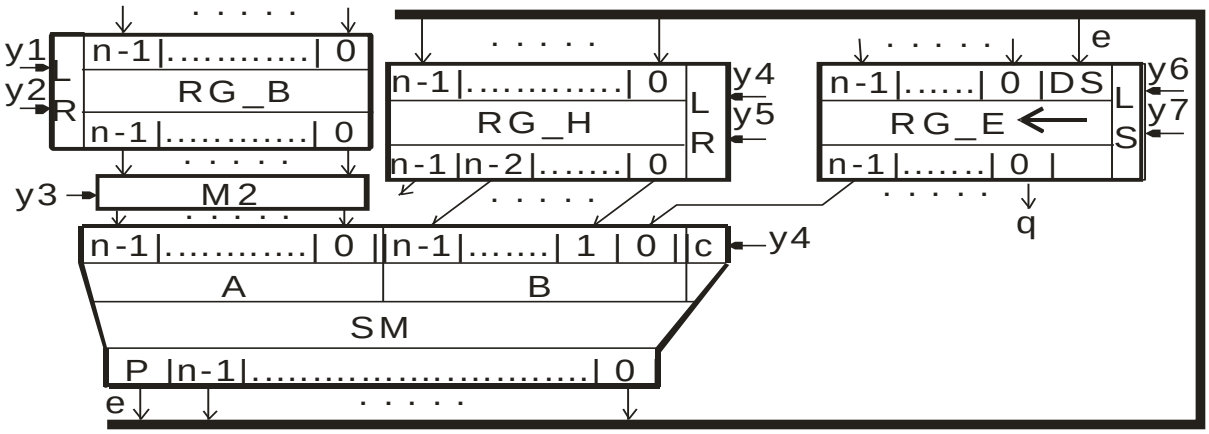


Рисунок 2.3. Схема деления

(Счетчик циклов и управляющая часть на схеме не показаны, см. схему умножения.)

Процедурное описание деления в реализации рис.3 будет следующим:

RG_E:=A -- делимое (частное)

RG_B:=B -- делитель

RG_H:=0 -- старшие разряды делимого (остаток)

-- В дальнейшем вместо RG_X будем писать X

-- ***Деление без восстановления остатка***

CT:= n–1

GoTo mm

```
          -- q = E[0]
loop:  if (q ≠0)      then
 mm:                        {(e , H:) = (2•H , E[n-1]) – B
                             (x , E:)=SL(E , e)
                             CT:=CT–1}
                       else {(e , H:) = (2•H , E[n-1]) + B
                             (x , E:)=SL(E , e)
                             CT:=CT–1}
          z  =(CT≠0)  --  значение  CT  как  в  правой  части  операции
                        -- присваивания
          if (z ≠0) then GoTo loop
```

Операция «2•»- умножение на 2 обозначает косую передачу влево. Операция «SL» означает сдвиг влево (Shift Left).

Сравним алгоритмы с восстановлением и без восстановления остатка. Основной критерий, по которому следует их сравнивать это – суммарное время

выполнения деления. Основной вклад в это время вносит цикл, выполняемый n раз. В алгоритме «с восстановлением» он выполняется минимум за два такта, в алгоритме «без восстановления» – минимум за один такт.

2.8.2. Деление целых со знаком в дополнительном коде.

Можно выполнять деление чисел со знаками непосредственно в дополнительных кодах.

Начнём с примера –13/3.

Частное получилось равное –5. Всегда, если остаток не нулевой, то при целочисленном делении отрицательное частное на единицу меньше правильного результата. (Сравните с делением на 2 отрицательного числа в дополнительном коде сдвигом вправо.) Знак остатка должен быть равен знаку делимого.

	делимое		делитель
	11111	10011	00011
частное	остаток		
	11111	0011x	сдвиг остатка
	00011		сложение с делителем
1	00010		положительный остаток
	00100	011xx	сдвиг остатка
	11101		вычитание делителя
1	00001		положительный остаток
	00010	11xxx	сдвиг остатка
	11101		вычитание делителя
0	11111		отрицательный остаток
	11111	1xxxx	сдвиг остатка
	00011		сложение с делителем
1	00010		положительный остаток
	00101	xxxxx	сдвиг остатка
	11101		вычитание делителя
1	00010		положительный остаток
	11111	восстановленный отрицательный остаток	

Рассмотрим различные варианты сочетания знаков операндов:

A≥0, B>0, частное и остаток – положительные числа.

A≥0, B<0 частное должно быть отрицательными, а получится положительным, придётся изменять знак частного, остаток должен быть положительным.

A<0, B>0 частное должно быть отрицательными и получится отрицательным. Ответ правильный, если остаток ноль (остаток равный –|B| эквивалентен нулевому), если остаток не нулевой, то для получения правильного результата

к отрицательному частному надо прибавить единицу. Остаток должен быть отрицательными.

A<0, B<0 при этом сочетании знаков лучше перед делением изменить знак делимого, тогда частное должно быть положительным и получится положительным. Остаток должен быть отрицательными.

2.8.3. Деление дробей

При делении дробей должно выполняться A<B. Разрядность делимого не надо увеличивать в два раза. При делении дробей без знака надо добавить два разряда знаковый разряд и 0 в старшие разряды дробей, иначе деление на число более ½ может быть неверным. При делении дробей со знаком в дополнительном коде надо разделить обе дроби на 2 (аналог добавления 0 в старшие разряды). Если делимое отрицательное и остаток не нулевой, то полученное отрицательное частное на единицу меньше младшего разряда дроби, чем правильный результат. В некоторых приложениях ошибка в младшем разряде может быть не существенной и тогда её не надо исправлять. Выше описанные алгоритмы деления дробей называют «необратимыми», в том смысле, что полученный в результате работы алгоритма не нулевой остаток неверен. Но при делении дробей чаще всего остаток игнорируется.

частное	делимое		делитель	
	01001	= +9/16	01101	= +13/16
	остаток ± делитель		001101	= +13/32
	001001	= +9/32	110011	= –13/32
	110011	вычитание делителя		
0	111100	отрицательный остаток		
	111000	сдвиг остатка		
	001101	сложение с делителем		
1	000101	положительный остаток		
	001010	сдвиг остатка		
	110011	вычитание делителя		
0	111101	отрицательный остаток		
	111010	сдвиг остатка		
	001101	сложение с делителем		
1	000111	положительный остаток		
	001110	сдвиг остатка		
	110011	вычитание делителя		
1	000001	положительный остаток		

Частное = 01011b = +11/16

Сравним деление модулей чисел со знаком с делением в дополнительном коде. При делении модулей наихудшим надо считать случай A<0, B>0 – при-

дется изменять знак у четырёх чисел. При делении в дополнительном коде надо менять знак только одного числа (не считая остатка), либо скорректировать отрицательное частное на единицу.

2.9. Числа в формате с плавающей точкой

Числа в этом формате содержат два слова:

- поле порядка (характеристики) Pc (binary biased exponent (characteristic)), основанием порядка будем считать 2 (самый распространённый вариант, хотя возможно и 2^k), сам порядок кодируется смещенным кодом. Истинный порядок числа P=(|Pc|–C), где C – смещение.
- поле мантиссы M (mantissa);

Мантисса нормализованная двоичная дробь в дополнительном коде. Понятие *нормализованная* означает, что $1 > |M| \geq ½$.

Истинное значение числа X определяется выражением:

$$X = M \cdot 2^{|Pc|-C}$$

2.9.1. Сложение и вычитание

Пусть $A = m_a \cdot 2^{Pa}$, $B = m_b \cdot 2^{Pb}$, $A \pm B = m \cdot 2^P$,

Где m_a , m_b , m – нормализованные мантиссы ($1 > |m| \geq ½$) со знаком.

