

М.А. АМЕЛИНА

e-mail: amelina.marina@gmail.com

**Конспект лекций
по курсу**

**«Цифровая техника»
4 курс
7 семестр**

2013

ОГЛАВЛЕНИЕ

СПИСОК СОКРАЩЕНИЙ	3
1 ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ	4
1.1 Общие сведения о цифровых устройствах	4
1.2 Алгебра логики	6
1.3 Основные логические операции и способы их аппаратной реализации	8
1.4 Универсальные логические операции и их особенности	11
1.5 Формы записи логических функций (ДНФ, КНФ)	12
1.6 Переход от логической функции к логической схеме	13
1.7 Минимизация логических функций	14
1.8 Запись и реализация логических функций в универсальных базисах	17
1.9 Коды и системы счисления	19
1.10 Компьютерные форматы данных	26
2 ЭЛЕМЕНТЫ ЦИФРОВЫХ УСТРОЙСТВ	33
2.1 Комбинационные и последовательностные устройства	33
2.2 Шифраторы, дешифраторы, преобразователи кодов	35
2.3 Мультиплексоры и демультиплексоры	38
2.4 Компараторы кодов	41
2.5 Двоичные полусумматор и сумматор	42
2.6 Арифметико-логические устройства	47
2.7 Триггеры	50
2.8 Счетчики	60
2.8.1 Основные параметры и классификация счетчиков	60
2.8.2 Двоичные счетчики	61
2.8.3 Двоично-кодированные счетчики	73
2.8.4 Счетчики с двоичным кодированием состояний	80
2.9 Регистры и регистровые файлы	91
2.9.1 Параллельные регистры	91
2.9.2 Регистровые файлы	92
2.9.3 Сдвигающие регистры	94
2.9.4 Универсальные регистры	96
2.10 Запоминающие устройства	100
3 СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ	101
3.1 Синтез асинхронных автоматов на RS-триггерах	101
3.1.1 Пример 1	101
3.1.2 Пример 2	103
3.1.3 Пример 3 — Автомат Мили	105
3.1.4 Пример 4 — автомат Мура	107
3.2 Синтез асинхронных автоматов на мультиплексорах	110
3.2.1 Пример 1. Асинхронный автомат Мили	110
3.2.2 Пример 2. Асинхронный автомат Мура	112
3.3 Синтез синхронных автоматов	115
3.3.1 Пример 3. Синтез счетчика с изменяемым коэффициентом пересчёта	115
ЛИТЕРАТУРА	118

СПИСОК СОКРАЩЕНИЙ

- АЛУ — арифметическо-логическое устройство
ASCII — Американский Стандартный Код для Информационного Обмена
(American Standard Code for Information Interchange)
БИС — большая интегральная схема
ГТИ — генератор тактовых импульсов
ИМС — интегральная микросхема
ЛЭ — логический элемент
МПС — микропроцессорная система
ОЗУ — оперативное запоминающее устройство
ПЗУ — постоянное запоминающее устройство
ПЛМ — программируемая логическая матрица
СБИС — сверхбольшая интегральная схема
СДНФ — совершенная дизъюнктивно-нормальная форма записи логических выражений
СИ — синхроимпульс
СКНФ — совершенная конъюнктивно-нормальная форма записи логических выражений
УГО — условное графическое обозначение
ФАЛ — функция алгебры логики

1 ЛОГИЧЕСКИЕ ОСНОВЫ ЦИФРОВЫХ УСТРОЙСТВ

1.1 Общие сведения о цифровых устройствах

Цифровыми называют устройства, предназначенные для формирования, преобразования и передачи кодовых слов. При этом кодовые слова (коды или числа) в электронных цифровых устройствах представляются в виде последовательностей электрических импульсов (сигналов с двумя уровнями напряжения: высоким и низким), а их преобразования осуществляются арифметическими, логическими, запоминающими и вспомогательными устройствами.

Элементами и узлами цифровых устройств, служащими основой для построения микропроцессоров, микропроцессорных систем, компьютеров, автоматизированных систем управления объектами, технологическими процессами и информационными потоками являются: *дешифраторы, шифраторы, мультиплексоры, сумматоры, триггеры, регистры, счетчики* и многие другие.

В современных устройствах цифровой обработки информации используется два класса переменных: числа и логические переменные. *Числа* несут информацию о количественных характеристиках процесса, объекта, системы, над ними можно производить арифметические действия. *Логические переменные* определяют состояние системы или принадлежность её к определенному классу состояний.

Главная особенность цифровых устройств (по отношению к аналоговым и импульсным устройствам) состоит в том, что объектами информации являются *двоичные числа (кодовые слова)* и логические переменные, а не *непрерывные функции* времени. И числа, и логические переменные могут принимать конечное множество значений, в отличие от обычных аналоговых сигналов — непрерывных функций времени.

Числа и логические переменные связаны друг с другом при решении задач управления и обработки информации. В вычислительных задачах вначале определяются совокупность и значения входных воздействий на объект управления. Предполагается, что существует математическая модель объекта в виде набора формул, таблиц, графиков и несколько логических условий. При решении задач необходимо вести анализ логических условий с выдачей логических команд. Для решения таких задач необходим специальный математический аппарат и соответствующие устройства (микропроцессоры, микроконтроллеры).

Устройство на цифровой элементной базе, выполняющее арифметические и логические операции называют *арифметико-логическим устройством (АЛУ)*.

АЛУ, выполняющее также функции управления — *центральным процессором*.

Арифметические устройства (сумматоры, умножители) предназначены для выполнения арифметических операций над бинарными кодовыми словами.

Логическими устройствами называют схемные элементы, с помощью которых осуществляется преобразование поступающих на их входы двоичных (бинарных) сигналов и непосредственное выполнение предусмотренных логических операций.

Запоминающими называют такие устройства, которые обладают свойствами длительно сохранять поступающую в них информацию без изменения её содержания и отправлять её по команде в другие устройства.

Вспомогательными являются все прочие устройства, предназначенные для образования надёжных связей между арифметико-логическими и запоминающими функциональными узлами и внешними устройствами. К ним относят тактовые генераторы, устройства приёма и распределения данных, таймеры, усилители, повторители, инверторы и др.

Функционирование цифровых (в том числе и микропроцессорных) устройств можно представить следующим образом:

- посредством генератора тактовых импульсов производится синхронизация начала выполнения отдельных операций преобразования входного кодового слова и отводится время для выполнения команды (в течение одного или нескольких периодов тактовых импульсов);
- после активизации начала операции осуществляется преобразование всех входных кодовых слов (состоящих из логических нулей и единиц) в требуемые выходные кодовые слова;
- выходные кодовые слова отправляются на хранение в память цифрового устройства и/или во внешние устройства для выполнения определенных действий.

Операции над кодовыми словами, представленными в виде электрических сигналов, в цифровом устройстве могут выполняться следующими двумя способами:

1) *Последовательное* (поразрядное, побитовое) выполнение операций, при котором символы 1 и 0 кодового слова поступают последовательно по времени на единственный вход цифрового устройства и по завершении операции последовательно символ за символом выводятся из него. На рис. 1.1, а показано выполнение операции цифровым устройством ЦУ (инвертором) над трехразрядным входным словом $x_2x_1x_0=100$, при котором биты выходного слова $y_2y_1y_0=011$ принимают противоположные значения;

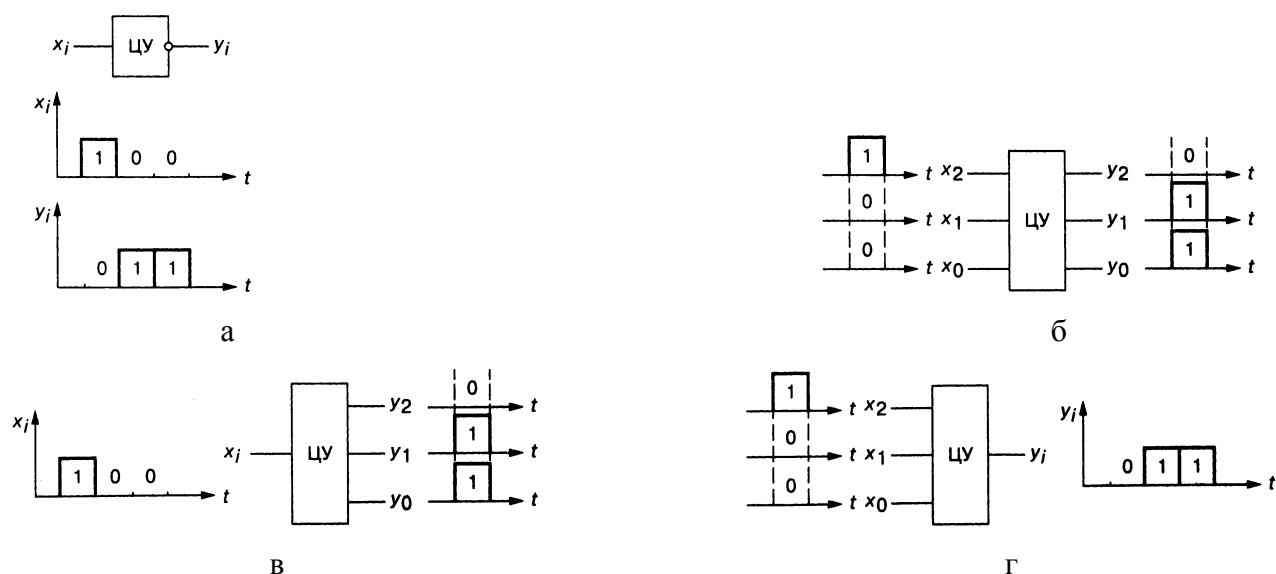


Рисунок 1.1 — Способы выполнения операций в цифровых устройствах

2) *Параллельное* выполнение операций, при котором символы 1 и 0 кодового слова поступают одновременно на три входа ЦУ и по завершении операции одновременно выводятся из него (рис. 1.1, б).

В рассмотренных устройствах для выполнения операций над кодовыми словами использовались устройства последовательного и параллельного действия, а входные и выходные слова представлялись в виде последовательного и параллельного кодов.

3) В ряде случаев используют *комбинированные* способы обработки информации: с последовательным вводом и параллельным выводом (рис. 1.1, в) и с параллельным вводом и последовательным выводом (рис. 1.1, г).

1.2 Алгебра логики

Работа любого логического устройства подчиняется законам формальной логики, которые не допускают уклончивых ответов. Решение логических задач осуществляется с помощью логических элементов, базирующихся на математическом аппарате алгебры логики (булевой алгебры, разработанной английским математиком Джорджем Булем (1815-1864), в которой все переменные величины (и аргументы, и функции) могут принимать только два логических значения: «1» (логическая единица) и «0» (логический ноль). Во многих случаях эти символы простейшего алфавита, состоящего из двух букв, отождествляют с арабскими цифрами 1 и 0, не вкладывая в них смысла количества.

Понятия "1" и "0" являются условными, символизирующими состояния, например, релейного устройства: "включено", "выключено", высказывания «истинные» или «ложные». Как отмечалось, в цифровых электронных устройствах применяют сигналы двух уровней напряжения: низкого и высокого.

Логика называется положительной, если высокий потенциал отображает единицу, а низкий, – ноль. Если наоборот, высокий потенциал отображает ноль, а низкий, – единицу, то логика называется отрицательной. Данное правило называют логическим соглашением. Далее будет использоваться способ кодирования, называемый *соглашением положительной логики*.

В общем случае логическое устройство может иметь n входов и m выходов. Рассматривая входные сигналы x_1, x_2, \dots, x_n в качестве аргументов, можно соответствующие выходные сигналы представлять в виде функций $y_i = f(x_0, x_1, x_2, \dots, x_n)$ с помощью элементарных операций алгебры логики.

Функции алгебры логики (ФАЛ), иногда называемые переключательными функциями, представляют в нескольких формах:

- в алгебраической (в виде математического выражения):

$y = (\bar{x}_0 \wedge x_1) \vee (x_1 \wedge x_2) = \bar{x}_0 \cdot x_1 + x_1 x_2$ \wedge, \cdot – операция «И»; $\vee, +$ – операция «ИЛИ», $\bar{}$ — инверсия;

- в виде таблиц истинности или комбинационных таблиц;
- в виде временных диаграмм;
- встречается также *абстрактный* вид записи функций алгебры логики: $y = (2, 6, 7)$, где в скобках приведены десятичные эквиваленты, например

3-разрядных двоичных кодовых слов, которые соответствуют значениям функции $y=1$ (табл. 1.1).

Таблица содержит всевозможные комбинации (наборы) бинарных значений входных переменных с соответствующими им бинарными значениями выходных переменных; каждому набору входных сигналов соответствует определенное значение выходной логической функции, (таблица 1.1). Максимальное число возможных различных наборов (строк) зависит от числа входных переменных n и равно 2^n ;

Таблица 1.1 Представление логической функции в виде таблицы истинности

x_2	x_1	x_0	y_1
0	0	0	0
0	0	1	0
0	1	0	1
•	•	•	0
1	1	0	1
1	1	1	1

На рис. 1.2 изображена временная диаграмма логической операции сложения двух кодовых слов по модулю 2:

$$y = x_1 \bar{x}_2 + \bar{x}_1 x_2 = x_1 \oplus x_2.$$

Практическим препятствием для повсеместного использования табличной формы задания логических функций является быстрый рост числа строк таблицы.

При $n=1$ входной сигнал x может принимать лишь два значения 0 и 1. Вполне возможно, что для обоих значений x (1 и 0) выходной сигнал y может принимать значение, равное 0; в другом случае y равен 1 при $x = 0$ и при $x=1$ и т.д., т.е. цифровое устройство с одним выходом способно сформировать четыре различных варианта выходного сигнала, которые приведены в таблице 1.2.

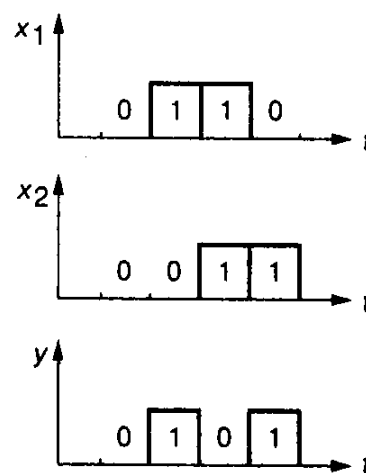


Рисунок 1.2

Таблица 1.2. Варианты функций одной логической переменной

Комбинация вх. x	0, 1	0, 1	0, 1	0, 1
Значение вых. функции y	0	1	0, 1	1, 0
Название операции	Постоянный 0	Постоянная 1	Повтор x	Инверсия x

Для цифрового устройства с двумя входными переменными x_1 и x_2 ($n=2$) возможно четыре варианта комбинаций аргументов (входных слов): 00, 01, 10 и 11 и шестнадцать различных выходных функций y_i , (таблица 1.3).















В учебных примерах (как правило) ограничиваются рассмотрением логических функций двух-четырёх аргументов.

Название и обозначение функции y , в какой-то мере отображает особенности выполнения логических операций (см. последнюю строку таблицы 1.3). Нулевая y_0 и единичная y_{15} функции тривиальны, функции y_3 , y_5 , y_{10} и y_{12} не зависят от одного из аргументов: $y_3=x_2$, $y_5=x_1$, $y_{10}=\bar{x}_1$, и $y_{12}=\bar{x}_2$. И только оставшиеся 10

функций являются функциями двух переменных.

Отметим, что многие функции имеют несколько названий. Например, логическая операция неравнозначности для функции y_6 имеет название «*исключающее ИЛИ*», «*сложение по модулю 2*»; функция y_7 имеет название «*сложение*», «*дизъюнкция*», «*ИЛИ*». Для обозначения операций логических функций используются специальные символы. Например, в качестве знака операции ИЛИ-НЕ используется символ " \downarrow " (стрелка Пирса), условное обозначение функции $y_8 = x_1 \downarrow x_2$; для операции И-НЕ принят символ " $|$ " (штрих Шеффера), обозначение функции $y_{14} = x_1 | x_2$; для операции неравнозначности — символ \oplus (сложения по модулю 2), обозначение функции $y_6 = x_1 \oplus x_2$ и т. д.

Таблица 1.3 Перечень всевозможных функций для логического устройства с двумя входами и одним выходом

		Значение выходной функции y_i для всех комбинаций вх. сигналов															
Комбинации входных сигналов x_2, x_1	00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Логическое выражение для выходной функции		$y_0 = 0$	$y_1 = x_2 \cdot x_1$	$y_2 = x_2 \cdot \bar{x}_1$	$y_3 = x_2$	$y_4 = \bar{x}_2 \cdot x_1$	$y_5 = x_1$	$y_6 = y_4 = \bar{x}_2 \cdot x_1 + x_2 \cdot \bar{x}_1$	$y_7 = x_2 + x_1$	$y_8 = \overline{x_2 + x_1}$	$y_9 = \bar{x}_2 \cdot \bar{x}_1 + x_2 \cdot x_1$	$y_{10} = \bar{x}_1$	$y_{11} = x_2 + \bar{x}_1$	$y_{12} = \bar{x}_2$	$y_{13} = \bar{x}_2 + x_1$	$y_{14} = \overline{x_2 \cdot x_1}$	$y_{15} = 1$
Название функции		Постоянный 0	Умножение, конъюнкция, И	Запрет по x_1	Тождественность x_2	Запрет по x_2	Тождественность x_1	Неравнозначность	Сложение, дизъюнкция, ИЛИ	Стрелка Пирса, ИЛИ-НЕ	Равнозначность	Инверсия x_1 , НЕ	Импликация от x_1 к x_2	Инверсия x_2 , НЕ	Импликация от x_2 к x_1	Штрих Шеффера, И-НЕ	Постоянная 1
																	
			AND & ^					XOR \oplus < +	OR < +	NOR \downarrow	XNOR	NOT		NOT		NAND 	

1.3 Основные логические операции и способы их аппаратной реализации

В булевой алгебре выделяют три основные функции: *конъюнкция*, *дизъюнкция*, *отрицание*. Остальные функции являются производными от приведенных выше.

Основные логические операции состоят из следующих элементарных преобразований двоичных сигналов:

- *логическое сложение* или *дизъюнкция* (от английского "disjunction" — разъединение), обозначаемое символом " \vee " и называемое также операцией ИЛИ. При этом число аргументов (слагаемых x) может быть любым. Эта операция для функции двух переменных x_1 и x_2 описывается в виде логической формулы:

$$y = x_1 \vee x_2 \quad (1.1)$$

Запись (1.1) формулируется следующим образом: y равен x_1 ИЛИ x_2 . Это значит, что y истинно (равно 1), если истинно хотя бы одно из слагаемых x_1 или x_2 . И только в случае, когда все слагаемые x равны 0, результат логического сложения y также равен 0.

Для удобства записи сложных логических функций символ дизъюнкции " \vee " можно условно отождествлять со знаком обычного сложения "+". Для функции двух переменных:

$$y = x_1 \vee x_2 = x_1 + x_2.$$

Условное обозначение, таблица истинности и другие показатели этой логической функции приведены в 8-ом столбце (функция y_7) таблицы 1.3;

- *логическое умножение* или *конъюнкция* (от английского слова "conjunction" — соединение), обозначаемое символом " \wedge " и называемое также операцией И. При этом число аргументов (сомножителей x) может быть любым. Эта операция для функции двух переменных x_1 и x_2 описывается в виде логической формулы:

$$y = x_1 \wedge x_2 \quad (1.2).$$

Запись (1.2) формулируется следующим образом: y равен x_1 И x_2 . Это значит, что y истинно (равно 1), если истинны оба сомножителя x_1 и x_2 . В случае если хотя бы один из сомножителей равен 0, результат логического умножения y тоже равен 0.

Для удобства записи сложных логических функций символ конъюнкции \wedge можно условно отождествлять со знаком обычного умножения « \cdot ». Для функции двух переменных в этом случае:

$$y = x_1 \wedge x_2 = x_1 \cdot x_2.$$

Условное обозначение, таблица истинности логической функции И приведены во втором столбце таблицы 1.3 (функция y_1).

- *логическое отрицание* или *инверсия*, обозначаемое чёрточкой над переменной и называемое операцией "НЕ". Формулируется так: y равен НЕ x . Это значит, что y истинно (равно 1), если x ложно (равно 0), и наоборот. Операция y выполняется над одной переменной x и её значение всегда противоположно этой переменной (см. 11-ый и 13-й столбцы таблицы 1.3, функции y_{10} , y_{12}). Операция записывается в виде:

$$y = \bar{x} \quad (1.3)$$

Основные логические операции ИЛИ, И и НЕ и цифровые элементы их выполняющие (дизъюнктор, конъюнктор, инвертор) позволяют реализовать цифро-

вое устройство без памяти, называемое *комбинационным*, любой степени сложности.

Примеры контактной и простейшей схемной реализаций дизъюнктора, конъюнктора и инвертора приведены на рис. 1.3.

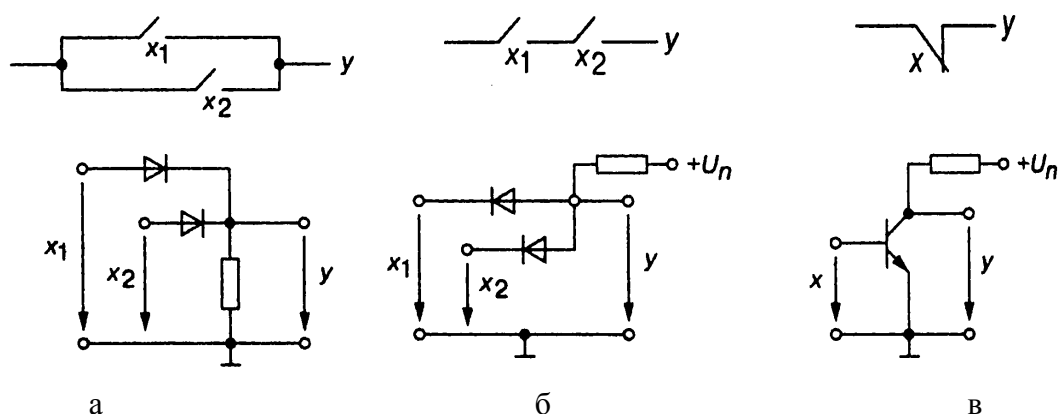


Рисунок 1.3 — Примеры контактной и простейшей схемной реализаций: а – дизъюнктора, б – конъюнктора, в — инвертора

Сопоставляя таблицы истинности для операций «ИЛИ» и «И» (см. таблицу 1.3), можно получить некоторые соотношения булевой алгебры, имеющие большое практическое значение. Например, принцип дуальности (двойственности) булевой алгебры записывается в виде двух следующих положений:

$$\text{если } x_1 x_2 = y, \quad \text{то} \quad \bar{x}_1 + \bar{x}_2 = \bar{y} \quad (1.4)$$

$$\text{если } x_1 + x_2 = y, \quad \text{то} \quad \bar{x}_1 \cdot \bar{x}_2 = \bar{y} \quad (1.5)$$

Из этих соотношений вытекает **теорема (правило) де Моргана**: *инверсия выражения может быть представлена тем же выражением без инверсии с изменением всех знаков конъюнкции на знаки дизъюнкции, знаков дизъюнкции на знаки конъюнкции и инверсией всех аргументов*, т. е.

$$\overline{x_1 \cdot x_2} = \bar{x}_1 + \bar{x}_2; \quad \overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2 \quad (1.6)$$

Эти соотношения позволяют взаимно заменять операции дизъюнкции и конъюнкции, а это даёт возможность построить любую переключательную функцию, используя только две операции: «И» и «НЕ» или «ИЛИ» и «НЕ». Однако при использовании только двух элементов не всегда удастся получить логические устройства наипростейшего типа. Поэтому в логических схемах находят применение и другие типовые элементы, реализующие иные логические операции, приведенные в таблице 1.3.

Приведем основные аксиомы и теоремы алгебры логики (без вывода):

Коммутативный закон:

$$x_1 \cdot x_2 = x_2 \cdot x_1 \quad x_1 + x_2 = x_2 + x_1 \quad (1.7)$$

Ассоциативный закон:

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 \quad x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 \quad (1.8)$$

Дистрибутивный закон:

$$x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3 \quad x_1 + x_2 \cdot x_3 = (x_1 + x_2)(x_1 + x_3) \quad (1.9)$$

Правило поглощения:

$$x_1 \cdot (x_1 + x_2) = x_1 \quad x_1 + x_1 \cdot x_2 = x_1 \quad (1.10)$$

Правило склеивания:

$$x_1 \cdot x_2 + x_1 \cdot \bar{x}_2 = x_1 \quad (x_1 + x_2)(x_1 + \bar{x}_2) = x_1 \quad (1.11)$$

Правило повторения:

$$x \cdot x = x \quad x + x = x \quad (1.12)$$

Правило отрицания:

$$x \cdot \bar{x} = 0 \quad x + \bar{x} = 1 \quad (1.13)$$

Правило двойного отрицания:

$$\overline{(\bar{x})} = x \quad (1.14)$$

Операции с 0 и 1:

$$x \cdot 1 = x \quad x + 0 = x \quad (1.15)$$

$$x \cdot 0 = 0 \quad x + 1 = 1$$

$$\bar{0} = 1 \quad \bar{1} = 0$$

Формулы (1.12...1.15) представляют собой тождества, в справедливости которых легко убедиться прямой подстановкой $x=0$ и $x=1$. В преобразованиях логических выражений важную роль играют формулы (1.6, 1.9... 1.11), . Формулы де Моргана (1.6), как отмечалось, используют для того, чтобы перейти от логического произведения к логической сумме и обратно.

При использовании приведенных соотношений можно упростить записи и минимизировать логические выражения в смысле уменьшения числа символов.

Приведём несколько примеров.

$$a + \bar{a}b = a(b + \bar{b}) + \bar{a}b = ab + a\bar{b} + \bar{a}b = ab + ab + a\bar{b} + \bar{a}b = a(b + \bar{b}) + b(a + \bar{a}) = a + b$$

$$(a + c)(b + \bar{c}) = ab + a\bar{c} + bc + c\bar{c} = ab + a\bar{c} + bc$$

Здесь использованы тождества $a + \bar{a} = 1$, $b + \bar{b} = 1$, $c\bar{c} = 0$.

$$a + ab + ac = a(1 + b + c) = a$$

$$\overline{(ab)} + \overline{(bc)} = \bar{a} + \bar{b} + \bar{b} + \bar{c} = \bar{a} + \bar{b} + \bar{c} = \overline{abc}$$

1.4 Универсальные логические операции и их особенности

Особое значение в цифровой электронике имеют универсальные (базовые) логические элементы, способные образовать функционально полный набор, с

помощью которых можно реализовать все другие элементы логических базисов. В интегральной технологии удобство изготовления одного базового элемента имеет решающее значение. Поэтому базовые логические устройства составляют основу большинства цифровых ИМС. К универсальным логическим операциям (устройствам) относят две разновидности базовых элементов:

- функцию Пирса, обозначаемую символически вертикальной стрелкой \downarrow (стрелка Пирса) и отображающую операцию ИЛИ-НЕ. Этой операции соответствует столбец y_8 в таблице 1.3. Для простейшей функции двух переменных x_1 и x_2 функция $y=1$ тогда и только тогда, когда $x_1=x_2=0$:

$$y = x_1 \downarrow x_2 = \overline{x_1 + x_2} \quad (1.16)$$

- функцию Шеффера, обозначаемую символически вертикальной черточкой $|$ (штрих Шеффера) и отображающую операцию И-НЕ. Этой операции соответствует столбец y_{14} в таблице 1.3. Для простейшей функции двух переменных x_1 и x_2 функция $y=0$ тогда и только тогда, когда $x_1=x_2=1$:

$$y = x_1 | x_2 = \overline{x_1 \cdot x_2} \quad (1.17)$$

Для построения двухвходовой схемы ИЛИ-НЕ (рис. 1.4, а) к нагрузочному резистору подключены коллекторы двух параллельно включенных биполярных транзисторов п-р-п-типа, эмиттеры которых заземлены, а для построения схемы И-НЕ на два входа потребовалось последовательное (ярусное) включение двух биполярных транзисторов п-р-п типа (эмиттер нижнего транзистора подключен к земле) и нагрузочный резистор R.

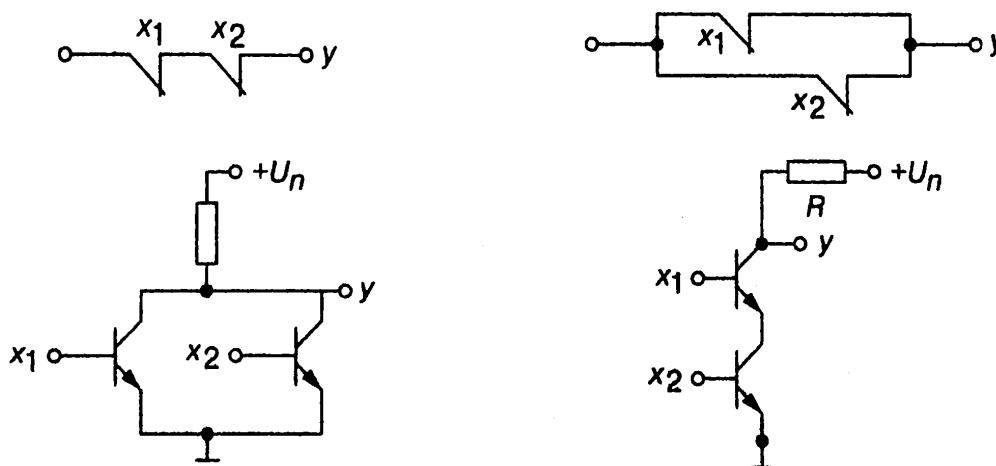


Рисунок 1.4 — Примеры контактной и простейшей схемной реализаций:
а — элемента ИЛИ-НЕ; б — элемента И-НЕ

1.5 Формы записи логических функций (ДНФ, КНФ)

Наиболее распространенным способом задания логических функций является табличная форма. Таблицы истинности позволяют полно и однозначно установить все существующие логические связи.

При табличном представлении логических функций их записывают в одной из канонических форм: совершенной дизъюнктивной нормальной форме (СДНФ) или совершенной конъюнктивной нормальной форме (СКНФ).

Математическое выражение логической функции в СДНФ получают из таблицы истинности следующим образом: для каждого набора аргументов, при котором функция равна 1, записывают элементарные произведения переменных, причем переменные, значения которых равны нулю, записывают с инверсией. Полученные произведения, называемые *конституентами единицы* или *минтермами*, суммируют.

Запишем логическую функцию y трех переменных a , b и c , представленной в виде таблицы 1.4, в СДНФ:

$$y(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + abc$$

Таблица 1.4 — Таблица истинности функции 3-х переменных

№	a	b	c	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Для сокращения записи СДНФ представляют последовательностью номеров (десятичных чисел) конституент единицы:

$$y(a,b,c) = \Sigma(2,3,5,7).$$

Совершенной конъюнктивной нормальной формой (СКНФ) называют логическое произведение элементарных сумм, в каждую из которых аргумент или его отрицание входят один раз. При этом для каждого набора аргументов таблицы истинности, при котором функция y равна 0, составляют элементарную сумму, причем переменные, значение которых равно 1, записывают с отрицанием. Полученные суммы, называемые *конституентами нуля* или *макстермами*, объединяют операцией логического умножения. Для функции (таблица 1.4) СКНФ:

$$y(a,b,c) = (a + b + c)(a + b + \bar{c})(\bar{a} + b + c)(\bar{a} + \bar{b} + c).$$

Для сокращения записи СКНФ представляют последовательностью номеров (десятичных чисел) конституент нуля:

$$y(a,b,c) = \Pi(0,1,4,6).$$

1.6 Переход от логической функции к логической схеме

Для построения логической схемы необходимо логические элементы, предназначенные для выполнения логических операций, располагать, начиная от входа, в порядке, указанном в булевом выражении.

Построим структуру логического устройства, реализующего логическую функцию трех переменных, заданную таблицей истинности 1.4:

$$y(a,b,c)=(a+b+c)(a+b+\bar{c})(\bar{a}+b+c)(\bar{a}+\bar{b}+c).$$

Слева располагаем входы a , b и c с ответвлениями на три инвертора, затем четыре элемента ИЛИ и, наконец, элемент И на выходе (рис. 1.5).

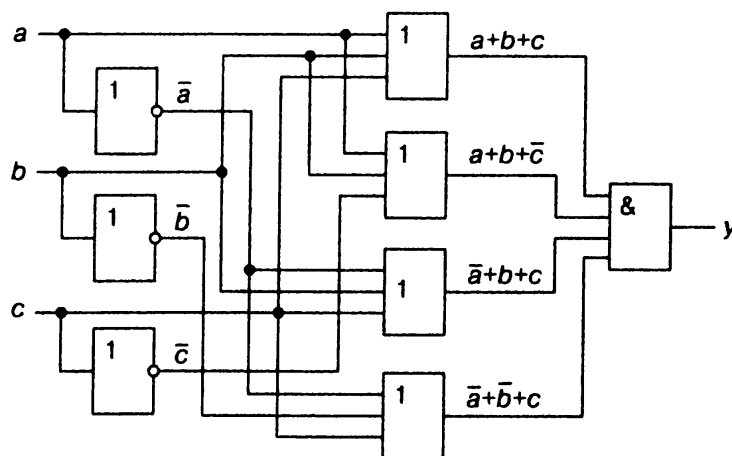


Рисунок 1.5 — Реализация логической функции, заданной табл. 1.4

Итак, любую логическую функцию можно реализовать непосредственно по выражениям, представленным в виде СДНФ или СКНФ. Однако, полученная таким образом схема, как правило, не оптимальна с точки зрения её практической реализации: она громоздка, содержит много элементов, и возникают трудности в обеспечении её высокой надёжности.

Можно минимизировать выражение для этой функции, записанное, например в форме СДНФ, используя теоремы алгебры логики:

$$y(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + abc = \bar{a}b(\bar{c} + c) + ac(\bar{b} + b) = \bar{a}b + ac \quad (1.18)$$

1.7 Минимизация логических функций

О необходимости минимизации структурных формул СДНФ и СКНФ. Структурные формулы в виде логических функций в СДНФ и СКНФ однозначно определяют структуру логической схемы комбинационного устройства. Однако непосредственное их использование, как правило, не позволяет построить устройство, наиболее полно удовлетворяющее заданным требованиям. Недостаток построения схем комбинационных устройств по структурным формулам в СДНФ и СКНФ проявляется в том, что чаще всего устройства содержат большое количество логических элементов, имеют низкие экономические показатели и надёжность. Путём тождественных преобразований структурной формулы, приводящих только к изменению её формы, а не значений, можно значительно упростить схему комбинационного устройства (см. соотношение 1.18).

Преобразование структурной формулы с целью упрощения комбинационного устройства называется её *минимизацией*. Критерием минимизации может служить число операций над переменными логической функции или логических элементов, реализующих эти операции.

Следует отметить, что критерии, в соответствии с которыми на практике осуществляют минимизацию логической функции, неоднозначны. Это миними-

зация стоимости её технической реализации, уменьшение количества элементарных логических элементов, использование только однородных базовых элементов, например, типа И-НЕ (ИЛИ-НЕ) и др.

Для интерпретации любых логических функций и их наглядной минимизации широко используют карты Карно-Вейча, базирующиеся на табличном представлении логических функций с числом переменных, не превышающих 4...5.

		b		0	1
		a			
0				$f(\bar{a}\bar{b})$	$f(\bar{a}b)$
1				$f(a\bar{b})$	$f(ab)$
		a		а	

		bc			
		00	01	11	10
0			$\bar{a}\bar{b}c$		
1				abc	
		б			

		cd			
		00	01	11	10
00		0	1	0	1
01		0	0	0	0
11		1	0	1	0
10		0	0	0	0
		в			

Рисунок 1.6. Изображения карт Карно для функций: а — 2-х переменных; б — 3-х переменных, в — 4-х переменных

Карта Карно — графическое представление всех возможных минтермов (2^n) для данного числа переменных (n). Каждый минтерм изображается в виде клетки, расположенной так, что минтермы, соответствующие соседним клеткам, отличаются друг от друга только одной переменной. На рис. 1.6 представлены изображения карт Карно для функций двух, трех и четырех переменных. Карта для двух переменных содержит четыре клетки (рис. 1.6, а), для трех — восемь (рис. 1.6, б), для четырех — шестнадцать (рис. 1.6, в). Множество клеток позволяет отобразить все наборы аргументов, а карту Карно можно рассматривать как упорядоченное представление подмножеств. Так, на рис. 1.6, б, в нижней строке и в третьем столбце имеем пересечение аргументов a , b и c , в верхней строке и втором столбце пересечение аргументов \bar{a} , \bar{b} и c и т. д.

Если требуется представить на карте Карно логическую функцию, заданную в виде СДНФ, то в соответствии с необходимым значением функции ставятся в надлежащих клетках 1. Остальные клетки остаются незаполненными (если для соответствующей комбинации аргументов не задано значение функции) или заполняются нулями.

Итак, логическая функция в СДНФ на карте Карно представляется совокупностью клеток, заполненных единицами, а инверсия функции представляется совокупностью пустых клеток или заполненных нулями.

На рис. 1.6, в изображена карта Карно полностью определенной логической функции в СДНФ

$$y = ab\bar{c}\bar{d} + abcd + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}\bar{c}d.$$

Как отмечалось, для минимизации логической функции применяют метод последовательного исключения переменных с помощью законов и тождеств алгебры логики, минимизационных карт Карно и др.

Основу минимизации логических функций с помощью карт Карно составляет следующее: два минтерма, находящиеся в соседних клетках карты, могут быть заменены одной конъюнкцией, содержащей на одну переменную меньше.

Если соседними являются две пары минтермов, то такая группа из четырех соседних минтермов может быть заменена конъюнкцией, которая содержит на две переменные меньше.

При минимизации функции следует помнить, что одна и та же клетка карты Карно может входить в несколько групп и что соседними клетками являются не только клетки, расположенные рядом по горизонтали и вертикали, но и клетки на противоположных границах карты Карно.

На рис. 1.7 приведены примеры минимизации логических функций трех (а) и четырех (б) переменных.

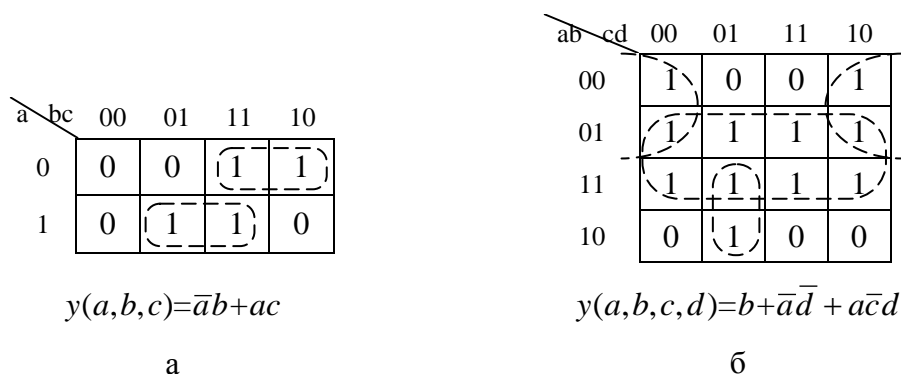


Рисунок 1.7 — Примеры минимизации логических функций: а — 3-х переменных; б — 4-х переменных

Минимизацию ФАЛ можно проводить и с использованием нулевых значений функции. Так, для ФАЛ, представленной на рис. 1.7, а, $\bar{y} = \bar{a}\bar{b} + a\bar{c}$, а для функции, представленной на рис. 1.7, б, $\bar{y} = a\bar{b}\bar{d} + \bar{a}\bar{b}d + \bar{b}cd$.

Таким образом, объединение областей карты Карно по единичным и нулевым значениям функции приводит к равносильным, но различным минимальным выражениям. Следовательно, может отличаться и схема, реализующая заданный алгоритм. Поэтому для получения минимально простой технической реализации целесообразно проводить минимизацию как единичных, так и нулевых значений функции, и из полученных минимальных форм выбрать простейшую.

На практике часто встречаются логические функции, часть значений которых не задана, т. е. эти значения могут быть произвольными. Такие ФАЛ называют недоопределенными. При различном доопределении ФАЛ и минимизации по единичным и нулевым значениям функции, могут быть получены различные минимальные формы. При доопределении ФАЛ необходимо стремиться к тому, чтобы на карте Карно было выделено минимальное число максимально больших областей.

На рис. 1.8 приведены примеры доопределения ФАЛ, приводящие к получению наиболее простой реализации заданного алгоритма.

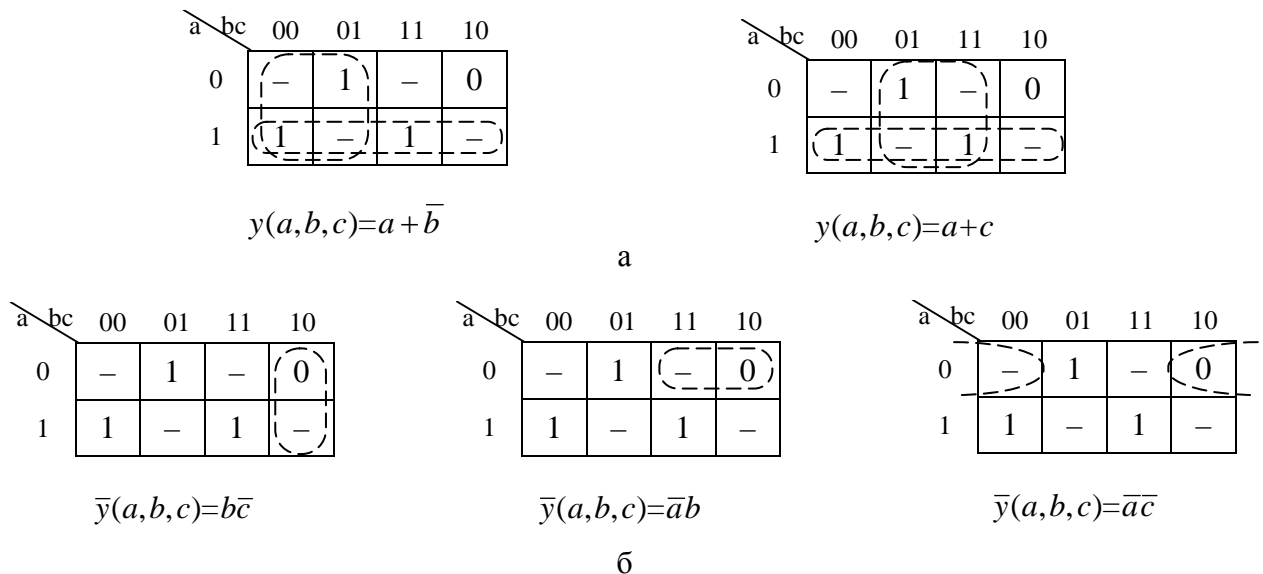


Рисунок 1.8 — Доопределение ФАЛ: а — единицами; б — нулями

1.8 Запись и реализация логических функций в универсальных базисах

Запись логических функций в универсальных базисах ИЛИ-НЕ и И-НЕ производится в такой последовательности:

- заданная логическая функция минимизируется в базисе ИЛИ, И, НЕ;
- над полученным выражением логической функции ставят двойное отрицание и с помощью правила де Моргана осуществляют переход в универсальный базис ИЛИ-НЕ или И-НЕ;
- при преобразовании логической функции используют следующие выражения:

в базисе И-НЕ преобразование выражения $\bar{a}b + \bar{a}b$:

$$\bar{a}b = a \cdot \bar{a} + a \cdot \bar{b} = a(\bar{a} + \bar{b}) = a(\overline{a \cdot b}); \quad \bar{a} = \overline{a \cdot a}; \quad \bar{a} = \overline{a \cdot 1}; \quad \Rightarrow \bar{a}b + \bar{a}b = [\overline{a \cdot b}][\overline{b \cdot a}];$$

в базисе ИЛИ-НЕ преобразование выражения $\bar{a}b + \bar{a}b$:

$$a + \bar{b} = (a + \bar{b})(a + \bar{a}) = a + \bar{b}\bar{a} = a + \overline{a \cdot b}; \quad \bar{a} = \overline{a \cdot a}; \quad \bar{a} = \overline{a \cdot 0}; \quad \bar{a}b + \bar{a}b = \overline{a \cdot b} + \overline{a \cdot b} = \overline{a \cdot b + a \cdot b} = \overline{a + b}.$$

При построении функциональных схем на элементах Шеффера (ИЛИ-НЕ) логическую функцию представляют в минимальной КНФ, а при построении функциональных схем на элементах Пирса (И-НЕ) — в минимальной ДНФ. В этих случаях функциональные схемы содержат минимальное количество элементов и более просты при построении.

Пример. Запишем логическую функцию $y = \bar{a}\bar{d} + a\bar{c}d + acd + b\bar{c}\bar{d} + \bar{b}c\bar{d}$ в базисе И-НЕ и ИЛИ-НЕ в минимальных ДНФ и КНФ.

Вычерчиваем карту Карно для четырех переменных a, b, c и d (рис. 1.9) и отметим в ней единицей (1) минтермы, содержащие конъюнкции, входящие в заданную функцию. В результате склеивания минтермов в карте Карно, для ко-

торых заданная функция $y=1$, получим выражение для выходной функции в минимальной ДНФ:

$$y_{DNF} = a + b\bar{c}\bar{d} + \bar{b}c\bar{d} \quad (1.19),$$

а в результате склеивания минтермов, для которых функция $y=0$, получим выражение для исходной функции в минимальной КНФ:

$$y_{KNF} = \overline{\bar{a}d + \bar{a}bc + \bar{a}\bar{b}\bar{c}} = (a + \bar{d})(a + \bar{b} + \bar{c})(a + b + c) \quad (1.20).$$

Для записи логической функции $y(a, b, c, d)$ в базисе И-НЕ применим к правым частям выражений (1.19), (1.20) правила двойного отрицания и правило де Моргана в разной последовательности. После преобразований получим:

$$y_{DNF} = \overline{\overline{a + b\bar{c}\bar{d} + \bar{b}c\bar{d}}} = \overline{\bar{a} \cdot (\overline{b\bar{c}\bar{d}}) \cdot (\overline{\bar{b}c\bar{d}})} \quad (1.21),$$

$$y_{KNF} = (a + \bar{d})(a + \bar{b} + \bar{c})(a + b + c) = (\overline{\bar{a}d}) \cdot (\overline{\bar{a}bc}) \cdot (\overline{\bar{a}\bar{b}\bar{c}}) = \overline{(\bar{a}d) \cdot (\bar{a}bc) \cdot (\bar{a}\bar{b}\bar{c})} \quad (1.22).$$

Анализ выражений (1.21) и (1.22) показывает, что функциональная схема (рис. 1.10, а), реализующая заданную функцию в базисе И-НЕ, будет содержать меньшее количество элементов Шеффера ($3 < 5$), если её строить, используя выражение (1.21).

Для записи логической функции $y(a, b, c, d)$ в базисе ИЛИ-НЕ применим к правой части выражений (1.19), (1.20) правило де Моргана и правило двойного отрицания в разной последовательности. После преобразований получим:

$$y_{DNF} = a + b\bar{c}\bar{d} + \bar{b}c\bar{d} = a + (\overline{\bar{b} + c + d}) + (\overline{b + \bar{c} + d}) = \overline{\overline{a + (\bar{b} + c + d) + (b + \bar{c} + d)}} \quad (1.23),$$

$$y_{KNF} = \overline{\overline{(a + \bar{d})(a + \bar{b} + \bar{c})(a + b + c)}} = \overline{\overline{(a + \bar{d}) + (a + \bar{b} + \bar{c}) + (a + b + c)}} \quad (1.24).$$

Из анализа выражений (1.23) и (1.24) следует, что функциональные схемы, реализующие эти выражения, будут содержать одинаковое (4) количество элементов Пирса (для реализации 1.23 понадобится дополнительный элемент для заключительной инверсии). На рис. 1.10, б приведена функциональная схема, реализующая логическую функцию (1.24).

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

Рисунок 1.9 — Минимизация функции в форме ДНФ и КНФ

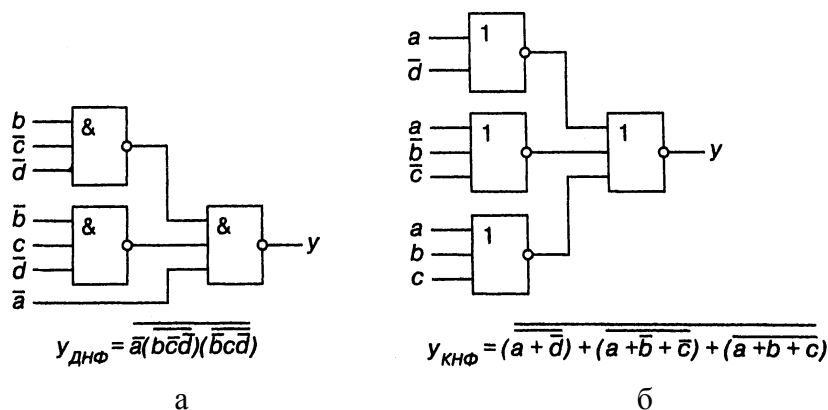


Рисунок 1.10 — Аппаратная реализация логической функции: а — в базисе И-НЕ; б — в базисе ИЛИ-НЕ

1.9 Коды и системы счисления

Существующие системы счисления подразделяются на *позиционные* и *непозиционные*. В непозиционных системах значение конкретной цифры постоянно и не зависит от ее расположения в записи числа. Примером такой системы счисления является Римская система записи числа. Например, в числе XXXVII значение цифры X не зависит от ее местоположения в записи числа. Оно везде равно 10.

В позиционных системах счисления значимость конкретной цифры определяется ее местоположением в записи числа. Так, произвольное число A в позиционной системе счисления с основанием q в общем случае можно представить в виде полинома:

$$A = (a_{n-1} a_{n-2} \dots a_0)_q = a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \dots + a_0 \cdot q^0,$$

где a_i — разрядный коэффициент $\{a_i=0, \dots, q-1\}$; q — весовой коэффициент или основание системы счисления.

Числа (кодовые слова) в цифровых устройствах обычно представляют в *позиционной двоичной* системе счисления.

Запись натурального числа A в позиционной двоичной системе счисления:

$$A = (a_{n-1} a_{n-2} \dots a_0)_2 = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_0 \cdot 2^0 \quad (1.25).$$

Представление дробных чисел, значения которых не превышают единицы:

$$A = (0, a_{n-1} a_{n-2} \dots a_0)_2 = \frac{a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_0 \cdot 2^0}{2^n} = a_{n-1} \cdot 2^{-1} + \dots + a_0 \cdot 2^{-n} \quad (1.26).$$

В формулах (1.25, 1.26) a_i — разрядные коэффициенты, принимающие значения 1 и 0; n — число разрядов в коде. Например, $26_{(10)} = 11010_{(2)}$, $n = 5$.

Код, построенный по (1.25, 1.26), принято относить к *арифметическим кодам*, на которые распространяются арифметические операции сложения, вычитания, умножения и деления.

Число символов в кодовом слове цифрового устройства обычно фиксировано, т. е. кодовые слова имеют одинаковую длину. Если кодовое слово имеет n символов (разрядов), то из них можно составить $N=2^n$ комбинаций кодовых слов. Например, в 32-разрядном вычислительном устройстве можно закодировать $2^{32} = 4\,294\,967\,296$ слов.

Для оценки количества цифровой информации используют бит и байт (1 байт=8 бит). 1 бит — это мера информации, выражающая такое её количество, которое может передать один символ двоичного алфавита при равной вероятности появления каждого символа алфавита:

$$I = \log_2 m = \log_2 2 = 1 \text{ бит} \quad (1.27),$$

где $m=2$ — число символов в бинарном алфавите.

В цифровой электронике это понятие распространяют на двоичные системы с любым числом разрядов, полагая при этом информационную ёмкость каждого разряда, равной 1 биту. Так, в 8-разрядном слове информационная ём-

кость равна 8 битам или 1 байту.

В цифровых вычислительных системах наряду с двоичной BIN широко используют также восьмеричную OCT (для кодирования адресов и команд), шестнадцатеричную HEX (цифробуквенную), двоичную-десятичную BCD системы счисления.

Ниже приведены символы алфавитов вышеперечисленных систем счисления.

BIN — 0, 1

OCT — 0, 1, 2, 3, 4, 5, 6, 7

HEX — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

BCD₈₋₄₋₂₋₁ — 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001

Преобразование целой части числа из десятичной в другую форму счисления. Для преобразования целой части десятичного числа в иную систему счисления нужно последовательно делить ее на основание новой системы счисления и записывать остатки в качестве значащих цифр результата. Алгоритм такого преобразования имеет следующий вид:

1. Поделить десятичное число на основание новой системы счисления.
2. Сохранить остаток (первый остаток будет цифрой, стоящей в самом младшем значащем разряде).
3. Повторять шаги 1 и 2 до тех пор, пока частное не будет нулевым.
4. Записать остатки от деления на основание в обратном порядке (последний остаток будет старшим (крайним слева) разрядом преобразованного числа).

Пример. Преобразовать $25_{(10)}$ в двоичную систему счисления.

Решение.

$$25:2 = 12 + 1_{(ост)}$$

$$12:2 = 6 + 0_{(ост)}$$

$$6:2 = 3 + 0_{(ост)}$$

$$3:2 = 1 + 1_{(ост)}$$

$$1:2 = 0 + 1_{(ост)}$$

$$\text{Итак, } 25_{(2)} = 11001$$

Преобразование дробной части десятичного числа. Преобразование дробной части десятичного числа в другую систему счисления выполняет посредством умножения на значение основания системы счисления. Алгоритм для преобразования дробной части десятичного числа имеет следующий вид:

1. Умножить дробную часть на основание новой системы счисления.
2. Сохранить целую часть результата (даже если она нулевая) в качестве цифры. Обратите внимание на то, что первый результат записывается непосредственно справа от точки, разделяющей целую и дробную часть числа.
3. Повторять шаги 1 и 2, используя дробную часть шага 2 до тех пор, пока дробная часть не станет равной нулю. (Следует обратить внимание на то, что некоторые числа бесконечны — периодические дроби, следовательно, в этом случае процесс может продолжаться бесконечно).

4. Записать целые части от последовательных умножений в прямом порядке, начиная с первой справа позиции от точки, разделяющей целую и дробную части.

Каким образом можно представить десятичное число 926 в двоичной форме? Преобразование этого числа из десятичной системы в двоичную можно осуществить, пользуясь способом, описанным выше на стр. 20. Полученное двоичное число 1110011110 большинству из нас мало что говорит. Код, в котором двоичная система счисления используется несколько иным образом, чем в предыдущем примере, называется двоично-десятичным кодом 8421. Именно этот код часто имеют в виду, когда говорят просто о *двоично-десятичном коде*.

Преобразование десятичного числа 926 в этот код дает результат в виде последовательности двоичных тетрад 1001 0010 0110, каждая из которых кодирует соответствующую десятичную цифру.

Представьте, что вам дано число 0001 1000 0111 0001, записанное в BCD-коде 8421. Какому десятичному числу оно соответствует? Чтобы узнать это, надо разбить число на тетрады *слева направо* и считать закодированные в двоичном коде десятичные цифры (1871_{10}).

В **коде 8421** каждая десятичная цифра кодируется обычным позиционным 4-хразрядным двоичным числом. Этот код употребляется чаще других. Его преимущество перед другими заключается, во-первых, в том, что двоичные комбинации для одnorазрядных десятичных чисел в нем такие же, как и при обычном двоичном счете, и, во-вторых, в том, что этот код однозначен: для каждого десятичного числа существует только одна соответствующая ему кодовая комбинация. Другие же коды неоднозначны. В табл. 1.5, например, показаны два кода 4-2-2-1, т. е. два кода с одинаковыми весами разрядов, но с разными двоичными комбинациями для одних и тех же десятичных чисел.

Наряду с кодом 8-4-2-1 находят широкое применение также и другие двоично-десятичные коды. При этом взвешенные коды (2-4-2-1, 5-4-2-1 и др.) применяются чаще, чем невзвешенные (код с изб. 3, $3a+2$, 2 из 5), что объясняется тем, что взвешенные коды более приемлемы при построении цифро-аналоговых преобразователей.

Особую группу двоично-десятичных кодов образуют *самодополняющиеся коды* (код с изб. 3, $3a+2$, 2-4-2-1, 4-2-2-1). Для самодополняющихся кодов характерно то, что при поразрядном инвертировании кодовой комбинации данного десятичного числа получается кодовая комбинация, дополняющая данное число до девяти (*дополнение*). Иначе говоря, *в самодополняющемся коде обратные двоичные числа соответствуют обратным десятичным числам*. Это свойство кода очень удобно при построении цифровых приборов, измеряющих как положительные, так и отрицательные величины, а также при реализации арифметических операций.

В коде с избытком 3 каждое десятичное число кодируется двоичным числом, на три единицы превышающим десятичное число, поэтому числу 0 соответствует код 0011, отображающий число 3; числу 9 соответствует код 1100, отображающий число 12;

Код 8421 очень широко применяется в цифровых системах. Как уже было отмечено, его часто называют просто двоично-десятичным кодом. Здесь, однако, следует соблюдать определенную осторожность, поскольку существуют двоично-десятичные коды и с другими весами числовых разрядов, например

код 4221, код с избытком 3 и др.

Помехозащищенные коды. Код Грея, код с контролем паритета, код Хемминга.

Код Грея. В табл. 1.5 код Грея сопоставляется с некоторыми уже известными вам кодами. Важной особенностью кода Грея является то, что при переходе к следующему, ниже расположенному числу в предыдущем числе изменяется только одна цифра (см. правый столбец табл. 1.5). Код Грея нельзя использовать в арифметических схемах. Этот код применяется во входных и выходных устройствах цифровых систем (чаще всего для кодирования пространственного (углового) перемещения), в так называемых датчиках-энкодерах, рис. 1.11. Его использование удобно тем, что два соседних значения шкалы сигнала отличаются только в одном разряде. Также он используется для кодирования номеров дорожек в жёстких дисках. Помехозащищённость кода Грея заключается в том, что при ошибочном кодировании ошибка составляет единицу младшего разряда.

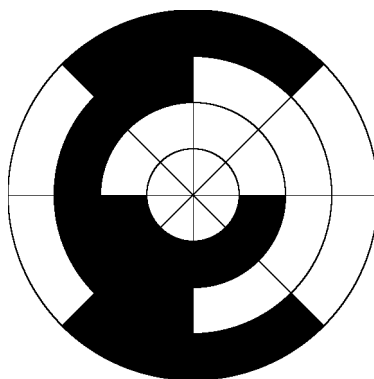


Рисунок 1.11 — Датчик-энкодер углового перемещения

Из табл. 1.1 видно, что код Грея нельзя считать одним из многочисленных вариантов двоично-десятичного кода. Заметьте также, что довольно трудно переводить десятичные числа в код Грея и переходить обратно от кода Грея к десятичным числам. Конечно, есть способы такого перевода, но обычно для этой цели используют электронные дешифраторы.

Алгоритм перехода от числа в позиционном двоичном коде к коду Грея следующий. Двоичное число надо сложить по модулю 2 (выполнить поразрядную операцию Искл-ИЛИ) с этим же числом, сдвинутым на один разряд вправо. Младший разряд при сложении не учитывается:

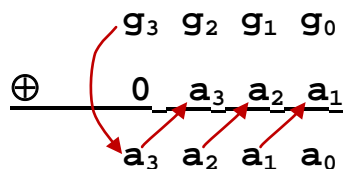
$$\begin{array}{rcccc}
 & a_3 & a_2 & a_1 & a_0 \\
 \oplus & 0 & a_3 & a_2 & a_1 \\
 \hline
 g_3 & g_2 & g_1 & g_0
 \end{array}$$

Пример. Преобразовать число 6 в двоичном позиционном коде в код Грея

$$\begin{array}{rcccc}
 & 1 & 1 & 0 & \text{BIN} \\
 \oplus & 0 & 1 & 1 & \\
 \hline
 & 1 & 0 & 1 & \text{Gray}
 \end{array}$$

Алгоритм обратного перехода следующий. Старший разряд кода Грея (g_i)

переносится в соответствующий старший разряд числа в преобразуемом двоичном позиционном коде (a_i). Для получения следующего, более младшего разряда преобразуемого позиционного кода (a_{i-1}), соответствующий разряд в коде Грея (g_{i-1}) складывается по модулю 2 с полученным ранее на предыдущем шаге более старшим разрядом позиционного кода (a_i). Процесс повторяется до нахождения самого младшего разряда позиционного кода.



Пример. Преобразовать число 110 в коде Грея в двоичный позиционный код.

$$\begin{array}{r} \oplus \quad \begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array} \end{array} \quad \begin{array}{l} \text{Gray} \\ \\ \text{BIN} \end{array}$$

Помехозащищенный код с контролем паритета. Относится к самоконтролирующимся кодам, т.е. кодам, позволяющими автоматически обнаруживать ошибочную передачу данных. Для его построения достаточно приписать к каждому слову один добавочный (контрольный) двоичный разряд и выбрать цифру этого разряда так, чтобы общее количество единиц в изображении любого числа было, например, четным. Одиночная ошибка в каком-либо разряде передаваемого слова (в том числе, может быть, и в самом контрольном разряде) изменит четность общего количества единиц. Счетчики по модулю 2, подсчитывающие количество единиц, которые содержатся среди двоичных цифр числа, могут давать сигнал о наличии ошибок.

При этом, разумеется, мы не получаем никаких указаний о том, в каком именно разряде произошла ошибка, и, следовательно, не имеем возможности исправить её. Остаются незамеченными также ошибки, возникающие одновременно в двух, в четырёх или вообще в четном количестве разрядов.

k	a₆	a₅	a₄	a₃	a₂	a₁	a₀
1	1	0	1	1	0	1	1

$$P = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

k	a₆	a₅	a₄	a₃	a₂	a₁	a₀
1	1	0	1	0	0	1	1

$$P = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 1 \Rightarrow$$

⇒ произошла ошибка!!!

Код Хемминга. Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися. Для построения самокорректирующегося кода, рассчитанного на исправление одиночных ошибок, одного контрольного разряда недостаточно. Количество контрольных разрядов k должно быть выбрано так, чтобы выполнялось неравенство $2^k \geq k + m + 1$ или $k \geq \log_2(k + m + 1)$, где m — количество основных (информационных) двоичных разрядов кодового слова. Т.е. число, кодируемое контрольными разрядами, должно описывать $(m+k+1)$ различных событий, включая отсутствие ошибки (1 событие) или искажение одного из $(m+k)$ разрядов ($m+k$ событий).

Таблица 1.6 Минимальные значения k при заданных значениях m для построения кода с исправлением одиночной ошибки

Диапазон m	k_{\min}
1	2
2-4	3
5-11	4
12-26	5
27-57	6

Имея $m+k$ разрядов, самокорректирующийся код можно построить следующим образом. Присвоим каждому из разрядов свой номер — от 1 (крайнему справа) до $m+k$ (крайнему слева); запишем эти номера в двоичной системе счисления. Поскольку $2^k > m + k$, номер любого разряда можно представить, очевидно, k -разрядным двоичным числом.

Предположим далее, что все $m+k$ разрядов кода разбиты на контрольные группы, которые частично перекрываются, причем так, что *единицы в двоичном представлении номера разряда указывают на его принадлежность к определённым контрольным группам*. Например: разряд №5 принадлежит к 1-ой и 3-ей контрольным группам, потому что в двоичном представлении его номера $5_{10} = 0101_2$ — 1-й и 3-й разряды содержат единицы.

Среди $m+k$ разрядов кода при этом имеется k разрядов, каждый из которых принадлежит только к одной контрольной группе:

Разряд № 1: $1_{10} = \dots 00001_2$ принадлежит только к 1-й контрольной группе.

Разряд № 2: $2_{10} = \dots 00010_2$ принадлежит только к 2-й контрольной группе.

Разряд № 4: $4_{10} = \dots 00100_2$ принадлежит только к 3-й контрольной группе.

...

Разряд № 2^{k-1} принадлежит только к k -ой контрольной группе.

Эти k разрядов мы и будем считать контрольными. Остальные m разрядов, каждый из которых принадлежит по меньшей мере к двум контрольным группам, будут информационными разрядами.

В каждой из k контрольных групп будем иметь по одному контрольному разряду. В каждый из контрольных разрядов поместим такую цифру (0 или 1), чтобы общее количество единиц в его контрольной группе было четным (табл. 1.6).

Например, довольно распространен код Хемминга с $m=7$ и $k=4$.

Пусть исходное слово (кодовое слово без контрольных разрядов) — 1010110_2 .

Обозначим P_i — контрольный разряд № i ; а D_j — информационный разряд № j , где $i = 1, 2, \dots, k$, $j = 1, 2, \dots, m$.

Предположим теперь, для примера, что при передаче данного кодового слова 10100110001 произошла ошибка в 11-м разряде, так, что было принято новое кодовое слово 0100110001 . Произведя в принятом коде проверку четности внутри контрольных групп, мы обнаружили бы, что количество единиц нечетно в 1-й, 2-й и 4-й контрольных группах, и четно в 3-й контрольной группе. Это указывает, во-первых, на наличие ошибки, во-вторых, означает, что номер ошибочно принятого разряда в двоичном представлении содержит единицы на первом, втором и четвертом местах и нуль — на третьем месте, т.к. ошибка только одна, и 3-я контрольная сумма оказалась верной (табл. 1.7).

Таблица 1.6 — Формирование значений контрольных разрядов кода Хемминга

№ разряда BIN:		1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Распределение контрольных и информационных разрядов		d ₇	d ₆	d ₅	p ₄	d ₄	d ₃	d ₂	p ₃	d ₁	p ₂	p ₁
Информационное кодовое слово:		1	0	1		0	1	1		0		
Вычисление контр. разрядов в контр. группах	p ₁	1		1		0		1		0		1
	p ₂	1	0			0	1			0	0	
	p ₃					0	1	1	0			
	p ₄	1	0	1	0							
Кодовое слово с контрольными разрядами:		1	0	1	0	0	1	1	0	0	0	1

Таблица 1.7 — Проверка контрольных сумм на приемной стороне

№ разряда:	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001		
Распределение контрольных и информационных разрядов	d₇	d₆	d₅	p₄	d₄	d₃	d₂	p₃	d₁	p₂	p₁	Кон- троль по четности в группе	Кон- троль ный бит
Переданное кодовое слово:	1	0	1	0	0	1	1	0	0	0	1		
Принятое кодовое слово:	0	0	1	0	0	1	1	0	0	0	1		
p ₁	0		1		0		1		0		1	Fail	1
p ₂	0	0			0	1			0	0		Fail	1
p ₃					0	1	1	0				Pass	0
p ₄	0	0	1	0								Fail	1

p₄ p₃ p₂ p₁

В двоичном представлении 1 0 1 1

В десятичном представлении 8+2+1=11

Из таблицы следует, что ошибка произошла в 11-м разряде и её можно исправить (проинвертировав принятый 11-ый разряд d₇). Построенный код, разумеется, не рассчитан на возможность одновременной ошибки в двух разрядах.

1.10 Компьютерные форматы данных

Успешное программирование требует полного понимания форматов данных. Обычно данные представляются в микрокомпьютерной системе в следующих основных видах: формате ASCII, как двоично-кодированные десятичные числа (BCD), целые числа со знаком и без, а также иногда как числа с плавающей точкой (действительные числа). Могут быть использованы и другие формы представления данных, однако они здесь не описываются, поскольку не являются столь же широко распространенными.

Представление отрицательных чисел. Дополнительный код.

При записи двоичного числа в *прямом коде* старший разряд является знаковым разрядом. Если его значение равно 0 — то число положительное, если 1 — то отрицательное. В остальных разрядах (которые называются цифровыми разрядами) записывается двоичное представление модуля числа.

При представлении отрицательных чисел, данные сохраняются в форме дополнения. При этом используются две системы. Одна использует дополнение до основания системы счисления (*дополнительный код*), а вторая — дополнение до основания системы счисления -1 (*обратный код*). Первым появился обратный код, в котором при представлении отрицательного числа каждая цифра числа вычитается из основания системы счисления минус 1 с целью получения значения дополнения до основания системы счисления -1 .

Получим дополнение 8-разрядного двоичного числа 01001100 до значения основания системы счисления минус 1 с целью отображения его как отрицательного значения. Обратите внимание на то, что каждая цифра числа вычитается из единицы для генерирования дополнения до значения основания системы счисления минус 1 (единица). В данном примере отрицательное число $-01001100_{(2)} = -76_{(10)}$ представляется как $10110011_{(2)}$. Та же самая методика может применяться для любой системы счисления, но в *применении к двоичной системе* счисления получение *обратного* кода может быть достигнуто путем инвертирования каждого двоичного разряда числа.

Сегодня само по себе дополнение до основания системы счисления -1 (*обратный код*) не используется; оно используется только в качестве этапа на пути к получению дополнения до основания системы счисления. Основная проблема дополнения до основания системы счисления -1 заключается в том, что при использовании этой системы возникают два нуля: положительный (00000000) и отрицательный (11111111); в системе, использующей дополнение до основания системы счисления, существует только положительный ноль.

Таким образом, дополнение до основания системы счисления (*дополнительный код*) используется для представления отрицательных чисел в современных компьютерных системах.

Для формирования дополнения до основания системы счисления сначала находят дополнение до основания системы счисления -1 (*обратный код*), а затем добавляют единицу к результату. Пример ниже показывает, как число $01001000_{(2)} = 72_{(10)}$ преобразуется в отрицательное значение переводом его в дополнение до двух (основание системы счисления):

$$01001000 \rightarrow 10110111 \rightarrow 10110111 + 1 = 10111000 = -72_{(10)}$$

Чтобы убедиться в том, что 1011 1000 является инверсией (отрицательным значением 0100 1000), сложим их для формирования 8-разрядного результата:

$$\begin{array}{r} 01001000 \\ + 10111000 \\ \hline 100000000 \end{array}$$

Девятый разряд отбрасывается и результат будет нулем, потому что 01001000 — это положительное 72, в то время как 10111000 — это отрицательное 72 в дополнительном коде. Та же самая методика применима для любой системы счисления.

При использовании дополнительного кода изменяется диапазон представления чисел машинным словом заданной длины.

Еще для представления отрицательных чисел используют так называемый *смещенный код*. Для его получения прибавляют к числу в обычном дополнитель-

ном коде половину диапазона, для байта — это 128_{10} . Такое преобразование эквивалентно инверсии старшего (знакового) разряда, при этом положительные числа имеют десятичный эквивалент от 129 до 255, а отрицательные — от 0 до 127. Такой код используется при цифро-аналоговом преобразовании чисел со знаком.

Байтовые данные хранятся как в форме *целых чисел без знака*, так и в форме *целых чисел со знаком*, при этом отрицательные числа хранятся в дополнительном коде и старший разряд байта является знаковым (1 — для отрицательных чисел, 0 — для положительных). Рис. 1.12 иллюстрирует как знаковую, так и беззнаковую формы байтовых целых чисел. Разница между этими формами заключается в весе самого левого разряда. Его значение равно 128 для беззнакового целого и -128 для целого со знаком. В формате целого со знаком самый левый разряд одновременно является как знаковым разрядом числа, так и разрядом с весом минус 128. Например, $0x80=10000000_{(2)}$ представляет значение 128 как беззнакового числа; в качестве же числа со знаком оно представляет число -128 . Целые числа без знака имеют диапазон значений $0x00 - 0xFF$ ($0 - 255_{10}$). Целые числа со знаком имеют диапазон значений -128_{10} до $+127_{10}$.

128	64	32	16	8	4	2	1	
0	0	0	0	1	0	0	0	$8_{(10)}$

-128	64	32	16	8	4	2	1	
1	1	1	1	1	0	0	0	$-8_{(10)}$

Рисунок 1.12 — Байтовые (8-битные) данные

Метод оценки чисел со знаком путем использования весов позиций каждого разряда является намного более легким, чем операция нахождения дополнения до двух числа с целью определения его абсолютного значения.

Всякий раз, когда выполняется нахождение дополнения до двух числа, знак этого числа изменяется с отрицательного на положительный либо с положительного на отрицательный. Например, число 00001000 — это $+8$. Его отрицательное значение (-8) находится посредством определения дополнения до 2-х числа $+8$. Для нахождения дополнения числа до 2-х сначала находится его дополнение до одного. Для нахождения дополнения числа до одного, каждый разряд этого числа инвертируется — с нуля на единицу либо с единицы на ноль. Как только дополнение до единицы будет сформировано, определяется дополнение до двух посредством добавления единицы к дополнению до одного (рис. 1.12).

Данные в формате ASCII (Американский Стандартный Код для Информационного Обмена — American Standard Code for Information Interchange) представляют алфавитно-цифровые символы в памяти компьютерной системы (табл. 1.8). Стандартный ASCII-код является 7-битным кодом, в котором восьмой (самый старший бит), используется как бит паритета в некоторых устаревших системах. Если ASCII-данные используются для вывода на принтер, то нулевое значение самого старшего бита будет означать режим алфавитно-цифровой печати, а единичное — режим графической печати. В персональном

компьютере расширенный набор символов ASCII выбирается установкой единичного состояния самого левого бита.

Управляющие символы ASCII, приведенные в табл. 1.8, выполняют функции управления в компьютерной системе, включая очистку экрана, символ «шаг назад», перевод строки и т.д. Для ввода управляющих кодов с компьютерной клавиатуры нужно, удерживая в нажатом состоянии клавишу Ctrl (Control), вводить соответствующие символы. Так для ввода 0x01, нужно ввести Control-A; а код 0x02 вводится как Control-B и т.д. Обратите внимание на то, что управляющие коды часто появляются на экране как ^A для Control-A, ^B для Control-B и т.д. Также обратите внимание на то, что код возврата каретки (CR) в большинстве современных клавиатур соответствует клавише <Enter> (Ввод). Назначением символа CR является возврат курсора или печатающей головки на левое поле. Другим кодом, который часто появляется во многих программах, является код перевода строки (LF), который перемещает курсор вниз на одну строку.

Таблица 1.8 — ASCII коды

1-я часть кода	Вторая часть кода															
	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	XA	XB	XC	XD	XE	XF
0X	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1X	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EMS	SUB	ESC	FS	GS	RS	US
2X	SP	!	"	#	\$	%	&	'	()	*	+	-	>		/
3X	0	1	2	3	4	5	6	7	8	9		'	<	=	>	?
4X	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	
6X	'	a	b	c	d	e	F	g	h	i	j	k	l	m	n	o
7X	P	q	r	s	t	u	V	w	x	y	z	{	I	}	~	

ASCII-данные чаще всего сохраняются в памяти при использовании специальной команды Ассемблера, которая называется `declare byte` (Объявить байт) или `DB`, либо командой `DATA` в случае строк.

Двоично-кодированные десятичные (BCD) данные могут сохраняться как в *упакованной*, так и *неупакованной* форме. *Упакованные BCD-данные* сохраняются в форме двух цифр на байт, а *неупакованные BCD-данные* сохраняются как одна цифра на байт. Диапазон BCD-цифр лежит от 0000_2 до 1001_2 , что соответствует десятичным цифрам от 0 до 9. Неупакованные BCD-данные поступают от микрокомпьютерной клавиатуры. Упакованные BCD-данные используются в некоторых командах набора команд микропроцессора, включая BCD-сложение и вычитание.

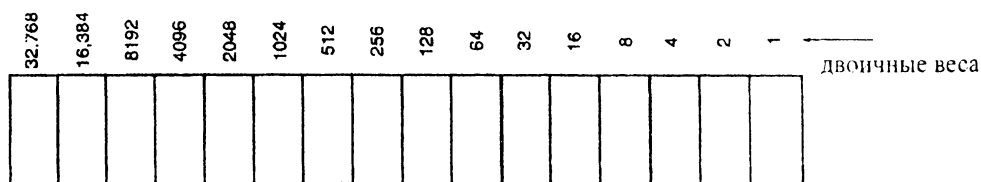
Табл. 1.9 показывает некоторые десятичные числа, преобразованные, как в упакованную, так и в неупакованную BCD-форму. Приложениями, которые в типичном случае требуют использования BCD-данных, являются локальные торговые терминалы, а также почти любые иные устройства, выполняющие минимальное количество простых арифметических операций. Если системы требуют выполнения сложных арифметических операций, то BCD-данные в них используются редко, поскольку нет простого и эффективного метода выполнения сложных арифметических BCD-операций.

Таблица 1.9 — Упакованные и неупакованные BCD-данные

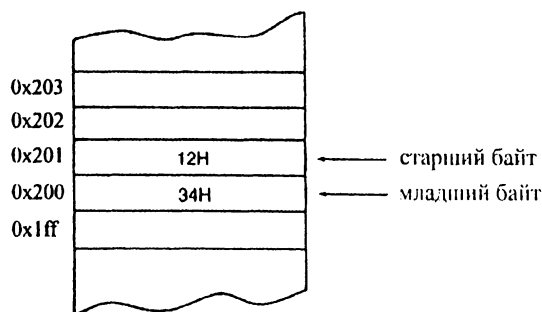
Десятичные	Упакованные		Неупакованные		
12	0001 0010		0000 0001	0000 0010	
623	0000 0110	0010 0011	0000 0110	0000 0010	0000 0011
910	0000 1001	0001 0000	0000 1001	0000 0001	0000 0000

Во всех случаях придерживаются соглашения о порядке записи данных в память, при котором первыми (по младшему адресу) записываются младшие разряды чисел. Это означает, что для сохранения в памяти, например, числа 623 в неупакованном BCD, сначала нужно сохранить 3, затем — 2, и наконец — 6.

Пословные данные. Слово (16 битов) формируется двумя байтами данных. Младший байт всегда сохраняется в ячейке памяти с меньшим номером, а старший — в ячейке с большим номером (адресом). Этот метод сохранения числа называется форматом с обратным порядком байтов. Альтернативный метод, используемый в некоторых других системах, называется форматом с прямым порядком. При использовании этого формата числа сохраняются так, что ячейка с меньшим номером содержит старший байт данных.



(а) слово без знака



(b) в ячейках памяти 0x200 и 0x201 записано слово 0x1234

Рисунок 1.13 — Пословные (16 бит) данные

Рис. 1.13, сверху показывает веса для каждой битовой позиции слова данных, а рис. 1.13 внизу показывает, как число 1234H заносится в ячейки памяти 200H и 21H. Единственная разница между словом со знаком и беззнаковым словом связана с самой левой битовой позицией. В беззнаковой форме самый левый бит не связан со знаком и имеет вес 32768; в форме со знаком его вес будет равен -32768. Как и в случае байтовых данных со знаком, слово со знаком имеет форму дополнения до двух, когда оно представляет отрицательное число.

Действительные числа иногда встречаются в приложениях — здесь они рассматриваются в силу того, что иногда встроенные контроллеры должны иметь дело с действительными числами. *Действительное число или число с плавающей точкой*, как его часто называют, состоит из двух частей: мантиссы — значащей или дробной части и экспоненты. Рис. 1.14 показывает как 4-

байтовую, так и 8-байтовую форму действительных чисел, как они сохраняются в системах Intel. Обратите внимание на то, что 4-байтовая форма называется формой числа *одинарной точности*, а 8-байтовая форма называется формой числа *двойной точности*. Представленные здесь формы сохранения действительных чисел являются теми же, которые специфицированы стандартом IEEE-754, версия 10.0. Этот стандарт был принят в качестве стандартной формы представления действительных чисел практически всеми языками программирования, а также многими прикладными пакетами. Данный стандарт также применяется при манипулировании данными в арифметических сопроцессорах персональных компьютеров. Рис. 1.14, а показывает форму числа одинарной точности, которая содержит бит знака, 8-битную экспоненту и 24-битную дробную часть (мантиссу). Обратите внимание на то, что в силу того, что приложения часто требуют использования чисел с плавающей точкой двойной точности, соответствующая форма также представлена на рис. 1.14, б.

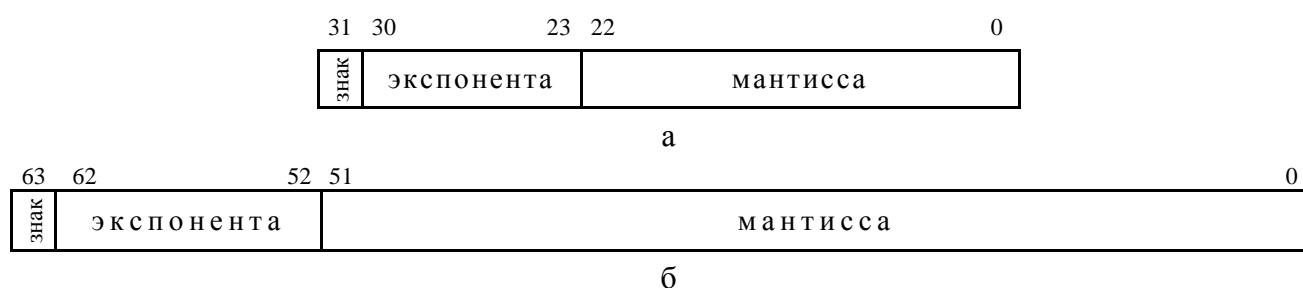


Рисунок 1.14 — Данные, заданные в формате с плавающей точкой: а — одинарной точности; б — двойной точности

Простой арифметический подсчет показывает, что для сохранения всех названных фрагментов данных необходимо 33 бита. Однако это не так — 24-битная мантисса содержит подразумеваемый (неявный) бит, что позволяет мантиссе представлять 24 бита, будучи сохраняемой всего в 23 битах. Неявный бит — это первый бит нормализованного действительного числа. При нормализации числа, оно подстраивается таким образом, что его значение по крайней мере равно единице, однако меньше 2. Например, если 12 преобразовать в двоичную форму (1100_2), то после нормализации результатом будет: 1.1×2^3 . Целая часть нормализованного числа (1) не сохраняется при использовании 23-битной части числа, задающей мантиссу; 1 находится в единичном «скрытом» бите. Табл. 1.10 показывает форму этого и других чисел одинарной точности.

Таблица 1.10 — Примеры чисел с одинарной точностью

Десятичное	Двоичное	Нормализованное	Знак	Смещенная экспонента	Мантисса		
+12	1100	1.1×2^3	0	10000010	1000000	00000000	00000000
-12	1100	1.1×2^3	1	10000010	1000000	00000000	00000000
+100	1100100	1.1001×2^6	0	10000101	1001000	00000000	00000000
-1.75	1.11	1.11×2^0	1	01111111	1100000	00000000	00000000
+0.25	0.01	1.0×2^{-2}	0	01111101	0000000	00000000	00000000
+0.0	0	0	0	00000000	0000000	00000000	00000000
±∞ (-∞)	±∞ (-∞)		0 (1)	11111111	0000000	00000000	00000000
+0.15625	0.00101	1.01×2^{-3}	0	01111100	0100000	00000000	00000000

Экспонента сохраняется в форме смещенной экспоненты (специфичный для данного представления формат чисел со знаком). В форме действительного числа одинарной точности смещение равно 127 (0x7F), а при использовании формы действительного числа двойной точности оно равно 1023 (0x3FF). Значения смещения и экспоненты складываются перед сохранением экспоненты в соответствующей части числа с плавающей точкой. Например, экспонента, равная 2^3 , представляется в форме смещенной экспоненты, равной $(127+3)$ или 130 (0x82) в форме действительного числа одинарной точности или как 1026 (0x402) в форме действительного числа двойной точности. Экспонента, равная 2^{-3} , представляется в форме смещенной экспоненты, равной $(127-3)=124$ (0x7C=01111100₍₂₎) в форме действительного числа одинарной точности или как $(1023-3)=1020$ (0x3FC=111111100) в форме действительного числа двойной точности.

Число 0 сохраняется в форме, все разряды которой равны нулю. Это соответствует минимально возможной экспоненте -127_{10} (-1023_{10}). Число «бесконечность» сохраняется в форме, все разряды экспоненты которой имеют единичное значение (максимально возможная экспонента $+128_{10}$, $(+1024_{10})$), а мантиссы — нулевое. Знаковый бит в этом случае указывает либо положительную, либо отрицательную бесконечность. В скобках указаны значения для плавающего формата двойной точности.

Бывают отличия в форматах. Так Microchip использует другой формат для чисел с плавающей точкой в компиляторе с языка C C18. Рис. 1.15 иллюстрирует разницу между форматом чисел с плавающей точкой фирмы Microchip и форматом IEEE. Разница заключается в размещении знакового бита числа. Эта разница не оказывает воздействия на работу компилятора C.

IEEE-754			
s e e e e e e e	e f f f f f f f	f f f f f f f f	f f f f f f f f
MICROCHIP			
e e e e e e e e	s f f f f f f f	f f f f f f f f	f f f f f f f f

s — знаковый бит

e — экспонента

f — мантисса

Рисунок 1.15 — Форматы чисел с плавающей точкой IEEE и MicroChip

2 ЭЛЕМЕНТЫ ЦИФРОВЫХ УСТРОЙСТВ

2.1 Комбинационные и последовательностные устройства

В общем случае на вход цифрового устройства поступает n ($n > 1$) двоичных переменных $X(x_1, x_2, \dots, x_n)$, а с выхода снимается m ($m > 1$) двоичных переменных $Y(y_1, y_2, \dots, y_m)$. При этом устройства осуществляют (реализуют) определенную связь (ФАЛ) между входными и выходными переменными.

По способу функционирования цифровые устройства делят на *комбинационные* и *последовательностные*.

Комбинационными устройствами или *автоматами без памяти* называют логические устройства, выходные сигналы которых однозначно определяются только действующей в настоящий момент на входе комбинацией переменных и не зависят от значений переменных, действовавших на входе ранее.

В качестве примера на рис. 2.1, б приведена временная диаграмма работы логического элемента И-НЕ с двумя входами и одним выходом (рис. 2.1, а).

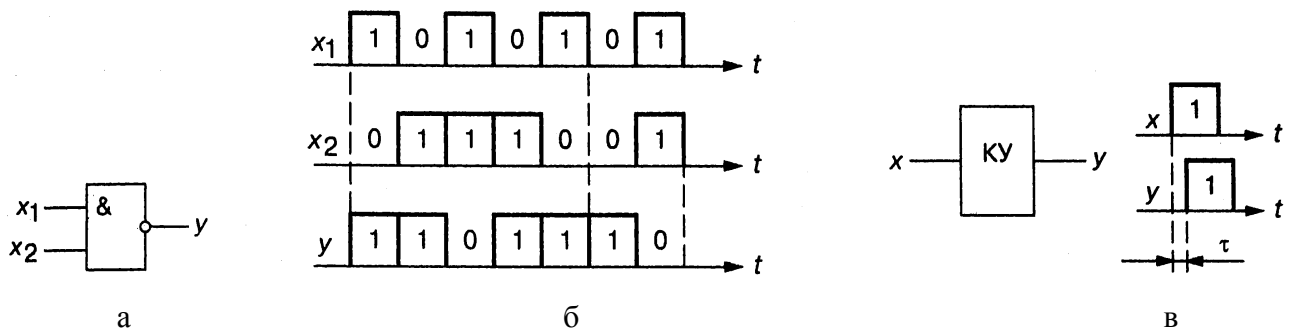


Рисунок 2.1 — Функционирование комбинационных цифровых устройств

В теоретических рассмотрениях (как и в данной теме) обычно считается, что сигналы на выходе комбинационного устройства (КУ) появляются в тот же момент, когда на вход устройства поступают инициирующие их входные сигналы, т. е. предполагается их быстроедействие бесконечным.

В реальных электронных приборах задержка выходного сигнала $\tau > 0$ ($\tau = 10^{-8} \dots 10^{-10}$ с) — естественный физический предел электронных приборов, (рис. 2.1, в), а в сложных устройствах с последовательным соединением логических элементов задержки суммируются.

Комбинационные узлы и блоки цифровых устройств либо собирают из отдельных микросхем малой степени интеграции (элементов И-НЕ, ИЛИ-НЕ и др.), либо изготавливают в виде систем средней интеграции, либо входят в состав БИС и СБИС. Различные типы комбинационных узлов и блоков широко используются в устройствах ввода-вывода и управления, в оперативных запоминающих устройствах современных цифровых систем.

По функциональному назначению можно выделить следующие классы КУ: *сумматоры, шифраторы и дешифраторы, цифровые компараторы, мультиплексоры и демультиплексоры, преобразователи кодов, программируемые логические матрицы (ПЛИМ), перемножители, арифметико-логические устройства.*

Для построения цифровых систем, кроме комбинационных узлов, требуются последовательностные устройства (автоматы с памятью), значения вы-

ходных двоичных переменных Y которых определяются как значениями входных переменных X в течение рассматриваемого такта работы, так и предысторией системы (предыдущим состоянием).

Последовательными устройствами, или автоматами с памятью, называют логические устройства, выходные сигналы которых определяются не только действующей в настоящий момент на входе комбинацией переменных, но и всей последовательностью входных переменных, действовавших в предыдущие моменты времени. Этот тип устройств часто называют цифровыми автоматами.

В последовательных устройствах, кроме логических, должны быть еще и запоминающие элементы — элементы памяти с множеством входов и выходов (рис. 2.2), включаемые в цепи обратной связи, в соответствии со стрелками, указывающими прохождение сигналов. При этом память системы может охватывать не бесконечно большое, а лишь конечное число тактов, формируемых генератором тактовых импульсов (ГТИ) (см. рис. 2.2). Поэтому цифровые устройства с памятью называют *конечными автоматами*, которыми являются все ЭВМ. Следует отметить, что сам элемент памяти содержит также внутренние обратные связи и является последовательным устройством.

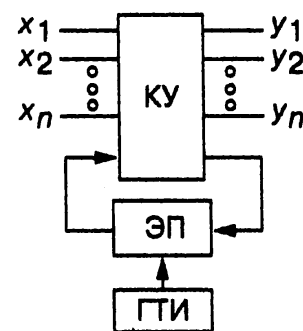


Рисунок 2.2 — Схема автомата с памятью

Обозначим t и $(t+1)$ два следующих друг за другом такта конечного автомата. Состояние элементов памяти в $(t+1)$ -й такт определяется множествами как входных x сигналов, так и сигналов q на выходах элементов памяти в предыдущий такт t , т. е.:

$$q_{t+1} = f(q_t, x_{t+1}) \text{ или } (q_1 q_2 \dots q_r)^{t+1} = f(x_1 x_2 \dots x_n, q_1 q_2 \dots q_r)^t$$

Это выражение называют *функцией переходов* автомата с памятью.

Выходные же сигналы Y могут формироваться двояко. В *автоматах Мили* они являются функциями как входных сигналов, так и сигналов элементов памяти в этом же такте, т. е. функция выходов автомата Мили:

$$y_t = \varphi(q_t, x_t).$$

В *автоматах Мура* они являются функциями только сигналов элементов памяти в этом же такте (состояния автомата), т. е. функция выходов автомата Мура:

$$y_t = \varphi(q_t)$$

Существуют правила перехода от одного вида автоматов к другому.

Функции переходов q_i и выходов y_i последовательных устройств представляют в виде таблиц переходов и выходов или объединенными таблицами состояний. Как отмечалось, реальные элементы всегда инерционны. Таблицы и алгебраические функции соответствуют статическим режимам. В динамических режимах (в переходной части тактов) связь между переменными может оказаться отличающейся от режима статики, что может привести к сбоям (ошибкам), т. е. к появлению ошибочных сигналов на входах памяти, а также к ошибочным состояниям автомата. Это явление называют «гонками» в автома-

тах, и его надо учитывать при синтезе конкретных цифровых блоков и устройств.

Учитывая это явление, во многих автоматах, называемых синхронными, элементы памяти управляются внешними тактовыми импульсами ГТИ (рис. 2.2), обеспечивающими переход элементов памяти из состояния t в состояние $(t+1)$ и выдачу сигналов y_i после завершения переходных процессов в момент подачи (или окончания) тактового импульса. В асинхронных автоматах изменение входных сигналов сразу влечет за собой соответствующие изменения выходных сигналов.

Простейшими конечными автоматами являются *триггеры*, они же, в свою очередь, являются элементами памяти более сложных цифровых устройств. Основными типами функциональных узлов последовательностных устройств являются *триггеры*, *регистры*, *счетчики* и генераторы чисел.

Познакомимся с некоторыми типовыми узлами и блоками комбинационных и последовательностных устройств в интегральном исполнении.

2.2 Шифраторы, дешифраторы, преобразователи кодов

Дешифратор или декодер — комбинационная схема с n входами и m выходами ($m > n$), преобразующая двоичный входной n -разрядный код (кодированное слово) в унитарный (один из m).

На одном из m выходов дешифратора появляется логическая единица (ноль, если выходы инверсные, с кружками), а именно на том, номер которого равен поданному на вход двоичному числу. На всех остальных выходах дешифратора выходные сигналы равны нулю (единице при инверсных выходах). Дешифратор используют, когда нужно обращаться к различным цифровым устройствам, и при этом номер устройства — его адрес представлен двоичным кодом.

Условное изображение дешифратора 4→16 (читаемого «четыре в шестнадцать») на схемах представлено на рис. 2.3. Полный дешифратор DC содержит число выходов, равное числу комбинаций входных пе-

ременных, например, при $n=4$, $m=2^n=16$, от $y_0 = \overline{x_8}\overline{x_4}\overline{x_2}\overline{x_1}$ до $y_{15} = x_8x_4x_2x_1$. Обратим внимание, что в конъюнкциях используется общая инверсия, поскольку выходы дешифратора — инверсные.

Каждый выход полного дешифратора реализует конъюнкцию входных переменных (код адреса) или их инверсий: при наборе $\overline{x_8}\overline{x_4}\overline{x_2}\overline{x_1}$ (0000), $y_0 = 0$; при $x_8\overline{x_4}\overline{x_2}\overline{x_1}$ (1000) $y_8=0$; при $x_8x_4x_2x_1$ (1111), $y_{15}=0$ и т. д.

Применяются также неполные дешифраторы с меньшим числом выходов (например 10 у двоично-десятичного дешифратора 4→10 при четырех переменных на входе). В этом случае ряд комбинаций на входе не используется (рис. 2.4).

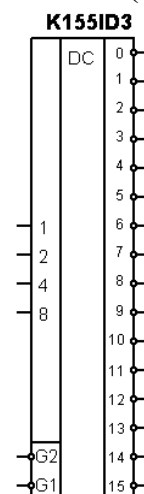


Рисунок 2.3 — Двоичный полный дешифратор 4-16

Дешифраторы часто имеют разрешающий (управляющий, стробирующий) вход G (G_1 , G_2 на рис. 2.3). При $G=1$ дешифратор функционирует как обычно, при $G=0$ на всех выводах устанавливается 0 независимо от поступающего кода адреса. Отметим, что для дешифратора ИДЗ (рис. 2.3) все происходит наоборот, поскольку и разрешающие входы G и выходы у него инверсные. Вход разрешения позволяет наращивать размерность дешифраторов, применяя многоступенчатую пирамидальную структуру. Дешифраторы широко используют во многих устройствах, в том числе в качестве преобразователей двоичного кода в унитарный, для выбора внешних устройств МПС по коду с шины адреса.

Шифратор (CD) или кодер (encoder) выполняет функцию, обратную дешифратору — преобразует унитарный код в двоичный (двоично-десятичный). Условное изображение шифратора $16 \rightarrow 4$ на схемах показано на рис. 2.5. Классический полный шифратор имеет n входов и m выходов ($m < n$), и при подаче активного уровня сигнала на один из входов (и не более) на его выходах появляется двоичный код номера возбуждённого входа. Число входов и выходов такого шифратора связано соотношением $n=2^m$.

Если $n < 2^m$, то шифратор называется неполным. Например, шифратор 9-4 может быть использован для преобразования нажатой клавиши 9-клавишного пульта в двоичный код (рис. 2.6).