Если $P_a \neq P_b$, то надо выравнивать порядки. Это означает, что меньший порядок надо увеличить на величину $\Delta P=|P_a - P_b|$, что означает сдвиг мантиссы числа с меньшим порядком вправо на количество разрядов равное ΔP. Если $\Delta P \geq n$, где n разрядность мантиссы, то результат равен числу с большим порядком с соответствующим знаком.

Порядки выровнены, т.е. $P = Max(P_a , P_b)$. сложение и вычитание , обозначим это действие как $m_a \pm m_b$.

1) Если $|m_a \pm m_b| \geq 1$ (разумеется $|m_a \pm m_b|<2$), то мантисса результата $m = (m_a \pm m_b)/2$, $P= Max(P_a , P_b) + 1$. При выполнении этой операции может произойти переполнение порядка в положительную сторону.
2) Если $1 > |m_a \pm m_b| \geq ½$, то $P = Max(P_a , P_b)$, $m = m_a \pm m_b$.
3) Если $|m_a \pm m_b| < ½$, т.е. мантисса не нормализована. Нормализуя мантиссу, т.е. сдвигая влево надо уменьшать порядок, при этом может произойти переполнение порядка в отрицательную сторону. Реакция, предусмотренная на такое событие, может быть различной, обычно в этом случае формируется код «машинного» нуля.

2.9.2. Умножение.

Пусть $A=m_a \cdot 2^{Pa}$, $B=m_b \cdot 2^{Pb}$, $A \cdot B = m \cdot 2^P$,

Где m_a , m_b , m – нормализованные мантиссы ($1 > m \geq ½$)

$1 > m_a \cdot m_b \geq ¼$, (см. п.2.3.2-умножение дробей)

1) Если $m_a \cdot m_b \geq ½$, то $m = m_a \cdot m_b$, $P = P_a + P_b$.

2) Если $½ > m_a \cdot m_b \geq ¼$, то $m = 2 \cdot m_a \cdot m_b$, $P = P_a + P_b - 1$.

Особые случаи. При выполнении операций с порядками возможны ситуации переполнения:

1) положительное переполнение,

2) отрицательное переполнение – в этом случае формируется код «машинного» нуля.

2.9.3. Деление.

Пусть $A = m_a \cdot 2^{Pa}$, $B = m_b \cdot 2^{Pb}$, $A/B = m \cdot 2^{P}$,

Где m_a , m_b , m – нормализованные мантиссы $(1 > m \geq ½)$

$$max/min > m_a/m_b > min/max$$

$$2 > m_a/m_b > ½,$$ (см. п.2.8.3-деление дробей)

1) Если $m_a = min = ½$, $m_b = max = (1 - 2^{-n})$, то $m = ½$, что не противоречит приведённому выше неравенству, т. к. строго говоря, $m_a/m_b = m + R$.

2) Если $m_a/m_b < 1$, то $m = m_a/m_b$, $P = P_a - P_b$.

3) Если $m_a/m_b \geq 1$, то $m = (½) \cdot m_a/m_b$, $P = P_a - P_b + 1$.

Особые случаи. При выполнении операций с порядками возможны ситуации переполнения:

1) положительное переполнение,

2) отрицательное переполнение – в этом случае формируется код «машинного нуля».

ЧАСТЬ III. УПРАВЛЯЮЩИЕ АВТОМАТЫ

1. Структура вычислительного устройства

При проектировании вычислительного устройства, выполняющего сложную обработку цифровой информации, одним из вариантов декомпозиции является представление синхронного вычислителя в виде композиции двух автоматов операционного и управляющего - рис.1.1.

Операционный автомат (ОА) это, в свою очередь, некоторая композиция из автоматов, но если все регистры этих автоматов синхронизируются одинаково, то ОА можно представлять себе как один автомат Мили. Для исключения гонок по замкнутым цепям управляющий автомат (УА) должен быть автоматом Мура.

Совместная работа этих двух автоматов может быть описана следующим образом. На границе такта (фронт, синхронизирующий регистры автоматов) изменяется содержимое регистров как ОА так и УА и соответственно выходные значения автоматов. Это приводит к тому, что УА формирует новые значения

сигналов для ОА (сигналы эти на рис.1 обозначены как *микрокоманда*). В том же такте ОА сформирует новые значения сигналов для УА (сигналы эти на рис.1.1 обозначены как *признаки*).

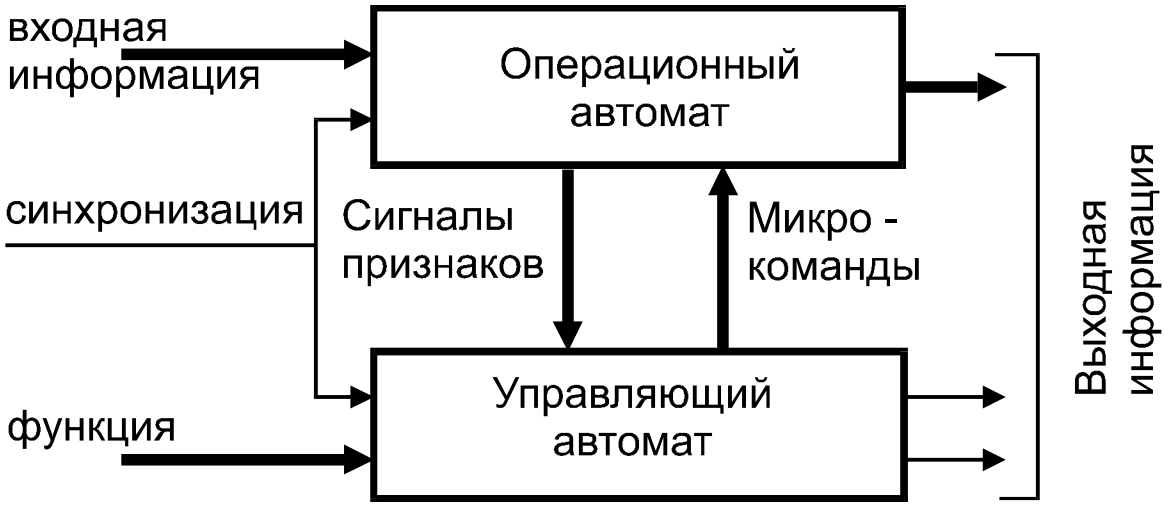


Рисунок 1.1. Структура вычислителя

Приведем также терминологию, сложившуюся в инженерной практике проектирования вычислительных устройств такого типа.

Микропрограмма - любая формализованная форма описания совместной работы операционного и управляющего автоматов - будем использовать *блок-схемы* и *блок-тексты*.

Микрооперация - базисное (элементарное) действие, выполняемое в ОА.

Микроблок - набор микроопераций, выполняемых в ОА одновременно (в одном такте).

Признак (условие) - логическое значение, используемое при переходе к одному из возможных шагов алгоритма; - результат выполнения микрооперации.

Микрокоманда - набор сигналов, поступающий из УА в ОА и интерпретируемый как управляющий, т.е. необходимый для выполнения всех микроопераций одного микроблока.

Микрокомандой также иногда называют слово управляющей памяти, являющейся частью УА. Для различения этих понятий слово управляющей памяти будем называть *микроинструкцией*.

2. Варианты взаимодействия операционного и управляющего автоматов

Схема взаимодействия автоматов, более детальная чем на рис.1.1, представлена на рис.2.1.

Отметим, что сигналы признаков могут быть двух типов

а) $PA(t)=f_A(Y(t))$ зависит без сдвига от сигналов управления,

b) PB(t+1):=f_B(Y(t)) зависит со сдвигом от сигналов управления.

Минимальная длительность такта работы схемы определяется наиболее длинными цепями между регистрами. Для схемы на рис.2.1, которую будем называть последовательной схемой взаимодействия, зададимся (так чаще всего бывает), что такой критической цепью является цепь (CLy,CLa,CLp,RG). Поэтому длительность такта определяется:

$T > t_y + t_a + t_p + t_{rg}$,

где t_j- время установления соответствующего компонента цепи.

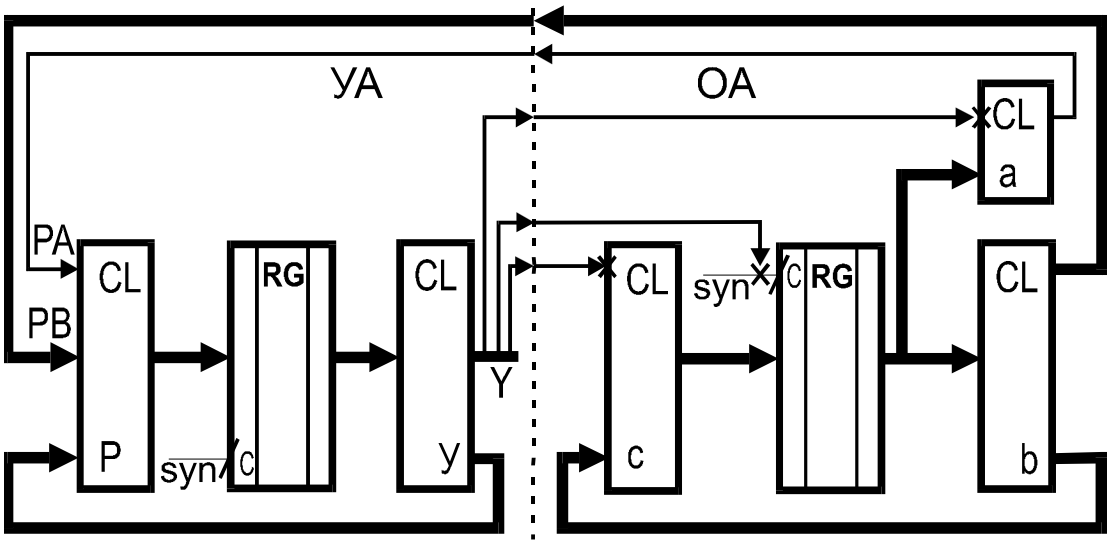


Рисунок 2.1. Последовательный вариант взаимодействия

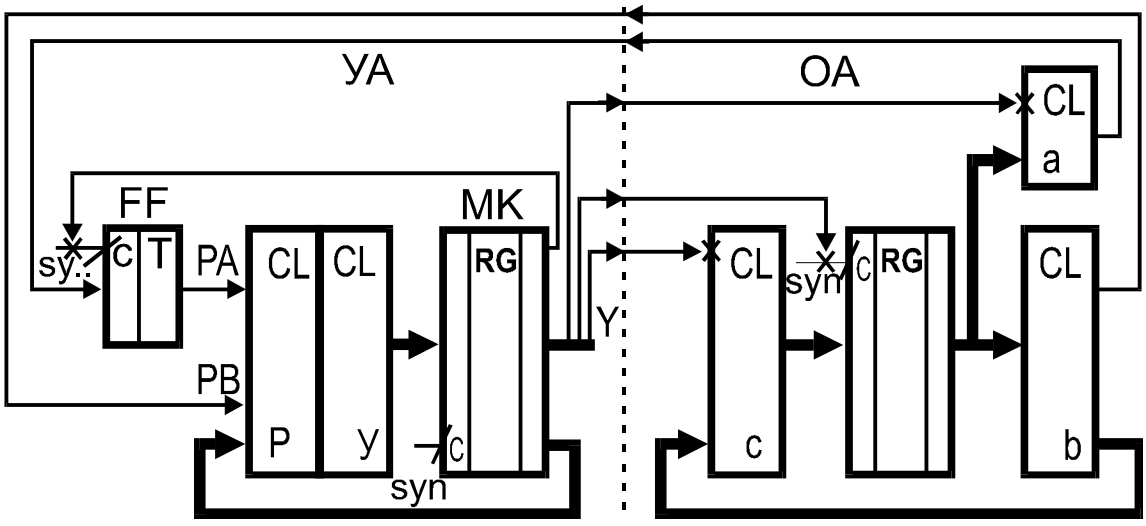


Рисунок 2.2. Конвейерный вариант взаимодействия

Чтобы сократить длину этой цепи, применяют другой вариант взаимодействия автоматов - *конвейерный*- рис.2.2. УА для этого варианта взаимодействия отличается от УА варианта последовательного взаимодействия добавленным регистром RG_FF (регистром флагов) и несколько иной компоновкой регистров и комбинационных схем. При этом варианте взаимодействия такой длинной

цепи, как в предыдущем случае, нет. Эта цепь разделена регистрами RG_FF и RG_MK (регистр микрокоманды) на две цепи. Продолжительность такта становится меньше и ее можно определить следующим образом: $T > \max[t_a , (t_p + t_y)] + t_{rg}$,

При конвейерном варианте взаимодействия PA(t+1):=f(Y(t)), т.е. и эти значения стали зависеть со сдвигом от сигналов управления. Тогда фрагмент микропрограммы

mS{...;pA=f(...)}
«GO(pA;mi,mj)»,

выполняемый в последовательной схеме за один такт, в конвейерном варианте за один такт выполнен быть не может, и должен быть модифицирован следующим образом:

mS {...,pA:=f(...)}
mS'{нет операции}
«GO(pA;mi,mj)»

Таким образом, время выполнения этого фрагмента не только не уменьшилось, но даже возросло, несмотря на уменьшение продолжительности каждого из тактов. Зато, во всех остальных случаях (при безусловных переходах, при переходах по значению РВ) время выполнения микропрограммы уменьшается.

Особенностью реально существующих структур является большая разрядность микрокоманды и малая разрядность (чаще всего 1) анализируемых на каждом шаге признаков. Поэтому является оправданной такая структуризация комбинационной части схемы управляющего автомата, в которой выделяется функция мультиплексирования одного признака на каждом шаге выполнения алгоритма и реализация остальной значительной части логики в виде ПЗУ.

3. Основные способы адресации микрокоманд

В простейшем варианте, в микропрограмме переходы только безусловные. В таком случае УА является автономным синхронным автоматом.

В общем случае функция переходов УА зависит от признаков. Условимся о некоторых ограничениях, позволяющих упростить схему на начальных этапах проектирования (от которых легко впоследствии и отказаться):

- на каждом шаге процесса вычислений ветвление может осуществляться только по одной двузначной переменной - условию (т.е. ветвление возможно лишь на два направления);

- начальные значения всех регистров УА являются нулевыми. Впредь на схемах УА не будем показывать цепей установки начальных значений.

Для реализации, в самом общем случае, микропрограмм произвольной структуры будем строить УА так, чтобы основным материальным носителем управляющей компоненты микропрограммы являлась бы управляющая память, реализованная, например, в виде ПЗУ. В этом случае слово управляющей памяти - *микроинструкция* - состоит из двух составных частей: *микрокоманды* и *адресной части*.

Адресная часть микроинструкции содержит информацию, позволяющую в следующем такте работы выбрать (указать) новый адрес управляющей памяти. Реализация именно этого момента является основным предметом дальнейшего рассмотрения и определяет, в основном, структуру, объем аппаратуры и быстродействие УА. При этом подлежит рассмотрению реализация следующих типов переходов как между шагами алгоритма, так, соответственно, и между микроинструкциями:

- безусловный переход,
- условный переход,
- функциональный переход,
- переход к микроподпрограмме с возвратом.