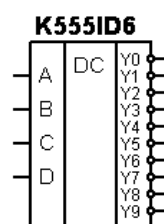


Рисунок 2.4 — Десятичный неполный дешифратор 4-10

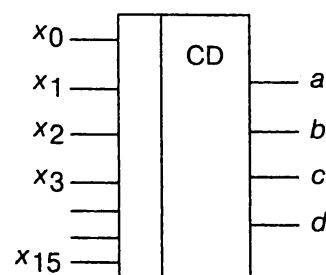
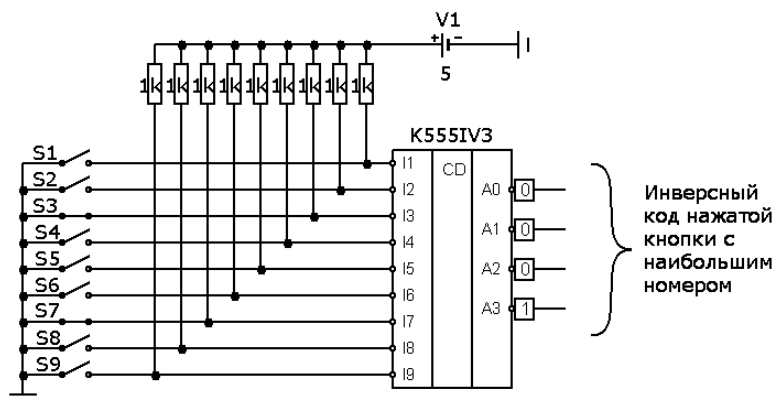


Рисунок 2.5 — Полный шифратор 16-4



Входы									Выходы			
I ₉	I ₈	I ₇	I ₆	I ₅	I ₄	I ₃	I ₂	I ₁	A ₃	A ₂	A ₁	A ₀
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	X	1	1	0	1
1	1	1	1	1	1	0	X	X	1	1	0	0
1	1	1	1	1	0	X	X	X	1	0	1	1
1	1	1	1	0	X	X	X	X	1	0	0	1
1	1	1	0	X	X	X	X	X	1	0	0	0
1	0	X	X	X	X	X	X	X	0	1	1	1
0	X	X	X	X	X	X	X	X	0	1	1	0

Рисунок 2.6 — Преобразование номера нажатой кнопки в ее двоичный код с помощью приоритетного шифратора

Области использования шифраторов — отображение в виде двоичного кода номера нажатой кнопки (рис. 2.6) или положения многопозиционного переключателя, а также номера устройства, подавшего сигнал на обслуживание в микропроцессорных системах, входят в состав микросхем контроллеров прерываний.

Как раз для случая контроллера прерываний характерно использование так называемого *приоритетного шифратора*. Он отличается от обычного тем, что при возбуждении одновременно нескольких входов на его выходе появится код соответствующий возбужденному входу с наибольшим номером (с наиболее высоким приоритетом, рис. 2.6). Примеры схем приоритетных шифраторов К555ИБ3 (аналог 74XXX147), К555ИБ1 (аналог 74148, но входы и выходы — инверсные).

Преобразователями кодов, в общем случае, называют устройства, предназначенные для преобразования одного кода в другой, при этом часто они выполняют нестандартные преобразования кодов. Преобразователи кодов обозначают через X/Y .

Рассмотрим особенности реализации преобразователя на примере преобразователя трехэлементного кода в пятиэлементный. Допустим, что необходимо реализовать таблицу соответствия кодов (табл. 2.1).

Таблица 2.1 — Таблица соответствия кодов кодопреобразователя

N	Трехэлементный код			Пятиэлементный код				
	x_3	x_2	x_1	y_5	y_4	y_3	y_2	y_1
0	0	0	0	0	0	0	1	1
1	0	0	1	0	0	1	1	0
2	0	1	0	0	1	0	0	1
3	0	1	1	1	0	0	0	1
4	1	0	0	0	1	1	0	0
5	1	0	1	1	0	1	0	0
6	1	1	0	1	1	0	0	0
7	1	1	1	0	0	1	1	0

Здесь через N обозначено десятичное число, соответствующее входному двоичному коду. Преобразователи кодов часто создают по схеме дешифратор — шифратор. Дешифратор преобразует входной код в некоторый унитарный код, а затем шифратор формирует выходной код. Схема преобразователя, созданного по такому принципу, приведена на рис. 2.7, где использован матричный диодный шифратор. Принцип работы такого преобразователя довольно прост. Например, когда на всех входах дешифратора логический «0», то на его выходе 0 появляется логическая «1», что приводит к появлению «1» на выходах y_2 и y_1 , т. е. реализуется первая строка таблицы соответствия кодов.

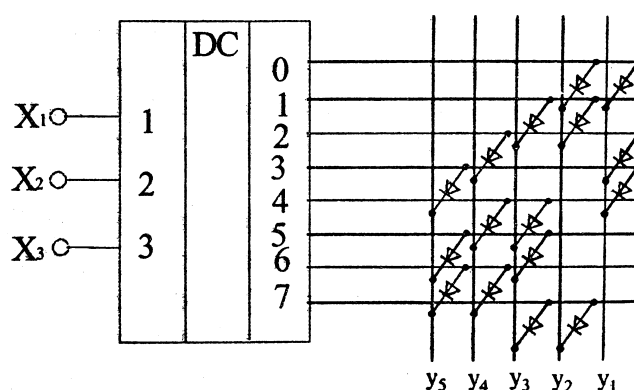


Рисунок 2.7 — Схема преобразователя кодов по структуре дешифратор-шифратор

При использовании в качестве второй ступени обычного шифратора (а не диодной матрицы) эффективно стыкуются друг с другом декодер и кодер, построенные на элементах И-НЕ: первый имеет инверсные выходы, а второй — инверсные входы. Если некоторым входным комбинациям соответствует одна и та же выходная, то соответствующие выходы декодера объединяют на элементе ИЛИ и выход последнего подают на нужный вход кодера.

Проектирование кодовой преобразовательной схемы на паре декодер-кодер оказывается в среднем более выгодным и по числу корпусов, и по быстродействию, чем при проектировании из готовых базовых логических микросхем И-НЕ и ИЛИ-НЕ при использовании методик минимизации. Однако потребляемая мощность в этом случае может оказаться больше, чем у схемы из отдельных элементов. Затраты времени разработчика на логическое проектирование по схеме декодер-кодер неизмеримо меньше, чем затраты на проектирование преобразователя кодов из россыпи логических микросхем.

Преобразователи кодов в виде микросхем выпускаются для выполнения таких операций, как преобразования двоично-десятичного кода в двоичный или обратного преобразования, для преобразования двоичного кода в код Грея, для преобразования двоичного кода в код управления шкальными или матричными индикаторами, для преобразования двоично-десятичного кода в код управления сегментными индикаторами.

Так, микросхемы 74LS47, 74LS48 (рис. 2.8) представляют преобразователь двоично-десятичного кода в код семисегментного индикатора для семисегментных светодиодных дисплеев с общим анодом и общим катодом соответственно.

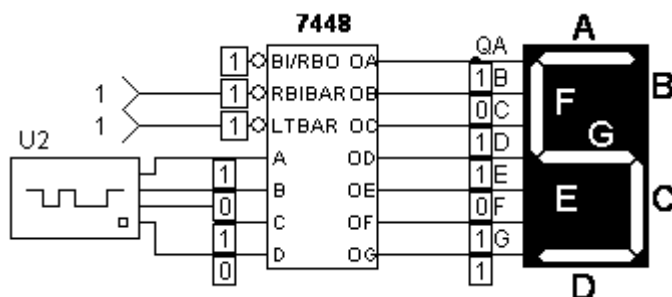


Рисунок 2.8 — Работа дешифратора индикатора на семисегментный светодиодный дисплей с общим катодом

2.3 Мультиплексоры и демультиплексоры

Мультиплексор (MS) — это функциональный узел, осуществляющий подключение (коммутацию) одного из нескольких входов данных к выходу. Номер выбранного входа соответствует коду, поданному на адресные входы мультиплексора. Условное изображение мультиплексора на четыре входа и возможный вариант его структурной схемы показаны на рис. 2.9, а, б.

При состоянии адресных входов 00 (код адреса $x_2=0$, $x_1=0$) $y=a$; при состоянии 01 ($x_2=0$, $x_1=1$) $y=b$; при состоянии 10 ($x_2=1$, $x_1=0$) $y=c$; в положении 11 ($x_2=1$, $x_1=1$) $y=d$. Функционирование мультиплексора описывается выражением:

$$y = a\bar{x}_2\bar{x}_1 + b\bar{x}_2x_1 + cx_2\bar{x}_1 + dx_2x_1$$

Вход E — разрешающий: при $E=1$ мультиплексор работает как обычно, при $E=0$ его выход находится в неактивном состоянии, мультиплексор заперт.

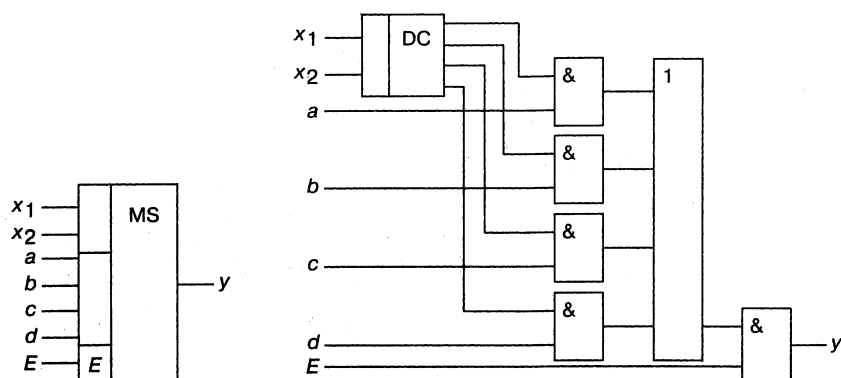


Рисунок 2.9 — Мультиплексор из 4 в 1: а — условное обозначение; б — внутренняя структура

Серийные ИМС выпускаются с числом адресных входов $n=2, 3$ и 4 при возможном числе 2^n коммутируемых входов (рис. 2.10). При необходимости коммутировать большее количество входов используют несколько мультиплексоров.

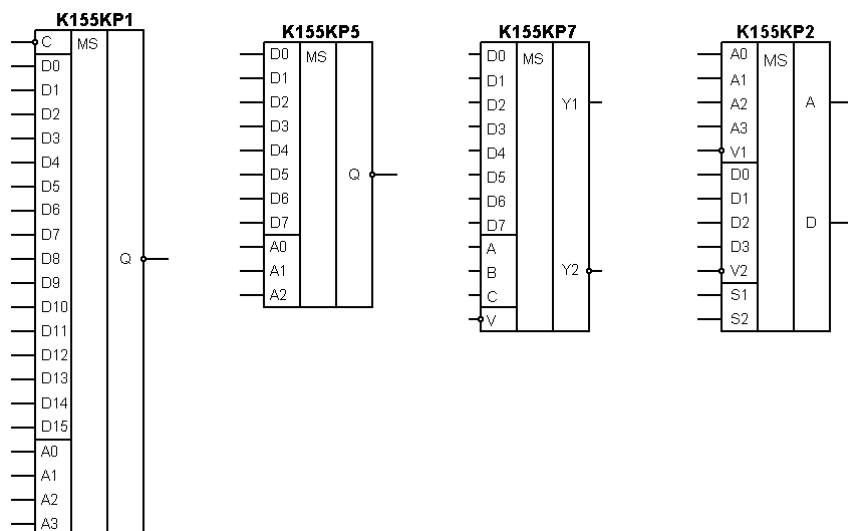


Рисунок 2.10 — Мультиплексоры разной размерности

Мультиплексоры находят широкое применение в устройствах отображения информации в различных микропроцессорных устройствах управления и ЭВМ.

Демультимплексоры выполняют функцию, обратную мультиплексорам, т. е. производят коммутацию одного входного сигнала на 2^n выходов, где n — число адресных входов x_i . Они могут осуществлять преобразование информации из последовательной формы (последовательно-параллельной) в параллельную. Демультимплексор имеет один информационный вход D и несколько выходов, причем вход подключается к выходу y_i имеющему адрес, заданный кодом на адресных входах.

В качестве примера на рис. 2.11, а дано условное графическое обозначение демультимплексора (DMS), имеющего четыре выхода, закон функционирования которого задан (табл. 2.2). Пользуясь таблицей 2.2, запишем переключательные

функции для выхода устройства:

$$y_0 = D\bar{x}_2\bar{x}_1; \quad y_1 = D\bar{x}_2x_1; \quad y_2 = Dx_2\bar{x}_1; \quad y_3 = Dx_2x_1.$$

Функциональная схема демультиплексора, реализующая эти выражения, приведена на рис. 2.11, б.

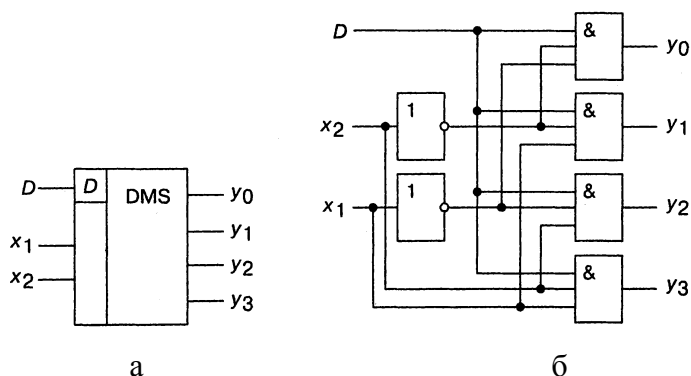


Рисунок 2.11 — Демультиплексор 1-4: а — условное обозначение; б — внутренняя структура

Таблица 2.2 — Таблица истинности демультиплексора 1-4

D	x_2	x_1	y_3	y_2	y_1	y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Функции демультиплексоров сходны с функциями дешифраторов. Дешифратор можно рассматривать как демультиплексор, у которого информационный вход поддерживает напряжение выходов в активном состоянии, а адресные входы выполняют роль входов дешифратора. Поэтому в обозначении как дешифраторов, так и демультиплексоров используются одинаковые буквы — ИД. Выпускают дешифраторы (демультиплексоры) К155ИД3, К531ИД7 и др.

Так, если в дешифраторе КР531ИД14А (рис. 2.13) вход разрешения Е использовать как вход данных, а информационные входы дешифратора считать за адресные входы демультиплексора, то получим демультиплексор 1→4, который работает аналогично демультиплексору рис. 2.11 (за небольшим исключением — у этих устройств разные пассивные уровни выходов).

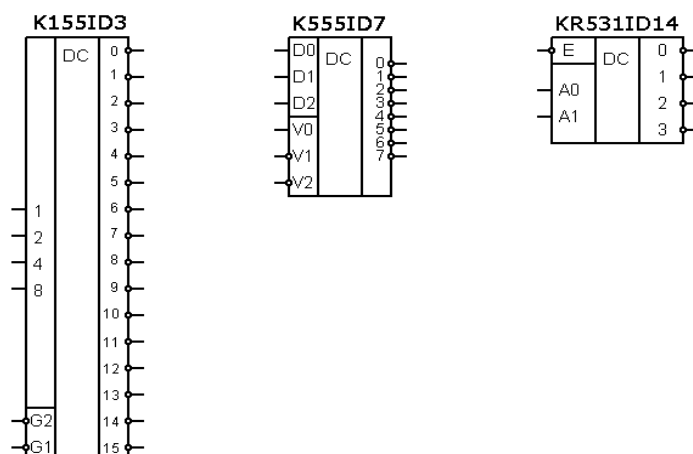


Рисунок 2.12 — Демультиплексоры

2.4 Компараторы кодов

Цифровой компаратор предназначен для определения равенства двоичных чисел. Операция поразрядного сравнения заключается в выработке признака равенства (равнозначности) или неравенства (неравнозначности) двух сравниваемых двоичных чисел. Два числа равны при равенстве цифр в одноименных разрядах: $a_i = b_i$, где a_i — цифра в i -ом разряде одного числа; b_i — цифра в i -ом разряде другого числа. Равенство $a_i = b_i$, имеет место при $a_i = 1, b_i = 1$ или при $a_i = 0, b_i = 0$. Поэтому логическая функция, выражающая это равенство, равна единице, если единице равно произведение этих цифр или произведение их инверсных значений, т. е.:

$$y_i = a_i b_i + \bar{a}_i \bar{b}_i,$$

а логическая функция, описывающая компаратор равенства, имеет вид:

$$y = (a_1 b_1 + \bar{a}_1 \bar{b}_1) (a_2 b_2 + \bar{a}_2 \bar{b}_2) \dots (a_n b_n + \bar{a}_n \bar{b}_n)$$

Для построения компаратора равенства только на элементах И-НЕ, запишем её в другой форме, воспользовавшись формулами де Моргана:

$$\bar{y} = \overline{(a_1 b_1 + \bar{a}_1 \bar{b}_1)} \overline{(a_2 b_2 + \bar{a}_2 \bar{b}_2)} \dots \overline{(a_n b_n + \bar{a}_n \bar{b}_n)}$$

Схема, реализующая это выражение, приведена на рис. 2.13.

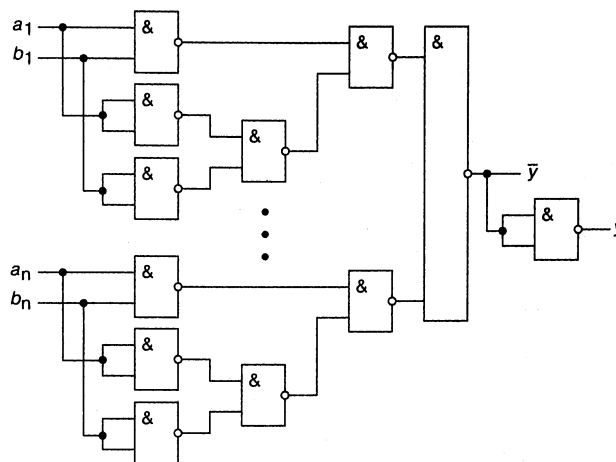


Рисунок 2.13 — Схема компаратора 2-х n -разрядных кодов на совпадение в базисе И-НЕ

Помимо совпадения кодов реальные ИМС цифровых компараторов могут определять и вид их неравенства $A > B$ или $A < B$. Т.е. реальные ИС компараторов имеют три выхода. Схема такого одноразрядного компаратора приведена на рис. 2.14.

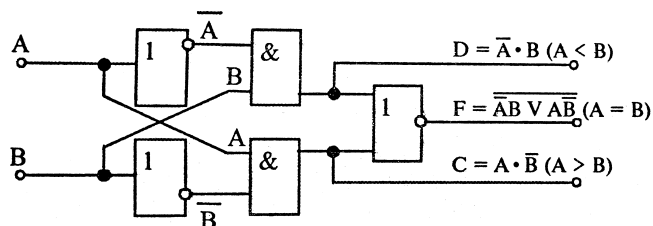


Рисунок 2.14 — Одноразрядный компаратор с определением равенства и вида неравенства

Цифровые компараторы выпускают, как правило, в виде самостоятельных микросхем. Так, например, имеется ТТЛШ микросхема К555СП1 (рис. 2.15) и КМОП К564ИП2, которые являются 4-хразрядными компараторами с определением типа неравенства и возможностью наращивания разрядности.

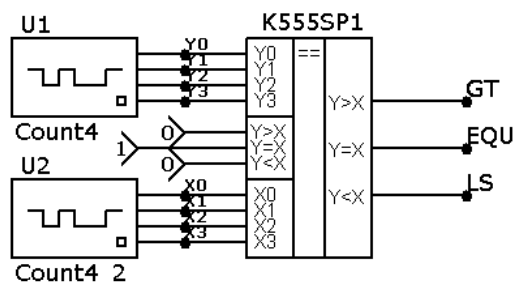


Рисунок 2.15 — Включение микросхемы компаратора 4-хразрядных кодов К555СП1

Если используется одна микросхема, то на ее вход $Y=X$ следует подать логическую 1.

2.5 Двоичные полусумматор и сумматор

Сумматор — это устройство, в котором выполняется арифметическая операция суммирования цифровых кодов двух двоичных чисел. Известно, что числа в любой позиционной системе счисления складываются поразрядно. Поэтому для сложения двух чисел нужно иметь типовые узлы, реализующие суммирование цифр одного разряда слагаемых с учетом возможного переноса единицы из соседнего младшего разряда. К таким узлам относят одноразрядные комбинационные полусумматоры и сумматоры.

Полусумматор предназначен для суммирования двух одноразрядных двоичных чисел. Он имеет два входа — a_i и b_i и два выхода — S_i и P_{i+1} , где S_i — выход суммы, а P_{i+1} является выходом переноса (табл. 2.3). Логические функции для $S_i = \bar{a}_i b_i + a_i \bar{b}_i$ и $P_{i+1} = a_i b_i$ и функциональную схему (рис. 2.16) этого узла легко построить на основе таблицы истинности (таблица 2.3), используя элементы И, ИЛИ и НЕ.

Таблица 2.3 — Таблица истинности одноразрядного полусумматора

a_i	b_i	P_{i+1}	S_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

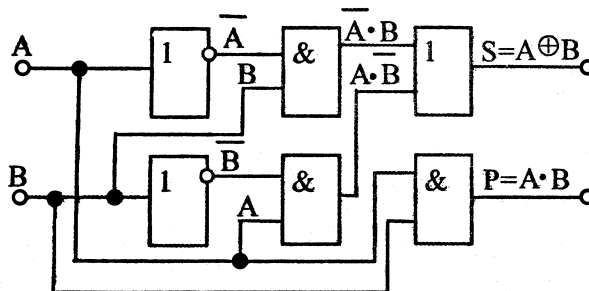


Рисунок 2.16 — Схема одноразрядного полусумматора

При построении сумматоров на интегральных микросхемах для обеспечения быстродействия и минимального количества однотипных логических элементов необходимо уменьшить число последовательно включенных элементов. Анализ показал, что более экономичной по количеству элементов и быстродействию является функциональная схема полусумматора, реализующая переключательную функцию:

$$S_i = a_i \bar{b}_i + \bar{a}_i b_i = \overline{a_i b_i + \bar{a}_i \bar{b}_i}, \quad P_{i+1} = a_i b_i$$

При суммировании двух многоразрядных чисел для каждого разряда (кроме младшего) необходимо использовать устройство, имеющее дополнительный вход переноса (из предыдущего более младшего разряда, вспомните правило сложения в столбик). Такое устройство (рис. 2.17) называют **полным сумматором** и его можно представить как объединение двух полусумматоров (P_{ex} — дополнительный вход переноса). Сумматор обозначают через SM.

В универсальных АЛУ, входящих в состав цифровых устройств, одноразрядные сумматоры проектируют из двух полусумматоров, объединенных в один выход S (рис. 2.17).

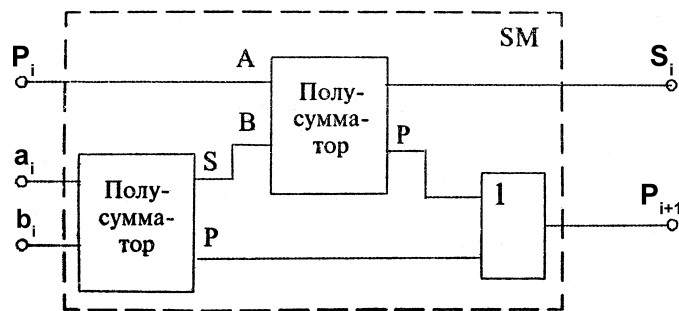


Рисунок 2.17 — Одноразрядный полный сумматор на основе двух полусумматоров

Многоразрядные комбинационные сумматоры последовательного или параллельного действия строятся на основе одноразрядных комбинационных сумматоров, реализующих функции для каждого разряда:

$$S_i = a_i \bar{b}_i \bar{p}_i + \bar{a}_i b_i \bar{p}_i + \bar{a}_i \bar{b}_i p_i + a_i b_i p_i, \quad P_{i+1} = a_i b_i \bar{p}_i + a_i \bar{b}_i p_i + \bar{a}_i b_i p_i + a_i b_i p_i$$

По этим функциям можно построить сумматор и на элементах И-НЕ или ИЛИ-НЕ.

Соединяя определенным образом полусумматоры и полные сумматоры друг с другом, получают устройство для выполнения сложения нескольких разрядов двоичных чисел. В качестве примера рассмотрим устройство для сложения двух трехразрядных двоичных чисел $A_2 A_1 A_0$ и $B_2 B_1 B_0$, где A_0 и B_0 — младшие разряды двоичных чисел (рис. 2.18).

На выходах $S_2 \dots S_0$ формируется код суммы чисел $A_2 A_1 A_0$ и $B_2 B_1 B_0$, а на выходе P_3 — сигнал переноса в следующую микросхему, так как при сложении двух трехразрядных двоичных чисел может получиться четырехразрядное число.

Следует отметить, что в рассмотренной структуре для суммирования в каждом разряде используется отдельный сумматор, но перенос из разряда в разряд осуществляется *последовательно*, что и определяет время выполнения

суммирования. Рассмотренный сумматор называется *параллельным сумматором с последовательным переносом* (К155ИМ3).

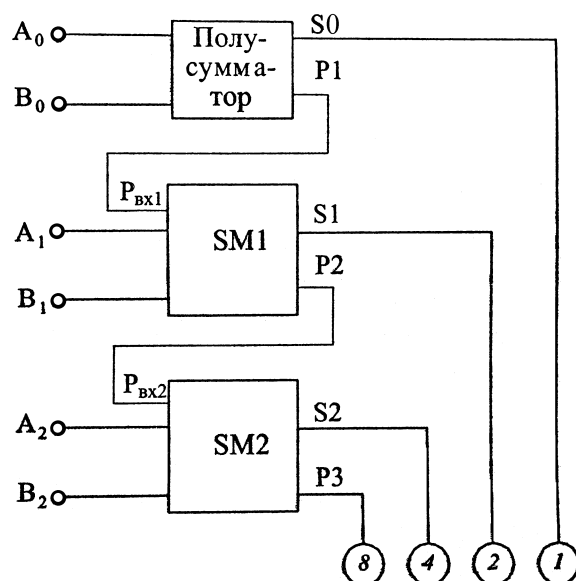


Рисунок 2.18 — Параллельный 3-хразрядный сумматор с последовательным переносом

Для повышения быстродействия сумматоров необходимо уменьшить время переноса, что достигается использованием вместо последовательного параллельного переноса. В этом случае в каждом разряде как сигнал суммы, так и сигнал переноса непосредственно формируются из входных переменных. Так микросхема К555ИМ6 (74LS283) представляет собой четырехразрядный сумматор с параллельным переносом.

В виде интегральных микросхем выпускаются одноразрядные, двухразрядные и четырехразрядные двоичные сумматоры (рис. 2.19).

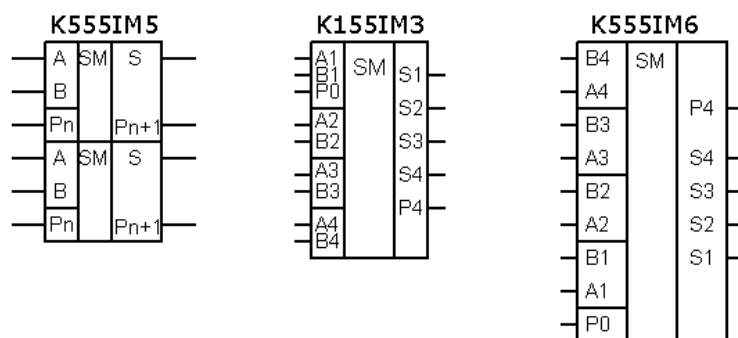


Рисунок 2.19 — Двоичные сумматоры

Рассмотренные сумматоры могут использоваться для вычитания двоичных чисел. В этом случае операция вычитания заменяется сложением уменьшаемого с вычитаемым, представленным в дополнительном коде, т. е. операцией:

$$A_{np} - B_{np} = A_{np} + B_{don} = A_{np} + B_{obr} + 1.$$

где $A = \{A_3 A_2 A_1 A_0\}$, и $B = \{B_3 B_2 B_1 B_0\}$ — многоразрядные двоичные числа, здесь для примера, четырехразрядные.

$$B_{obr} = \overline{B_3} \overline{B_2} \overline{B_1} \overline{B_0}, \quad -B = B_{don} = B_{obr} + 1$$

Рассмотрим 2 примера вычитания (10-5) и (5-10).

Двоичный эквивалент $+10_{10}=01010_2$, а $+5_{10}=00101_2$.

Числа в дополнительных кодах:

$-10_{10}=10_{\text{доп}}=10110\text{В}$, $-5_{10}=5_{\text{доп}}=11011\text{В}$

Для реализации описанного алгоритма вычитаемое нужно преобразовать в дополнительный код (см. выше) и сложить с уменьшаемым:

$$\begin{array}{r} 01010 \\ 11011 \\ \hline 00101 = 5_{10} \end{array} \qquad \begin{array}{r} 00101 \\ 10110 \\ \hline 11011 = -5_{10} \end{array}$$

Четыре младших разряда результата представляют собой результат в дополнительном коде, т. е. десятичное число 5 (в первом случае положительное, а во втором — отрицательное).

Следует подчеркнуть, что если $A > B$, т. е. результат — положительное число, то ответ формируется в прямом коде (знаковый разряд равен 0), при этом формируется 1 переноса в более старший разряд. При $A < B$ ответ формируется в дополнительном коде (знаковый разряд равен 1) и 1 переноса в более старший разряд не образуется.

Принципиально возможно построение функциональных схем сумматоров, работающих в любой системе счисления, отличающейся от двоичной.

Кроме двоичных, в микропроцессорной технике часто используются так называемые *двоично-десятичные коды*. Они отображают выраженные в виде последовательности двоичных разрядов десятичные числа. Очевидно, что для представления десятичных цифр необходим, как минимум 4-разрядный двоичный код. При этом из 16 возможных его комбинаций используется только 10. Это предполагает разработку большого числа различных двоично-десятичных кодов (см. табл. 1.5).

На практике большое распространение получил класс так называемых *взвешенных кодов*. В этих кодах каждому разряду двоичного числа присваивается вполне определенный весовой коэффициент (см. табл. 1.5). В табл. 2.4 приведено соответствие десятичных чисел и их двоичных и двоично-десятичных эквивалентов в коде 8-4-2-1. Весовые коэффициенты его двоичных разрядов соответственно равны 8, 4, 2, 1.

Таблица 2.4. Двоичные и двоично-десятичные коды чисел от 0 до 15

Двоичный код $x_3x_2x_1x_0$	Двоично-десятичный код		Десятичное число
0000	0000		0
0001	0001		1
0010	0010		2
0011	0011		3
0100	0100		4
0101	0101		5
0110	0110		6
0111	0111		7
1000	1000		8
1001	1001		9
1010	0001	0000	10
1011	0001	0001	11
1100	0001	0010	12
1101	0001	0011	13
1110	0001	0100	14
1111	0001	0101	15

Из приведенной таблицы следует, что 4-разрядные двоичные коды с 1010 по 1111 не имеют 4-разрядного двоично-десятичного эквивалента. Так, число 12 в двоично-десятичном коде представляется 8-разрядным упакованным кодом 00010010, а число 16 — кодом 00010110.

Описанная особенность двоично-десятичного кода предполагает использование для суммирования специальных логических схем. Смысл их построения состоит в том, что сначала двоично-десятичные коды суммируются как двоичные. Если результатом суммирования является несуществующий двоично-десятичный код, его необходимо уменьшить на 10_{10} , и дополнительно сформировать сигнал переноса. Уменьшение кода на 10_{10} может выполняться его суммированием с дополнительным кодом числа 10 (в двоичной системе $-10_{10} = 10110_2$, т.е. фактически прибавлением $6_{10} = 110_2$).

Необходимость выполнения такого суммирования согласно табл. 2.4 после минимизации выражается ФАЛ $F = x_3(x_2 + x_1)$.

Очевидно, что такое же суммирование необходимо выполнять и в случае, если в результате суммирования тетрад (BCD-кодов) получен сигнал переноса в старший разряд. С учетом сказанного, ФАЛ необходимости выполнения дополнительного суммирования имеет вид:

$$F = x_3(x_2 + x_1) + P \quad (2.1),$$

где x_3, x_2, x_1, x_0 — разряды полученного кода в порядке убывания весов, P — перенос, возникший при сложении текущих тетрад BCD-кодов.

Таким образом, для реализации операции сложения двух двоично-десятичных кодов необходимы два 4-хразрядных сумматора и логическая схема, обеспечивающая формирование выходного сигнала в соответствии с ФАЛ (2.1).

Пример реализации такого устройства показан на рис. 2.20. Четырехразрядный сумматор DD1 выполняет арифметическое сложение исходных двоично-десятичных кодов. Логическая схема на элементах DD2, DD3 и DD4 реализует ФАЛ (2.1), определяя необходимость дополнительного суммирования, выполняемого сумматором DD5.

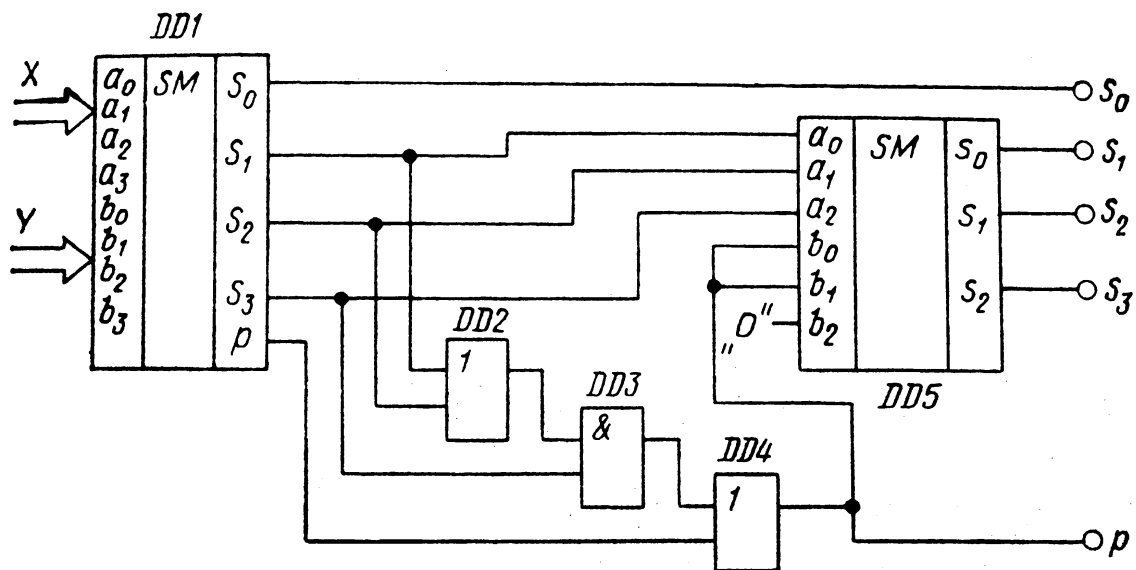


Рисунок 2.20 — Сумматор двух BCD-чисел

2.6 Арифметико-логические устройства

Арифметико-логическое устройство (АЛУ) — часть центрального процессора, которая в общем случае формирует функции двух входных переменных и порождает одну выходную переменную. Эти функции обычно состоят из простых арифметических операций, простых логических операций и операций сдвига.

Вне зависимости от того, насколько широк круг операций, реализуемых современными АЛУ, главными среди них остаются операции арифметического сложения и умножения, продолжительность выполнения которых указывается в качестве основных характеристик вычислительного устройства. Простейшие операции: арифметическое сложение (вычитание), логическое умножение, логическое сложение, сумма по модулю два, инверсия, сдвиг влево, сдвиг вправо, инкремент (положительное приращение), декремент (отрицательное приращение) выполняются в АЛУ с помощью только аппаратных средств (схем на логических элементах). Операция умножения (деления), как правило, выполняется программным способом (с применением последовательного исполнения нескольких операций сложения и сдвига, выполненных аппаратным способом). Однако, как будет показано далее, существуют и аппаратные умножители.

АЛУ классифицируют по различным признакам.

По **способу действия** над операндами АЛУ подразделяют на

- АЛУ параллельного действия
- АЛУ последовательного действия.

По способу **представления чисел** различают:

- АЛУ для чисел с фиксированной запятой
- АЛУ для чисел с плавающей запятой
- АЛУ для десятичных чисел.

По способу **организации работы**:

- синхронные АЛУ
- асинхронные АЛУ

По **характеру используемых элементов и узлов**:

- *Универсальные*, в которых операции для всех форм представления чисел выполняются одними и теми же схемами,
- *Функциональные*, в которых операции над операндами с фиксированной и плавающей запятой, с десятичными переменными и логическими переменными осуществляются в отдельных операционных блоках.

В настоящее время практически все АЛУ выполняются в интегральном исполнении и выпускаются различные серии соответствующих ИС. Проиллюстрируем функциональные возможности таких схем на примере ИС К531ИК2 (74LS381). Ее условное обозначение приведено на рис. 2.21.

Данная схема оперирует с двумя 4-разрядными входными кодами и способна выполнять 3 логических (поразрядные И, ИЛИ, исключающее ИЛИ) и 3

арифметических (сложение, вычитания (A-B), (B-A)) операций. Тип выполняемой операции определяется кодовой комбинацией на входах S2,S1,S0 (Табл. 2.5). Вход C0 представляет собой перенос из предыдущего разряда при сложении, инверсный заем из следующего разряда при вычитании. Выходы P, G представляют собой инверсные биты переноса для каскадирования и переполнения соответственно (схема Micro-CAP 381.cir).

Таблица 2.5 — Операции, выполняемые схемой АЛУ 74LS381

S2	S1	S0	Операция
0	0	1	B – A
0	1	0	A – B
0	1	1	A + B
1	0	0	A XOR B
1	0	1	A OR B
1	1	0	A AND B
1	1	1	PRESET

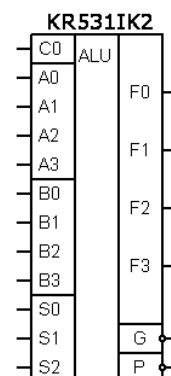


Рисунок 2.21

Логика построения **аппаратных умножителей** неразрывно связана с традиционным алгоритмом выполнения операции умножения, базирующемся на суммировании частных произведений разрядов сомножителей. Проиллюстрируем сказанное на примере умножения 2-разрядных двоичных кодов:

$$\begin{array}{r}
 \begin{array}{c} \mathbf{x} \\ \mathbf{+} \end{array}
 \begin{array}{r}
 \begin{array}{cc}
 & \begin{array}{cc} \mathbf{a_1} & \mathbf{a_0} \end{array} \\
 \begin{array}{cc} \mathbf{b_1} & \mathbf{b_0} \end{array} \\
 \hline
 \begin{array}{ccc}
 & \mathbf{b_0 a_1} & \mathbf{b_0 a_0} \\
 \mathbf{b_1 a_1} & \mathbf{b_1 a_0} & \\
 \hline
 \mathbf{M_3} & \mathbf{M_2} & \mathbf{M_1} & \mathbf{M_0}
 \end{array}
 \end{array}
 \end{array}$$

Структурная схема устройств показана на рис. 2.22. Частные произведения разрядов сомножителей формируются ЛЭ 2И — DD1...DD4. Суммируя эти произведения сумматорами DD5 и DD6 находят значение кода результата. Приведенная структура носит название матричного множительного блока.

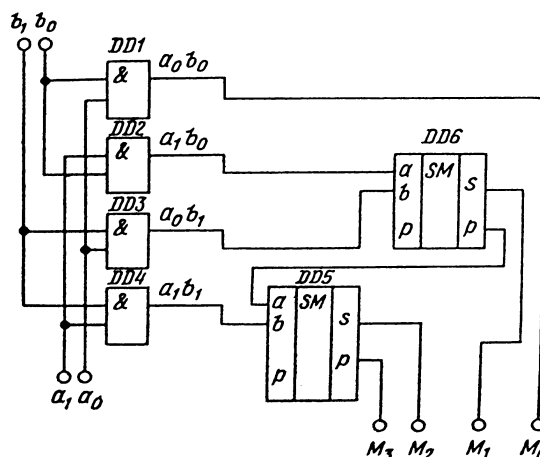


Рисунок 2.22 — Основная ячейка матричного множительного блока

2.7 Триггеры

Триггером называется устройство, имеющее два устойчивых состояния и сохраняющее любое из них сколь угодно долго после снятия внешнего воздействия, вызвавшего переход триггера из одного состояния в другое. Поэтому говорят, что триггер обладает памятью. Триггер можно представить в общем случае состоящим из ячейки памяти и устройства управления (порой весьма сложного), преобразующего входную информацию в комбинацию сигналов, под воздействием которых ячейка памяти принимает одно из двух устойчивых состояний.

По способу записи информации триггеры могут быть *асинхронными* и *синхронными*. Триггер называют асинхронным, если сам сигнал, несущий информацию, вызывает его переключение. В синхронных (тактируемых) триггерах информация записывается при одновременном воздействии информационного сигнала и синхронизирующего (разрешающего) импульса. Синхронизация может осуществляться импульсом (потенциалом) или перепадом потенциала (фронтом или срезом импульса). В первом случае (*статическое управление*) сигналы на информационных входах оказывают влияние на состояние триггера в течение всего времени наличия синхроимпульса. Во втором случае (*динамическое управление*) воздействие информационных сигналов проявляется только в моменты изменения потенциала на входе синхронизации, т.е. при переходе его от 0 к 1 (фронт) или от 1 к 0 (срез).

По функциональному признаку различают *RS-триггеры*, *D-триггеры*, *T-триггеры* и *JK-триггеры*, а также их комбинации.

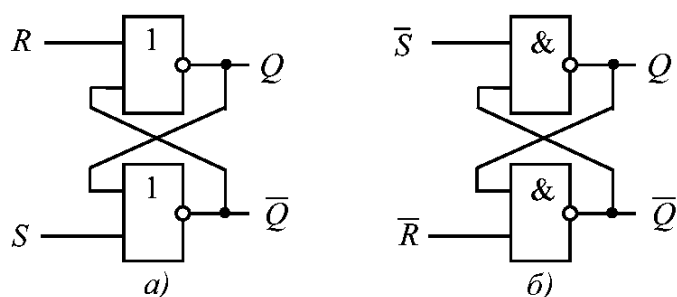


Рисунок 2.23 — Асинхронный RS-триггер с прямым (а) и инверсным (б) управлением

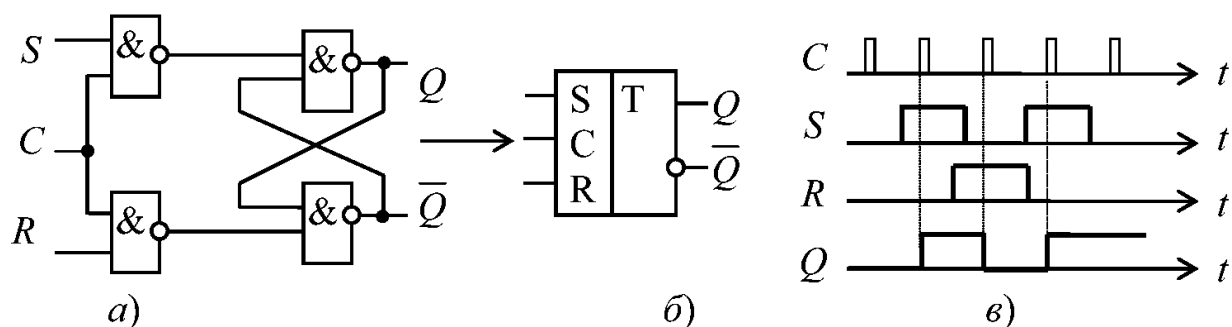


Рисунок 2.24 — Тактируемый RS-триггер: а) построение триггера на логических элементах И-НЕ; б) обозначение триггера на функциональных схемах; в) временные диаграммы, поясняющие работу триггера

D-триггер (от английского *Delay* – задержка) имеет один информационный (D – Data – данные) и один тактируемый (C – Clock – тактовая последовательность) вход. Такой триггер можно получить из RS-триггера, подав на R -вход инвертированный сигнал с S -входа (рис. 2.25, а). Условное обозначение D-триггера со статическим управлением показано на рис. 2.25, б. Из временных диаграмм, приведенных на рис. 2.25, в можно увидеть, что при $C=1$ триггер работает как повторитель ($Q=D$), а при переходе сигнала на входе C от логической единицы к логическому нулю триггер запирается и переходит в режим хранения (защелкивает информацию со входа D). В отличие от RS-триггера, D-триггер не имеет запрещенного состояния.

Статический синхронный триггер реагирует на входные сигналы в течение всего времени, пока тактовый сигнал C равен единице. Часто, однако, требуется триггер, в котором считываемая информация не передавалась бы непосредственно на выход, а появлялась там только тогда, когда все схемы уже заперты. Этим свойством обладают триггеры с динамическим управлением.

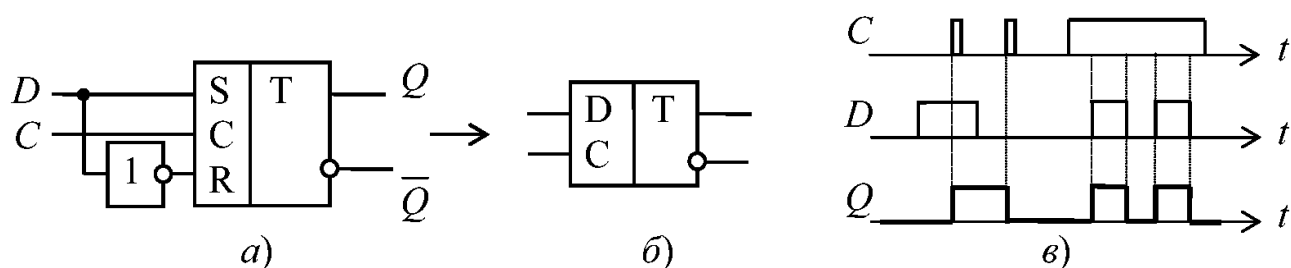


Рисунок 2.25 — Синхронный D-триггер со статическим управлением

D-триггер с динамическим управлением, тактируемый фронтом тактового импульса, можно выполнить по двухступенчатой схеме, показанной на рис. 2.26, а. При $C=0$ триггер первой ступени повторяет сигнал D , но триггер второй ступени защелкнут (находится в режиме хранения). При переходе к $C=1$ первый триггер защелкивает на своем выходе информацию со входа D , а второй повторяет ее на своем выходе. Таким образом, двухступенчатый триггер по фронту тактового импульса защелкивает на своем выходе уровень сигнала с входа D и сохраняет его до следующего фронта. Триггеры с динамическим управлением необходимы для построения счетчиков и регистров сдвига.

Из двух вариантов УГО динамического входа, приведенных на рис. 2.26, б и в, в дальнейшем будем использовать вариант б, меняя направление кривой черты для обозначения входа при тактировании по срезу импульса.

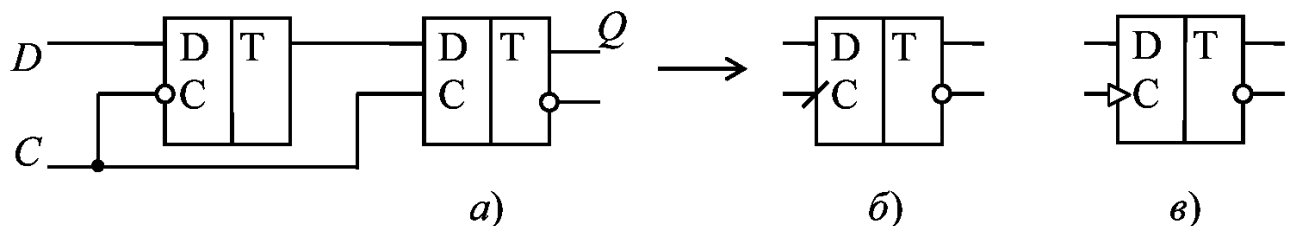


Рисунок 2.26 — Синхронный триггер задержки с динамическим управлением (а) и варианты его УГО (б, в)

Серийно производимые в семействах ТТЛ D-триггеры, переключающиеся по положительному фронту, устроены не по принципу «ведущий-ведомый», на основе которого действует схема, приведенные на рис. 2.26, а. Вместо этого в триггерах типа 74LS74 (555TM2) реализована схема на 6 вентилях (рис. 2.27), которая меньше по объему и быстрее и обычно в литературе называется шести-элементным триггером или схемой трех триггеров [1, 2]. Для уяснения работы такого триггера запишем систему функций алгебры логики ФАЛ, обозначив сигналы на выходах элементов D1–D4 X_1 – X_4 соответственно. При этом будем считать, что триггер разблокирован по входам предустановки, т.е. $\bar{R} = \bar{S} = 1$.

$$\begin{aligned} X_1 &= \overline{X_2 \cdot X_4} = \overline{X_2} + \overline{X_4} \\ X_2 &= \overline{X_1 \cdot C} = \overline{X_1} + \overline{C} \\ X_3 &= \overline{X_2 \cdot X_4 \cdot C} = \overline{X_2} + \overline{X_4} + \overline{C} \\ X_4 &= \overline{X_3 \cdot D} = \overline{X_3} + \overline{D} \end{aligned} \quad (2.2)$$

Решим систему уравнений (2.2) относительно входных сигналов выходного асинхронного триггера X_2, X_3 .

$$\begin{aligned} X_2 &= \overline{X_1} + \overline{C} = \overline{\overline{X_2 \cdot X_4}} + \overline{C} = X_2 \cdot (\overline{X_3} + \overline{D}) + \overline{C} \quad (2.3) \\ X_3 &= \overline{X_2} + \overline{X_4} + \overline{C} = \overline{X_2} + \overline{C} + \overline{\overline{X_3 \cdot D}} = \overline{X_2} + \overline{C} + X_3 \cdot D \end{aligned}$$

Полученные выражения (2.3) содержат сигналы X_2 и X_3 как в правой, так и в левой частях. Это означает, что, подставляя в правую часть уравнений значения сигналов в момент времени t_n , в левой части получим их значения в момент времени t_{n+1} .

Допустим, что в исходном состоянии $C=0$. Тогда согласно (2.3) $X_2=X_3=1$, независимо от значения сигнала D . На входах выходного асинхронного RS-триггера D5-D6 будут действовать пассивные логические уровни (единицы) и он будет находиться в режиме хранения. При поступлении синхронизирующего сигнала $C=1$ из (2.3) получаем:

$$\begin{aligned} X_2 &= X_2 \cdot (\overline{X_3} + \overline{D}) + \overline{C} = (0 + \overline{D}) + 0 = \overline{D} \\ X_3 &= \overline{X_2} + \overline{C} + X_3 \cdot D = 0 + 0 + 1 \cdot D = D \end{aligned}$$

Данное состояние будет устойчивым, что легко проверить, подставив полученные значения X_2 и X_3 в (2.3). Таким образом, при появлении синхронизирующего сигнала $C=1$, на выходах выходного RS триггера будут присутствовать сигналы, определенные информационным уровнем на входе D . При $D=1$ получается $Q=1$, при $D=0$ — $Q=0$.

Пусть $D=1$, тогда сразу после прихода синхроимпульса имеем $X_2=0, X_3=1$. Подставив значения в (2.3), и, считая, что уровень на входе D может быть любым, находим:

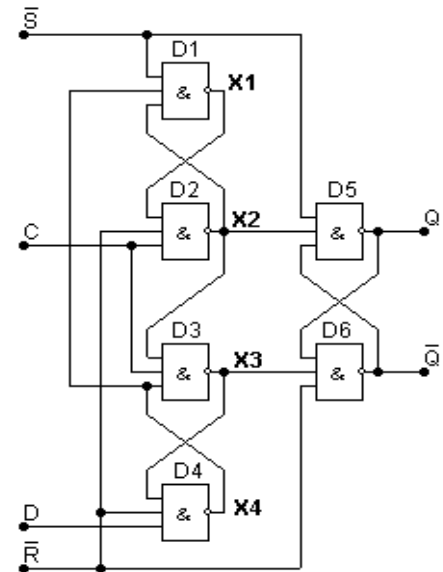


Рисунок 2.27 — Шестиэлементный D-триггер (555TM2)

$$X_2 = 0 \cdot (0 + \overline{D}) + 0 = 0$$

$$X_3 = 1 + 0 + 1 \cdot D = 1$$

Таким образом, после прихода синхроимпульса $C=1$ состояние выходного асинхронного RS-триггера становится нечувствительным к изменениям сигнала на информационном входе D .

Для следующего переключения триггера необходимо подать $C=0$, при этом вентили D_2 , D_3 устанавливают сигналы на своих выходах $X_2=X_3=1$, переводя выходной RS-триггер в режим хранения ранее записанной информации. Фронтом следующего импульса синхронизации триггер будет переведен в новое состояние, определяемое значением на информационном входе D .

Рассмотренный триггер является *триггером с прямым динамическим управлением* (принимает информацию по фронту синхросигнала). На практике в него вводят дополнительно асинхронные входы предустановки \overline{R} и \overline{S} , которые имеют приоритет над информационным входом D .

T-триггер (от английского *Toggle* – опрокидываться, кувыраться) имеет только тактовый вход T и меняет свое состояние на противоположное по фронту или срезу каждого нового тактового импульса (рис. 2.28). На рисунке показано УГО T-триггера и как можно выполнить T-триггер на базе RS- или D-триггеров с динамическим управлением. Каждый раз по фронту сигнала T изменяется уровень напряжения на выходе Q . Частота изменения потенциала на выходе T-триггера в два раза меньше частоты импульсов на его тактовом входе. Это свойство T-триггера используется при построении двоичных счетчиков, а T-триггер называют также счетным триггером.

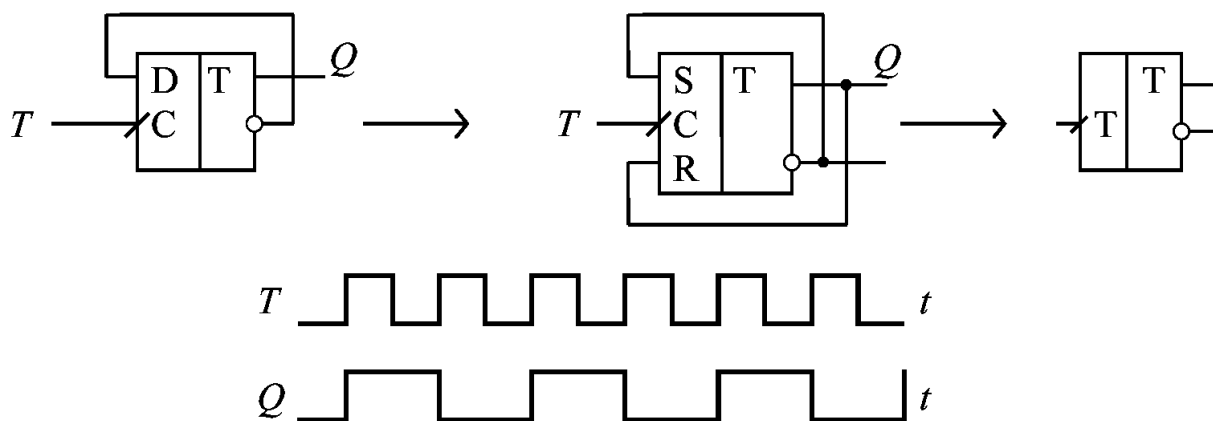


Рисунок 2.28 — Счетный триггер, синхронизируемый фронтом и временные диаграммы его работы

JK-триггер выполняет наиболее универсальные функции (J – *Jerk* – резкое движение, толчок; K – *Kill* – ликвидировать). Он строится на базе RS-триггера (рис. 2.29), но, в отличие от него, в JK-триггере устранено запрещенное состояние при $J=K=1$. При совпадении логических единиц на информационных входах J и K он работает как счетный (режим переключения), т.е. меняет свое состояние на противоположное при каждом новом такте. Логическая 1 на входе J устанавливает триггер в состояние единицы (режим записи 1, установка), логическая 1 на входе K переводит триггер в состояние логического нуля (режим

записи 0, сброс) при наличии тактирования. При наличии логических нулей на входах J и K тактовый импульс не меняет состояние триггера (режим хранения). Во избежание режима генерации здесь требуется применять RS-триггер двухступенчатого типа или с динамическим управлением (рис. 2.30). Все работоспособные триггеры JK-типа имеют динамическое управление (переключаются по фронту или срезу синхроимпульса).

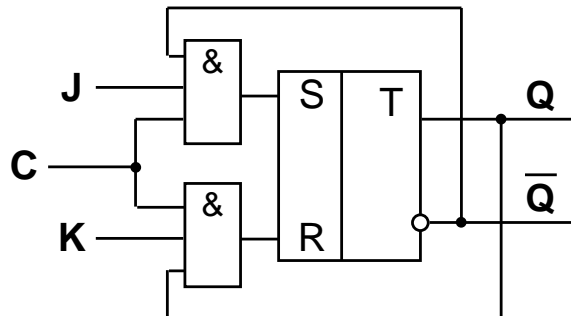


Рисунок 2.29 — Схема информационных связей JK-триггера

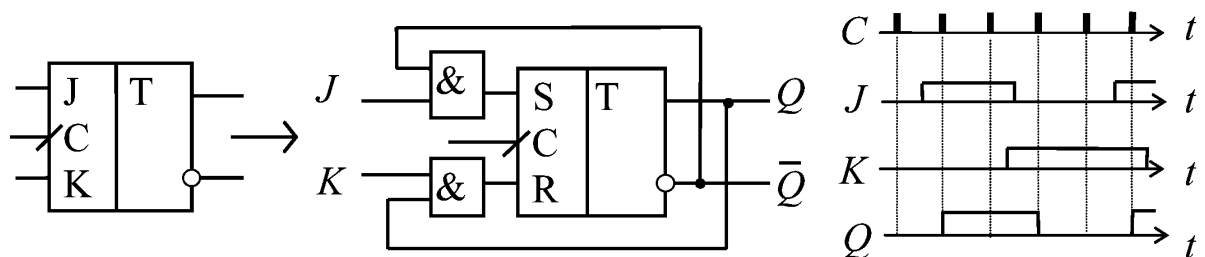


Рисунок 2.30 — JK-триггер с динамическим синхровходом (тактируемый фронтом СИ)

Синтез JK-триггера (Master-slave), запоминающего состояние J и K входов при высоком уровне синхроимпульса и меняющий состояние на выходе по отрицательному перепаду синхроимпульса (переход из высокого в низкое состояние).

$$S_1 = \overline{Q_2} \cdot J \cdot C \quad R_1 = Q_2 \cdot K \cdot C$$

$$S_2 = Q_1 \cdot \overline{C} \quad R_2 = \overline{Q_1} \cdot \overline{C}$$

Схема синтезированного двухтактного триггера представлена на рис. 2.32.

Предназначен для использования при управлении короткими импульсами (имеется в виду длительность единичного состояния) со стороны тактового входа. В противном случае при изменении состояния информационных входов во время высокого уровня синхроимпульса, в первом (Master) RS-триггере может быть защелкнута промежуточная информация, а не непосредственно предшествующая спаду синхроимпульса (см. граф переходов синтезированного триггера). Данное явление носит название *захвата нуля* и *захвата единицы* [1, 2] и иллюстрируется рис. 2.33.

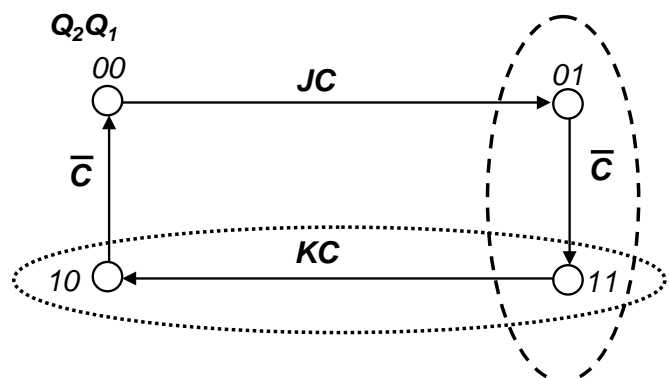


Рисунок 2.31 — Граф переходов асинхронного автомата (JK-триггера MS)

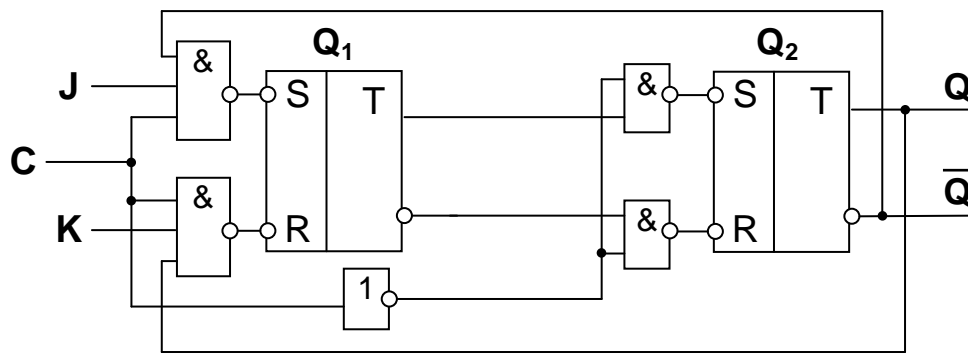


Рисунок 2.32 — Синхронный двухтактный JK-триггер, меняющий состояние выхода по срезу (заднему фронту) синхроимпульса, синтезированный в соответствии с рис. 2.31

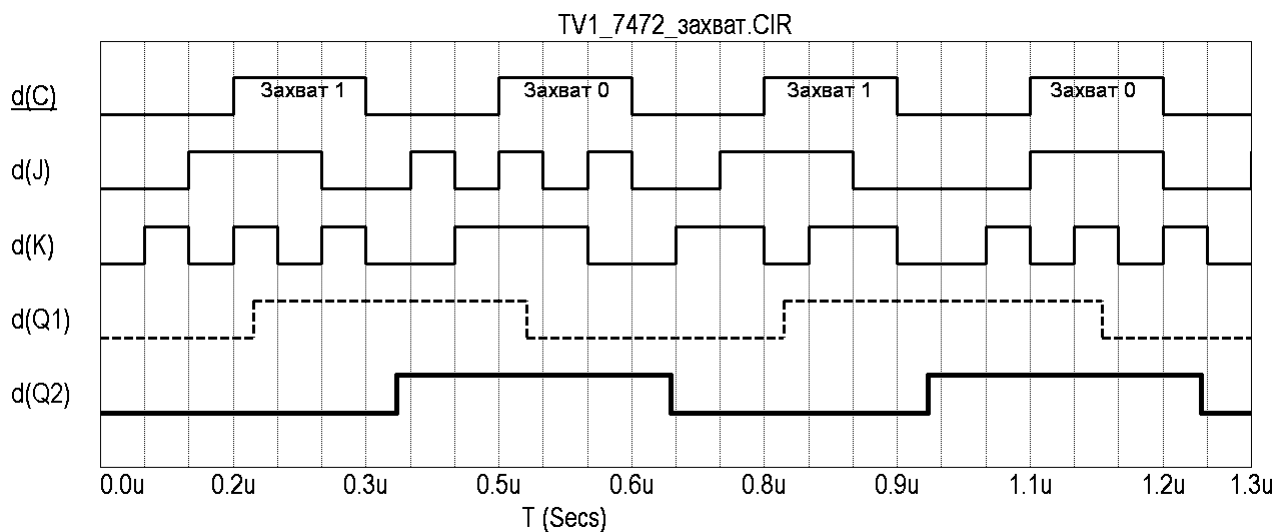


Рисунок 2.33 — Явления захвата нулей и единиц в двухступенчатом JK-триггере

Для того чтобы информация менялась на выходе в соответствии с сигналами на информационных входах по заднему (переднему) фронту (а точнее в течение короткого временного интервала, следующего за фронтом) надо изменить граф переходов, а, следовательно и принципиальную схему JK-триггера (рис. 2.34, есть задания в л.р. 8 сем).

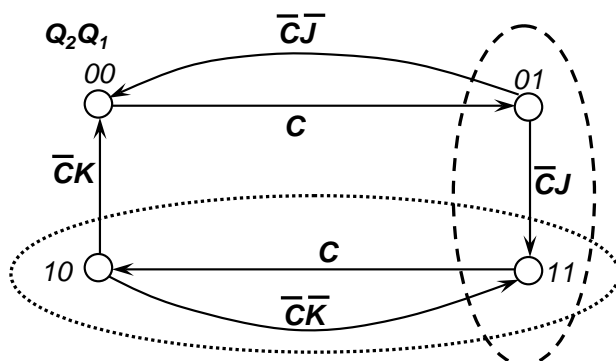


Рисунок 2.34 — Граф переходов асинхронного автомата (JK-триггера срабатывающего по заднему фронту СИ)

JK-триггер, устанавливающийся в соответствии с управляющими сигналами на J и K входах по отрицательному перепаду синхроимпульса (переход из низкого в высокое состояние).

$$S_1 = \overline{Q_2} \cdot C \vee Q_2 \cdot \overline{C} \cdot \overline{K}$$

$$R_1 = Q_2 \cdot C \vee \overline{Q_2} \cdot \overline{C} \cdot \overline{J}$$

$$S_2 = Q_1 \cdot \overline{C} \cdot J \quad R_2 = \overline{Q_1} \cdot \overline{C} \cdot K$$

Синтез принципиальной схемы в соответствии с графом переходов рис.

2.34 после некоторых упрощений приводит к схеме 6-элементного JK-триггера. Структура 6-элементного JK-триггера, срабатывающего по фронту синхроим-

пульса и имеющего инверсный информационный вход K (ТВ15, XXXX109A) приведена ниже на рисунке 2.38.

Схема реального двухступенчатого JK-триггера со входами предустановки и логики на входе 555ТВ1 (7472) приведена на рис. 2.35. За исключением наличия установочных входов, и логики на информационных входах J и K , ее работа аналогична схеме рис. 2.32 со свойственными недостатками, состоящими в вышеописанных явлениях захвата нулей и единиц.

Данный триггер из-за рассмотренного недостатка считается устаревшим, новые разновидности свободны от захватов и по существу изменяют состояние в соответствии с сигналами на J и K информационных входах по срезу (фронту) синхросигнала.

Одной из таких структур является схема триггера, использующая внутренние задержки составляющих его логических элементов (рис. 2.36).

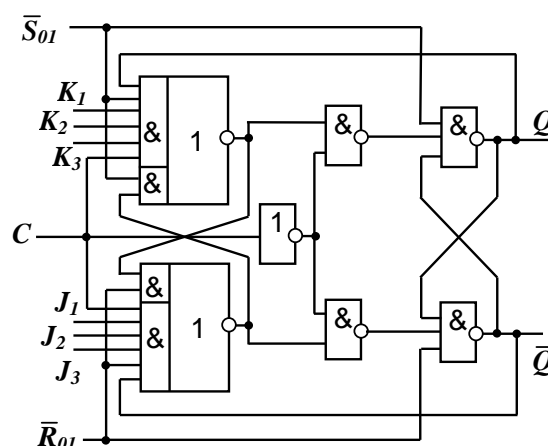


Рисунок 2.35 — Схема триггера 555ТВ1 (7472)

Функционирование одноступенчатого триггера с внутренней задержкой (рис. 2.36, а) можно рассмотреть с помощью временных диаграмм (рис. 2.36, б).

Рассмотрим только счетный режим, т. к. процесс переключения по информационным синхронным входам (J, K) аналогичен для любых их комбинаций. Работа же входов сброса и установки рассмотрена ранее. Так как в счетном режиме $J=K=1$, соответствующие входы не влияют на работу элементов D1 и D2. Исходное состояние триггера примем нулевым. Поскольку схема симметрична, достаточно рассмотреть только один процесс переключения (из нуля в единицу, см. рис. 2.36, б).

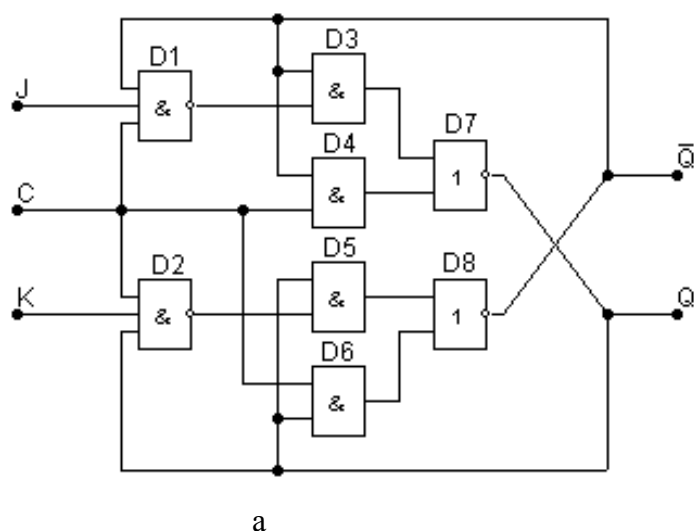


Рисунок 2.36 — JK-триггер с внутренними задержками

Работоспособность триггера обеспечивается только при условии $t_{3,1,2} > 2 \cdot t_{3,4,5,6} + t_{3,7,8}$ (задержки вентилях D1 и D2 превышают суммарную за-

держку вентилях «И» и «ИЛИ-НЕ»), которое и отражено на временных диаграммах. Как видно из диаграмм, триггер переключается по отрицательному перепаду тактирующего сигнала. По такой схеме с добавлением входов (входа) предварительной установки строятся триггеры TB6 (74LS107), TB9 (74LS112), TB10 (74LS113), TB11 (74LS114). На рис. 2.37 показана внутренняя структура триггера TB9.

Также распространены шестиэлементные JK-триггеры с управлением фронтом. Принцип работы шестиэлементных триггеров рассмотрен ранее, приведем лишь внутреннюю структуру триггера TB15 (74LS109), использующего этот вариант построения (рис. 2.38).

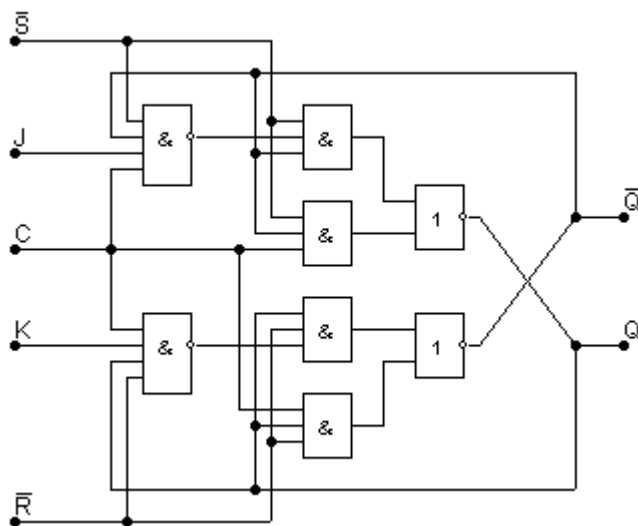


Рисунок 2.37 — JK-триггер TB9

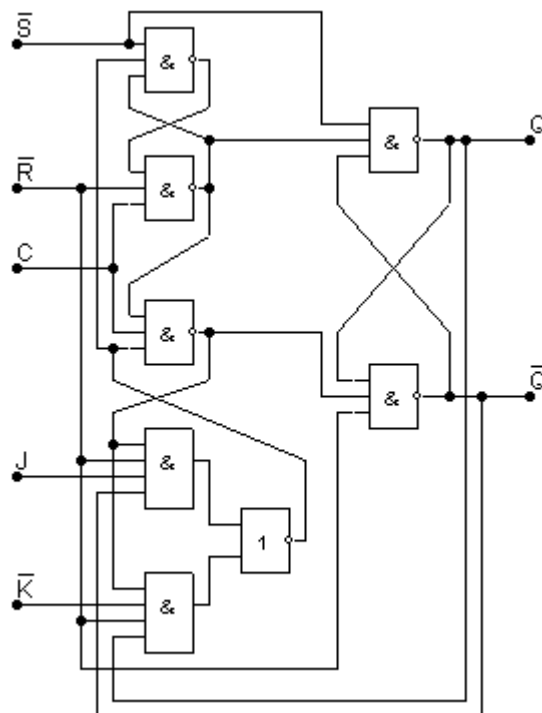
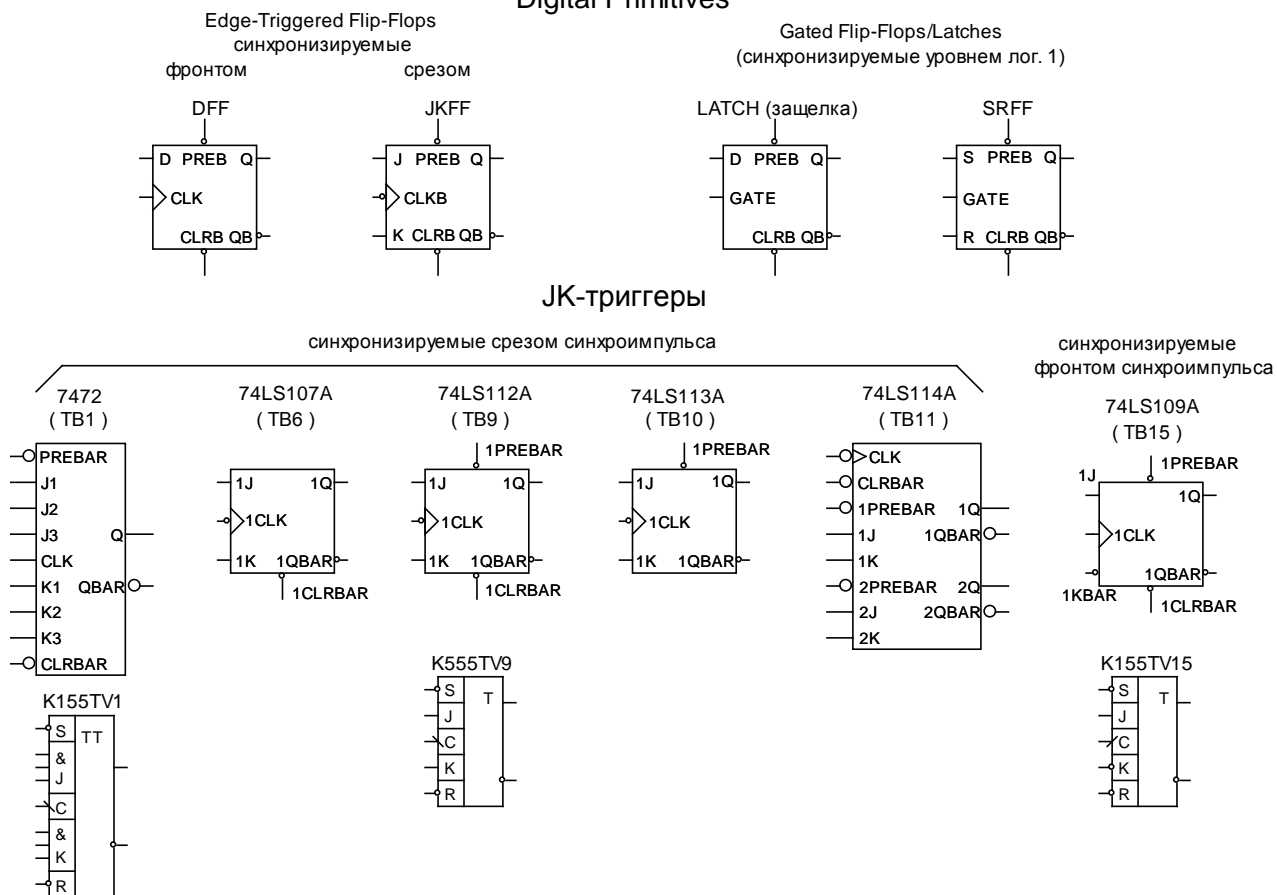


Рисунок 2.38 — JK-триггер TB15

Сводная карта всех триггеров-примитивов и ТТЛ-триггеров (которые также можно найти в системе моделирования Micro-CAP 9, 10) приведена на рис. 2.39. Триггеры серий КМОП и ЭСЛ представлены на рис. 2.40.

Digital Primitives



RS и D-триггеры

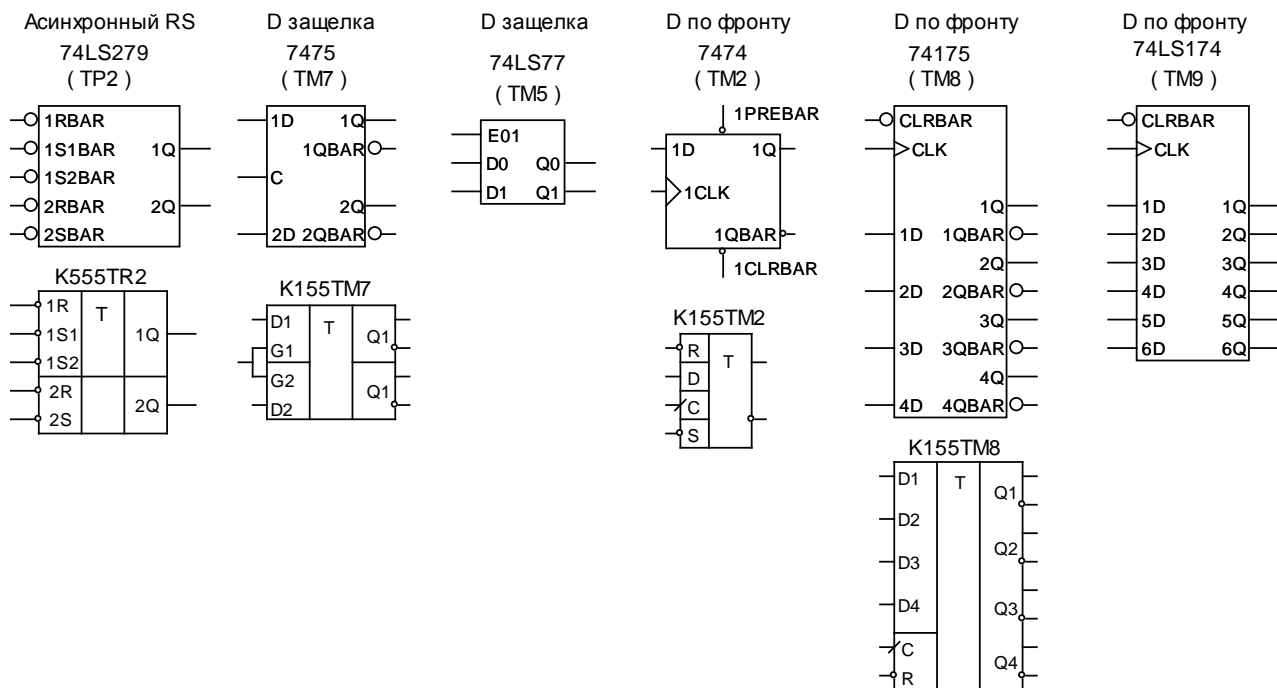
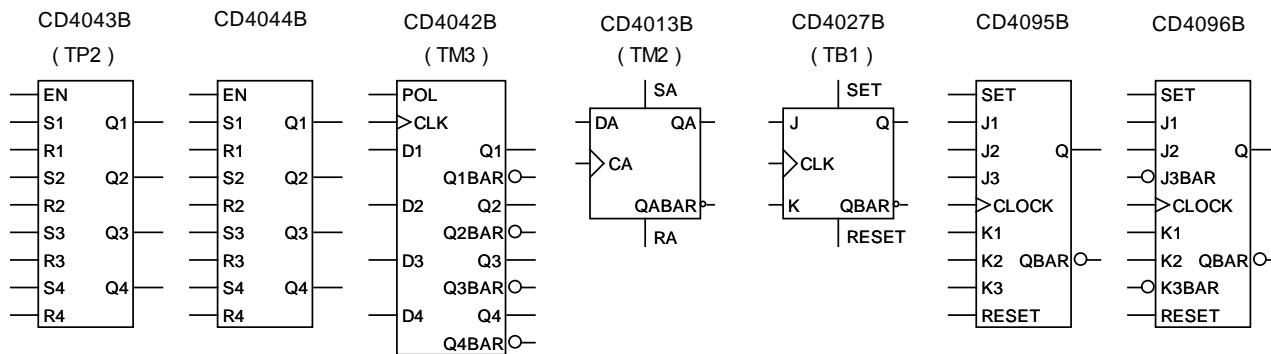


Рисунок 2.39 — Разновидности триггеров ТТЛ, имеющиеся в программе Micro-CAP

Триггера КМОП



Триггера ЭСЛ

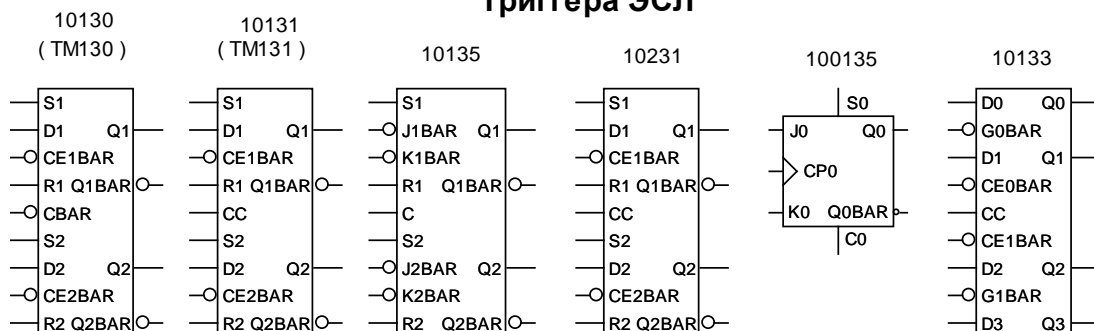
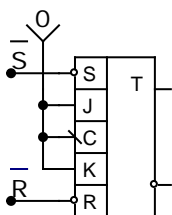


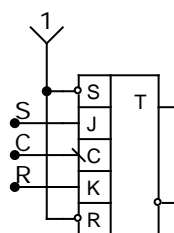
Рисунок 2.40 — Разновидности триггеров КМОП и ЭСЛ

2.7.1 Взаимные преобразования триггеров

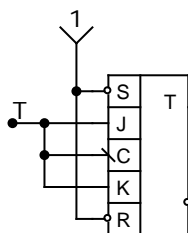
Асинхронный RS-триггер



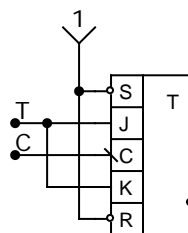
Синхронный RS-триггер



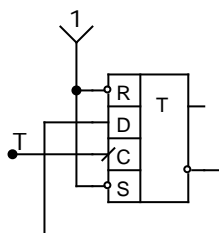
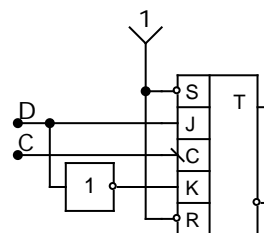
Асинхронный T-триггер



Синхронный T-триггер



Синхронный D-триггер



2.8 Счетчики

Счетчиком называется последовательностное устройство, предназначенное для счета входных импульсов и фиксации их числа в двоичном коде.

Понятие «счетчик» является очень широким. К счетчикам относят автоматы, которые под действием входных импульсов переходят из одного состояния в другое, фиксируя тем самым число поступивших на их вход импульсов в том или ином коде.

Специфичной для счетчиков операцией является изменение их содержимого на единицу (возможно и условную). Прибавление такой единицы соответствует операции инкремента, вычитание — операции декремента. Обычно счетчиками выполняются также и другие операции — сброс, установка, параллельная загрузка и пр.

Счетчики строятся на основе N однотипных связанных между собой разрядных схем, каждая из которых в общем случае состоит из триггера и некоторой комбинационной схемы, предназначенной для формирования сигналов управления триггером.

В цифровых схемах счетчики могут выполнять следующие микрооперации над кодовыми словами:

- установка в исходное состояние (запись нулевого кода);
- запись входной информации в параллельной форме;
- хранение информации;
- выдача хранимой информации в параллельной форме;
- инкремент — увеличение хранящегося кодового слова на единицу;
- декремент — уменьшение хранящегося кодового слова на единицу.

2.8.1 Основные параметры и классификация счетчиков

Основным статическим параметром счетчика является модуль счета M , определяющий число возможных состояний счетчика. После поступления на счетчик M входных сигналов начинается новый цикл, повторяющий предыдущий.

Существуют следующие режимы работы счетчика:

- регистрация числа поступивших на счетчик сигналов;
- деление частоты.

В первом режиме результат — содержимое счетчика, во втором режиме выходными сигналами являются импульсы переполнения счетчика.

Основным динамическим параметром для первого режима, определяющим быстродействие счетчика, является *время установления выходного кода* $t_{ycm\ K}$ — временной интервал между моментом подачи входного сигнала и моментом установления нового кода на выходе. Иногда быстродействие счетчика характеризуется *максимальной частотой входных сигналов* f_{max} .

Счетчики могут классифицироваться по многим параметрам. Рассмотрим основные из них.

По значению **модуля счета** счетчики подразделяют на:

- двоичные, модуль счета которых равен целой степени числа 2 ($M=2^n$);
- двоично-кодированные, модуль счета которых может принимать любое, не равное целой степени числа 2, значение.

По способу кодирования внутренних состояний различают:

- двоичные счетчики
- счетчики Джонсона
- счетчики с кодом "1 из N" и др.

По **направлению счета** счетчики подразделяют на:

- суммирующие, выполняющие микрооперацию инкремента над хранящимся кодовым словом;
- вычитающие, выполняющие микрооперацию декремента над хранящимся кодовым словом;
- реверсивные, выполняющие в зависимости от значения управляющего сигнала над хранящимся кодовым словом либо микрооперацию декремента, либо инкремента.

Счетчики строятся из разрядных схем, имеющих межразрядные связи. По **способу организации межразрядных связей** счетчики делятся на:

- счетчики с последовательным переносом, в которых переключение триггеров разрядных схем осуществляется последовательно один за другим;
- счетчики с параллельным переносом, в которых переключение всех триггеров разрядных схем осуществляется одновременно по сигналу синхронизации;
- счетчики с комбинированным последовательно-параллельным переносом, при котором используются различные комбинации способов переноса.

По **принадлежности к тому или иному классу автоматов** говорят о синхронных или асинхронных счетчиках (более подробную классификацию по этому признаку не затрагиваем, учитывая реальный состав микросхем счетчиков).

2.8.2 Двоичные счетчики

Как и любой автомат, счетчик можно строить на триггерах любого типа, однако удобнее всего использовать для этого триггеры типа Т (счетные) и JK, имеющие при $J=K=1$ счетный режим (режим Т-триггера).

Состояние счетчика читается по выходам разрядных схем как слово $Q_{n-1}Q_{n-2}\dots Q_0$, входные сигналы поступают на младший разряд счетчика.

Двоичный счетчик — счетчик, имеющий модуль $M=2^n$, где n — целое число, и естественную последовательность кодов состояний (его состояния отображаются последовательностью двоичных чисел, десятичными эквивалентами которых будут числа 0, 1, 2, 3, $M-1$; см. табл. 1.1).

Для определения структуры двоичного счетчика рассмотрим последовательность двоичных чисел, ограничившись для простоты 3-разрядным кодом (табл. 2.6).

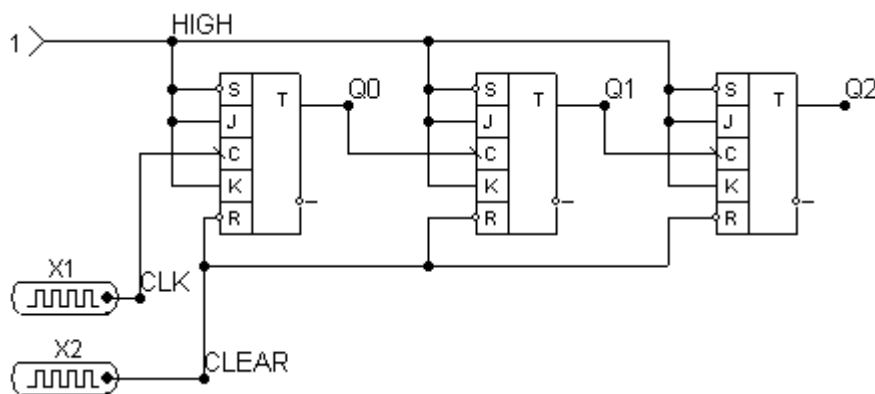
Т а б л и ц а 2.6 — Таблица 3-разрядных двоичных чисел

C	Q2	Q1	Q0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

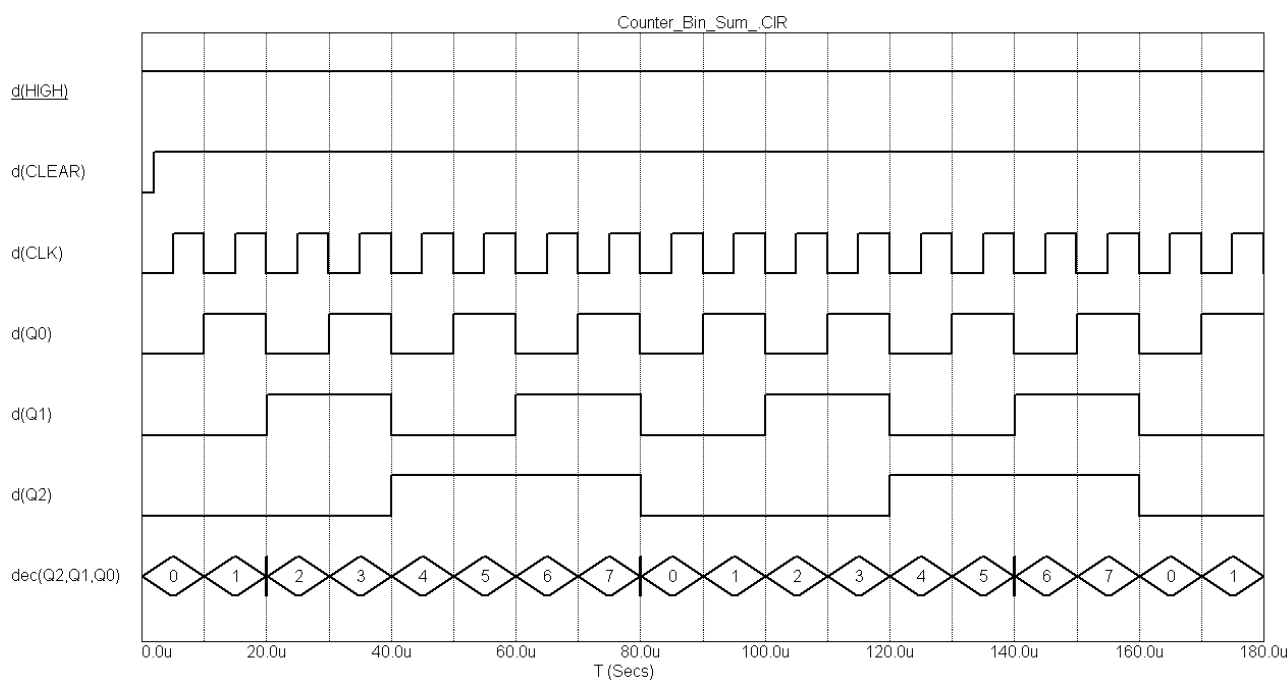
Схему двоичного счетчика можно получить с помощью формального синтеза, однако более наглядным и быстрым является эвристический метод. Таблица истинности двоичного счетчика — последовательность двоичных чисел от нуля до $M-1$. Наблюдение за разрядами чисел, составляющих таблицу (табл. 2.6), приводит к пониманию структурной схемы двоичного счетчика. Состояния младшего разряда Q_0 при просмотре по соответствующему столбцу таблицы показывают чередование нулей и единиц вида 01010101. В следующем разряде наблюдается последовательность пар нулей и единиц вида 00110011... . В третьем разряде образуется последовательность из четверок нулей и единиц 00001111... и т.д. Из этого наблюдения видно, что следующий по старшинству разряд переключается с частотой, в два раза меньшей, чем данный.

Так Q_0 , соответствующее младшему разряду двоичного числа, изменяет свое значение с приходом каждого импульса синхронизации; Q_1 — с приходом каждого второго импульса синхронизации, а Q_2 — с приходом каждого четвертого импульса. Данный алгоритм можно легко реализовать, используя асинхронные Т-триггеры, причем синхронизацию каждого последующего триггера осуществляют выходным сигналом предыдущего, а переключение первого триггера, формирующего значение Q_0 — непосредственно последовательностью синхроимпульсов (рис. 2.41, а). Так как в счетчике значения Q ассоциируются с выходными сигналами соответствующих триггеров, то для получения счетчика с модулем счета $M=8$ необходимо как минимум три триггера. Временные диаграммы, поясняющие такой алгоритм работы, приведены на рис. 2.41, б.

Представление счетчика цепочкой Т-триггеров справедливо как для суммирующего, так и для вычитающего вариантов, поскольку закономерность по соотношению частот переключения разрядов сохраняется как при просмотре таблицы сверху вниз (прямой счет), так и снизу вверх (обратный счет). Различия при этом состоят в направлении переключения предыдущего разряда, вызывающего переключение следующего. При прямом счете следующий разряд переключается при переходе предыдущего в направлении от 1–0, а при обратном — при переключении 0–1 (см. табл. 2.6). Следовательно, различие между вариантами заключается в разном подключении входов последующих триггеров к выходам предыдущих.



a



6

Рисунок 2.41 — Суммирующий счетчик с последовательным переносом: а — схема; б — временные диаграммы

Так для получения суммирующего счетчика, последовательно включенные асинхронные Т-триггеры должны быть снабжены инверсными динамическими входами (рис. 2.41). Для получения вычитающего счетчика на основе цепочки триггеров с инверсными динамическими входами, необходимо на вход последующего триггера подавать сигнал с инверсного выхода предыдущего (рис. 2.42).

Если исходные асинхронные Т-триггеры снабжены *прямыми динамическими входами*, то счетчик, собранный по схеме рис. 2.41, а превращается в вычитающий и выполняет микрооперацию декремента. Счетчик же, собранный по схеме рис. 2.42, а, превращается в суммирующий и выполняет микрооперацию инкремента.

Результаты приведенных выше рассуждений можно свести в таблицу (см. табл. 2.7).