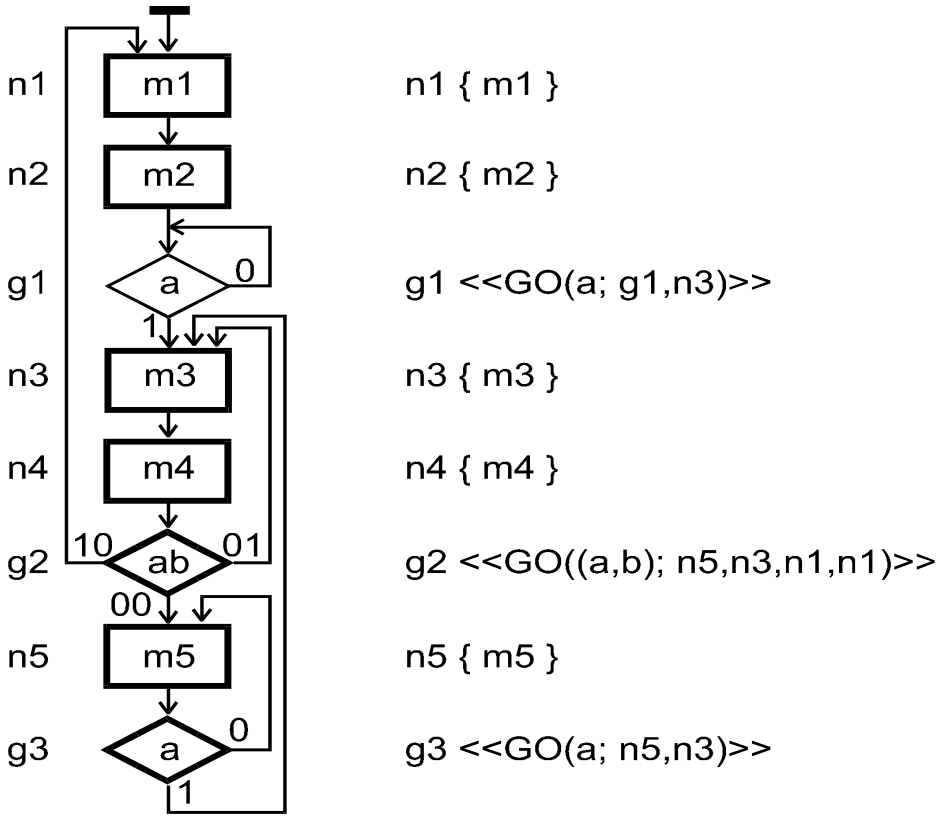


Рисунок 3.1. Блок-схема и блок-текст микропрограммы

Будем изучать работу управляющих автоматов различной структуры, демонстрирующие основные применяемые варианты адресации микроинструкций, на алгоритме, показанном на рис.3.1.

Укажем на некоторые особенности этого алгоритма: Оператор перехода (условная вершина), помеченный меткой g1, называют ждущим. Оператор, помеченный меткой g2, использует для перехода 4-значный признак, что не удовлетворяет вышеуказанному ограничению. Поэтому потребуются эквивалентные преобразования алгоритма для того, чтобы удовлетворить этому ограничению.

Алгоритмы эквивалентны, если они преобразуют информацию одинаковым образом, не обязательно за одно и тоже число тактов. Наиболее распространенным приемом эквивалентного преобразования алгоритмов и микропрограмм является включение микроблоков и, соответственно, тактов, в которых не выполняется модификация памяти ОА - "**нет операции**".

3.1. Схема с адресным ПЗУ

Начнем рассмотрение с управляющего автомата, который называют микропрограммным автоматом Уилкса. Функцию перехода и функцию выхода реализуем в виде ПЗУ.

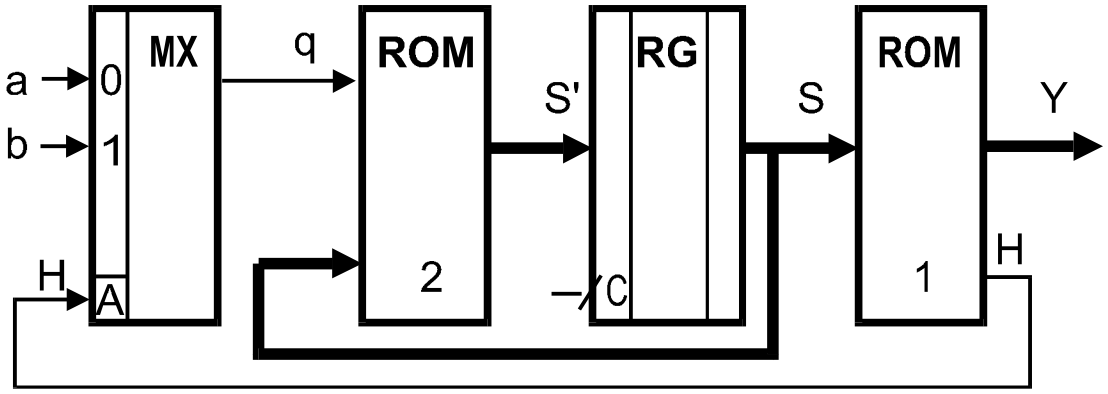


Рисунок 3.2. УА с адресным ПЗУ; последовательный вариант

В ПЗУ (ROM_1), реализующем функцию выхода, следует разместить микрокоманды; при этом их распределение по определенным адресам совершенно произвольно, за исключением начальной микрокоманды, которая должна располагаться по нулевому адресу в силу вышеуказанного ограничения (сброс в ноль RG УА в начальный момент времени). ПЗУ (ROM_2), реализующее функцию переходов автомата, можно трактовать как адресное ПЗУ. Ячеек в адресном ПЗУ в два раза больше, чем в ПЗУ микрокоманд. Каждой ячейке ПЗУ микрокоманд соответствуют две ячейки в адресном ПЗУ, в которых записываются два альтернативных адреса.

Схема УА для конвейерного варианта взаимодействия может остаться прежней или может быть как на рис.2. Выбор схемы зависит от быстродействия цепей ОА, вычисляющих признаки без сдвига. Микропрограмма для конвейерного варианта может измениться, даже если схема УА останется прежней, в си-

лу замечаний, сделанных в разделе "Варианты взаимодействия ОА и УА". Для схемы на рис.2 ограничения на расположение стартовой микрокоманды иное.

		ROM_1			ROM_2		
		S	Y	H	S	q	S’
n1	{ m1 }	0	m1	X	0	0	1
					0	1	1
n2	{ m2 }	1	m2	0	1	0	2
	«GO(a; d1,n3)»				1	1	3
d1	{ m0 }	2	m0	0	2	0	2
	«GO(a; d1,n3)»				2	1	3
n3	{ m3 }	3	m3	X	3	0	4
					3	1	4
n4	{ m4 }	4	m4	0	4	0	5
	«GO(a; d2,n1)»				4	1	0
d2	{ m0 }	5	m0	1	5	0	6
	«GO(b; n5,n3)»				5	1	3
n5	{ m5 }	6	m5	0	6	0	6
	«GO(a; n5,n3)»				6	1	3

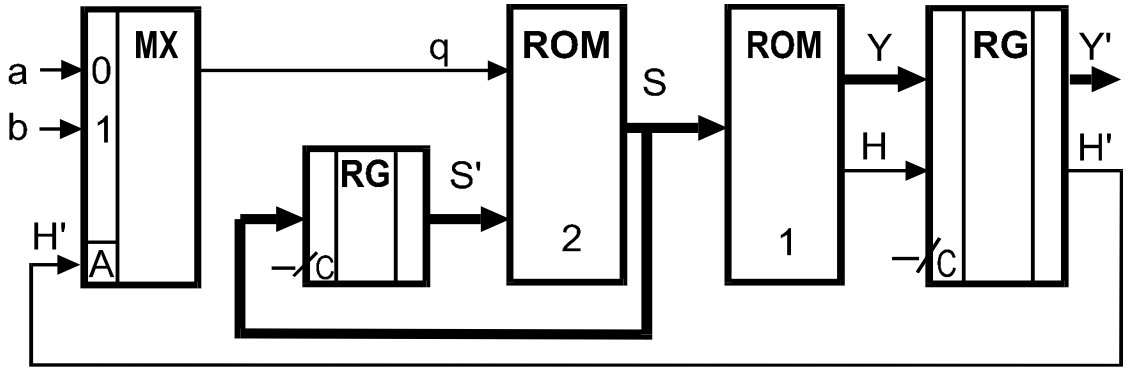
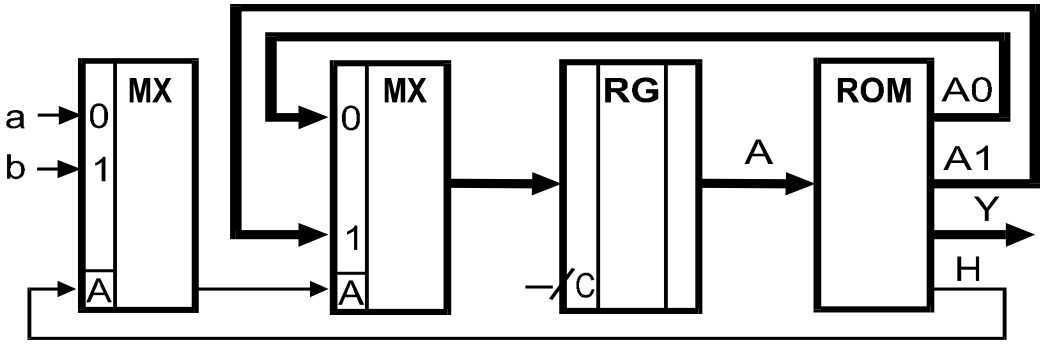


Рисунок 3.3. УА с адресным ПЗУ; конвейерный вариант

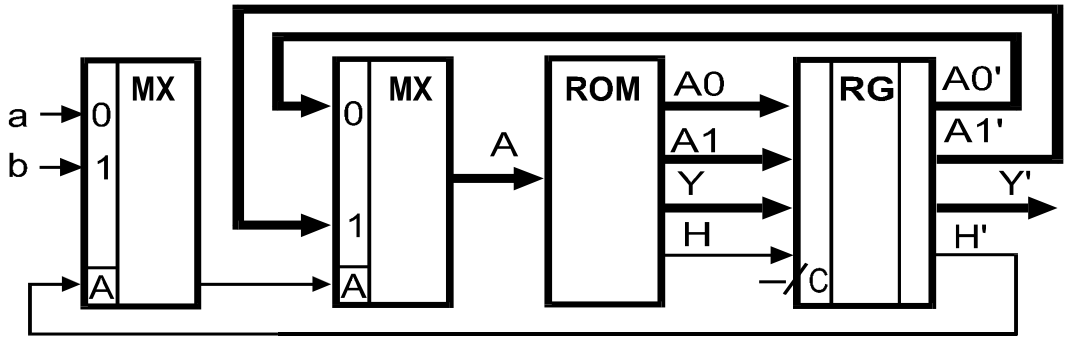
3.2. Схема с двумя адресами в памяти

Эта схема отличается от предыдущей тем, что, по существу, тот же способ адресации выполнен с использованием только одного ПЗУ. В этом варианте альтернативные адреса записываются в той же микроинструкции, что и микрокоманда.

		A	Y	H	A0	A1
n1	{ m1 }	0	m1	x	1	1
n2	{ m2 }«GO(a; d1,n3)»	1	m2	0	2	3
d1	{ m0 }«GO(a; d1,n3)»	2	m0	0	2	3
n3	{ m3 }	3	m3	x	4	4
n4	{ m4 }«GO(a; d2,n1)»	4	m4	0	5	0
d2	{ m0 }«GO(b; n5,n3)»	5	m0	1	6	4
n5	{ m5 } «GO(a; n5,n3)»	6	m5	0	6	4



а) последовательный вариант



b) конвейерный вариант

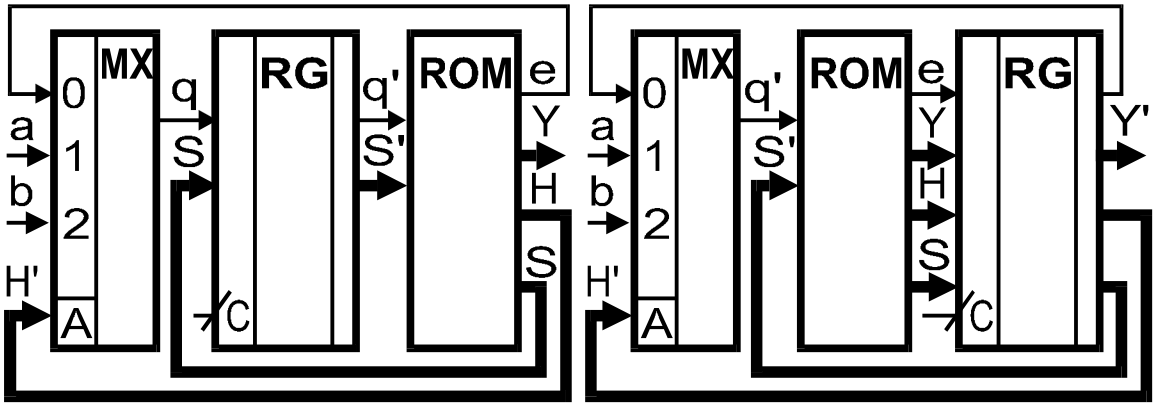
Рисунок 3.4. УА с явным указанием альтернативных адресов

3.3. Схема с одним адресом в памяти

		Адрес
n1	{ m1 }	(0,0),(2,1)
n2	{ m2 }	(4,0)
	«GO(a; d1,n3)»	1,X
d1	{ m0 }	(1,0)
	«GO(a; d1,n3)»	1,X
n3	{ m3 }	(1,1),(3,1)
n4	{ m4 }	(0,1)
	«GO(a; d2,n1)»	2,X
d2	{ m0 }	(2,0)
	«GO(b; n5,n3)»	3,X
n5	{ m5 }	(3,0)
	«GO(a; n5,n3)»	3,X

S’q’	Y	H	S	e
0 0	m1	0	4	0
0 1	m4	1	2	x
1 0	m0	1	1	x
1 1	m3	0	0	1
2 0	m0	2	3	x
2 1	m1	0	4	0
3 0	m5	1	3	x
3 1	m3	0	0	1
4 0	m2	1	1	x

При этом способе адресации альтернативные адреса отличаются только одним разрядом (в данном варианте - младшим), формируемым входным сигналом. Остальные разряды адреса указываются вместе с микрокомандой в одной и той же микроинструкции. При безусловном переходе в данном варианте схемы младший разряд также указывается в микроинструкции.