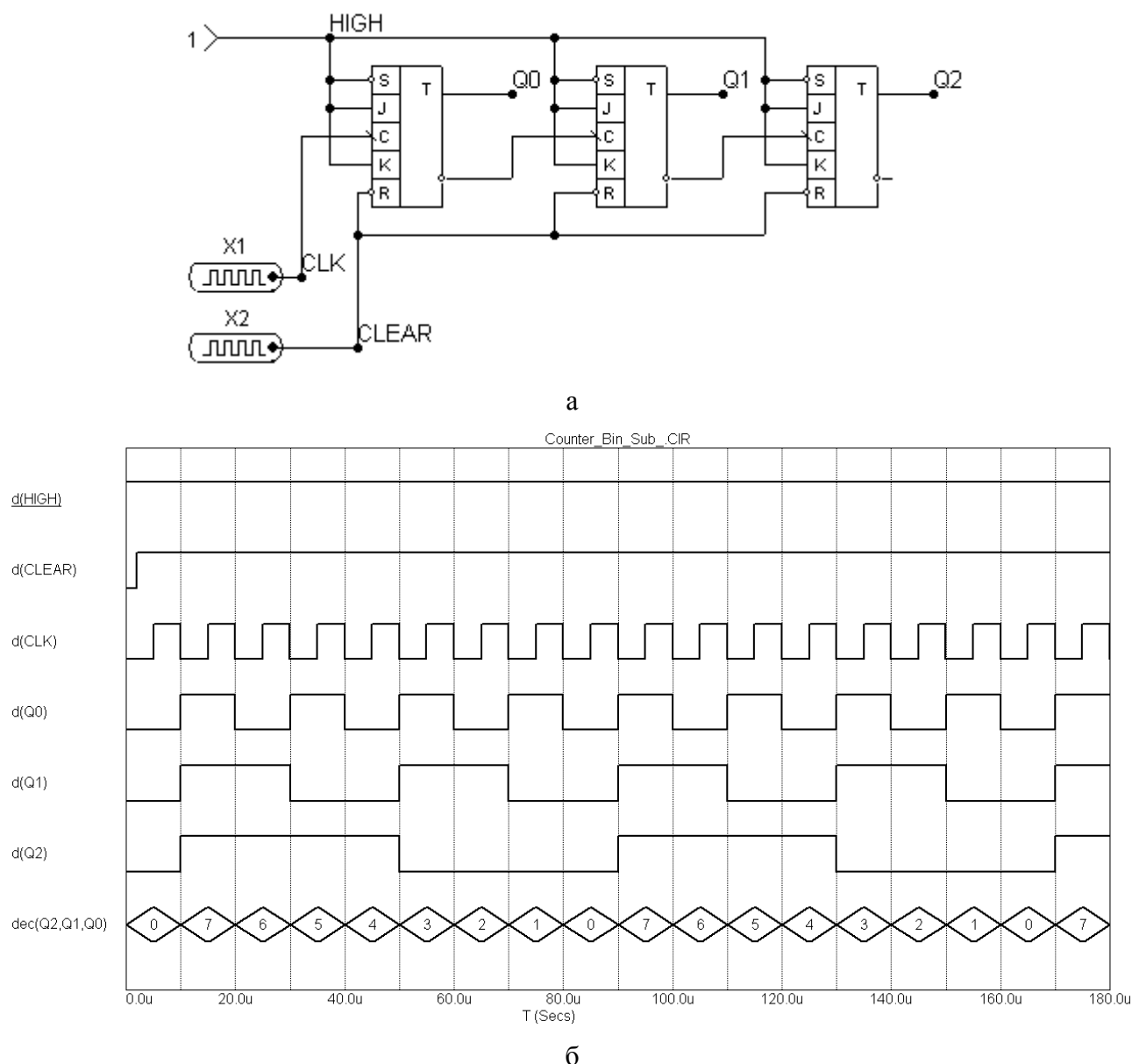


Рисунок 2.42 — Вычитающий счетчик с последовательным переносом: а — схема;
б — временные диаграммы

Таблица 2.7 — Зависимость типа выполняемой микрооперации от вида межразрядных связей и типа синхровхода

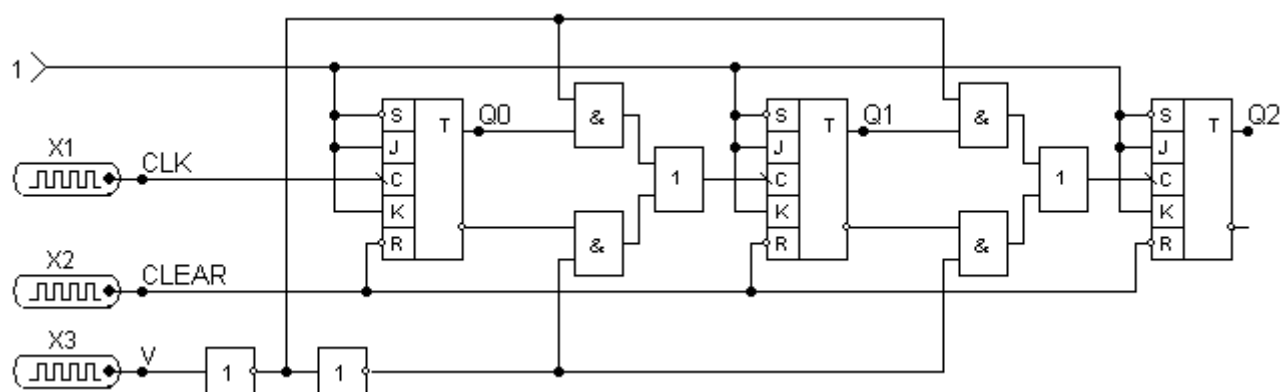
Тип входа Т	Используемый выход	
	Q	\bar{Q}
Прямой динамический ↑	Декремент	Инкремент
Инверсный динамический ↓	Инкремент	Декремент

Таким образом, в счетчиках на синхронных Т-триггерах направление счета зависит как от того, какой из выходов используется для синхронизации последующего триггера, так и от типа входа синхронизации. В табл. 2.7 приведены все возможные комбинации соединения триггеров с различными типами входов синхронизации и получаемые при этом виды счетчиков.

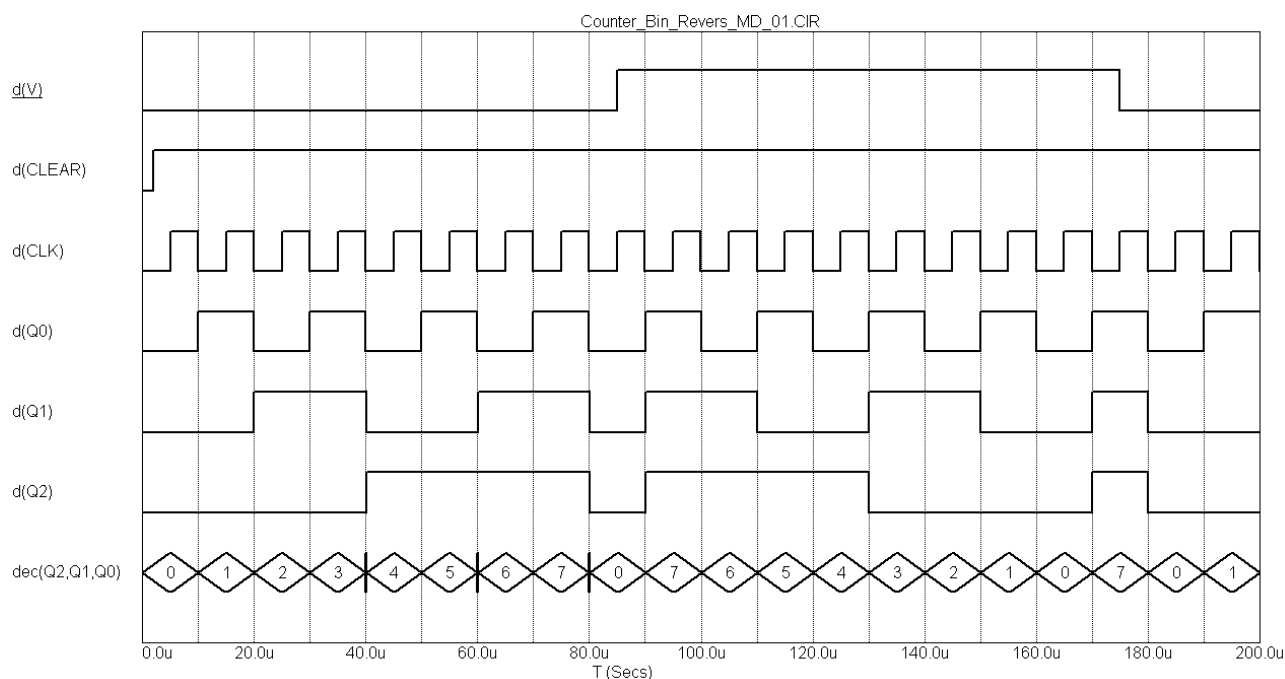
Из сказанного также следует, что направление счета счетчика может изменяться (реверсирование) путем изменения вида межразрядных связей. Последнее легко достигается включением в состав каждой разрядной схемы счетчика

мультиплексора (например на элементах И-ИЛИ, рис. 2.43), управление которым дает возможность реверсировать счет.

Сигнал на входе V данного счетчика определяет вид межразрядных связей, а, следовательно, и тип получаемого счетчика. Схема и временные диаграммы, поясняющие работу такого счетчика при различных значениях сигнала V , приведены на рис. 2.43.



а



б

Рисунок 2.43 — Реверсивный двоичный счетчик с последовательным переносом: а — схема; б — временные диаграммы

Полученные структуры относятся к асинхронным счетчикам, т. к. в них каждый триггер переключается выходным сигналом предыдущего, и эти переключения происходят не одновременно. Переключение одного триггера за другим есть не что иное, как распространение переноса по разрядам числа при изменении содержимого счетчика. В худшем случае перенос распространяется по всей разрядной сетке от младшего разряда к старшему, т. е. для установления нового состояния должны переключиться последовательно все триггеры. Отсюда видно, что время установления кода в асинхронном счетчике удовлетво-

бенно заметно при высокой частоте работы счетчика. Так в случае, показанном на рис. 2.44, период тактовых импульсов, составляет 80 нс, в то время как время переключения триггеров счетчиков составляет 15-20 нс. Опасность воздействия коротких ложных импульсов на последующие цифровые схемы заставляет прибегать при необходимости к стробированию выхода счетчика.

Все рассмотренные выше счетчики являются счетчиками с последовательным переносом, так как переключение каждого последующего триггера может произойти только после переключения предыдущего. Данный тип счетчика отличается простотой внутренней структуры. Однако следствием такой организации является большое время установления выходного кода, которое к тому же не остается постоянным в процессе работы и зависит от конкретного значения его выходного кода. Максимальное значение $t_{уст\ к}$ наблюдается в случае необходимости переключения всех триггеров счетчика, например при изменении выходного кода со значения 111 на значение 000, или наоборот (см. рис. 2.44, б). Максимально возможное время установления кода определяется соотношением:

$$t_{уст\ к\ max} = N \cdot t_{пт\ p} \quad (2.4),$$

где N — число разрядов счетчика; $t_{пт\ p}$ — время переключения (установления выходного кода) одного триггера счетчика.

Уменьшить время установления выходного кода счетчика можно при условии, что все триггеры его разрядных схем будут переключаться одновременно. Для этого необходимо отказаться от применения асинхронных триггеров в пользу синхронных и *сформировать сигналы, регламентирующие требуемый порядок переключения триггеров разрядных схем до прихода импульса синхронизации.*

Максимальным быстродействием обладают **синхронные счетчики с параллельным переносом**, структуру которых можно определить, рассмотрев процессы прибавления единицы к двоичным числам и вычитания ее из них, например:

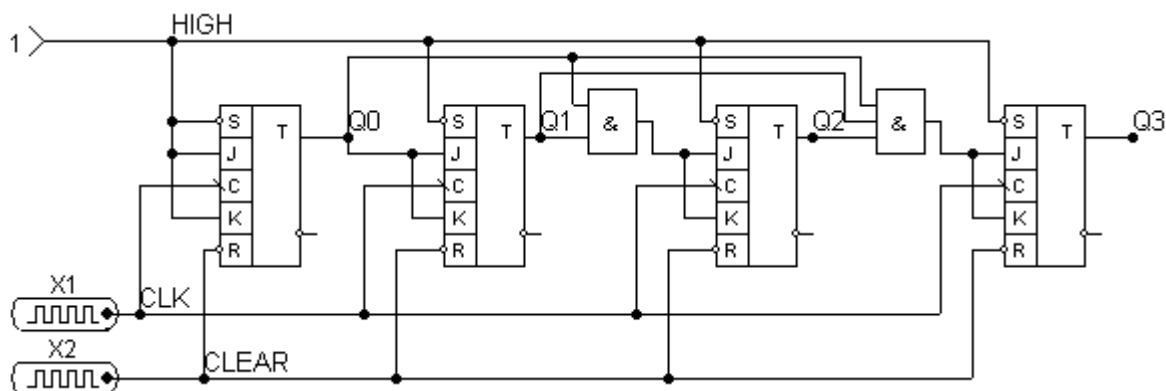
$$\begin{array}{r} 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1 \\ + \quad \quad \quad \quad \quad \quad 1 \\ \hline 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0 \end{array} \qquad \begin{array}{r} 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0 \\ - \quad \quad \quad \quad \quad \quad 1 \\ \hline 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1 \end{array}$$

Результат всегда отличается от исходного числа только в нескольких младших разрядах, значения которых инвертируются. Для суммирующего счетчика требуется инверсия разрядов до первого разряда, равного логическому нулю, включая и его, а для вычитающего аналогично до разряда, равного логической единице. Таким образом, в суммирующем счетчике должны переключиться все младшие последовательные разряды в единичном состоянии, в вычитающем — все младшие последовательные разряды в нулевом состоянии.

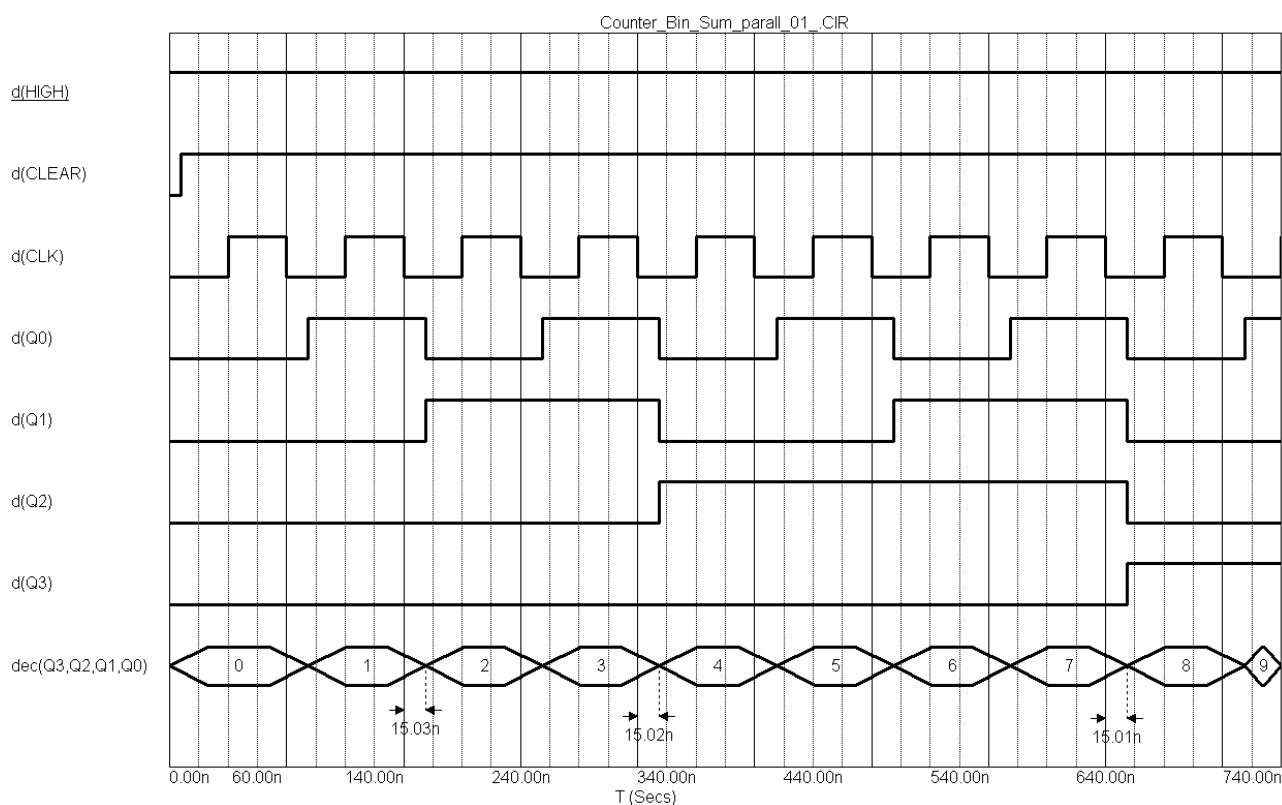
Эти задачи и должны решать счетчики с параллельным переносом. Время установления таких счетчиков не зависит от разрядности счетчика n и равно времени переключения одного триггера:

$$t_{уст\ K} = t_{пт\ p} \quad (2.5).$$

Структура суммирующего синхронного счетчика с параллельным переносом, реализованного на триггерах с управлением срезом, показана на рис. 2.45, а. Следует отметить, что в данной структуре триггер, формирующий сигнал Q_0 , по-прежнему остался асинхронным. Поэтому его входной сигнал $T \equiv 1$.



а



б

Рисунок 2.45 — Суммирующий счетчик с параллельным переносом: а — схема; б — временные диаграммы работы

Однако если счетчик с последовательным переносом непосредственно после установления нового значения выходного кода готов к следующему переключению, то при реализации параллельного алгоритма для подготовки счетчика к следующему переключению должно пройти некоторое время t_{nod} . Это время необходимо для формирования нового сигнала переноса и определяется временем задержки распространения сигнала через логические элементы И ($t_{nod} = t_{\&}$), используемых в цепях формирования сигналов переноса. Так как

это время всегда меньше времени установления выходного кода одиночного триггера, быстродействие полученного счетчика всегда выше быстродействия счетчика с последовательным переносом.

Следует отметить, что в счетчиках с параллельным переносом направление счета не зависит от того, какой (прямой или инверсный) динамический вход имеют триггеры, составляющие его разрядные схемы. Направление счета определяется исключительно тем, какой (прямой или инверсный) выход триггера используется для формирования сигнала переноса. Так, счетчик, схема которого показана на рис. 2.45, будет суммирующим. Если же для формирования сигнала переноса будут использованы инверсные выходы триггеров разрядных схем, счетчик будет вычитающим (табл. 2.8).

Таблица 2.8 — Зависимость типа операции, реализуемой счетчиком с параллельным переносом, от вида межразрядных связей

Сигнал связи триггеров	Операция
Q	Инкремент
\overline{Q}	Декремент

Следовательно, и при использовании параллельного переноса введением в разрядные схемы мультиплексоров на элементах И—ИЛИ можно легко построить реверсивный счетчик.

Схема межразрядной связи для реверсивного счетчика с сигналом V ($V=1$ — прямой счет, $V=0$ — обратный счет) показана на рис. 2.46.

$V=1$ - суммирование, $V=0$ - вычитание.

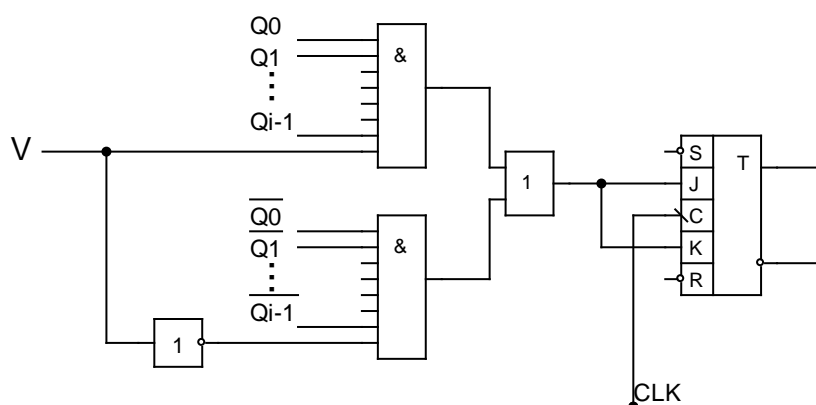


Рисунок 2.46 — Схема межразрядной связи для реверсивного счетчика с параллельным переносом

Сложность практической реализации счетчиков с параллельным переносом состоит в том, что при увеличении числа разрядов пропорционально увеличивается число входов логических элементов И, используемых в цепях формирования сигнала переноса. Поэтому при увеличении числа разрядов используют структуры **счетчиков с комбинированным переносом**. Это, как правило, триггерные структуры с *последовательно-параллельным* или *параллельно-параллельным переносом*.

Идея построения счетчиков с комбинированным переносом состоит в разбиении разрядных схем счетчика на группы, внутри которых осуществляют ли-

либо последовательный, либо параллельный перенос. Формирование сигнала переноса между группами (для последовательно-параллельного варианта) выполняется элементами И лишь в случае, когда триггеры всех входящих в данную группу разрядных схем установлены в единичное состояние, т. е. по параллельному принципу (рис. 2.47, а). Время установки выходного кода в такой структуре

$$t_{\text{уст к max}} = t_{\text{кзр}} \quad (2.6),$$

где $t_{\text{кзр}}$ — время установки выходного кода в пределах одной группы. При использовании в пределах одной группы последовательного переноса (рис. 2.47, а):

$$t_{\text{кзр}} = t_{\text{н mp}} \cdot N_j \quad (2.7),$$

где N_j — число триггеров в j -ой группе.

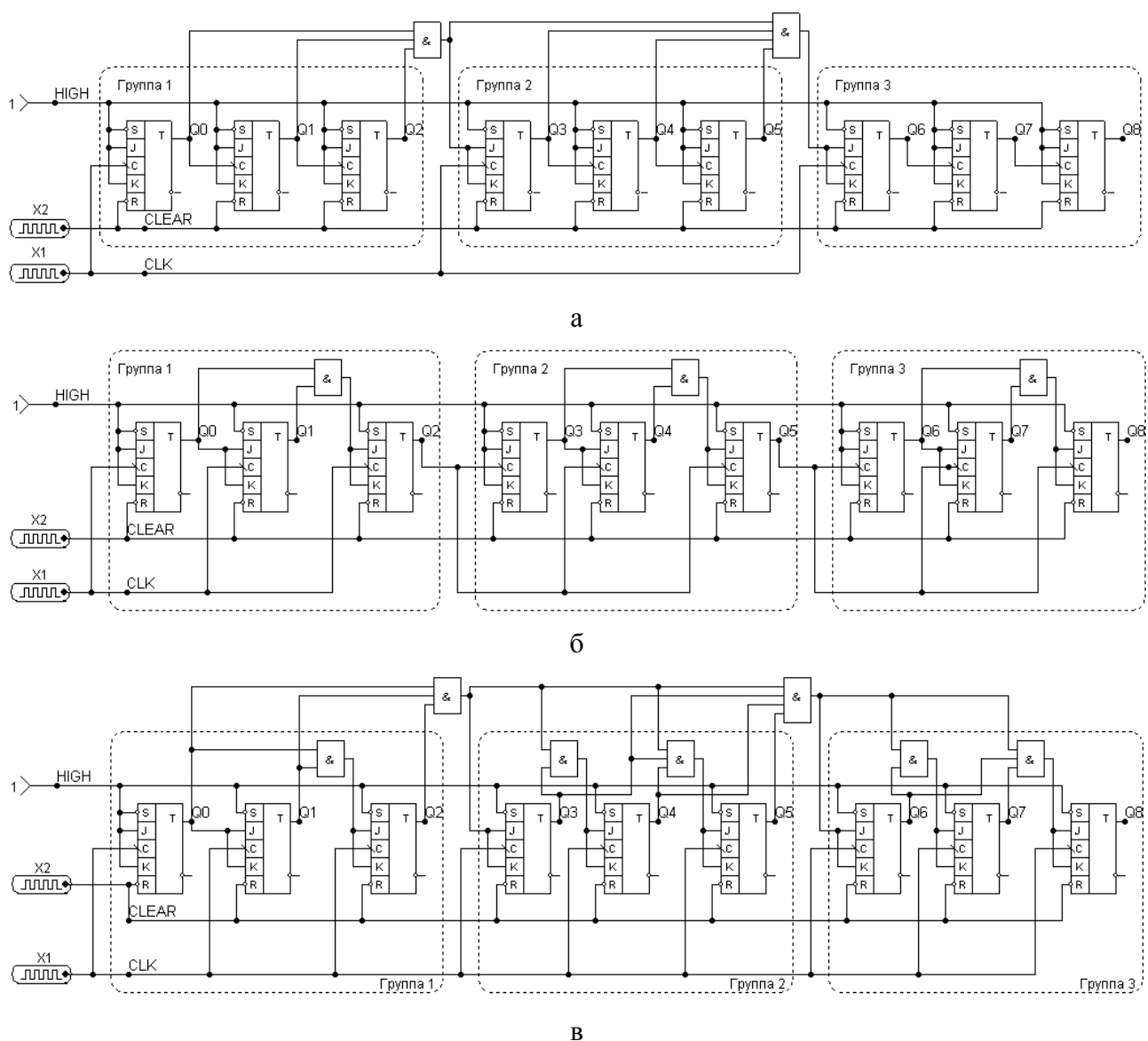


Рисунок 2.47 — Счетчики с комбинированным переносом: а — с последовательно-параллельным; б — с параллельно-последовательным; в — с параллельно-параллельным

Очевидно, что упрощение счетчика с последовательно-параллельным переносом достигается за счет некоторого снижения его быстродействия.

Максимальное время подготовки счетчика с комбинированным переносом (последовательно-параллельным) к следующему переключению

$$t_{\text{под max}} = t_{\&}(L-1) \quad (2.8).$$

где L — число групп в счетчике.

При организации внутри группы параллельного переноса (рис. 2.47, б), а между группами — последовательного, быстродействие счётчика увеличивается (при условии, что число триггеров в группе не меньше чем количество счетных групп). Это происходит за счет уменьшения времени $t_{\text{к гр}}$, которое в данном случае равно:

$$t_{\text{к гр}} = t_{\text{п тр}} \quad (2.9)$$

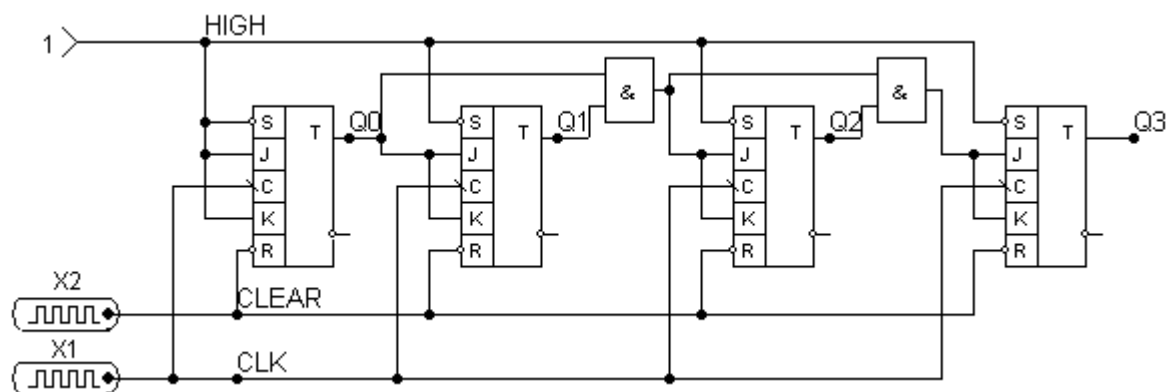
Но при этом за счет организации последовательного межгруппового переноса максимальное время установления выходного кода:

$$t_{\text{уст к max}} = t_{\text{к гр}} \cdot L \quad (2.10)$$

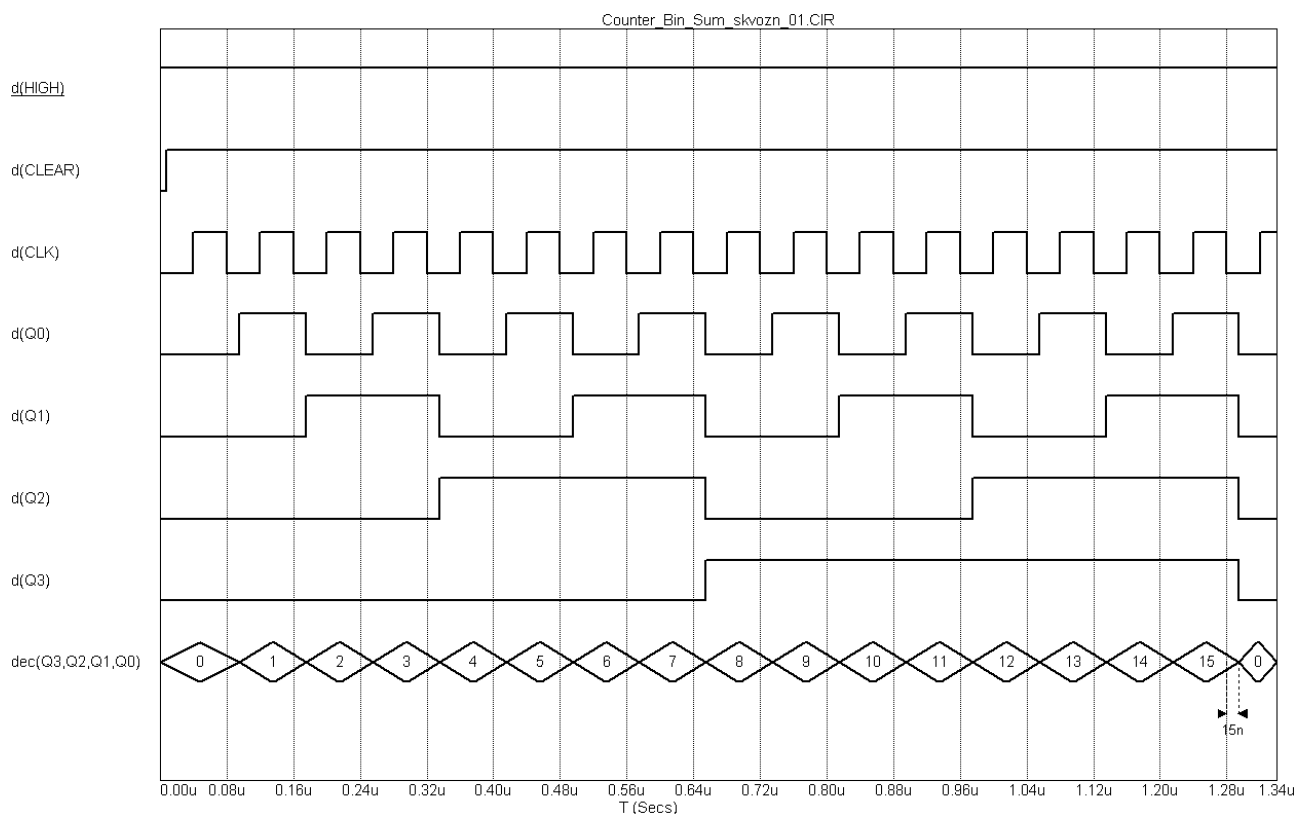
Наилучшими показателями быстродействия (и отсутствием ложных состояний) характеризуется схема с комбинированным параллельно-параллельным переносом (рис. 2.47, в). При организации и внутри группы и между группами параллельного переноса (рис. 2.47, в) время установления выходного кода снижается до времени переключения одного триггера. Однако данная схема характеризуется увеличенным временем подготовки за счет прохождения сигнала через более длинную цепочку последовательно-соединенных вентилях «И», что приводит к отказу переключения старших разрядов счетчика при меньшем значении тактовой частоты, чем в схемах рис. 2.47, а, б.

Если уменьшить разрядность группы до единицы и использовать синхронные Т-триггеры, то получится схема синхронного счетчика с последовательным переносом или *счетчик со сквозным переносом* (см. рис. 2.48, а). Схема относится к числу синхронных, т. к. все триггеры срабатывают одновременно под действием единого входного сигнала. В этом проявляется быстрая реакция схемы на входной сигнал, такая же, как и в счетчике с параллельным переносом. Однако по максимальной частоте входных сигналов эта схема существенно отличается от схемы с параллельным переносом, т. к. до подачи нового входного сигнала требуется дать цепочке вентилях установиться в новое состояние путем их последовательного переключения.

В данной схеме переключение всех триггеров происходит одновременно, однако для подготовки к следующему переключению должно пройти время, необходимое для последовательного формирования на выходах всех элементов $И$ новых значений сигнала переноса. Это время, как и время установления выходного кода в счетчиках с последовательным переносом, зависит от конкретного кода, записанного в счетчик. Получаемый в данном случае выигрыш по быстродействию определяется меньшим временем распространения сигнала в элементе $И$ по сравнению с временем установления выходного кода в отдельном триггере.



a



6

Рисунок 2.48 — Счетчик со сквозным переносом: а — принципиальная схема;
б — временные диаграммы работы

В развитых сериях ИС обычно имеется по 5...10 вариантов двоичных счетчиков, выполненных в виде 4-х разрядных групп (секций). Каскадирование секций может выполняться путем их последовательного включения по цепям переноса, организации параллельно-последовательных переносов.

Особенностью двоичных счетчиков синхронного типа является наличие ситуаций с одновременным переключением всех его разрядов (например, для суммирующего счетчика при переходе от кодовой комбинации 11...1 к комбинации 00...0 при переполнении счетчика и выработке сигнала переноса). Одновременное переключение многих триггеров создает значительный токовый импульс в цепях питания цифровых схем и может привести к сбою в их работе. Поэтому в руководящих материалах по использованию некоторых БИС (СБИС) программируемой логики, в частности, имеется ограничение на разрядность

двоичных счетчиков, заданной величиной k (например, 16). При необходимости применения счетчика большей разрядности рекомендуется переходить к коду Грея, для которого переходы от одной кодовой комбинации к другой сопровождаются переключением всего одного разряда. Правда, для получения результата счета в двоичном коде придется использовать дополнительно преобразователь кода, но это является платой за избавление от токовых импульсов большой интенсивности в цепях питания.

2.8.3 Двоично-кодированные счетчики

Как уже отмечалось ранее, двоично-кодированный счетчик имеет модуль счета, отличный от целой степени числа 2. Примером такого счетчика может служить счетчик с модулем счета 10, т. е. двоично-десятичный счетчик.

Для построения счетчика с произвольным модулем M используется разрядность (количество триггеров) $n = \lceil \log_2 M \rceil$, где $\lceil \rceil$ — знак округления до ближайшего большего целого числа. Иными словами, исходной структурой как бы служит двоичный счетчик с модулем 2^n , превышающим заданный и ближайшим к нему. Такой двоичный счетчик имеет $2^n - M = L$ лишних (неиспользуемых) состояний, подлежащих исключению.

Способы исключения лишних состояний многочисленны, и для любого M можно предложить множество реализаций счетчика. Исключая некоторое число первых состояний, получим ненулевое начальное состояние счетчика, что приводит к отсутствию естественного порядка счета и регистрации в счетчике кода с избытком. Исключение последних состояний позволяет сохранить естественный порядок счета. Трудоемкость синтеза и схемотехнические затраты обоих вариантов приблизительно одинаковы, поэтому далее будем ориентироваться на схемы с естественным порядком счета. Состояния счетчиков во всех случаях предполагаем закодированными двоичными числами, т. е. будем рассматривать *двоично-кодированные счетчики*.

В счетчиках с исключением последних состояний счет ведется обычным способом вплоть до достижения числа $M-1$. Далее последовательность переходов счетчика в направлении роста регистрируемого числа должна быть прервана, и следующее состояние должно быть нулевым. При этом счетчик будет иметь M внутренних состояний (от 0 до $M-1$), т. е. его модуль счета будет равен заданному значению M .

При построении таких счетчиков получили распространение в основном два метода:

- метод исключения лишних состояний (или как его еще называют метод модификации межразрядных связей);
- метод управляемого сброса.

Первый метод заключается в использовании стандартной методики синтеза последовательностных устройств. При построении счетчика с модифицированными межразрядными связями последние, лишние, состояния исключаются непосредственно из таблицы функционирования счетчика. При этом после построения схемы обычным для синтеза автоматов способом получается счетчик, специфика которого состоит в нестандартных функциях возбуждения тригге-

ров, и, следовательно, в нестандартных связях между триггерами, что и объясняет название способа. Схема получается как специализированная, изменение модуля счета требует изменения самой схемы, т. е. легкость перестройки с одного модуля на другой отсутствует. В то же время реализация схемы счетчика может оказаться простой.

Поэтому такой подход используется только при проектировании часто встречающихся устройств, выпускаемых большими партиями. В иных случаях его применение становится экономически нецелесообразным.

Рассмотрим более подробно этот метод на примере построения счетчика с модулем счета $M=5$. Отметим, что методика синтеза может быть полностью стандартизирована при использовании для минимизации логических функций управления J и K входами триггеров счетчика метода карт Карно (см. пример синтеза счетчика-делителя на 3 в методических указаниях к ЛР №2 по ЭПУ). Карты Карно делают синтез более надежным и наглядным, хотя при достаточной внимательности можно ограничиться и таблицей состояний с сигналами возбуждения используемых триггеров [2] (см. табл. 2.9).

Построение счетчика первым способом (методом исключения лишних состояний).

Построение счетчика первым способом проиллюстрируем примером для $M=5$, начав с таблицы (табл. 2.9).

Таблица 2.9 — Состояния счетчика с модулем счета 5 и значения сигналов для составляющих его JK триггеров

Исходное состояние			Следующее состояние			Функции возбуждения					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

$Q_i \rightarrow Q_{i+1}$	J	K
$0 \rightarrow 0$	0	X
$0 \rightarrow 1$	1	X
$1 \rightarrow 1$	X	0
$1 \rightarrow 0$	X	1

При нахождении функций возбуждения триггеров использована таблица переходов JK-триггера (табл. 2.9, справа). Имея в виду, что вместо символа произвольного сигнала X можно подставлять любое значение (0 или 1), воспользуемся методом минимизации карт Карно с целью получения функций сигналов возбуждения триггеров в форме ДНФ.

$$J_2 = Q_1 Q_0$$

$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$$K_2 = 1$$

$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	X	X	X	X
1	1	X	X	X

$$J_1 = Q_0$$

$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	0	1	X	X
1	0	X	X	X

$$K_1 = Q_0$$

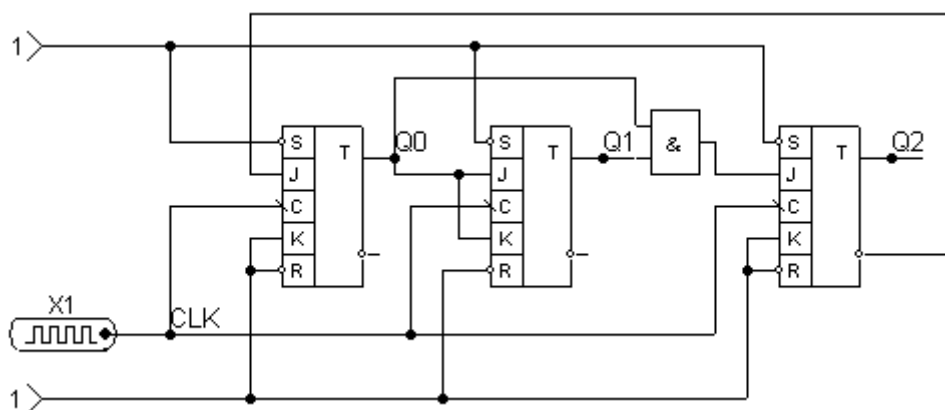
$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	X	X	1	0
1	X	X	X	X

$$J_0 = \bar{Q}_2$$

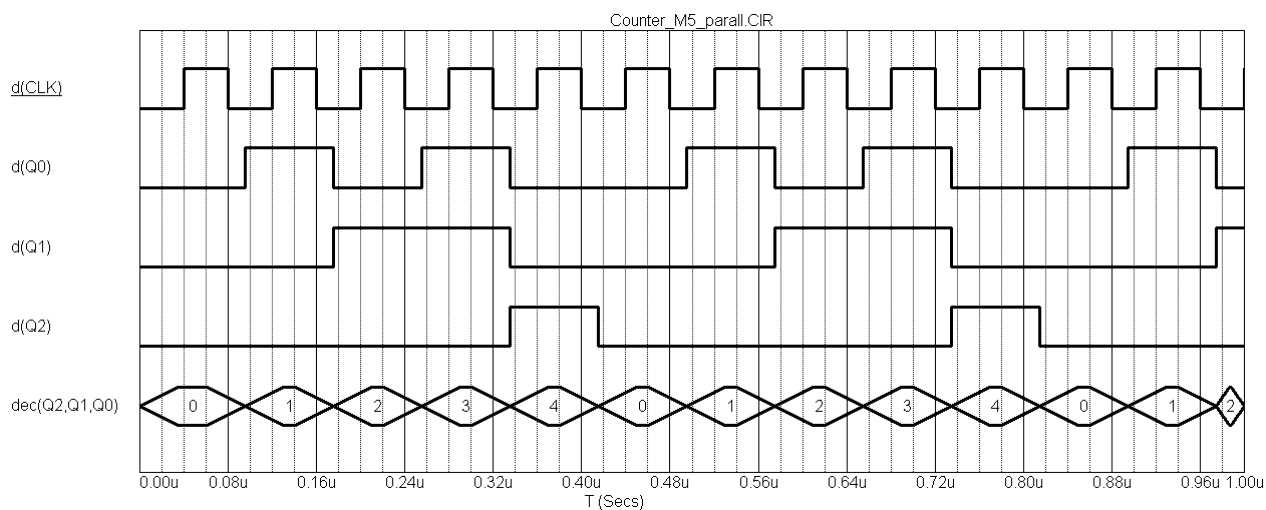
$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	1	X	X	1
1	0	X	X	X

$$K_0 = 1$$

$Q_2 \backslash Q_1 Q_0$	00	01	11	10
0	X	1	1	X
1	X	X	X	X



а



б

Рисунок 2.49 — Двоично-кодированный счетчик с модулем счета 5: а — схема;
б — временные диаграммы работы

В спроектированной схеме счетчика лишние состояния исключены в том смысле, что они не используются при нормальном функционировании счетчика. Но при сбоях или после подачи на схему напряжения питания в начале ее

работы лишние состояния могут возникать. Поэтому полезно определить поведение схемы (автомата), в которой возникло лишнее состояние. Имея схему, можно полностью предсказать поведение схемы во всех возможных ситуациях. Сделаем это для полученной схемы счетчика с модулем 5.

Взяв каждое лишнее состояние, найдем для него функции возбуждения триггеров, определяющие их переходы в следующее состояние. При необходимости найдем таким же способом следующий переход и т. д. Для взятого примера лишними являются состояния 101, 110 и 111.

В состоянии 101 $Q_2=1$, $Q_1=0$ и $Q_0=1$. Зная функции возбуждения триггеров, находим, что $J_0=0$, $K_0=1$, $J_1=K_1=1$, $J_2=0$, $K_2=1$. Следовательно, триггеры 0 и 2 сбросятся, а триггер 1 переключится в противоположное текущему состоянию и из лишнего состояния 101 счетчик перейдет в состояние 010.

Аналогичным способом можно получить результаты для состояний 110 и 111. В итоге удобно построить диаграмму состояний счетчика (граф переходов), в которой учтен не только рабочий цикл (его состояния покажем кружками), но и поведение автомата, попавшего в неиспользуемые состояния (эти состояния показаны прямоугольниками). Такая диаграмма состояний показана на рис. 2.50. Из диаграммы видно, что рассматриваемый счетчик обладает свойством самозапуска (самовосстановления после сбоя) — независимо от исходного состояния он приходит в рабочий цикл после начала работы. Этим свойством обладают не все схемы. В некоторых схемах автоматический вход в рабочий цикл не происходит.

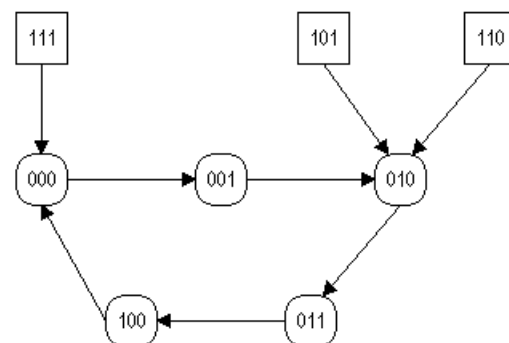


Рисунок 2.50 — Диаграмма состояний счетчика с модулем 5

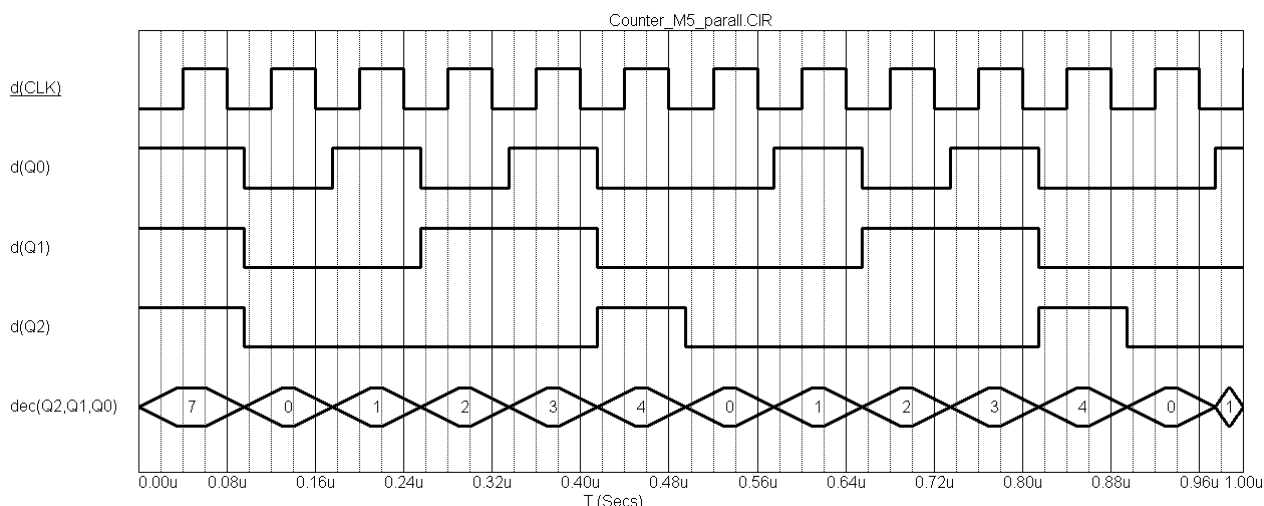


Рисунок 2.51 — Самовосстановление счетчика с модулем счета $M=5$ из состояния 111

При разработке некоторых схем в них вводят специальные элементы или подсхемы для придания свойств самозапуска.

Синтезированный блок используется в счетчике ИЕ2, который появился в самых первых сериях ИС и до сих пор повторяется во всех более новых. Он со-

стоит фактически из двух секций с модулями 2 и 5, представленных триггером T_0 , делящим частоту на 2 и группой $T_1 T_2 T_3$, на которой собран счетчик с модулем 5. Секции можно использовать по отдельности или соединять последовательно с помощью внешней коммутации выводов для получения двоично-десятичного счетчика. Соединение счетчиков в порядке mod2-mod5 дает двоично-десятичный счетчик с естественной последовательностью счета, который в режиме делителя частоты формирует импульсы со скважностью 5. Соединение в порядке mod5-mod2 дает, естественно, тот же модуль счета, но состояния счетчика образуют последовательность чисел 0, 1, 2, 3, 4, 8, 9, 10, 11, 12, после которой цикл повторяется. В режиме делителя частоты формируются импульсы со скважностью 2. Таким образом, разбиение счетчика на две секции предоставляет возможность получить как обычный двоично-десятичный счетчик, так и два делителя частоты на 10 — с формированием узких импульсов ($t_{\text{и}}=T/5$) и симметричных импульсов ($t_{\text{и}}=T/2$), где T — период повторения импульсов.

Данный счетчик также снабжен двумя парами равноценных входов установки R_1, R_2 и S_1, S_2 , позволяющими записать в триггеры всех разрядных схем либо нулевые, либо единичные значения.

Разрядные схемы некоторых счетчиков допускают запись не определенной (нулевой), а произвольной информации. Примером такого решения является счетчик типа ИЕ9 (74XXX160), в котором возможна параллельная асинхронная запись произвольного начального состояния. Кроме этого, данный счетчик, подобно параллельному регистру, снабжен входом сброса, что значительно расширяет его возможности.

Построение счетчика вторым способом (методом управляемого сброса)

На практике, как правило, желательно иметь схему-полуфабрикат, которая без особых сложностей и дополнительных элементов позволила бы гибко изменять алгоритм работы устройства.

Второй метод построения счетчиков с произвольным модулем — метод управляемого сброса — позволяет изменять модуль счета очень простым способом, не требующим изменений самой схемы счетчика.

Рассмотрим этот способ применительно к реализации *синхронного счетчика с параллельным переносом*. Функции возбуждения двоичного счетчика указанного типа, как известно, имеют вид $J_i=K_i=Q_0 Q_1 \dots Q_{i-1}$ (в младшем триггере $J_0=K_0=1$). Введем в эти функции сигнал сброса R , изменив их следующим образом:

$$J_i=(Q_0 Q_1 \dots Q_{i-1}) \cdot \bar{R}, \quad K_i=J_i \vee R.$$

Пока сигнал сброса отсутствует ($R=0$) функции J_i и K_i не отличаются от соответствующих функций двоичного счетчика с параллельным переносом ($J_i=K_i=Q_0 Q_1 \dots Q_{i-1}$). Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, K_i — единичными, что заставляет все триггеры *сброситься* по приходе следующего такта.

Если сигнал R появится как следствие появления в счетчике $(M-1)$ состояния, то будет реализована последовательность счета 0, 1, 2, ..., $M-1$, 0, ..., т. е. счетчик с модулем M .

Схемы всех разрядов счетчика с управляемым сбросом не зависят от модуля счета. Кроме разрядных схем, счетчик содержит один конъюнктор, вырабатывающий сигналы сброса R и \bar{R} при достижении содержимым счетчика значения $M-1 = \tilde{Q}_{n-1} \dots \tilde{Q}_1 \tilde{Q}_0$ *binary* (рис. 2.52).