Последовательный вариант Конвейерный вариант

Рисунок 3.5. УА с одним адресом в памяти

При адресации одного и того же микроблока различными операторами условного перехода может возникнуть *конфликт адресации*. В этом случае одну и ту же микроинструкцию приходится располагать в различных ячейках управляющей памяти. Если различные операторы условного перехода одними и теми же значениями признаков указывают на одни и те же микроблоки, то нет и конфликта адресации.

Распределять микроинструкции по ячейкам памяти удобно в следующем порядке:

- связать с различными операторами условного перехода, конфликтующими между собой по адресации, различающиеся между собой старшие разряды адреса;
- распределить микроблоки по ячейкам памяти с учетом назначенных старших разрядов адреса и входных значений, формирующих младший разряд адреса;
- оставшимся нераспределенным микроблокам назначить произвольные свободные адреса.

3.4. Схема с сокращенным тактом

Использование этой схемы позволяет при сохранении преимуществ последовательного варианта взаимодействия сократить наиболее длинные цепи, общие для ОА и УА, до длины цепей конвейерного варианта.

ROM 0

A’	Y	H	A	e
0	m1	0	4	0
1	m0	1	1	x
2	m0	2	3	x
3	m5	1	3	x
4	m2	1	1	x

ROM 1

A’	Y	H	A	e
0	m4	1	2	x
1	m3	0	0	1
2	m1	0	4	0
3	m3	0	0	1

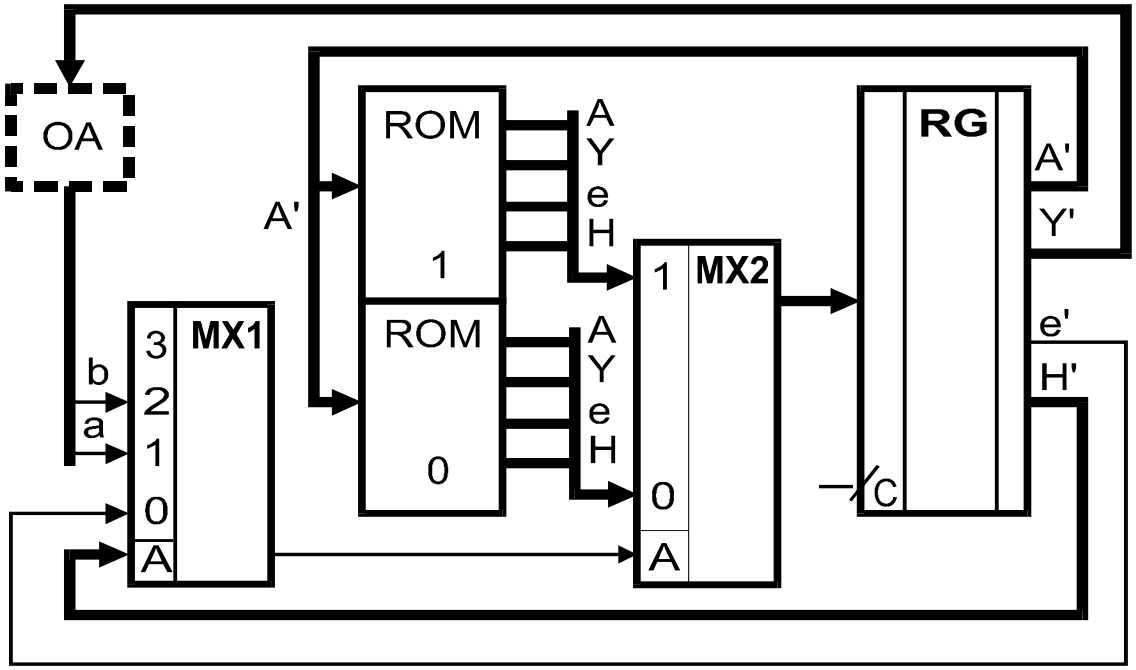


Рисунок 3.6. УА с сокращенным тактом

Части схемы критичные по длительности такта (ПЗУ и комбинационная часть операционного автомата) разнесены в разные контуры. Мультиплексор MX'2, функционально необходимый, реально может отсутствовать, т.к. мультиплексирование может быть реализовано использованием высокоимпедансного состояния выхода ПЗУ. Эта функция реализуется в ПЗУ обычно в 3 - 4 раза быстрее чем выбор содержимого ячейки по адресу. В этом случае объем аппаратуры остается таким же как и в схеме с «частичной записью адреса» (рис.3.6).

Способ адресации, по существу, такой же, как и в предыдущей схеме. Только в рассматриваемой схеме входной сигнал управляет выбором одного из двух блоков ПЗУ (можно интерпретировать этот сигнал как старший разряд адреса).

3.5. Схема с регулярной адресацией

В этой схеме рис.3.7. при разветвлении процесса вычислений пара альтернативных адресов получается следующим образом: один адрес всегда на единицу больше, чем текущий (т.е. изменяется «регулярным» образом), второй адрес - произвольный и содержится вместе с микрокомандой в микроинструкции.

Элементом, «вычисляющим» адрес, служит счетчик, управляемый входным для УА сигналом. При различных значениях входного сигнала счетчик выполняет две функции: либо прибавляет единицу к значению, которое хранилось в счетчике и являлось текущим адресом, либо загружается значением адреса из управляющей памяти. В схему введен элемент М2, позволяющий инвертировать значение входного сигнала, что облегчает распределение микроин-

струкций по ячейкам управляющей памяти. В схеме для конвейерного варианта взаимодействия регулярное изменение адреса приходится организовывать так, чтобы не увеличивать число мест конвейера.