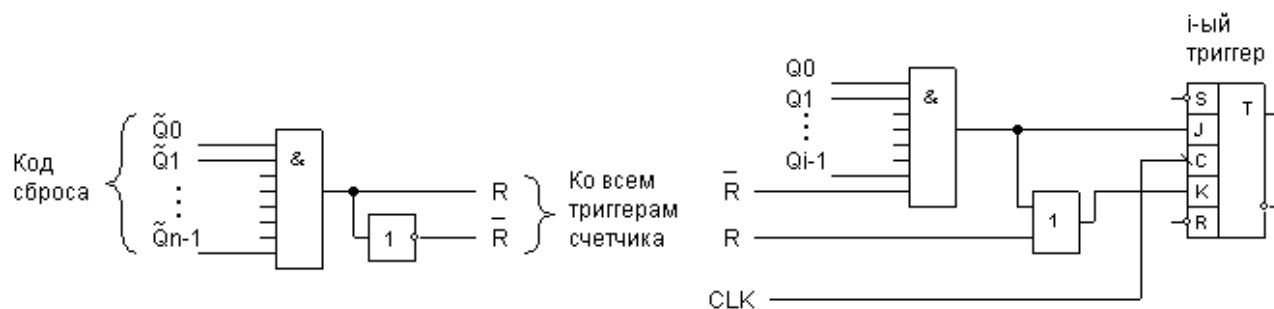
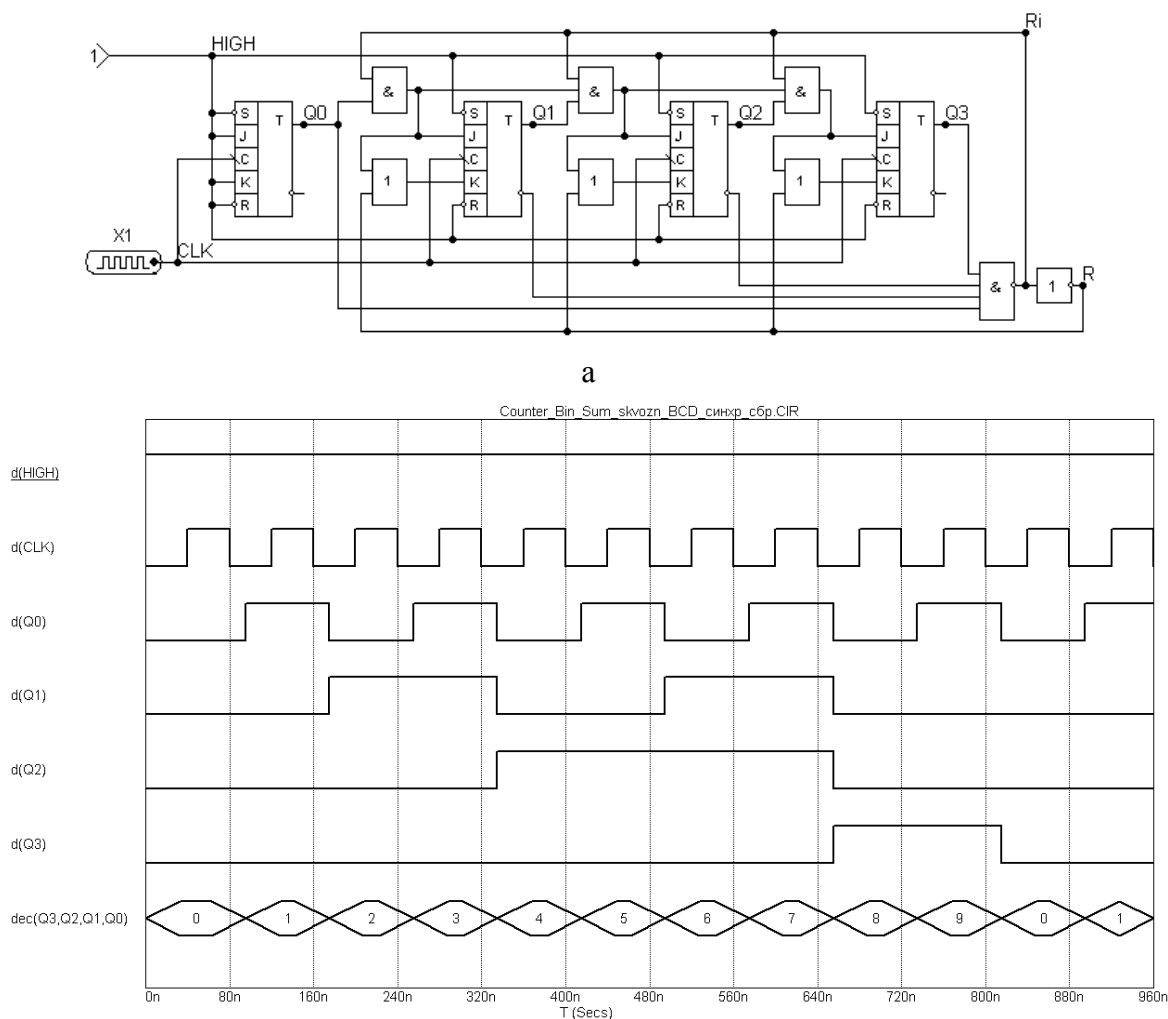


Рисунок 2.52 — Схема счетчика с управляемым сбросом

Если, например, имеется четырехразрядный счетчик, и на входы конъюнктора выработки сигнала сброса подключены выходы триггеров $Q_3 \bar{Q}_2 \bar{Q}_1 \bar{Q}_0$, то сброс произойдет после достижения счетчиком числа $1001_2 = 9_{10}$, т. е. счетчик будет работать как двоично-десятичный.



б

Рисунок 2.53 — Двоично-десятичный счетчик на основе схемы со сквозным переносом с синхронной схемой управляемого сброса: а — схема; б — временные диаграммы

Двоично-десятичный счетчик, полученный из двоичного со схемой сквозного переноса с помощью вышеописанного преобразования представлен на рис. 2.53.

Также сигнал сброса можно формировать не на информационные входы JK-триггеров счетчика в синхронном режиме, а на асинхронный вход сброса \bar{R} .

Идея данного метода состоит в принудительном формировании сигнала сброса триггеров по асинхронному входу для разрядных схем двоичного счетчика при появлении на его выходе кода, совпадающего с требуемым модулем счета M . Проиллюстрируем сказанное на примере преобразования двоичного 4-разрядного счетчика со сквозным переносом в двоично-десятичный счетчик (рис. 2.54, а).

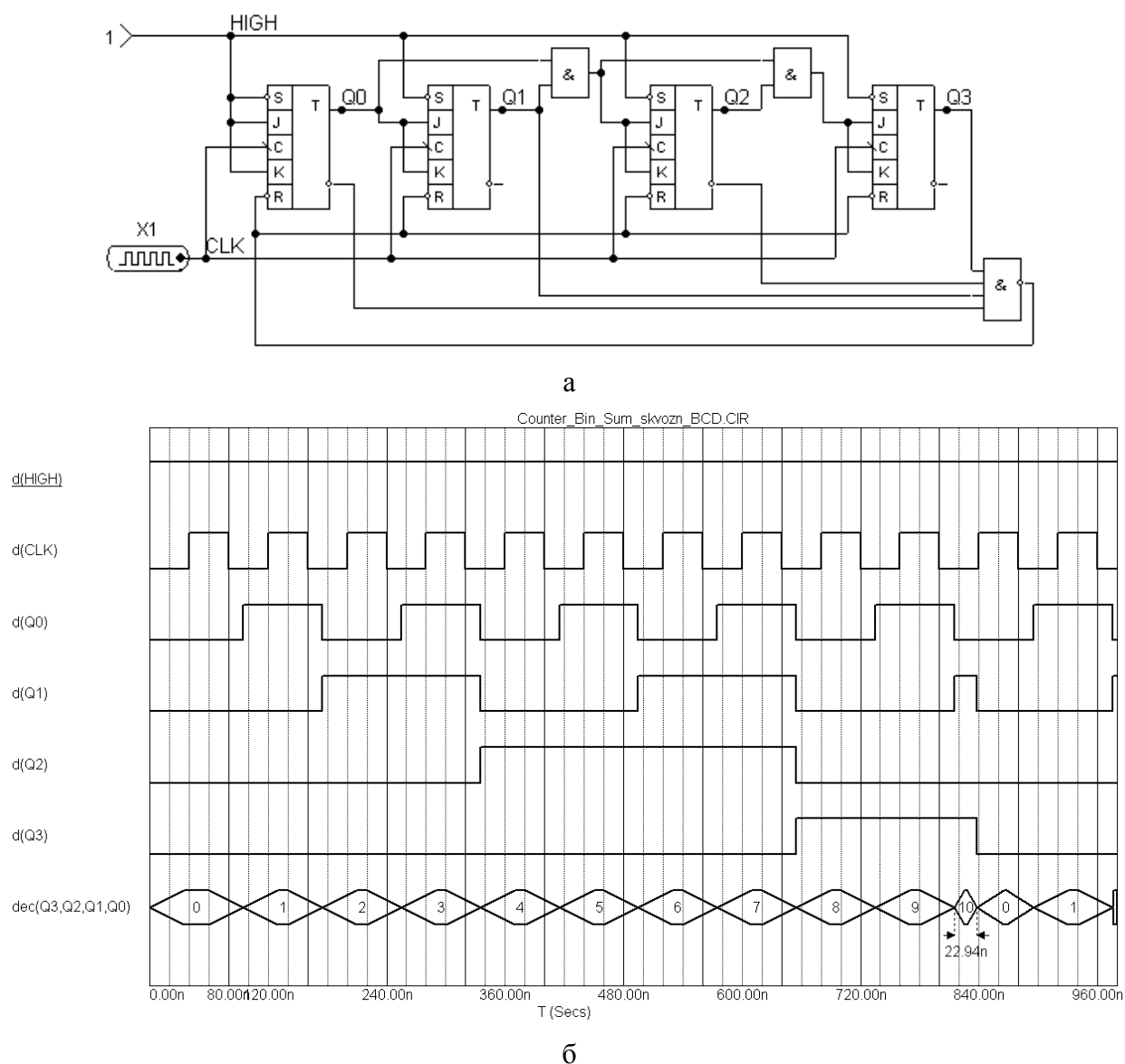


Рисунок 2.54 — Двоично-десятичный счетчик на основе схемы со сквозным переносом с асинхронной схемой управляемого сброса: а — схема; б — временные диаграммы

Для этого необходимо на входы дополнительного логического элемента 4И—НЕ подать комбинацию выходных сигналов разрядных схем, соответствующую коду 1010, т. е. $Q_3 \bar{Q}_2 Q_1 \bar{Q}_0$. В этом случае при появлении на выходе одиннадцатого по счету выходного кода (1010) элемент 4И—НЕ через время, равное времени его задержки распространения, сформирует сигнал сброса и на

выходе счетчика установится нулевой код. На рис. 2.54, б приведены временные диаграммы, иллюстрирующие работу такого счетчика.

Следует подчеркнуть, что при использовании метода управляемого сброса по асинхронным входам разрядных триггеров на выходе счетчика на время t_1 , равное сумме времени задержки распространения сигнала в дополнительном элементе 4И—НЕ и времени установления выходного кода счетчика по входу R , устанавливается лишнее выходное состояние (см. рис. 2.54, б). Это является платой за универсальность метода. Если по условию работы счетчика даже кратковременное появление на его выходе лишнего состояния является недопустимым, при проектировании необходимо пользоваться методом исключения лишних состояний с синхронной схемой сброса, например, по выходному коду ($M-1$) формировать управляющие сигналы, обеспечивающие по следующему импульсу синхронизации сброс всех триггеров счетчика (см. рис. 2.52, 2.53).

При соответствующем выборе исходной ИС счетчика решение описанной задачи можно упростить. Так, при проектировании на основе двоичного счетчика двоично-десятичного на его выходе не должны существовать любые коды, начиная с 1010. Поэтому число входов дополнительного элемента может быть уменьшено до двух, на которые достаточно подать значения Q_3 и Q_1 . Полная принципиальная схема счетчика со сквозным переносом (синхронного) с упрощенной схемой формирования управляемого асинхронного сброса представлена на рис. 2.55, а. Временные диаграммы работы этой схемы полностью аналогичны диаграммам, представленным на рис. 2.54, б.

Стандартные счетчики типа ИЕ4, ИЕ5 (правда они являются асинхронными) содержат по два входа асинхронного сброса, объединенных операцией И—НЕ. Поэтому, например, для получения двоично-десятичного счетчика на основе ИС ИЕ5, достаточно соответствующим образом соединить ее выходы (рис. 2.55, б).

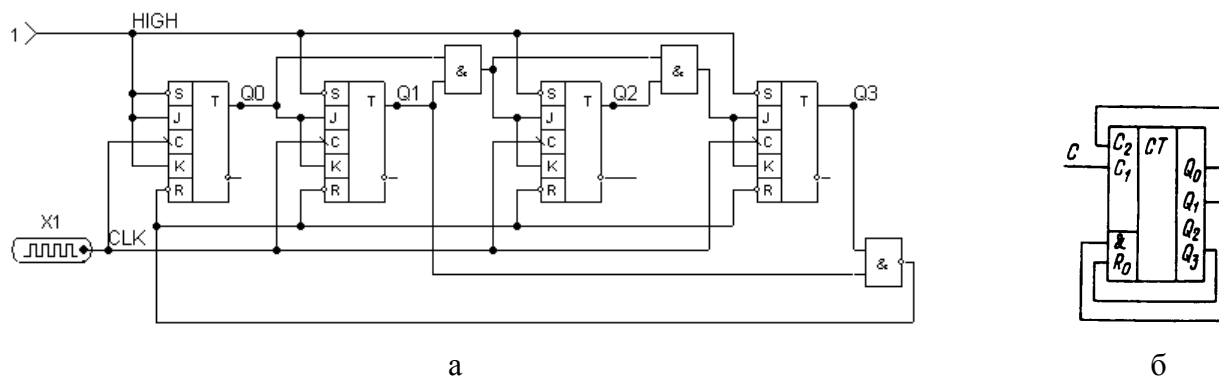


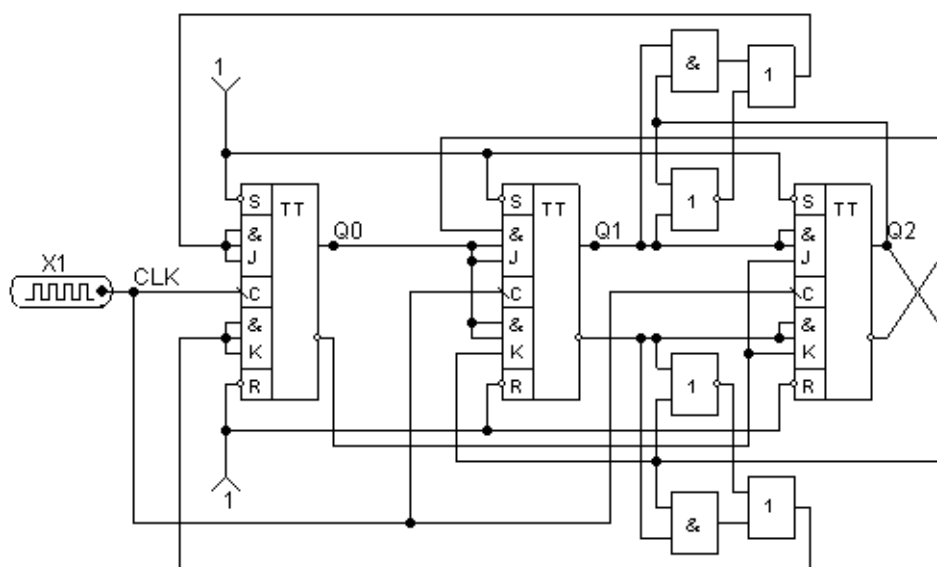
Рисунок 2.55 — Двоично-десятичный счетчик с упрощенной схемой управляемого сброса: а — принципиальная схема синхронного счетчика; б — схема соединения выводов ИЕ5

2.8.4 Счетчики с недвоичным кодированием состояний

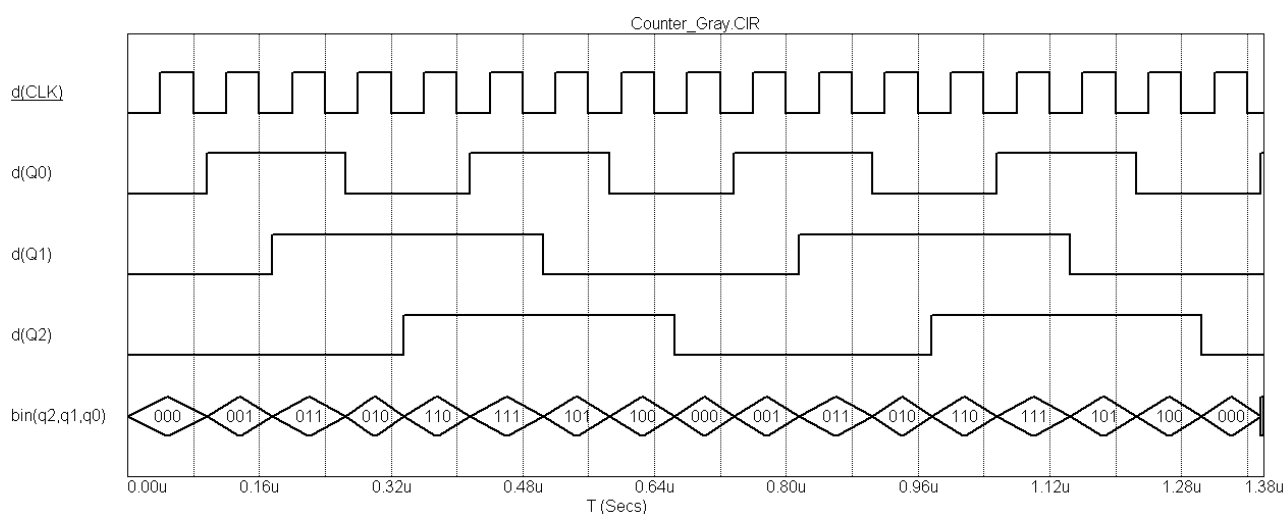
Наибольшее практическое значение среди счетчиков с недвоичным кодированием состояний имеют счетчики с кодом Грея, счетчики Джонсона и счетчики с кодом «1 из N».

Счетчики в коде Грея

Код Грея известен с 70-х годов XIX века, однако оказался связанным с именем Ф. Грея только в 50-х годах XX века, когда Ф. Грей применил его для построения преобразователя угловых перемещений в цифровой код, обладающего явными преимуществами перед преобразователем с двоичным кодом. Код Грея относится к таким, в которых при переходе от любой кодовой комбинации к следующей изменяется только один разряд. В схемотехнике счетчиков это свойство устраняет одновременное переключение многих разрядов, характерное для двоичных счетчиков при некоторых переходах. Одновременное переключение многих элементов создает такие токовые импульсы в цепях питания схем, которые могут вызывать сбои в работе устройства. В ряде БИС/СБИС применение двоичных счетчиков большой разрядности не разрешается, и они заменяются счетчиками с кодом Грея и последующим преобразованием кода Грея в двоичный.



а



б

Рисунок 2.56 — Счетчик в коде Грея

Сложность схемотехнической реализации счетчика с кодом Грея выше за счет усложнения логических функций возбуждения информационных входов триггеров (дополнительные логические элементы), преобразователь кодов относительно прост. Нетрудно построить счетчик с кодом Грея формальным способом с помощью минимизации на основе карт Карно, исходя из таблицы переходов счетчика (выполнить самостоятельно). Последовательность кодовых комбинаций для кода Грея можно получить по соотношению $g_i = b_i \oplus b_{i+1}$, где g_i — значение разряда кода Грея; b_i , значение разряда двоичного кода, преобразуемого в код Грея. Разряд левее старшего для двоичного кода считается нулевым.

Счетчики в коде «1 из N»

Счетчики в коде «1 из N» находят применение в системах синхронизации, управления и других ЦУ. На их основе получают импульсные последовательности с заданными временными диаграммами. Для этого можно вначале разбить период временной диаграммы на части («кванты»), соответствующие минимальному интервалу временной диаграммы, применив задающий генератор с частотой, равной m/T , где m — число «квантов» в периоде диаграммы T . Выходные импульсы задающего генератора затем распределяются во времени и пространстве так, что каждый «квант» появляется в свое время и в своем пространственном канале.

Счетчик в коде «1 из N» имеет один вход, на который подаются импульсы задающего генератора, и N выходов, причем первый импульс генератора передается на первый выход счетчика (канал), второй импульс во второй канал и т. д. Структура такого счетчика, называемого также распределителем тактов РТ, и временные диаграммы его работы показаны на рис. 2.57, а, б, в, причем диаграмма на рис. 2.57, б соответствует режиму распределения уровней (РУ) (паузы между активными состояниями каналов отсутствуют), а диаграмма на рис. 2.57, в — режиму распределения импульсов (РИ). Распределители импульсов не имеют самостоятельной схемотехники, они реализуются на основе распределителей уровней путем включения в их выходные цепи конъюнкторов, на вторые входы которых подаются импульсы задающего генератора.

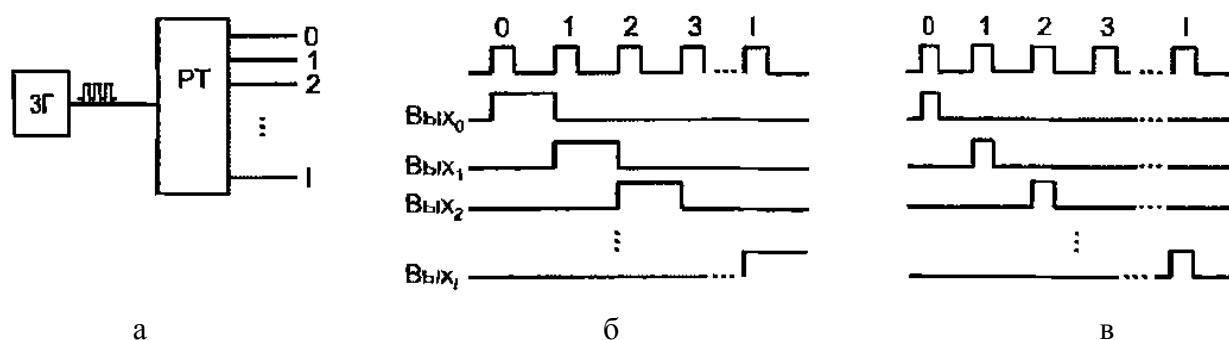


Рисунок 2.57 — Структура и временные диаграммы распределителей тактов

Счетчики в коде «1 из N» на основе кольцевых регистров

Распределителем тактов является сдвигающий регистр (рис. 2.58, а), замкнутый в кольцо, если записанное в регистр слово содержит всего одну единицу (рис. 2.58, б). При сдвиге единица перемещается с одного выхода на другой, циркулируя в кольце. Число выходов РТ равно разрядности регистра. Недоста-

Можно поставить задачу быстрого исправления сбоев, в том числе в ближайшем же такте. Для этого нужно задать и реализовать соответствующую диаграмму состояний распределителя. Сделаем это для трехканального распределителя. Диаграмма состояний с указанием рабочего цикла кружками и ложных состояний прямоугольниками приведена на рис. 2.59. Ей соответствует следующая таблица истинности (табл. 2.10).

Таблица 2.10 — Состояния распределителя тактов с самовосстановлением

$Q_2Q_1Q_0$	$Q_{2n}Q_{1n}Q_{0n}$
000	001
001	010
010	100
011	001
100	001
101	001
110	001
111	001

$$D_1 = \overline{Q_2}\overline{Q_1}Q_0$$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	0	1	0	0
1	0	0	0	0

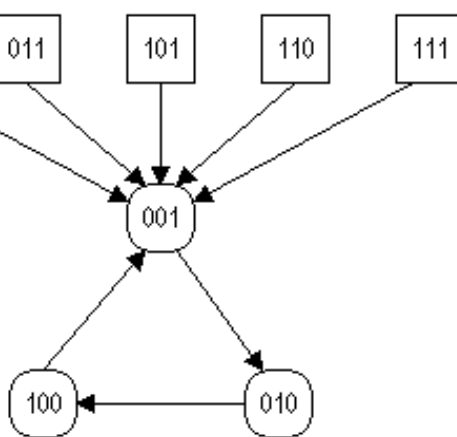


Рисунок 2.59 — Диаграмма состояний распределителя тактов с самовосстановлением

Выбрав для построения схемы триггеры типа D, учтем, что функция возбуждения этого триггера $D=Q_H$. Проведем минимизацию функций возбуждения D-входов триггеров с помощью карт Карно:

$$D_2 = \overline{Q_2}Q_1\overline{Q_0}$$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	0	0	0	1
1	0	0	0	0

$$D_0 = \overline{Q_1}\overline{Q_0} \vee Q_1Q_0 \vee Q_2$$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	1	0	1	0
1	1	1	1	1

Распределители на кольцевых регистрах находят применение при малом числе выходных каналов, когда необходимость иметь по триггеру на каждый канал не ведет к чрезмерно большим аппаратным затратам. Достоинством распределителей на кольцевых регистрах является отсутствие дешифраторов в их структуре и, как следствие, высокое быстродействие (задержка перехода в новое состояние равна времени переключения триггера).

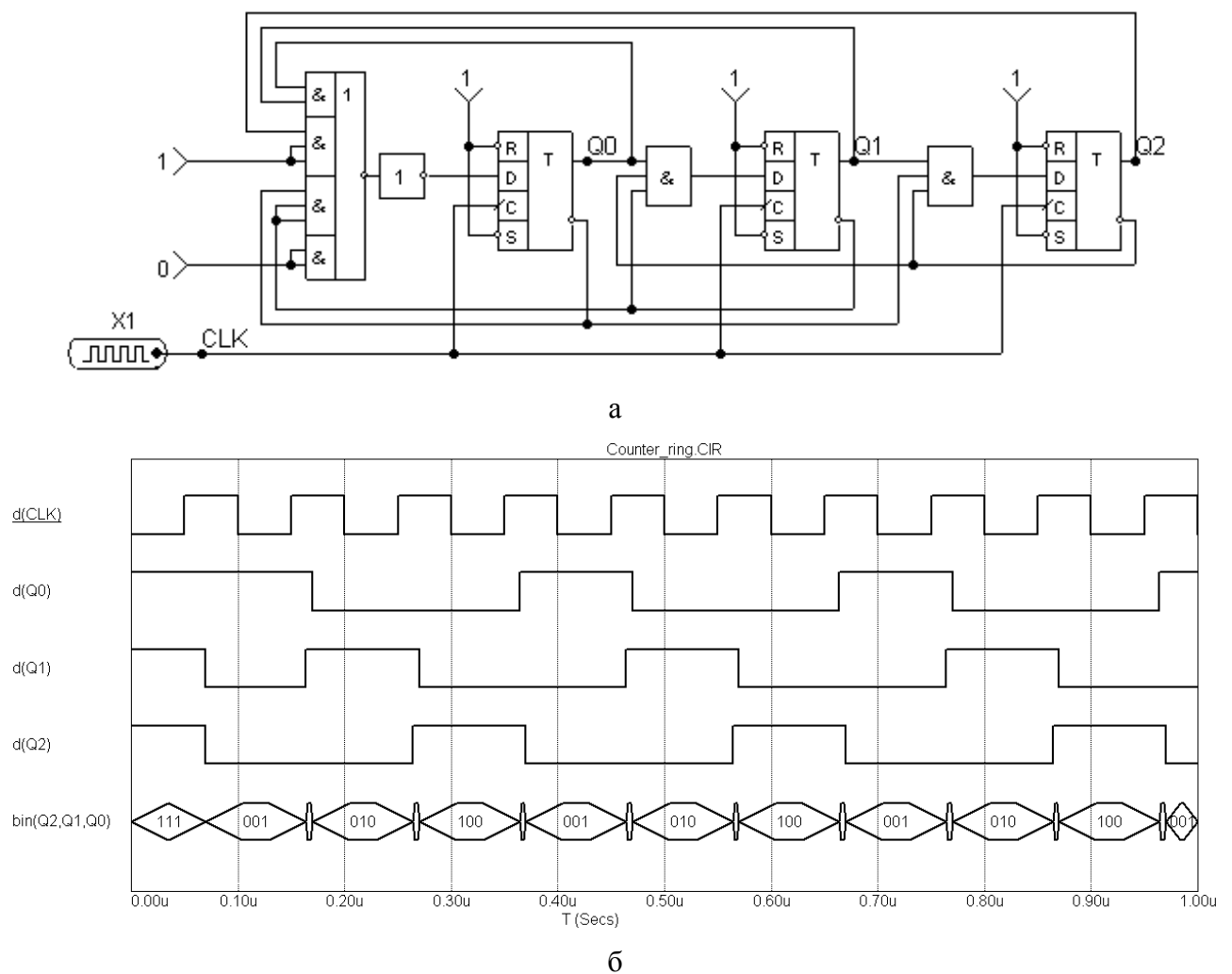


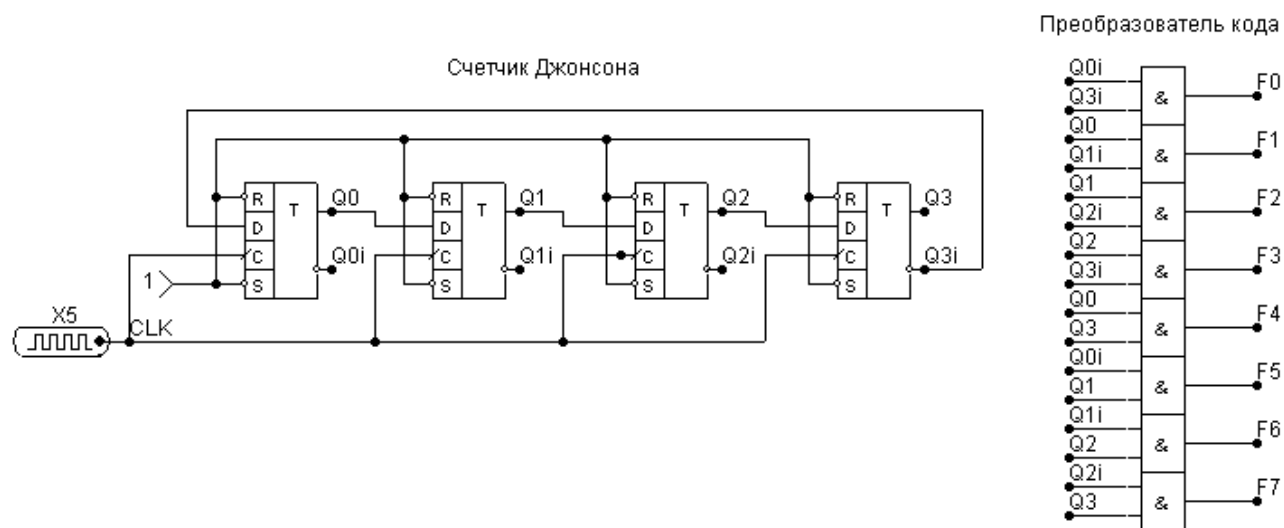
Рисунок 2.60 — Распределитель импульсов с самовосстановлением

Счетчики в коде «1 из N» на основе счетчиков Джонсона

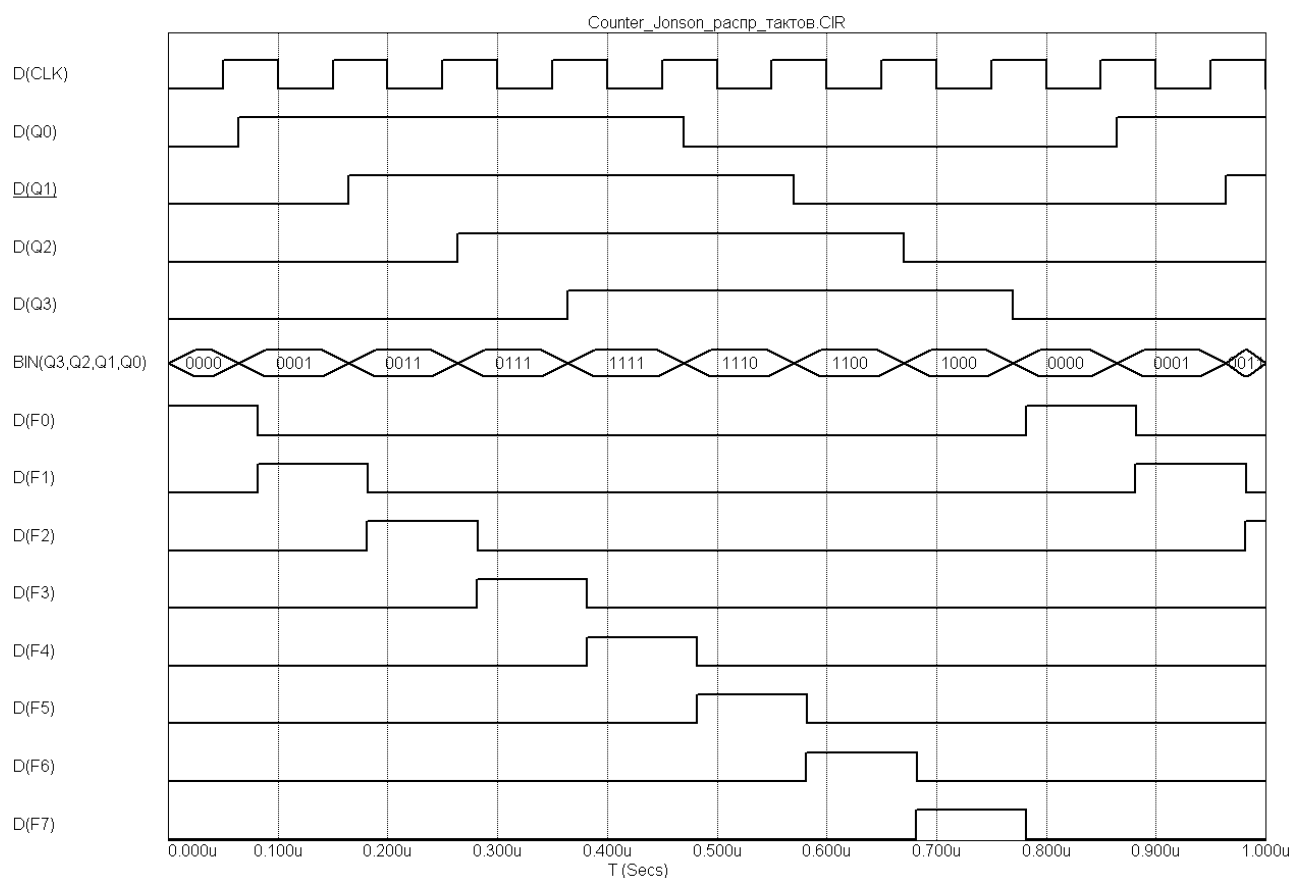
Кольцевой регистр с перекрестной обратной связью (счетчик Джонсона, счетчик Мебиуса, счетчик Либау-Крейга) имеет обратную связь на первый триггер от инверсии выходного сигнала (рис. 2.61, а). Он имеет $2n$ состояний, т. е. при той же разрядности вдвое больше, чем обычный кольцевой регистр. В то же время выход счетчика Джонсона представлен не в коде «1 из N», что требует преобразования кодов для получения выходов распределителя тактов. Такие преобразователи очень просты, что обуславливает применение счетчиков Джонсона в составе распределителей.

Показанный на рисунке четырехразрядный счетчик Джонсона при начальном нулевом состоянии работает следующим образом. Первый импульс сдвига C установит триггер нулевого разряда (первый слева) в единичное состояние ($Q_0=1$), т.к. $\overline{Q_3}=1$, в остальных разрядах будут нули как результат сдвига нулей от соседних слева разрядов. Второй импульс сдвига сохраняет единичное состояние триггера нулевого разряда, т. к. по-прежнему $\overline{Q_3}=1$. Первый разряд окажется в единичном состоянии ($Q_1=1$), поскольку примет единицу от триггера нулевого разряда. Остальные разряды будут нулевыми. Последующие сдвиги приведут к заполнению единицами всех разрядов счетчика, т. е. «волна еди-

ниц», распространяясь слева направо, приведет счетчик в состояние 1111. Следующий импульс сдвига установит триггер нулевого разряда в ноль, т. к. теперь $\overline{Q}_4 = 0$. Этим начинается процесс распространения «волны нулей». После восьми импульсов повторится состояние 0000, с которого было начато рассмотрение работы счетчика. Временные диаграммы описанных процессов показаны на рис. 2.61, б.



а



б

Рисунок 2.61 — Распределитель тактов на основе счетчика Джонсона

Особенность рассмотренной схемы четное число состояний при любом n ($2n$ — всегда число четное). Обычный кольцевой регистр такого ограничения не имеет.

Преобразование выходного кода счетчика Джонсона в код «1 из N» требует добавления всего одного двухвходового элемента И либо И-НЕ для каждого выхода распределителя тактов. Принцип дешифрации состоит в выявлении положения характерной координаты временной диаграммы — границы между зонами единиц и нулей. В двух случаях (для слов, состоящих только из нулей или только из единиц; состояние выявляется анализом крайних разрядов. В остальных случаях анализируются разряды на границе зоны единиц и нулей.

Как видно из таблицы, преобразование выходного кода счетчика Джонсона в код «1 из N» осуществляется согласно выражениям:

$$\begin{aligned} F_0 &= \overline{Q_0} \overline{Q_3}, \quad F_1 = Q_0 \overline{Q_1}, \quad F_2 = Q_1 \overline{Q_2}, \quad F_3 = Q_2 \overline{Q_3}, \\ F_4 &= Q_0 Q_3, \quad F_5 = \overline{Q_0} Q_1, \quad F_6 = \overline{Q_1} Q_2, \quad F_7 = \overline{Q_2} Q_3, \end{aligned}$$

где F_i ($i=0...7$) — выходы распределителя тактов.

По полученным выражениям строится дешифратор (рис. 2.61, а — справа).

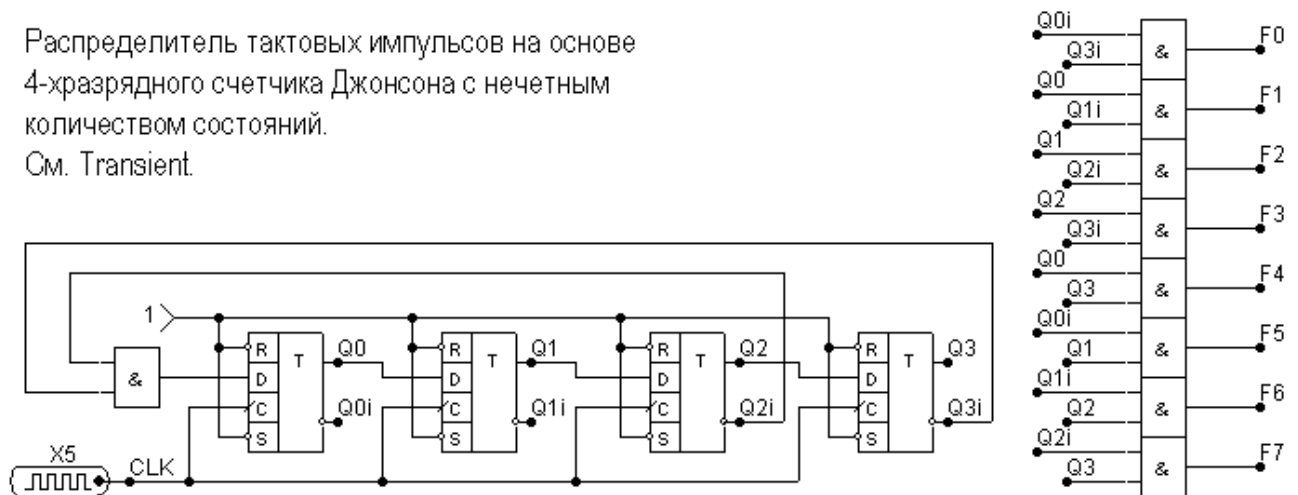
Распределитель тактов в целом (рис. 2.61, б, диаграммы для сигналов F_i) имеет выходные сигналы F_i в коде «1 из N».

Для схем со счетчиками Джонсона могут возникнуть вопросы преодоления ограничения обязательной четности числа состояний счетчика и обеспечения автоматического вхождения его в рабочий цикл (свойства самозапуска).

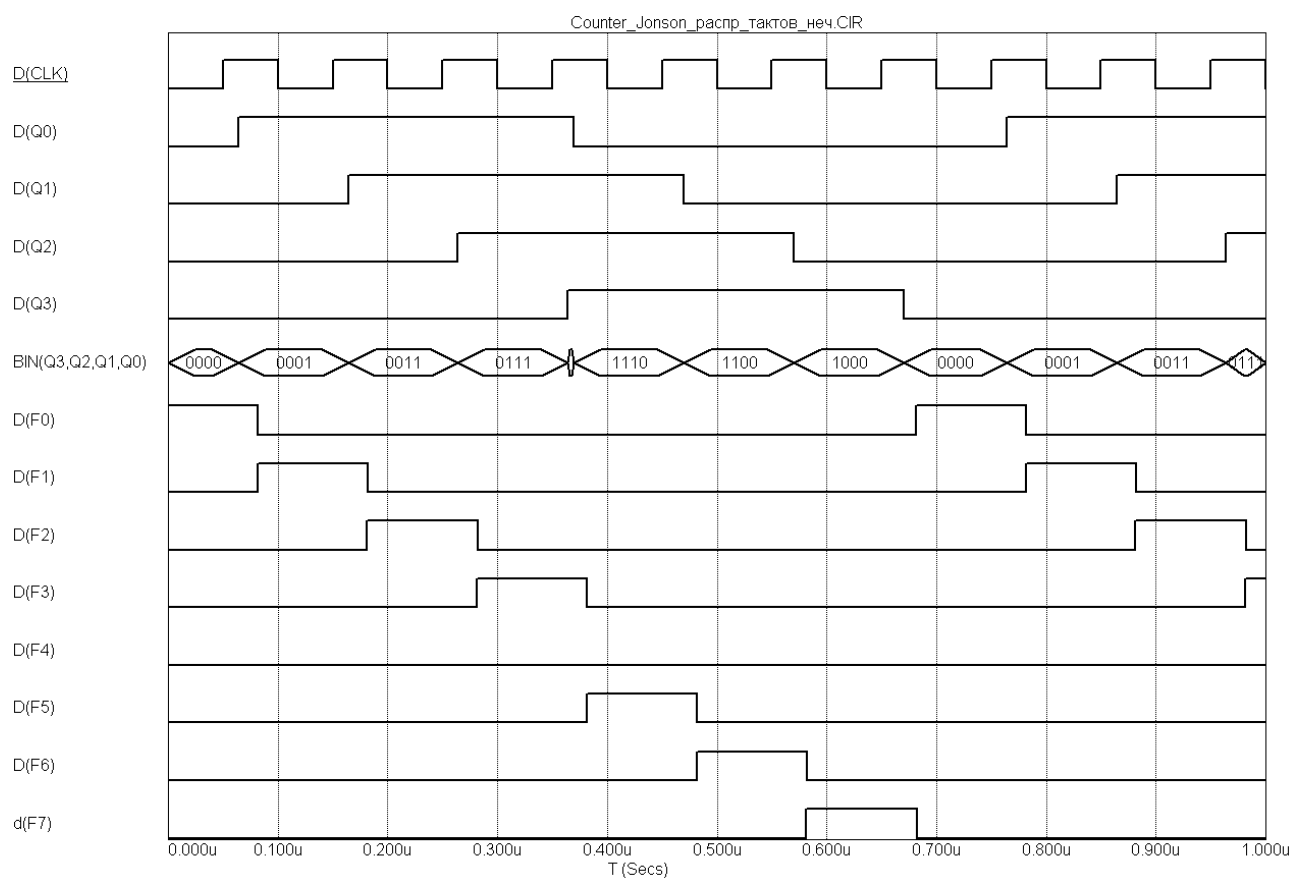
Первую задачу можно решить в рамках подхода, применявшегося при построении счетчиков с произвольным модулем, т. е. исключением одного «лишнего» состояния.

Получить схему с исключенным состоянием можно, уже не раз показанным способом, переходя от таблицы функционирования к функциям возбуждения триггеров и далее к схеме. Однако в данном случае нетрудно сократить этот путь, пользуясь простыми рассуждениями. Пусть исключению подлежит состояние 11...11. Чтобы его исключить, нужно перейти к следующему состоянию не от состояния «все единицы», а от предыдущего состояния 01...1, которое создает единицу в предпоследнем разряде счетчика, т.е. ноль на инверсном выходе этого разряда. Подачей этого нулевого сигнала на вход счетчика вместе с основным сигналом обратной связи через конъюнктор (см. рис. 1.40, а), исключается состояние 11...11 и в результате получается счетчик с нечетным числом состояний $2n-1$ (в примере на рис. 2.62 с числом состояний равным 7).

Распределитель тактовых импульсов на основе
4-хразрядного счетчика Джонсона с нечетным
количеством состояний.
См. Transient.



а



б

Рисунок 2.62 — Распределитель тактов с нечетным числом состояний на основе счетчика Джонсона

Задача обеспечения вхождения распределителя на основе счетчика Джонсона в рабочий цикл связана с тем, что базовая схема, рассмотренная выше, свойством самозапуска не обладает. Например, распределитель с трехразрядным счетчиком Джонсона (рис. 2.63, а) имеет общее число возможных состояний $2^3=8$, а число состояний в рабочем цикле $2 \cdot 3=6$. Неиспользуемыми являются два состояния: 010 и 101. Нетрудно видеть, что из состояния 010 счетчик перейдет в состояние 101, а из состояния 101 в состояние 010. Таким образом, на-

ряду с замкнутым рабочим циклом существует и замкнутый цикл из двух неиспользуемых состояний, попав в который, схема без постороннего воздействия не сможет перейти в рабочий цикл (рис. 2.63, б).

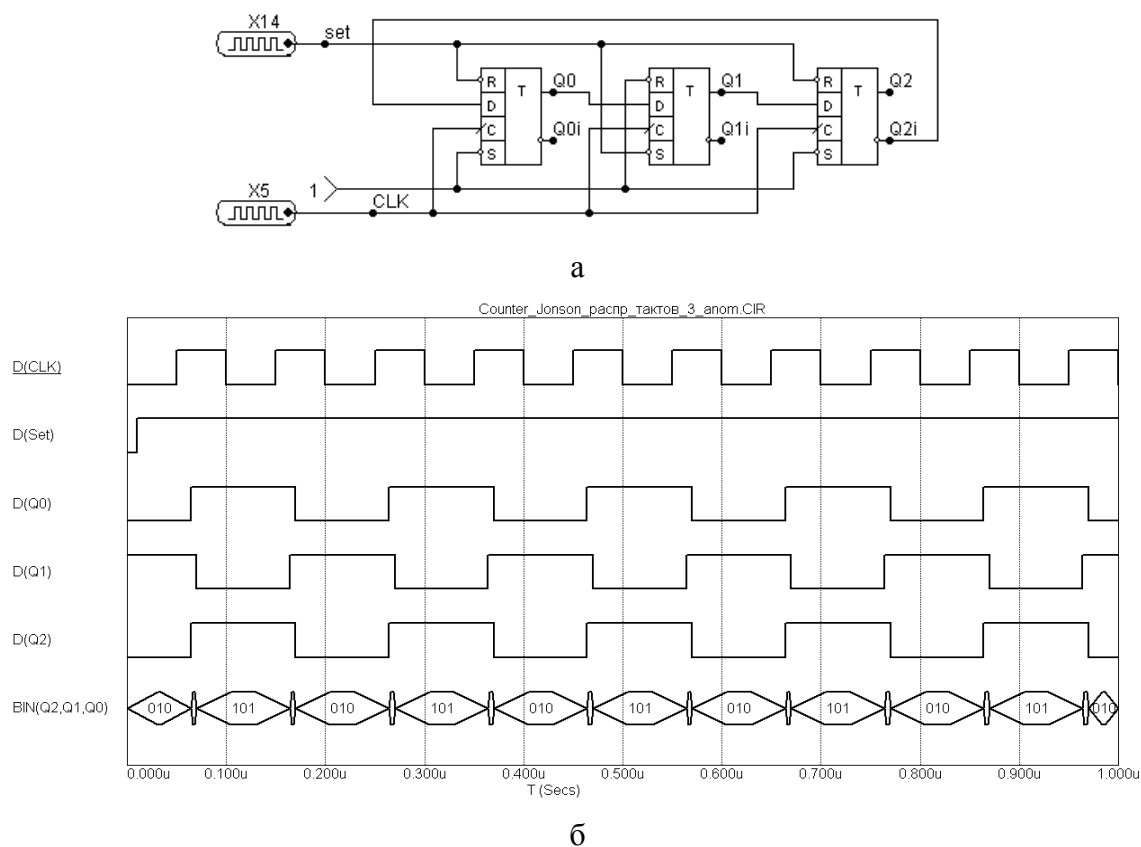


Рисунок 2.63 — Попадание счетчика Джонсона в цикл неиспользуемых состояний

Чтобы придать схеме свойство самозапуска, нужно модифицировать сигнал обратной связи, поступающий на вход счетчика. Понятно, что это можно сделать многими путями, поскольку траектория перехода из замкнутого цикла неиспользуемых состояний в рабочий неоднозначна.