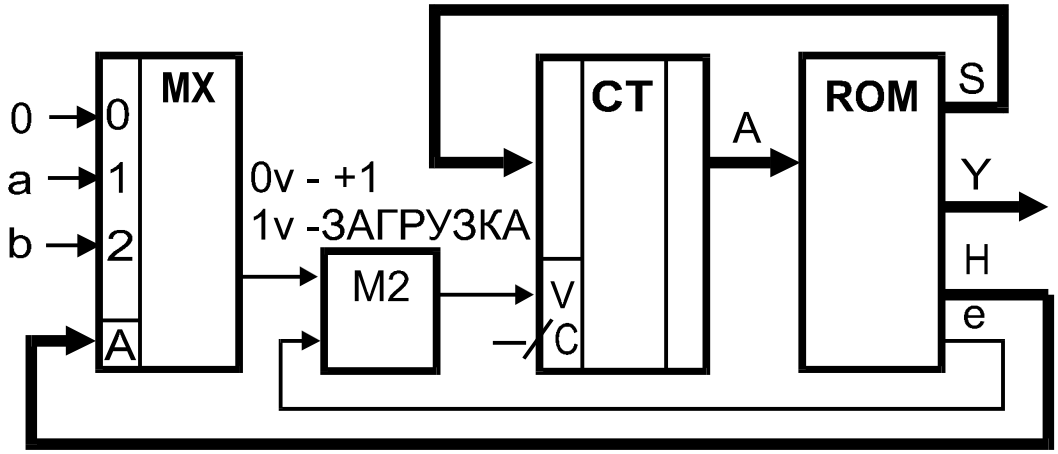


Рисунок 3.7. УА с регулярной адресацией; последовательный вариант

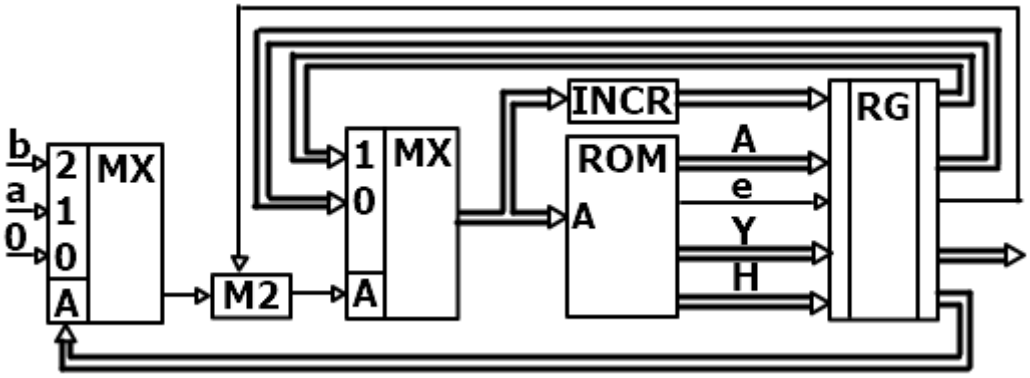


Рисунок 3.8. УА с регулярной адресацией; конвейерный вариант

		Адрес	A	Y	H	S	e
n1	{ m1 }	0	0	m1	x	1	x
n2	{ m2 }«GO(a; d1,n3)»	1	1	m2	1	3	0
d1	{ m0 }«GO(a; d1,n3)»	2	2	m0	1	2	1
n3	{ m3 }	3	3	m3	x	4	x
n4	{ m4 }«GO(a; d2,n1)»	4	4	m4	1	0	0
d2	{ m0 }«GO(b; n5,n3)»	5	5	m0	2	3	0
n5	{ m5 }«GO(a; n5,n3)»	6	6	m5	1	6	1
		7	7	m0	0	3	1

3.6. Схема с естественной адресацией

Эта схема используется только в конвейерном варианте взаимодействия. Метод вычисления адреса для следующего такта такой же, как и в схеме с регулярной адресацией. (Другой термин – «естественная» употреблен только ради различения самих схем.) В этой схеме, по сравнению с уже рассмотренными,

разряд управляющей памяти с одним и тем же номером (разрядный срез) в различных микроинструкциях может быть использован различным образом.

Будем различать микроинструкции двух типов:

- операционные,
- адресации (выбора).

В данном варианте схемы тип микроинструкции устанавливается разрядом k. При k=0 выполняется микроинструкция операционного типа. Все остальные разряды ячейки загружаются в регистр микрокоманды и управляют выполнением микроопераций в ОА. Следующий адрес всегда на единицу больше.

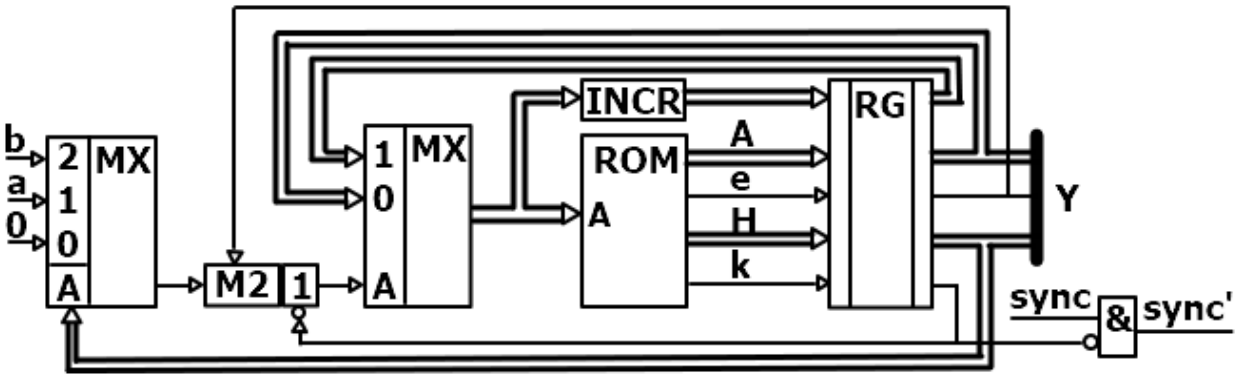


Рисунок 3.9. УА с естественной адресацией

	A	k	Y		
			H	e	S
n1 { m1 }	0	0	m1		
n2 { m2 }	1	0	m2		
g1 «GO(a; g1,n3)»	2	1	1	1	2
n3 { m3 }	3	0	m3		
n4 { m4 }	4	0	m4		
g2 «GO(a; g3,n1)»	5	1	1	0	0
g3 «GO(b; n5,n3)»	6	1	2	0	3
n5 { m5 }	7	0	m5		
g4 «GO(a; n5,n3)»	8	1	1	1	7
	9	1	0	1	3

При k=1 выполняется микроинструкция адресации. Все разряды микроинструкции могут быть использованы для вычисления следующего адреса. В данном варианте схемы, так же как и в схеме с регулярной адресацией, один из адресов явно записывается в микроинструкцию, другой альтернативный адрес на единицу больше текущего.

В этой схеме используется условная синхронизация, которая позволяет удлинить такт выполнения микрокоманды в ОА на время выполнения микроинструкций адресации.

3.7. УА с переходами функциональным и на микроподпрограмму.

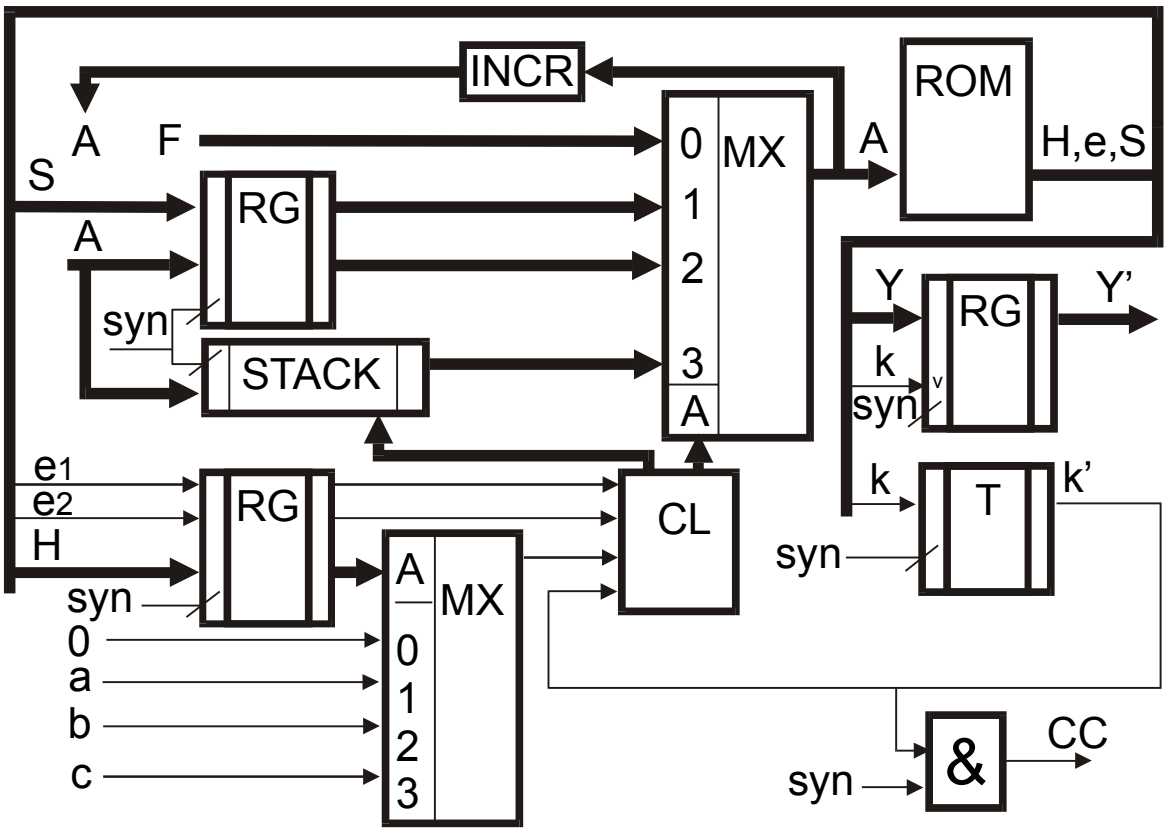


Рисунок 3.10. УА с вызовом микроподпрограмм

3.7.1. Функциональный переход

При необходимости выполнения нескольких вычислительных функций, управление которыми представляется в виде независимых микропрограмм, необходимо организовать независимый вызов этих микропрограмм.

Начальные адреса микропрограмм, управляющих вычислениями различных функций, обычно существуют вне управляющей памяти. В УА достаточно предусмотреть механизм коммутации, позволяющий сделать начальный адрес текущим. Это можно осуществить в любой из рассмотренных схем. (К механизму коммутации относятся, кроме аппаратуры, специальные разряды управляющей памяти и специальные микроинструкции.)

3.7.2. Переход к микроподпрограмме с возвратом

При реализации достаточно сложных вычислений удобно воспользоваться механизмом микроподпрограмм.

Одна и та же микроподпрограмма может быть вызвана из разных точек вызывающих микропрограмм. Поэтому при вызове микроподпрограммы необходимо запомнить адрес, с тем, чтобы восстановить его при возвращении из микроподпрограммы. Чтобы запомнить и восстановить адреса возврата от вложенных вызовов, используется безадресная память - стек (stack).

Принцип (дисциплина) работы стека описывается условием «последним вошел - первым вышел» (Last In - First Out, LIFO). чтобы описать этот принцип будем считать, что слова, хранящиеся в стеке, перенумерованы с помощью первых натуральных чисел 1,2,...,N. Слово с наибольшим номером называют вершиной стека.

Стек может выполнять следующие действия:

-«чтение» слова с наибольшим номером,

-«выталкивание» (стирание) из памяти слова с наибольшим номером,

-«запись» нового слова с присваиванием ему наибольшего номера.

При вызове микроподпрограммы выполняется операция «записи» в стек адреса возврата. При возвращении из микроподпрограммы выполняется операция «чтения» адреса возврата, затем «выталкивания» его же из стека.

Операция «чтения» без «выталкивания» выполняется при использовании стека для организации циклов.

Разряды с именем (k,e1,e2) определяют тип выполняемой микроинструкции. В связи с тем, что теперь в схеме существует 4 источника адреса для управляющей памяти, возможны 4 типа безусловных переходов, кроме того, возможны различные условные переходы в разных сочетаниях комбинирующие эти источники адреса и входные переменные.

С помощью комбинационной схемы CS разряды микроинструкции (k,e1,e2) преобразуются в сигналы управления стеком и мультиплексором.

3.8. Управление с предвосхищением

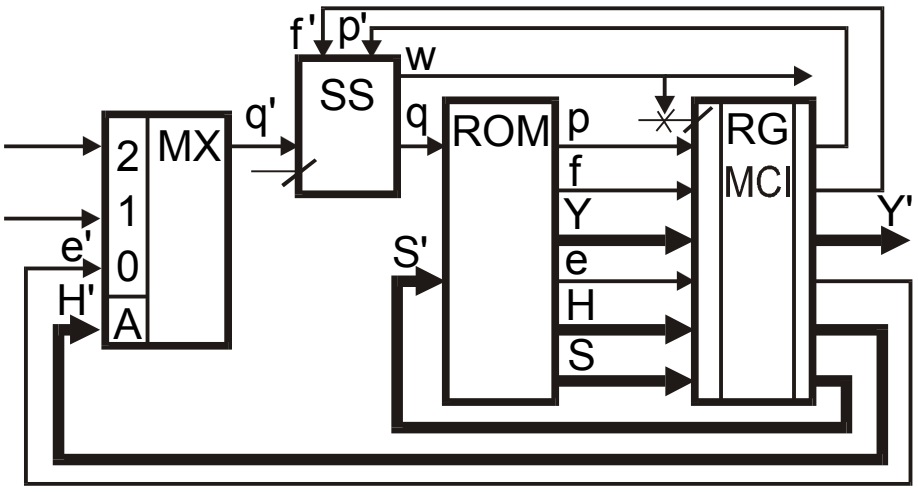


Рисунок 3.11. Автомат с предвосхищением

В конвейерном варианте при выполнении условных переходов по флагам, зависящим без сдвига от сигналов управления, приходится добавлять еще один такт для того, чтобы "увидеть" значение переменной, по которой выполняется

ветвление, и выбрать нужную микроинструкцию. Можно построить управляющий автомат, в котором для ускорения выполнения микропрограммы выполняются следующие действия. Предварительно выбирается из ПЗУ одна из двух альтернативных микроинструкций, соответствующая наиболее вероятному значению флага. Это значение должно храниться в той микроинструкции, после которой выполняется ветвление. В конце такта выработанное реальное значение переменной сравнивается с предсказанным, если они совпадают, то выбранная из ПЗУ микроинструкция записывается в выходной регистр, если нет, то предыдущий такт продлевается, т.е. не синхронизируются операционный автомат и RG'MCI.

Микропрограмма будет такой же, как и для последовательного варианта взаимодействия, но в микроинструкцию добавляются два разряда:

f={ 1, если используется предвосхищение; 0, если нет};

p - наиболее вероятное значение переменной ветвления.

В схему включается автомат SS, который вырабатывает два сигнала:

q- разряд адреса ПЗУ;

w - управляет условной синхронизацией операционного автомата и RG'MCI; эта переменная должна зависеть без сдвига от входных переменных автомата SS, т.к. установившееся значение w должно предшествовать рабочему фронту сигнала синхронизации.

Итак, SS - автомат, который может находиться в одном из двух состояний s0 и s1:

состояние	$f^!$	$p^! q^!$ (r)	q (a)	w	переход
s0	0	x	$q^!$ (0)	1	s0
s0	1	$p^!=q^!$ (1)	$p^!$ (1)	1	s0
s0	1	$p^!\neq q^!$ (0)	$q^!$ (0)	0	s1
s1	x	x	$q^!$ (0)	1	s0

Аппаратура микропрограммного управляющего автомата несколько увеличилась. При этом основная доля прироста приходится на ПЗУ, увеличенное на два разряда в каждой ячейке.

ЛИТЕРАТУРА

1. Антик М.И., Синхронные цифровые автоматы [Электронный ресурс]: монография / М.И.Антик, А.М.Романов. — М.: МГТУ МИРЭА, 2014. — Электрон. опт. диск (ISO), НТБ МИРЭА А72.
2. Антик М.И., Триггеры [Электронный ресурс]: учебное пособие для студентов, обучающихся по направлениям подготовки 230100 и спец. 230101 "Вычислительные машины, системы, комплексы и сети" / М.И.Антик, А.М.Романов. — М.: МГТУ МИРЭА, 2012. — Электрон. опт. диск (ISO), НТБ МИРЭА А72.
3. Зайцев Е.И., Прикладная теория цифровых автоматов : учебное пособие / Е. И. Зайцев, В. В. Макаров. — М.: МИРЭА, 2018. — 112 с.. — Библиогр.: с. 111 (7 назв.), НТБ МИРЭА З-17.
4. Теория автоматов / Карпов Ю.Г. – СПб: Питер. 2003. – 208 с.
5. Горбатов В.А. Теория автоматов: учеб. для студентов втузов – М.: АСТ: Астрель. 2008. – 559 с.