Распределители на основе счетчиков Джонсона характеризуются небольшими аппаратными затратами (1/2 триггера и один двухвходовой вентиль на канал) и достаточно высоким быстродействием (время установления — сумма задержек переключения триггера и вентиля). Счетчики Джонсона реализованы, в частности, в сериях элементов типа КМОП (микросхемы ИЕ9 и ИЕ19 серии К561 и др.), причем одной из причин их применения является отсутствие импульсов помех в выходном напряжении и пониженный уровень токовых импульсов в цепях питания, создаваемых микросхемами. Распределитель в целом реализован в ИС К561ИЕ8.

Следует заметить, что распределители могут быть получены без применения специализированных схем в виде сочетания обычного двоичного счетчика и дешифратора. Такое решение наиболее очевидно. При большом числе выходных каналов эта структура может выигрывать у других, но при малом числе каналов преимущество по аппаратной сложности и быстродействию, как правило, оказывается на стороне вариантов с кольцевыми регистрами или счетчиками Джонсона.

Таблица 2.11 — Интегральные микросхемы счетчиков

Тип микро- схем	Коли- чество разря- дов	$K_{сч}$	Макси- мальная тактовая частота, МГц	Наличие входов предвари- тельной установ- ки-сброса	Потреб- ляе- мый ток, мА	Двоично- десятичный код	Возмож- ность ре- версиро- вания	Дополнитель- ные данные
K155IE2	4	2, 5, 10	10	Нет	53	8-4-2-1	Нет	
K155IE4	4	2, 6, 12	10	Нет	51	—	Нет	
K155IE5	4	2, 8, 16	10	Нет	53	—	Нет	
K155IE6	4	10	25	Есть	102	8-4-2-1	Есть	
K155IE7	4	16	25	Есть	102	—	Есть	
K155IE8	6	$K_{сч \max} = 64$		Есть	120	—	Нет	Программи- руемый
K155IE9	4	$K_{сч \max} = 10$	25	Есть	94	—	Нет	Программи- руемый
K555IE10	4	$K_{сч \max} = 10$	25	Есть	32	—	Нет	Программи- руемый
K155IE14	4	2, 5, 10	25	Есть		8-4-2-1	Нет	
K155IE15	4	2, 8, 16		Есть		—	Нет	
K155IE16	4	10		Есть		8-4-2-1	Есть	
K155IE17	4	16		Есть		—	Есть	
K155IE18	4	16		Есть	101	—	Нет	
K531IE14	4	2, 5, 10	40 (80)	Есть	88	8-4-2-1	Нет	
K531IE16	4	2, 5, 10	40 (80)	Есть		8-4-2-1	Есть	
K531IE17	4	16	40 (80)	Есть		—	Есть	
K555IE18	4	$K_{сч \max} = 10$	25	Есть	32	—	Нет	Программи- руемый
K176IE2	5	10, 32	2	Есть				
K561IE8 (176IE8)		10	2	Нет		—		Счетчик Джонсона
K561IE9 (564IE9)		8	2	Нет				Счетчик Джонсона
K561IE10	4	16	4	Нет				
K561IE11	4	16		Есть		—	Есть	
K561IE14 (564IE14)	4	10, 16	2	Есть			Есть	
K564IE15	—	$3 \div 15999$ ($3 \div 21327$)	2	Есть		—	Нет	Программи- руемый
K561IE16 (564IE16)	14	16384	3	Есть			Нет	
K561IE19	5	10	2	Есть		—	Нет	Счетчик Джонсона
500IE136	4	16	100	Есть		—	Есть	
500IE137	4	10	100	Есть		—	Есть	
K1500IE136	4	16	450	Есть		—	Есть	
K1500IE137	4	10	450	Есть		—	Есть	

2.9 Регистры и регистровые файлы

Регистры — самые распространенные узлы цифровых устройств. Они оперируют с множеством связанных переменных, составляющих слово. Над словами выполняется ряд операций: прием, выдача, хранение, сдвиг в разрядной сетке, поразрядные логические операции.

Регистры состоят из разрядных схем, в которых имеются триггеры и, чаще всего, также и логические элементы.

По количеству линий передачи переменных регистры делятся на однофазные и парафазные, по системе синхронизации — на *однотактные, двухтактные и многотактные*. В *парафазных* регистрах информация записывается и считывается и в прямом (Q), и в обратном (\bar{Q}) кодах. В *однофазных* регистрах информация записывается и считывается либо в прямом (Q), либо в обратном (\bar{Q}) коде.

Однако главным классификационным признаком является способ приема и выдачи данных. По этому признаку различают параллельные (статические) регистры, последовательные (сдвигающие) и параллельно-последовательные.

В *параллельных* регистрах прием и выдача слов производятся по всем разрядам одновременно. В них хранятся слова, которые могут быть подвергнуты поразрядным логическим преобразованиям.

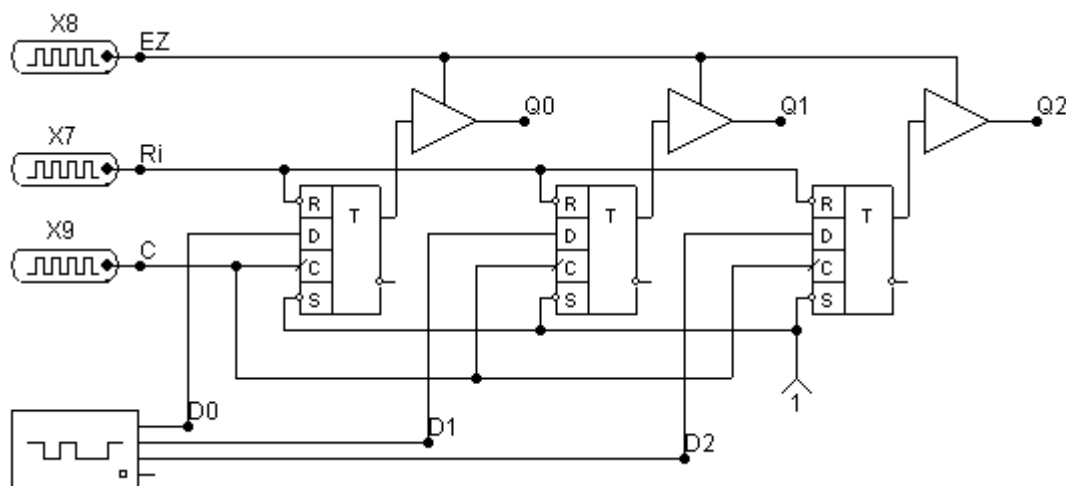
В *последовательных* регистрах слова принимаются и выдаются разряд за разрядом. Их называют сдвигающими, т. к. тактирующие сигналы при вводе и выводе слов перемещают их в разрядной сетке. Сдвигающий регистр может быть *нереверсивным* (с однонаправленным сдвигом) или *реверсивным* (с возможностью сдвига в обоих направлениях).

Последовательно-параллельные регистры имеют входы-выходы одновременно последовательного и параллельного типа. Имеются варианты с последовательным входом и параллельным выходом (SIPO, Serial Input — Parallel Output), параллельным входом и последовательным выходом (PISO), а также варианты с возможностью любого сочетания способов приема и выдачи слов.

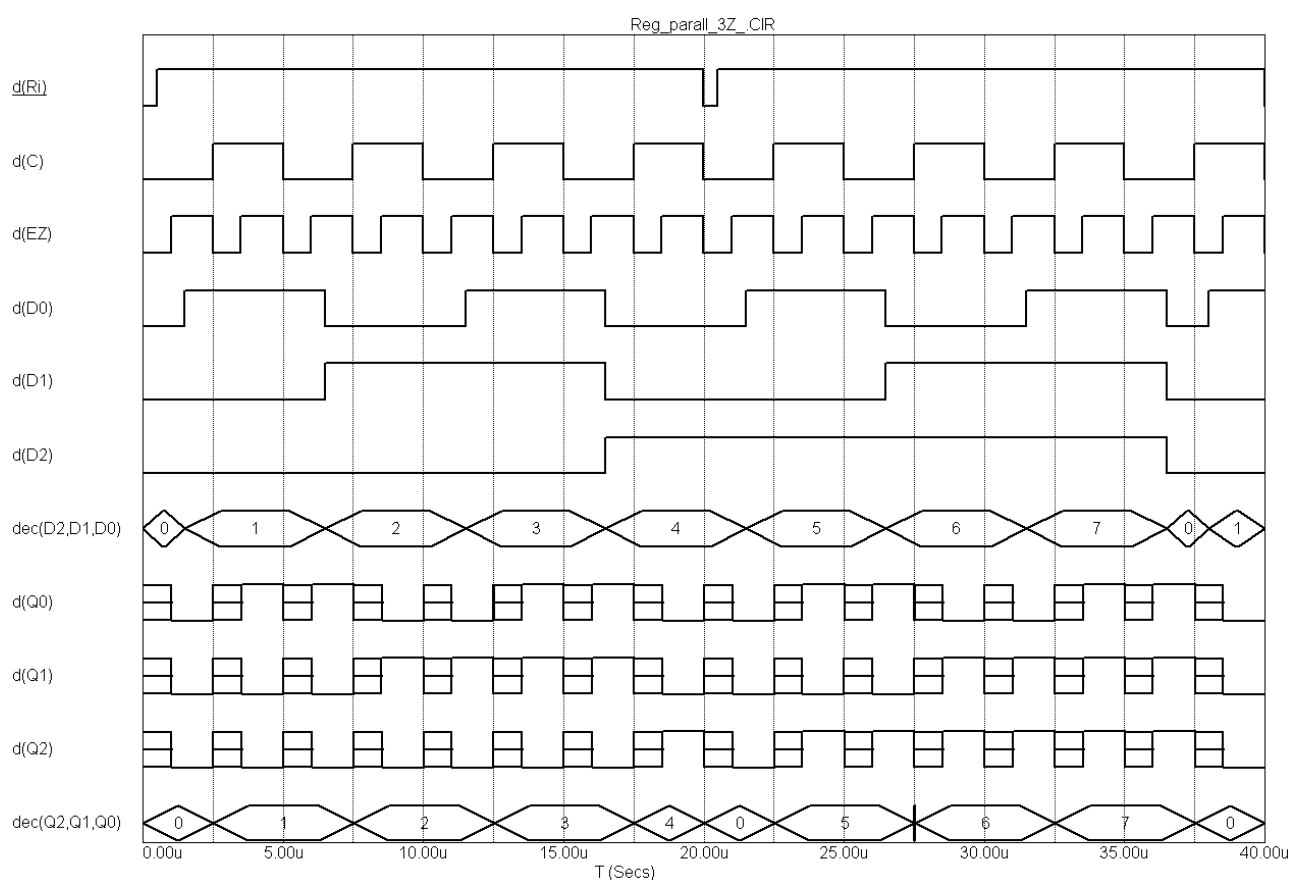
2.9.1 Параллельные регистры

В параллельных (статических) регистрах схемы разрядов не обмениваются данными между собой. Общими для разрядов обычно являются цепи тактирования, сброса/установки, разрешения вывода или приема, т. е. цепи управления. Пример схемы статического регистра, построенного на триггерах типа D с прямыми динамическими входами, имеющего входы сброса R и выходы с третьим состоянием, управляемые сигналом EZ, показан на рис. 2.64.

Для современной схемотехники характерно построение регистров именно на D-триггерах, преимущественно с динамическим управлением. Многие имеют выходы с третьим состоянием, некоторые регистры относятся к числу буферных, т. е. рассчитаны на работу с большими емкостными и/или низкоомными активными нагрузками. Это обеспечивает их работу непосредственно на магистраль (без дополнительных схем интерфейса).



а



б

Рисунок 2.64 — Параллельный регистр с 3-мя состояниями выходов, тактируемый фронтом:
а — схема; б — временные диаграммы

Подобную структуру (см. рис. 2.64) имеет параллельный регистр с 3-мя состояниями выхода КР1533ИР38 (74ALS874В), за исключением того, что выходы разрешаются низким уровнем (а не высоким как на рис. 2.64).

2.9.2 Регистровые файлы

Из статических регистров составляются блоки регистровой памяти — регистровые файлы. В микросхеме типа ИР26 (серии КР1533, К555 и др., зарубежный аналог 74LS670) можно хранить 4 четырехразрядных слова с возмож-

ностью независимой и одновременной записи одного слова и чтения другого. Информационные входы регистров соединены параллельно (рис. 2.65, а). Входы адресов записи WA и WB (от Write) дают четыре комбинации, каждая из которых разрешает «защелкнуть» данные, присутствующие в настоящее время на выводах $D_1...D_4$. Содержимое файла (регистра) вызывается на выходы блока $Q_1...Q_4$ с помощью дешифратора считывания (адресных входов мультиплексора) адресами RA и RB (от английского Read). Таких адресов также четыре.

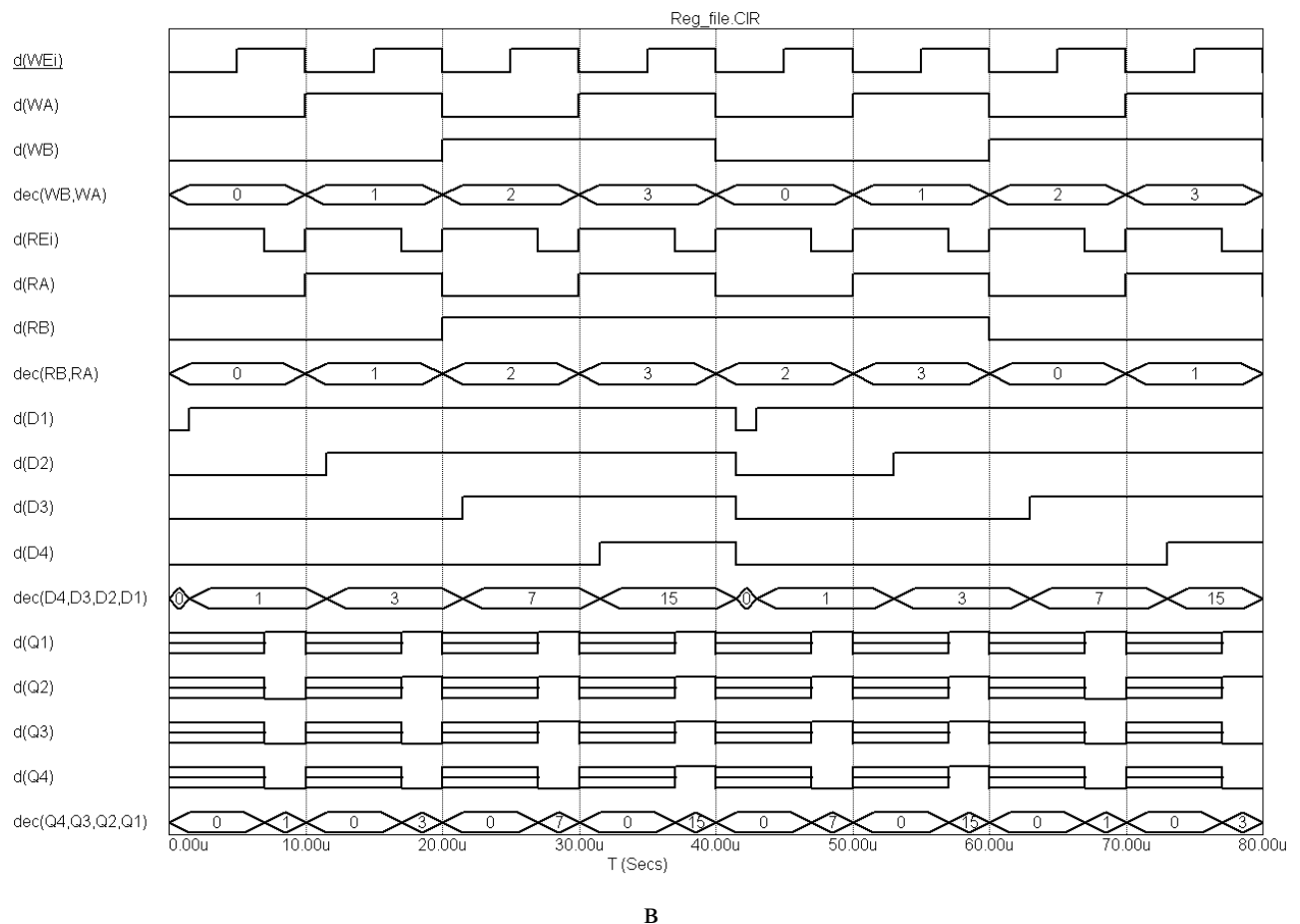
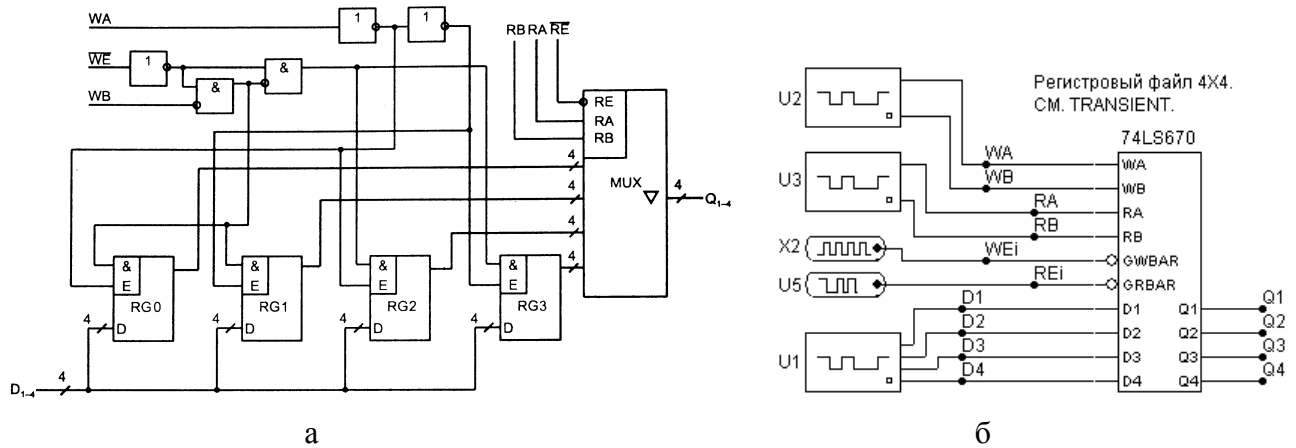


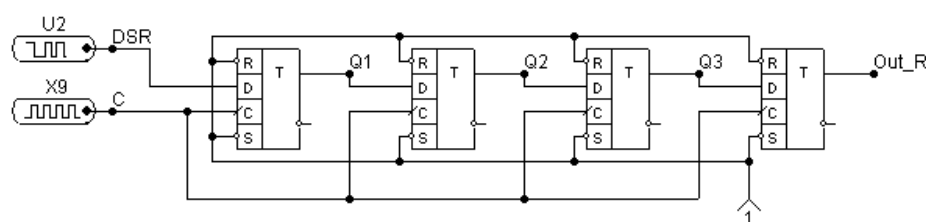
Рисунок 2.65 — Регистровый файл КР1533ИР26: а — внутренняя структура; б — схема для моделирования в среде Micro-CAP; в — временные диаграммы моделирования

Если на входе разрешения записи WE (Write Enable) действует активный низкий уровень, то данные поступают в соответствующий регистр, при высоком уровне WE входы для данных и адресов запрещены. Выходные данные выдаются в прямом коде. Внутренние D-триггеры регистрового файла представляют собой защелки Latch (т.е. информация записывается в них по высокому уровню синхросигнала, а по отрицательному фронту синхроимпульса — защелкивается).

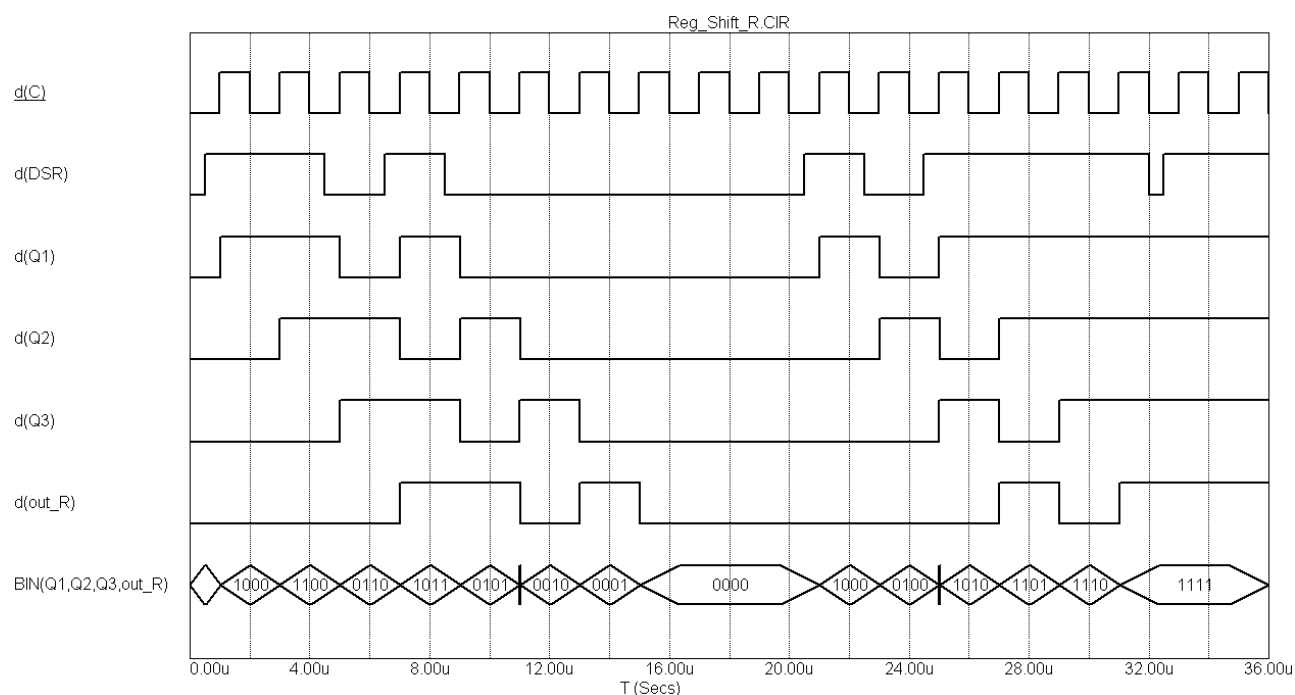
Размерность регистровой памяти можно наращивать, составляя из нескольких ИС блок памяти. При наращивании числа хранимых слов выходы отдельных ИС с тремя состояниями соединяются в одной точке. Допускается соединять непосредственно до 128 выходов, что дает 512 хранимых слов. Ограничение на число соединяемых в одной точке выходов вызвано токовым режимом выхода, оно может быть преодолено при подключении к выходной точке специальных внешних резисторов. При наращивании разрядности слова соединяют параллельно входы разрешения и адресации нескольких ИС, выходы которых в совокупности дают единое информационное слово.

2.9.3 Сдвигающие регистры

Последовательные (сдвигающие) регистры представляют собою цепочку разрядных схем, связанных цепями переноса.



а



б

Рисунок 2.66 — Регистр сдвига вправо на D-триггерах, управляемых фронтом

В одноктактных регистрах со сдвигом на один разряд вправо (см. рис. 2.66, а) слово сдвигается при поступлении синхросигнала. Вход и выход последовательные (DSR — Data Serial Right). На рис. 2.67, а показана схема регистра со сдвигом влево (вход данных DSL — Data Serial Left), а на рис. 2.68, а иллюстрируется принцип построения реверсивного регистра, в котором имеются связи триггеров с обоими соседними разрядами, но соответствующим сигналом R/L разрешается работа только одних из этих связей.

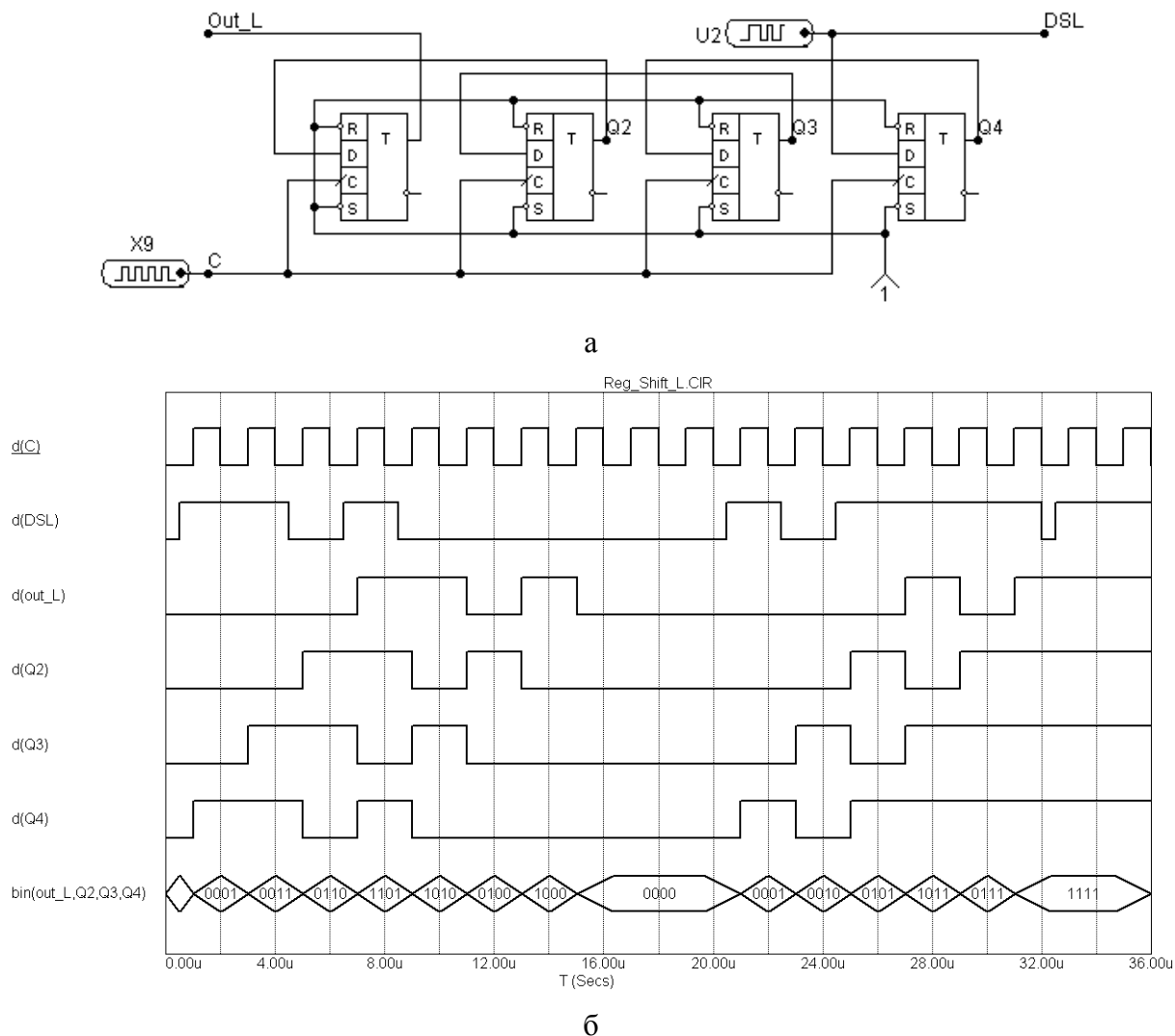
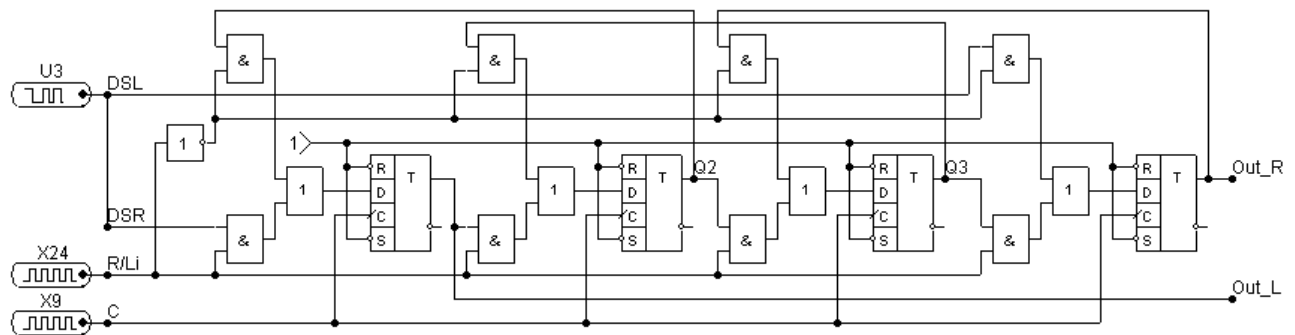


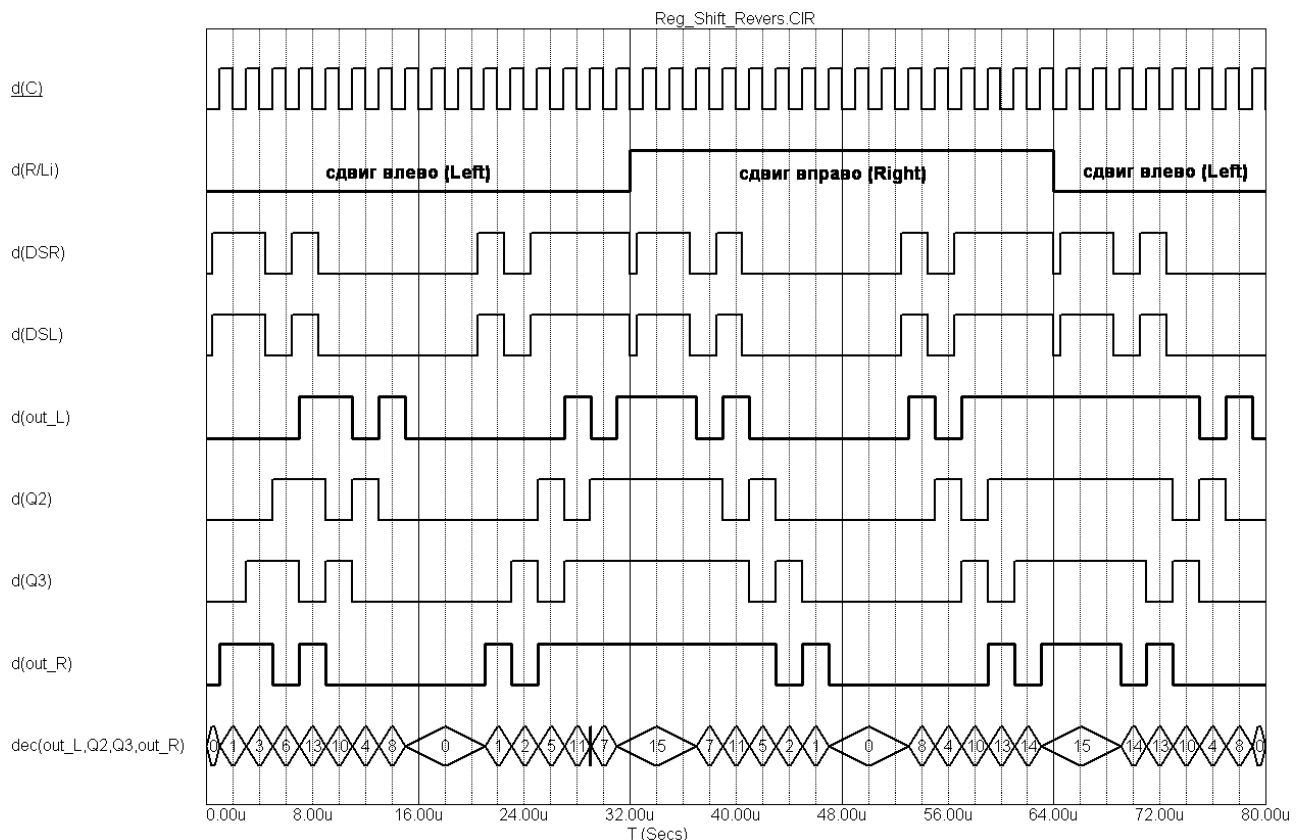
Рисунок 2.67 — Регистр сдвига влево на D-триггерах, управляемых фронтом

Согласно требованиям синхронизации, рассмотренным в предыдущем параграфе, в сдвигающих регистрах, не имеющих логических элементов в межразрядных связях, нельзя применять одноступенчатые триггеры, управляемые уровнем, поскольку некоторые триггеры могут за время действия разрешающего уровня синхросигнала переключиться неоднократно, что недопустимо. В сдвигающих регистрах могут использоваться триггеры с динамическим управлением или двухступенчатые.

Появление в межразрядных связях логических элементов и, тем более, логических схем неединичной глубины упрощает выполнение условий работоспособности регистров и расширяет спектр типов триггеров, пригодных для этих схем.



а



б

Рисунок 2.68 — Реверсивный сдвиговый регистр на D-триггерах, управляемых фронтом

Многотактные сдвигающие регистры управляются несколькими синхро-последовательностями. Из их числа наиболее известны двухтактные с основным и дополнительным регистрами, построенными на простых одноступенчатых триггерах, управляемых уровнем. По такту С1 содержимое основного регистра переписывается в дополнительный, а по такту С2 возвращается в основной, но уже в соседние разряды, что соответствует сдвигу слова. По затратам оборудования и быстродействию этот вариант близок к одноклапному регистру с двухступенчатыми триггерами.

2.9.4 Универсальные регистры

В сериях ИС и библиотеках БИС/СБИС программируемой логики имеется много вариантов регистров (в схемотехнике ТТЛШ их около 30). Среди них многорежимные (многофункциональные) или универсальные, способные вы-

полнять набор микроопераций. Многорежимность достигается композицией в одной и той же схеме частей, необходимых для выполнения различных операций. Управляющие сигналы, задающие вид выполняемой в данное время операции, активизируют необходимые для этого части схемы.

Типичным представителем многорежимных регистров является микросхема ИР13 серии КР1533 и других (рис. 2.69). Это восьмиразрядный регистр с возможностью двусторонних сдвигов с допустимой тактовой частотой до 25 МГц при токе потребления до 40 мА. Имеет также параллельные входы и выходы, вход асинхронного сброса R и входы выбора режима S0 и S1, задающие четыре режима (параллельная загрузка, два сдвига — влево и вправо, и хранение). Функционирование регистра определяется табл. 2.12.

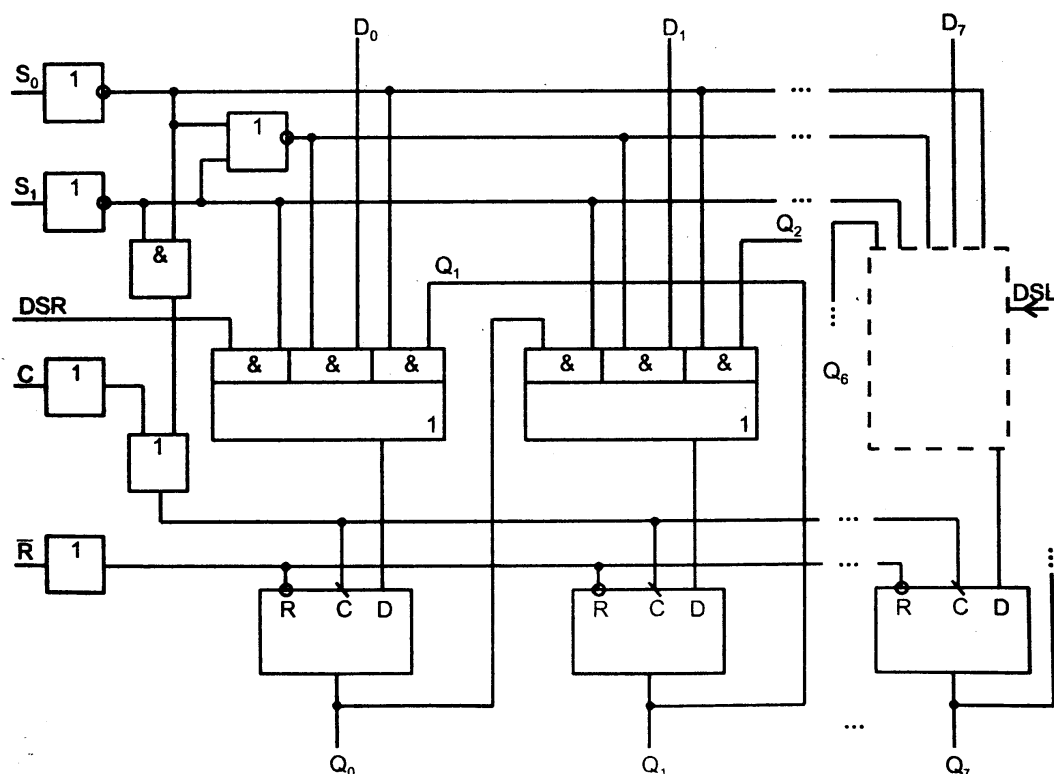


Рисунок 2.69 — Внутренняя структура универсального регистра ИР13

Таблица 2.12 — Функционирование многорежимного регистра ИР13

Режим	Входы							Выходы Q_{i+1}				
	C	\bar{R}	S1	S0	DSR	DSL	D_i	Q_0	Q_1	...	Q_6	Q_7
Сброс	X	L	X	X	X	X	X	L	L	...	L	L
Хранение	X	H	L	L	X	X	X	Q_0	Q_1	...	Q_6	Q_7
Сдвиг вправо (вниз)	┐	H	L	H	L	X	X	L	Q_0	...	Q_5	Q_6
		H	L	H	H	X	X	H	Q_0	...	Q_5	Q_6
Сдвиг влево (вверх)	┐	H	H	L	X	L	X	Q_1	Q_2	...	Q_7	L
		H	H	L	X	H	X	Q_1	Q_2	...	Q_7	H
Параллельная загрузка	┐	H	H	H	X	X	D_i	D_0	D_1	...	D_6	D_7

Примечание. Здесь сдвиг влево — от старшего к младшему, вправо — от младшего к старшему. Команды сдвига микропроцессоров и микроконтроллеров работают наоборот.

Условное обозначение регистра ИР13 на принципиальных схемах приведено на схеме для моделирования в среде Micro-CAP (рис. 2.70, а).

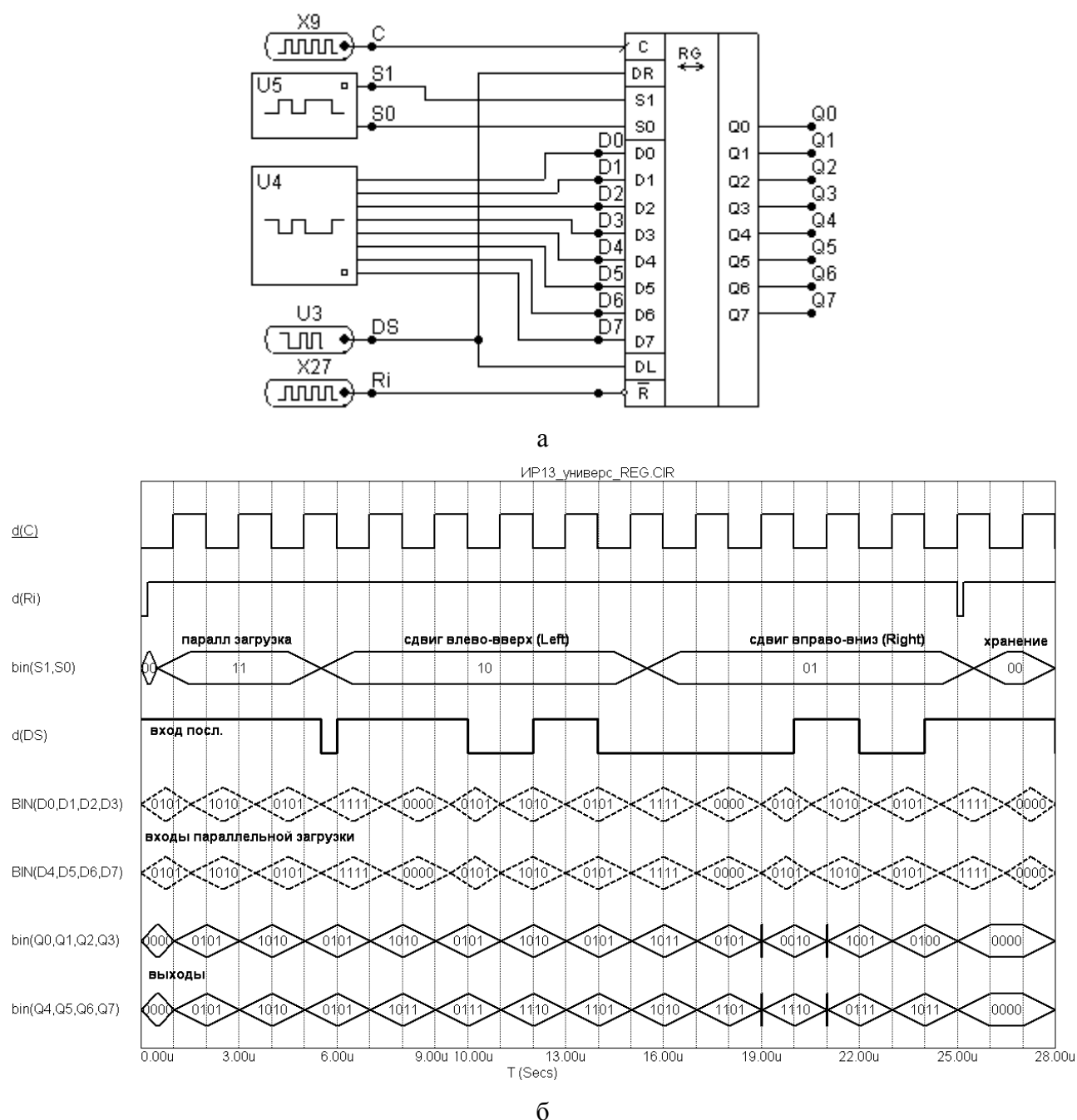
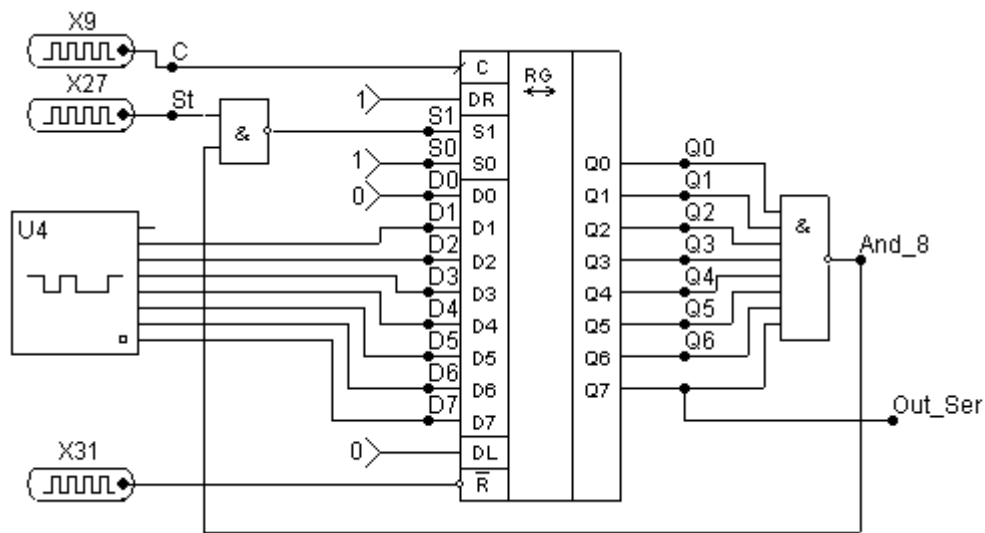
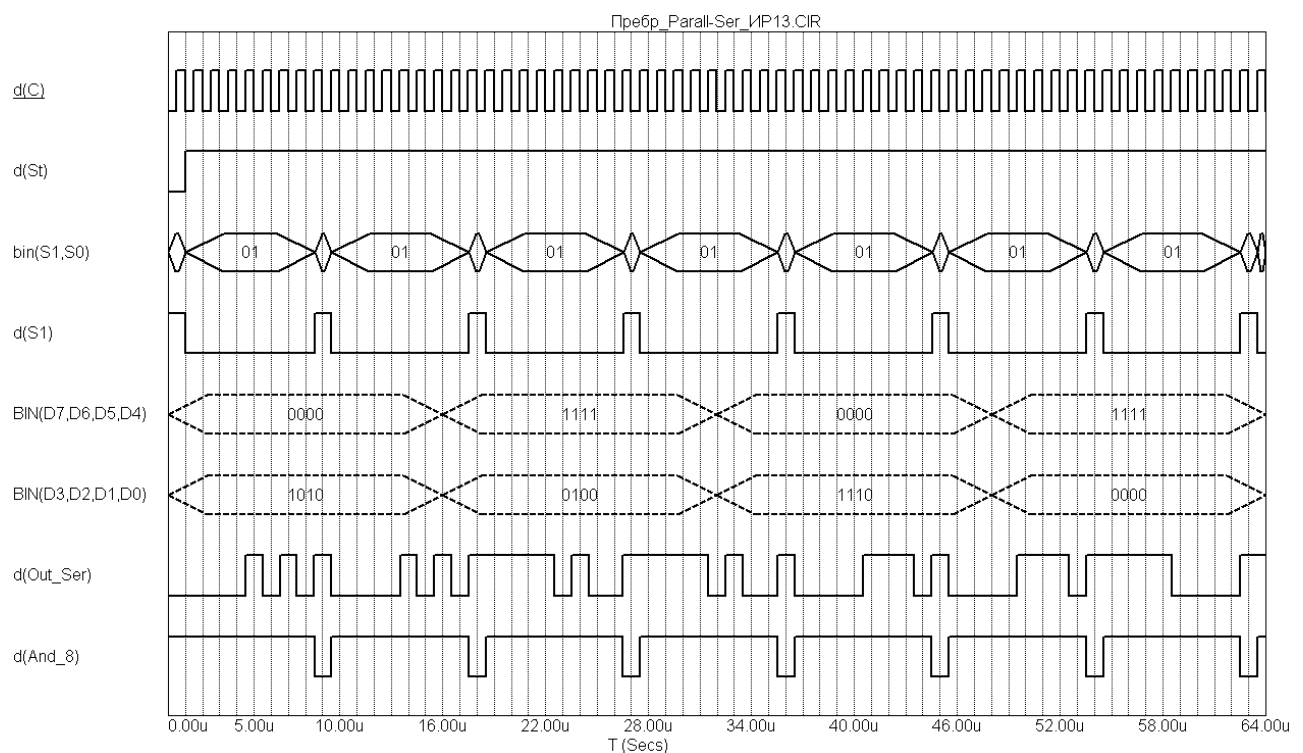


Рисунок 2.70 — Моделирование универсального регистра ИР13 в среде программы Micro-CAP

Регистры, имеющие разнотипные вход и выход, служат основными блоками преобразователей параллельных кодов в последовательные и обратно. На рис. 2.71 показана схема преобразователя параллельного кода в последовательный на основе восьмиразрядного регистра типа SI/PI/SO/PO (ИР13). В этой схеме отрицательный стартовый импульс St , задающий уровень логического нуля на верхнем входе двухвходового элемента И-НЕ, создает сигнал параллельного приема данных на входах $S_1S_0=11$ (см. табл. 2.12), по которому в разряды 1...7 регистра загружается преобразуемое слово D_1-D_7 , а в нулевой разряд — константа 0. На последовательный вход DR подана константа 1.



а



б

Рисунок 2.71 — Использование универсального регистра ИР13 в качестве преобразователя параллельного кода в последовательный: а — схема для моделирования; б — временные диаграммы

Таким образом, после загрузки в регистре формируется слово $0D_1...D_6D_7$. Тактовые импульсы, поступающие на вход C , вызывают сдвиги слова вправо (для условного обозначения это соответствует сдвигу вниз). Сдвиги выводят слово в последовательной форме через выход Q_7 (или узел схемы Out_Ser). Вслед за информационными разрядами идет ноль (константа "0"), после которого цепочка единиц ($DR=1$). Пока ноль не выведен из регистра, на выходе 8-входового элемента И-НЕ действует единичный сигнал. После вывода нуля все входы элемента становятся единичными, его выход (узел And_8) приобретает нулевое значение и через двухвходовой элемент И-НЕ формирует сигнал авто-

матической загрузки следующего слова ($S_I=1$), после чего цикл преобразования повторяется.

В перечне микроопераций, выполняемых регистрами, были указаны поразрядные логические операции. Современные регистры мало приспособлены для выполнения этих операций, однако при необходимости их можно выполнить, пользуясь регистрами на RS-триггерах. Для выполнения операции ИЛИ на S-входы статического регистра с исходным нулевым состоянием подается первое слово А, единичные разряды которого устанавливают соответствующие триггеры. Затем без сброса регистра на S-входы подается второе слово В. Ясно, что в результате получим побитную операцию «ИЛИ» над двумя словами $Q=A \vee B$.

При выполнении поразрядной операции И в первом такте на S-входы регистра подается слово А, устанавливающее те разряды регистра, а которых слово А имеет единицы. Затем следует подать на регистр слово В. Чтобы в регистре сохранились единицы только в тех разрядах, в которых единицы имеют оба слова, слово В подается на входы R триггеров в инверсном виде.

Сложение по модулю 2 может быть выполнено схемой с триггерами типа Т в разрядах путем последовательной во времени подачи на неё двух слов А и В.

Аббревиатуры цифровых сигналов, используемые в моделях последовательностных цифровых устройств программы Micro-CAP

...bar	активный низкий уровень входа или выхода(о)
Borrow	выход заема для счетчиков
Carry	выход переноса для счетчиков
Clk	вход тактовых импульсов
CS	вход выборки кристалла (корпуса микросхемы)
DOWN	вход счета на уменьшение
DS (DR, DL)	последовательные данные для сдвиговых регистров
Load	параллельная загрузка
Mode	режим работы
OE, outen	output enable, разрешение работы выходов ИМС для трехстабильных схем
R, Reset, Clear	вход сброса
R/W	выбор режима чтение/запись
Ser	последовательный вход
Shift	Сдвиг
Sh/Ld	сдвиг/параллельная загрузка
Strobe	строб, разрешение
UP	вход счета на приращение

2.10 Запоминающие устройства

3 СИНТЕЗ ЦИФРОВЫХ АВТОМАТОВ

3.1 Синтез асинхронных автоматов на RS-триггерах

3.1.1 Пример 1

На основе асинхронных RS-триггеров синтезировать JK-триггер (Master-slave), запоминающий состояние J и K входов при низком уровне синхроимпульса и меняющий состояние на выходе по положительному перепаду синхроимпульса (переход из низкого в высокое состояние).

Синтезируем устройство как асинхронный автомат, управляющими сигналами для которого являются уровень синхроимпульса ($0 \rightarrow \bar{C}$, $1 \rightarrow C$) и значения информационных сигналов на J и K входах (для J-входа $0 \rightarrow \bar{J}$, $1 \rightarrow J$; для K-входа $0 \rightarrow \bar{K}$, $1 \rightarrow K$).

Составим граф переходов разрабатываемого автомата (рис. 3.1), используя словесный алгоритм описания его работы, данный в техническом задании. Вершины графа (устойчивые состояния автомата) будем кодировать противоположно (с использованием кода Грея). Т.к. для формирования сигнала на выходе устройства необходим единичный уровень сигнала на синхровходе, а информация записывается при нулевом уровне сигнала C , вводятся промежуточные состояния на графе, переход в которые осуществляется отрицательным уровнем синхроимпульса \bar{C} . Таким образом, всего получается 4 устойчивых состояния, для кодирования которых необходимо 2 запоминающих элемента —

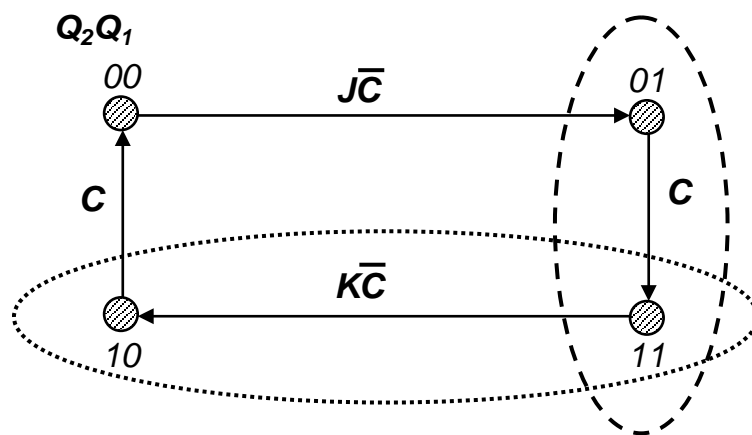


Рисунок 3.1 — Граф переходов асинхронного автомата (JK-триггера Master-Slave)

RS-триггера (см. рис. 3.1).

Охватим замкнутой линией все состояния на графе переходов, в которых значения одной и той же переменной (состояние одного элемента памяти) равны 1. Эти замкнутые кривые показаны штрих-пунктирной и пунктирной линиями соответственно.

Поскольку вход в подобную замкнутую область и выход из нее требуют линий сигналов возбуждения, обозначаются соответствующие сигналы возбуждения. Стрелками, входящими в область, указывают возбуждение установки (присваивающее переменной единичное значение), а стрелками, выходящими из области, — возбуждение сброса (присваивающее ей нулевое значение).

Составляются два выражения в форме ДНФ (суммы произведений): одно для функции возбуждения установки, а другое для функции возбуждения сброса. Каждое произведение должно содержать входные переменные (сигналы по которым совершается переход) и вторичные переменные, связанные с данным переходом, но не меняющие свои значения. В качестве вторичных переменных

выступают двоичные разряды кода состояния в коде Грея. Включение вторичных переменных гарантирует выполнение переходов в правильной последовательности.

$$S_1 = \overline{Q_2} \cdot J \cdot \overline{C}$$

$$R_1 = Q_2 \cdot K \cdot \overline{C};$$

$$S_2 = Q_1 \cdot C$$

$$R_2 = \overline{Q_1} \cdot C.$$

Строится схема на основе асинхронных RS-триггеров, реализующая полученные логические выражения. Для последующего моделирования с помощью программы Micro-CAP в качестве асинхронных RS-триггеров используются JK или D-триггера с асинхронными входами установки (PREBAR) и сброса (CLRBAR). При этом входы синхронизации и информационные входы не задействуются.

Схема для моделирования с обозначениями необходимых сигналов приведена на рис. 3.2, а, а на рис. 3.2, б приведены временные диаграммы работы автомата, доказывающие его работоспособность.

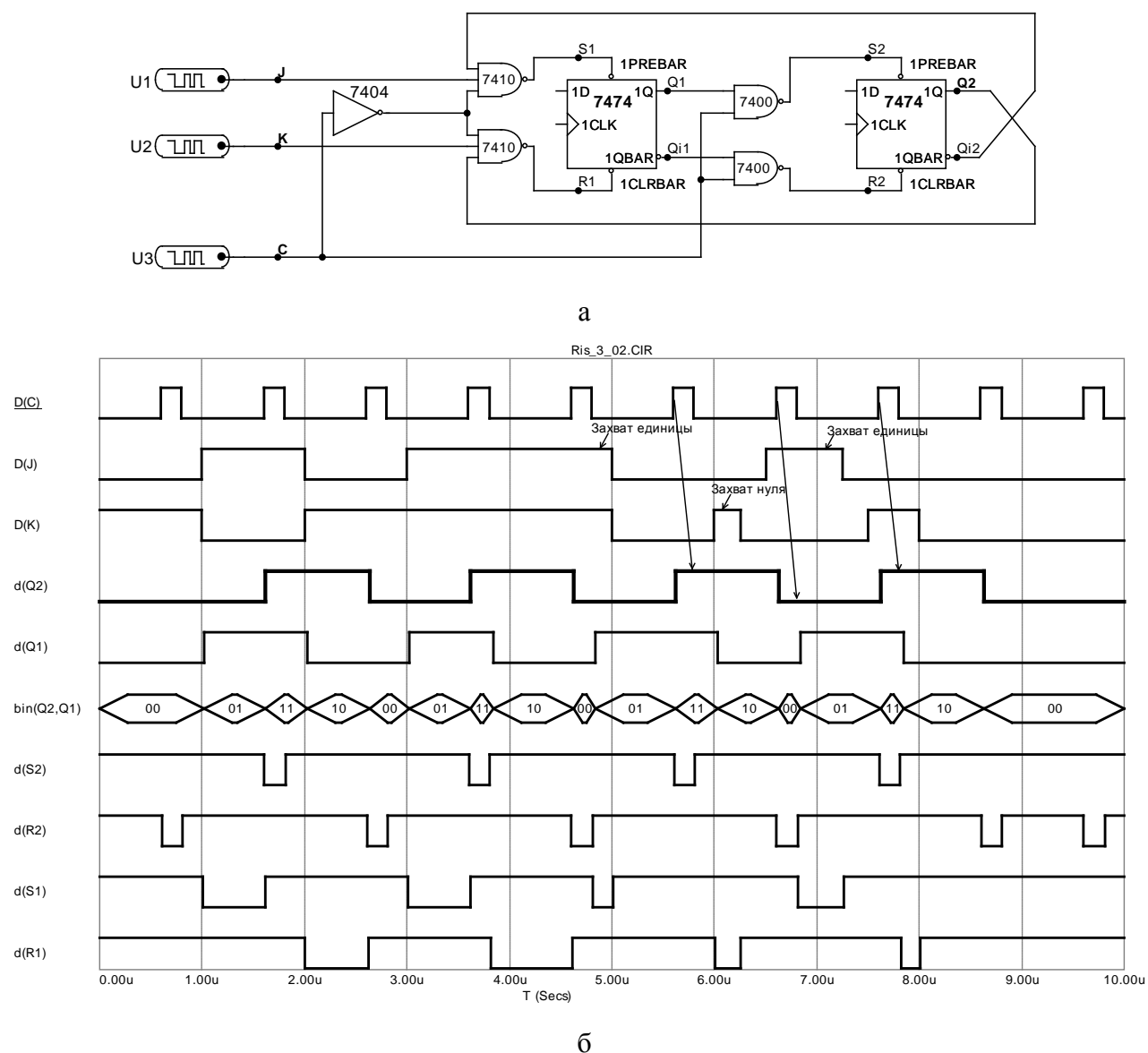


Рисунок 3.2 — Двухтактный JK-триггер, меняющий состояние по положительному фронту:
а — схема; б — временные диаграммы

3.1.2 Пример 2

На основе асинхронных RS-триггеров синтезировать JK-триггер, устанавливающийся в соответствии с управляющими сигналами на J и K входах по отрицательному перепаду синхроимпульса (переходу из высокого в низкое состояние).

Синтезируем устройство как асинхронный автомат, управляющими сигналами для которого являются уровень синхроимпульса ($0 \rightarrow \bar{C}$, $1 \rightarrow C$) и значение информационных сигналов на J и K входах (для J-входа $0 \rightarrow \bar{J}$, $1 \rightarrow J$; для K-входа $0 \rightarrow \bar{K}$, $1 \rightarrow K$).

Составим граф переходов разрабатываемого автомата (рис. 3.3), используя словесный алгоритм описания его работы, данный в техническом задании. Вершины графа (устойчивые состояния автомата) будем кодировать противоположно (с использованием кода Грея). Т.к. для формирования сигнала на вы-

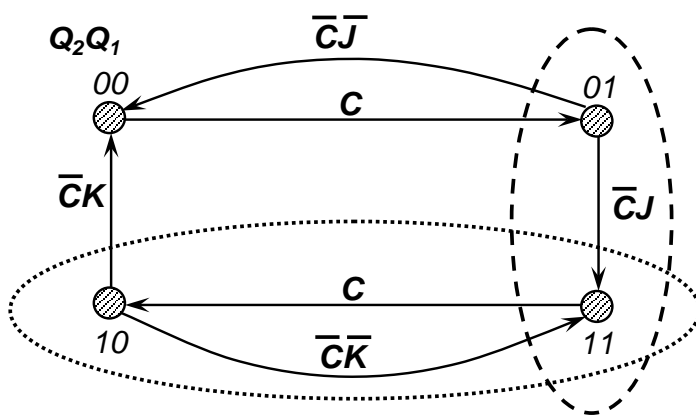


Рисунок 3.3 — Граф переходов асинхронного автомата (JK-триггера по отриц. фронт СИ).

ходе устройства необходимо выделить отрицательного фронта синхроимпульса, вводятся промежуточные состояния на графе, переход в которые осуществляется единичным уровнем синхроимпульса C . Таким образом, всего получается 4 устойчивых состояния, для кодирования которых необходимо 2 запоминающих элемента (RS-триггера).

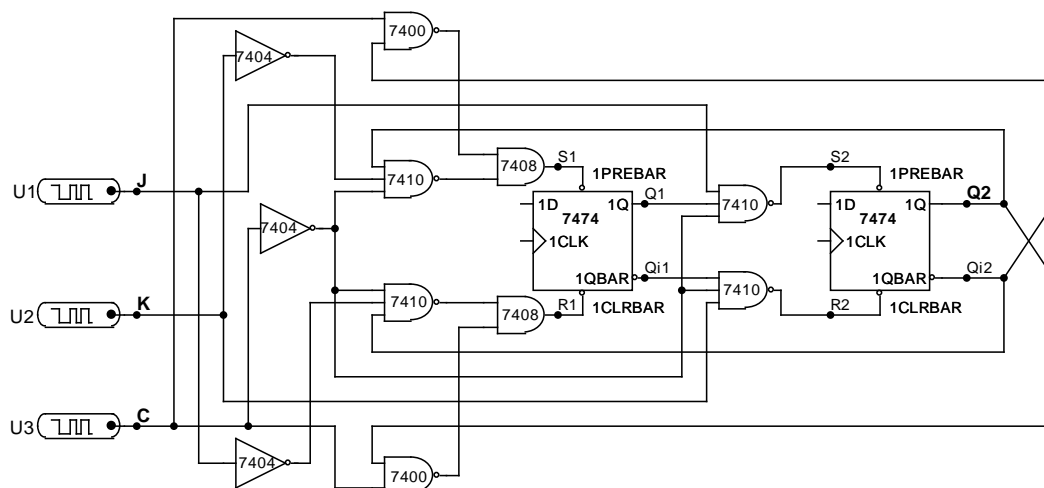
Охватим замкнутой линией все состояния на графе переходов, в которых значения одной и той же переменной (состояние одного элемента памяти) равны 1. Эти замкнутые кривые показаны штрих-пунктирной и пунктирной линиями соответственно.

Поскольку вход в подобную замкнутую область и выход из нее требуют линий сигналов возбуждения, обозначаются соответствующие сигналы возбуждения. Стрелками, входящими в область, указывают возбуждение установки (присваивающее переменной единичное значение), а стрелками, выходящими из области, — возбуждение сброса (присваивающее ей нулевое значение).

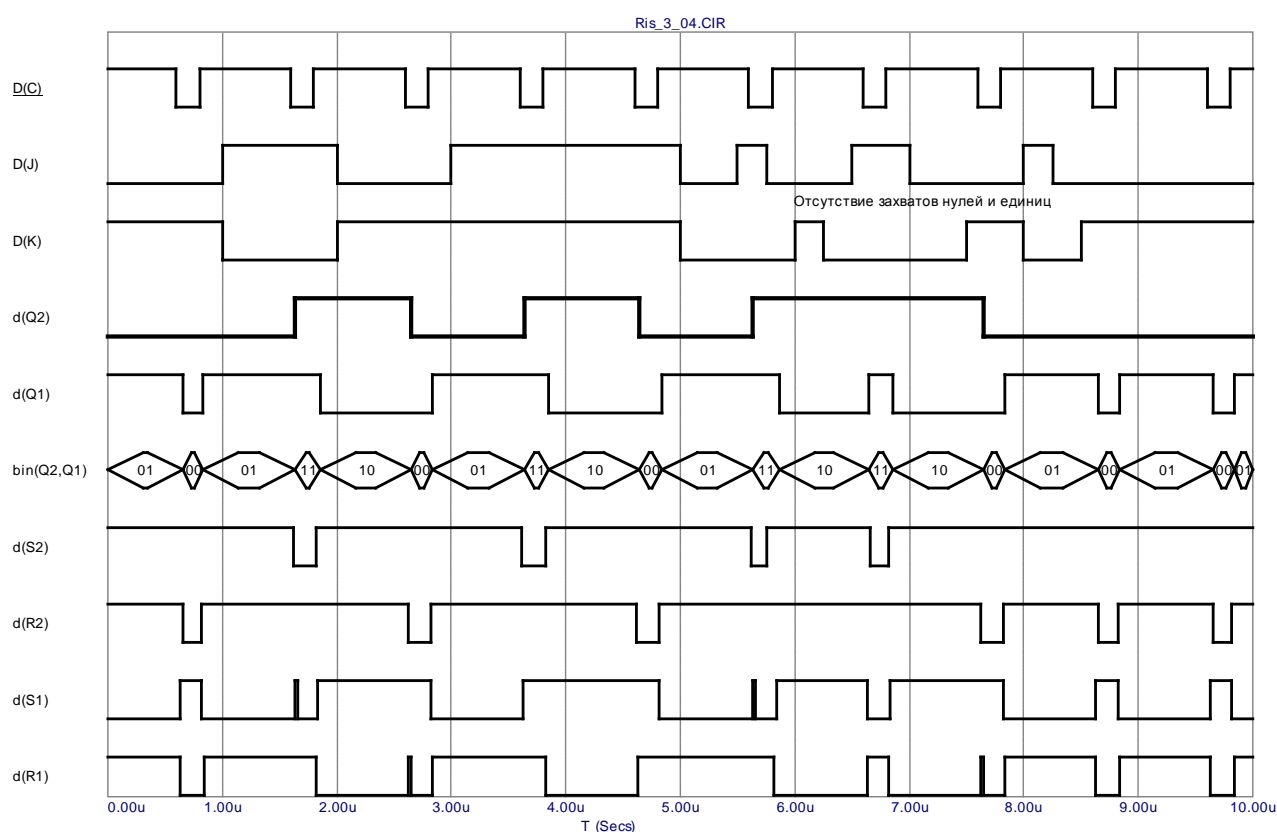
Составляются два выражения в форме ДНФ (суммы произведений): одно для функции возбуждения установки, а другое для функции возбуждения сброса. Каждое произведение должно содержать входные переменные (сигналы по которым совершается переход) и вторичные переменные, связанные с данным переходом, но не меняющие свои значения. В качестве вторичных переменных выступают двоичные разряды кода состояния в коде Грея. Включение вторичных переменных гарантирует выполнение переходов в правильной последовательности.

$$\begin{aligned} S_1 &= \bar{Q}_2 \cdot C \vee Q_2 \cdot \bar{C} \cdot \bar{K} \\ S_2 &= Q_1 \cdot \bar{C} \cdot J \end{aligned}$$

$$\begin{aligned} R_1 &= Q_2 \cdot C \vee \bar{Q}_2 \cdot \bar{C} \cdot \bar{J}; \\ R_2 &= \bar{Q}_1 \cdot \bar{C} \cdot K. \end{aligned}$$



a



6

Рисунок 3.4 — JK-триггер, переключающийся по заднему фронту: а — схема для моделирования с помощью программы Micro-CAP; б — временные диаграммы

Строится схема на основе асинхронных RS-триггеров, реализующая полученные логические выражения. Для последующего моделирования с помощью программы Micro-CAP в качестве асинхронных RS-триггеров используются JK или D-триггера с асинхронными входами установки (PREBAR) и сброса (CLRBAR). При этом входы синхронизации и информационные входы не задействуются.

Схема для моделирования с обозначениями необходимых сигналов приведена на рис. 3.4, а, а на рис. 3.4, б приведены временные диаграммы работы автомата, доказывающие его работоспособность.

3.1.3 Пример 3 — Автомат Мили

Асинхронный автомат Мили, выделяющий второй полный импульс из последовательности $U_{И1}$, если приходит сигнал запуска U_S ; и вторую полную паузу $U_{П1}$, если приходит сигнал запуска U_Z (см. рис. 3.5).

Автоматы Мура описываются функциями переходов и выходов:

$$a_{t+1} = f(a_t, x_t), \quad z_t = \varphi(a_t),$$

т.е. каждое новое состояние обусловлено предшествующим состоянием и входным сигналом, а выход в каждый момент однозначно определяется состоянием автомата. Таким образом, выходы однозначно определяются состояниями автомата и поэтому могут быть указаны в вершинах графа.

Автоматы Мили отличаются тем, что выход зависит не только от состояния, но и от входного сигнала:

$$a_{t+1} = f(a_t, x_t), \quad z_t = \varphi(a_t, x_t).$$

Для автомата Мили выходы указываются у концов дуг, т.к. они зависят как от входов, так и от состояний.

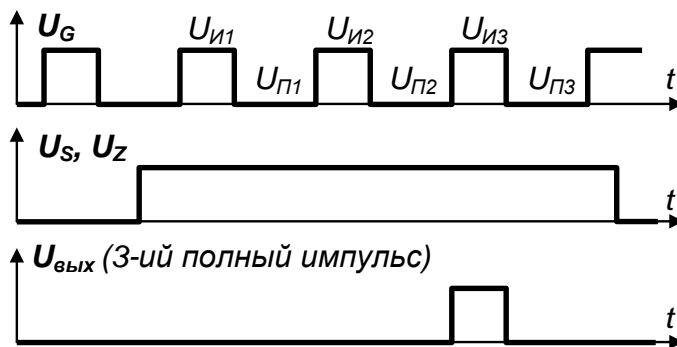


Рисунок 3.5 — Временные диаграммы работы асинхронного автомата

Составим граф переходов устройства, разрабатываемого как автомат Мили (см. определение выше). Вершины графа (устойчивые состояния автомата) будем кодировать противоположно (с использованием кода Грея). Необходимое нам состояние, в котором формируется выходной сигнал (либо второй импульс, либо 2-ая пауза) является 5-м состоянием автомата. Таким образом, для кодирования номеров состояний необходимо 3 запоминающих элемента — RS-триггера. При этом останутся неиспользуемые состояния, переходы через которые можно кодировать произвольно исходя из имеющихся в наличии сигналов и возможности минимизации аппаратной реализации. Часто переходы через неиспользуемые состояния осуществляются по сигналу «1». Это означает, что сразу же осуществляется переход между последовательными состояниями. В конце необходимо предусмотреть возврат автомата в исходное состояние, который обычно осуществляется при окончании действия управляющих сигналов.

Отметим, что в рассматриваемом примере переход из состояний 111, 101 осуществляется по сигналам Y и X соответственно, что сделано для минимизации аппаратной реализации устройства (как будет показано ниже).

Охватим замкнутой линией все состояния на графе переходов, в которых значения одной и той же переменной (состояние одного элемента памяти) равны 1. Эти замкнутые кривые показаны сплошной ($Q_1=1$), штрих-пунктирной ($Q_2=1$) и пунктирной линиями ($Q_3=1$) соответственно.

Поскольку вход в подобную замкнутую область и выход из нее требуют линий сигналов возбуждения, обозначаются соответствующие сигналы возбуж-

дения. Стрелками, входящими в область, указывают возбуждение установки (присваивающее переменной единичное значение), а стрелками, выходящими из области, — возбуждение сброса (присваивающее ей нулевое значение).

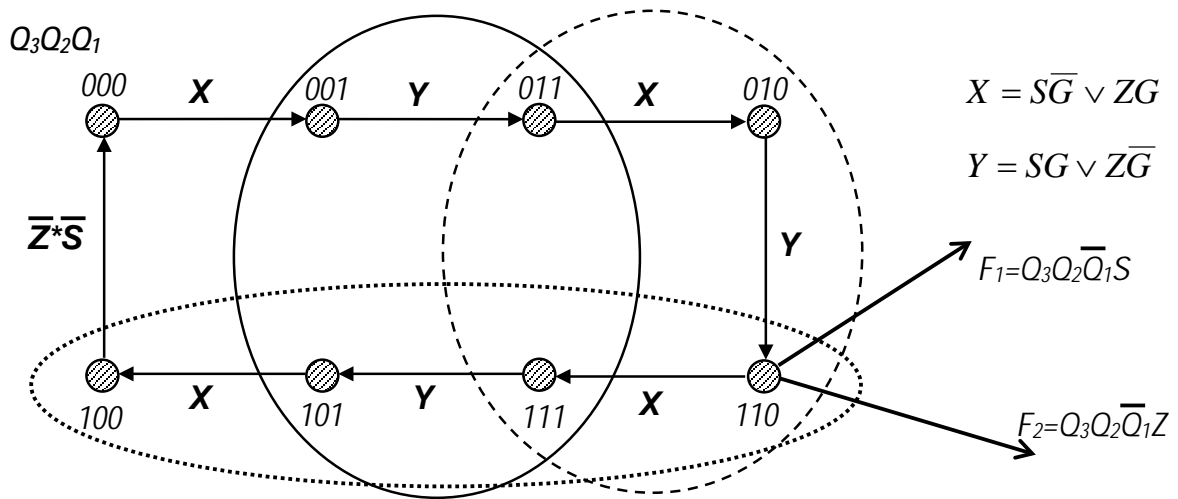


Рисунок 3.6 — Граф переходов асинхронного автомата МИЛИ

Составляются два выражения в форме ДНФ (суммы произведений): одно для функции возбуждения установки, а другое для функции возбуждения сброса. Каждое произведение должно содержать входные переменные (сигналы по которым совершается переход) и вторичные переменные, связанные с данным переходом, но не меняющие свои значения. В качестве вторичных переменных выступают двоичные разряды кода состояния в коде Грея. Включение вторичных переменных гарантирует выполнение переходов в правильной последовательности.

$$S_1 = \overline{Q_3}\overline{Q_2}X \vee Q_3Q_2X = X \cdot (\overline{Q_3} \vee Q_2) \quad R_1 = \overline{Q_3}Q_2X \vee Q_3\overline{Q_2}X = X \cdot (Q_3 \vee \overline{Q_2});$$

$$S_2 = \overline{Q_3}Q_1Y$$

$$R_2 = Q_3Q_1Y;$$

$$S_3 = Q_2\overline{Q_1}Y$$

$$R_3 = \overline{Q_2}\overline{Q_1}\overline{S}\overline{Z};$$

$$F_1 = Q_3Q_2\overline{Q_1}S$$

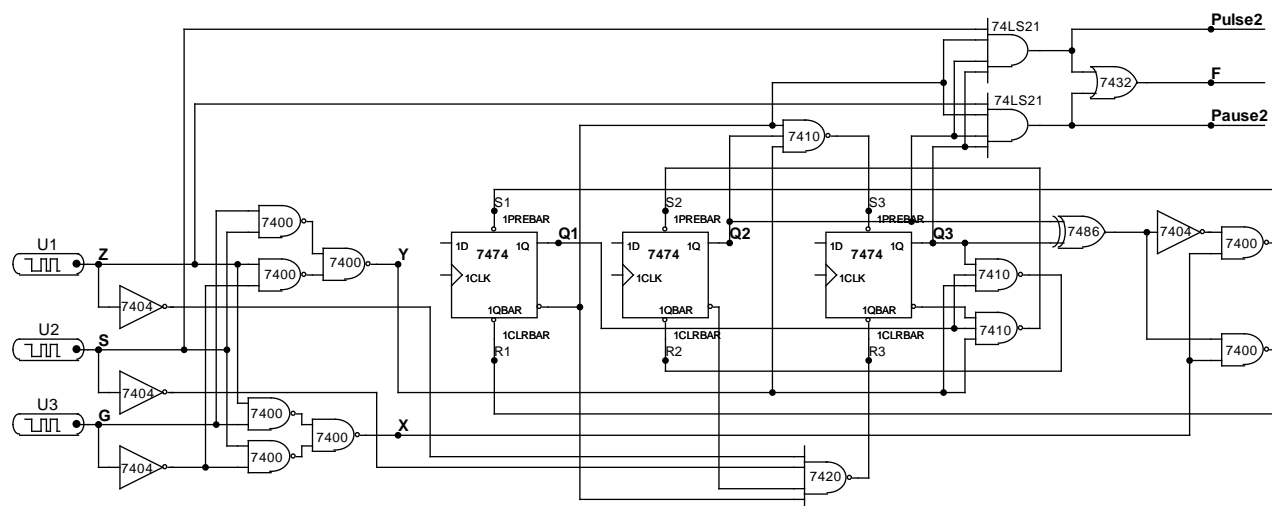
$$F_2 = Q_3Q_2\overline{Q_1}Z$$

$$F = F_1 \vee F_2 = Q_3Q_2\overline{Q_1}(Z \vee S).$$

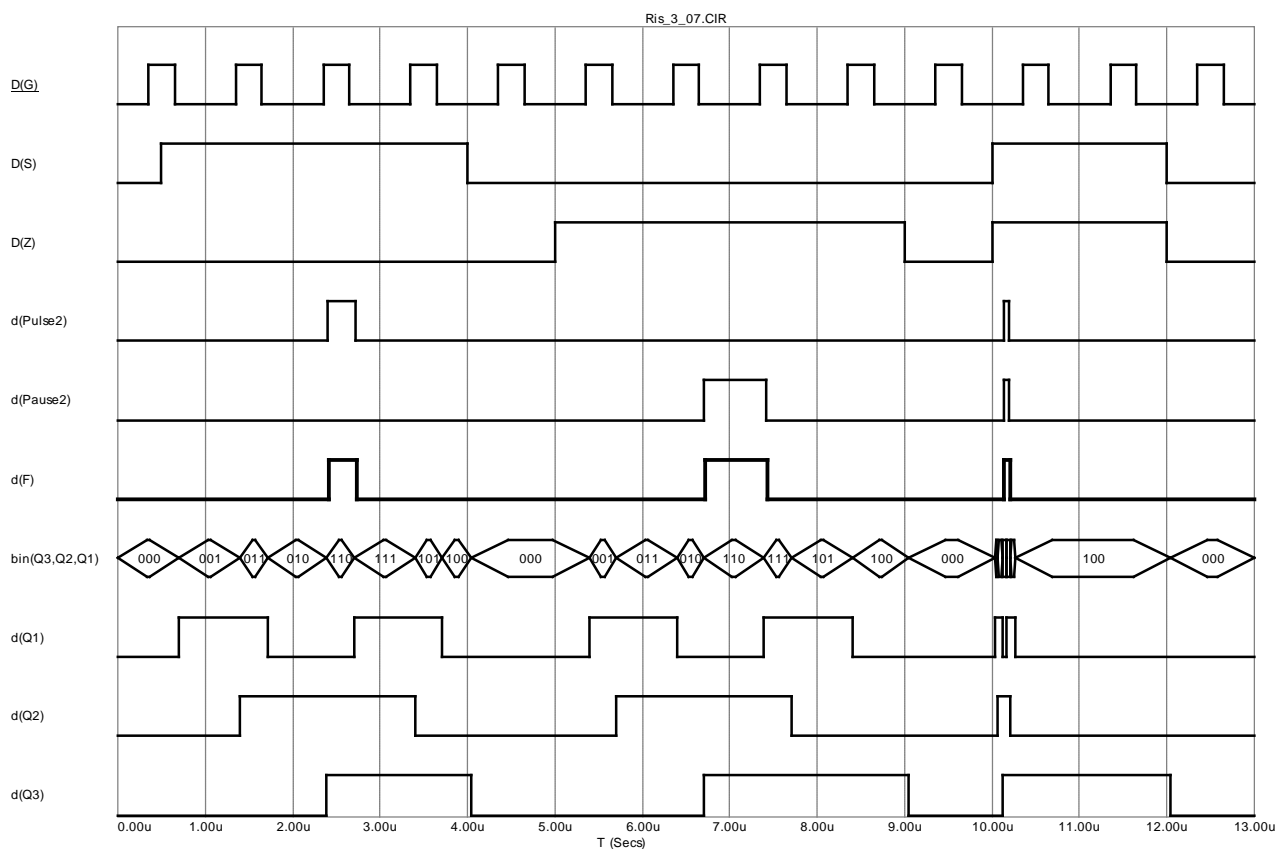
Строится схема на основе асинхронных RS-триггеров, реализующая полученные логические выражения. Для последующего моделирования с помощью программы Micro-CAP в качестве асинхронных RS-триггеров используются JK или D-триггера с асинхронными входами установки (PREBAR) и сброса (CLRBAR). При этом входы синхронизации и информационные входы не задействуются.

В рассматриваемом примере автомат проходит через неиспользуемые состояния 101, 100 по сигналам Y и X соответственно, при этом частично минимизируются логические выражения. Можно через эти состояния пройти автоматически, без внешнего сигнала (т.е. по «1»), как сделано в *слайдовом варианте лекций*.

Схема для моделирования с обозначениями необходимых сигналов приведена на рис. 3.7, а, а на рис. 3.7, б приведены временные диаграммы работы автомата, доказывающие его работоспособность.



a



6

Рисунок 3.7 — Асинхронный автомат Мили, выделяющий второй полный импульс при приходе управляющего сигнала S, и вторую полную паузу — при приходе Z

3.1.4 Пример 4 — автомат Мура

Асинхронный автомат Мура, выделяющий первый полный импульс из последовательности $U_{Ил}$, если приходит сигнал запуска U_S ; и вторую полную паузу $U_{Пл}$, если приходит сигнал запуска U_Z (см. рис. 3.5).

Автоматы Мура описываются функциями переходов и выходов:

$$a_{t+1} = f(a_t, x_t), \quad z_t = \varphi(a_t),$$

т.е. каждое новое состояние обусловлено предшествующим состоянием и вход-

ным сигналом, а выход в каждый момент однозначно определяется состоянием автомата. Таким образом, выходы однозначно определяются состояниями автомата и поэтому могут быть указаны в вершинах графа.

Составим граф переходов устройства, разрабатываемого как автомат Мура (см. определение выше). Вершины графа (устойчивые состояния автомата) будем кодировать противоположно (с использованием кода Грея). Исходное состояние автомата 000. Выход из него осуществляется либо по управляющему сигналу S и паузе в последовательности, либо по управляющему сигналу Z и импульсу в последовательности. Такая привязка осуществляется, т.к. в первом случае необходимо выделить 1-ый *полный* импульс, а во втором случае — 2-ую *полную* паузу. Указанные комбинации сигналов дают начало двум путям переходов автомата из состояния 000, показанным на рис. 3.8 стрелками. В конце необходимо предусмотреть возврат автомата в исходное состояние, который обычно осуществляется при окончании действия управляющих сигналов ($\bar{S} \cdot \bar{Z}$).

Охватим замкнутой линией все состояния на графе переходов, в которых значения одной и той же переменной (состояние одного элемента памяти) равны 1. Эти замкнутые кривые показаны сплошной ($Q_1=1$), штрих-пунктирной ($Q_2=1$) и пунктирной линиями ($Q_3=1$) соответственно.

Поскольку вход в подобную замкнутую область и выход из нее требуют линий сигналов возбуждения, обозначаются соответствующие сигналы возбуждения. Стрелками, входящими в область, указывают возбуждение установки (присваивающее переменной единичное значение), а стрелками, выходящими из области, — возбуждение сброса (присваивающее ей нулевое значение).

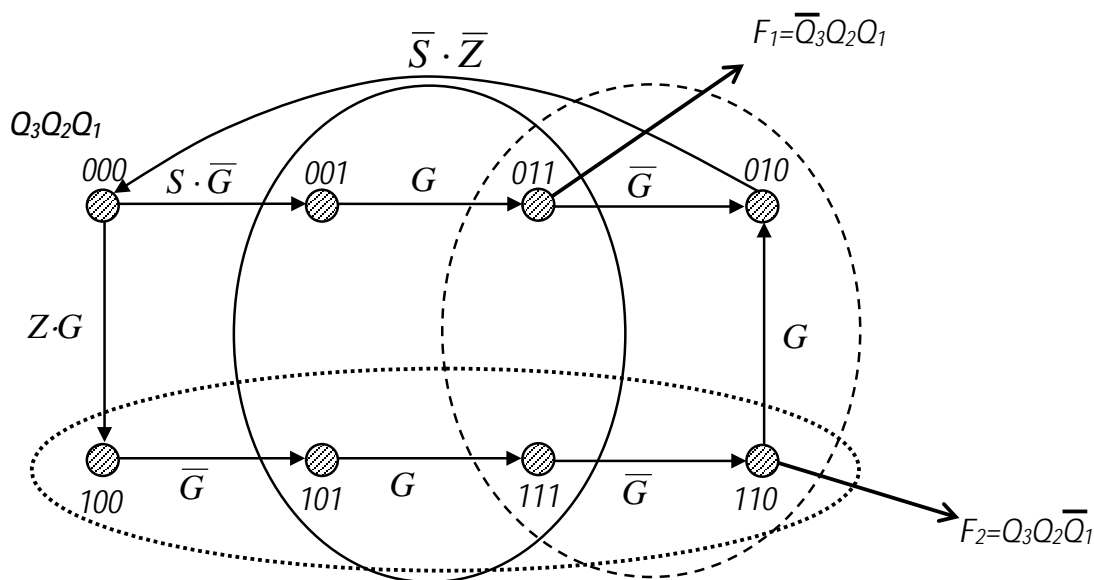


Рисунок 3.8 — Граф переходов асинхронного автомата МУРА

Составляются два выражения в форме ДНФ (суммы произведений): одно для функции возбуждения установки, а другое для функции возбуждения сброса. Каждое произведение должно содержать входные переменные (сигналы по которым совершается переход) и вторичные переменные, связанные с данным переходом, но не меняющие свои значения. В качестве вторичных переменных

выступают двоичные разряды кода состояния в коде Грея. Включение вторичных переменных гарантирует выполнение переходов в правильной последовательности.

$$S_1 = \overline{Q_3}\overline{Q_2}S\overline{G} \vee Q_3\overline{Q_2}\overline{G} \quad R_1 = \overline{Q_3}Q_2\overline{G} \vee Q_3Q_2\overline{G} = Q_2\overline{G};$$

$$S_2 = Q_1G$$

$$R_2 = \overline{Q_3}\overline{Q_1}\overline{S}\overline{Z};$$

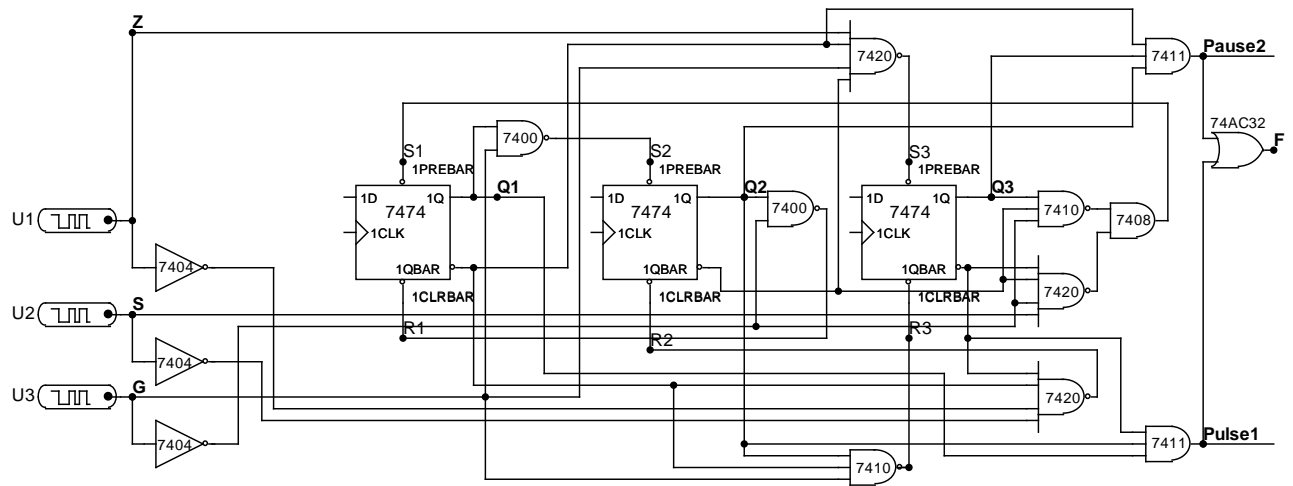
$$S_3 = \overline{Q_2}\overline{Q_1}ZG$$

$$R_3 = Q_2\overline{Q_1}G;$$

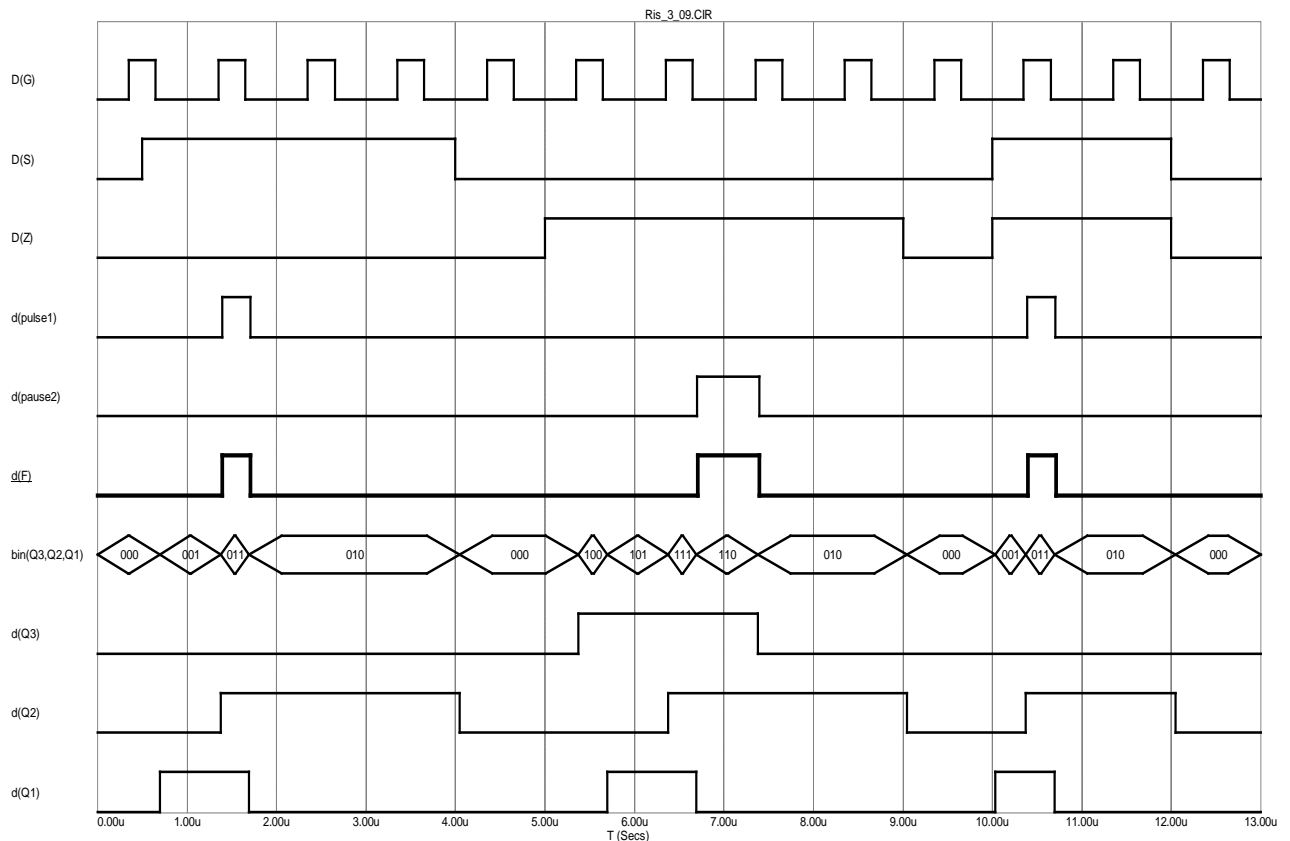
$$F_1 = \overline{Q_3}Q_2Q_1$$

$$F_2 = Q_3Q_2\overline{Q_1}$$

$$F = \overline{Q_3}Q_2Q_1 \vee Q_3Q_2\overline{Q_1}.$$



а



б

Рисунок 3.9 — Автомат Мура: а — схема для моделирования; б — временные диаграммы

Строится схема на основе асинхронных RS-триггеров, реализующая полученные логические выражения. Для последующего моделирования с помощью программы Micro-CAP в качестве асинхронных RS-триггеров используются JK или D-триггера с асинхронными входами установки (PREBAR) и сброса (CLRBAR). При этом входы синхронизации и информационные входы не задействуются.

Схема для моделирования с обозначениями необходимых сигналов приведена на рис. 3.9, а, а на рис. 3.9, б приведены временные диаграммы работы автомата, доказывающие его работоспособность.

3.2 Синтез асинхронных автоматов на мультиплексорах

3.2.1 Пример 1. Асинхронный автомат Мили

Устройство памяти, имеющее 2^k устойчивых состояний реализуется на k мультиплексорах, каждый из которых имеет 2^k информационных входов. Т.е. мультиплексоры должны иметь следующую организацию: 2^k входных направлений в 1 выходное направление (2^k в 1) [9, 10]. При построении автомата обратная связь с k выходов k мультиплексоров заводится на k селектирующих (адресных) входов всех k мультиплексоров. При этом на 2^k информационных входов мультиплексоров по определенным правилам подаются управляющие работой автомата сигналы. Таким образом, автомат с памятью, имеющий 16 устойчивых состояний может быть реализован на основе 4-х мультиплексоров с организацией 16 в 1. Автомат с памятью, имеющий 8 устойчивых состояний может быть реализован на основе 3-х мультиплексоров с организацией 8 в 1.

Пример 1. Асинхронный автомат Мили, выделяющий 3-ий полный импульс из последовательности $U_{ИЗ}$, если приходит сигнал запуска U_S ; и третью полную паузу $U_{ПЗ}$, если приходит сигнал запуска U_Z (см. рис. 3.5).

Методика синтеза асинхронных автоматов на основе мультиплексоров подробно рассмотрена в [7–9].

Составим граф переходов устройства, разрабатываемого как автомат Мили (см. определение выше). Вершины графа (устойчивые состояния автомата) будем кодировать противогоночно (с использованием кода Грея). Необходимое нам состояние, в котором формируется выходной сигнал (либо 3-ий импульс, либо 3-я пауза) является 7-м по счету состоянием автомата. Таким образом, для кодирования номеров состояний необходимо 3 запоминающих элемента — RS-триггера (или 3 мультиплексора с обратными связями). При этом не останется неиспользуемых состояний, поскольку необходим выход из состояния, в котором снимаются выходные сигналы и возврат автомата в исходное состояние.

В соответствии с графом переходов ($8=2^3$ устойчивых состояний) для синтеза устройства понадобится 3 мультиплексора из 8 в 1.

Составим таблицу программирования мультиплексоров по следующим правилам:

1. В первой строке таблицы указываются коды состояний автомата $Q_3Q_2Q_1$, которые соответствуют кодам на селектирующих входах мультиплексоров. Для последующего программирования информационных входов мультиплексоров

удобно указывать в соседних ячейках таблицы соседние коды состояний. Напомним, что соседние состояния кодируются противоположно, т.е. с использованием кода Грея (табл. 3.1, 1-ая строка);

2. Во второй строке таблицы указываются значения сигналов на информационных входах всех 3-х мультиплексоров $D_0...D_7$, которые формируются по следующим правилам:

а) в столбце соответствующем i -ому ($i=0...7$) состоянию автомата значения сигналов D_j^i ($j=1...3, i=0...7$) равны значениям Q_j для тех разрядов, которые остаются неизменными при переходе в следующее (соседнее) состояние. Отметим, что в случае если из данного состояния возможно несколько переходов, D_j^i копирует те разряды Q_j ($Q_3Q_2Q_1$), которые остаются неизменными при всех возможных переходах из данного состояния.

б) остальным информационным сигналам D_j^i , подаваемым на i -ые входы j -ых мультиплексоров, присваиваются значения переменных, вызывающих данный переход по следующим правилам. Если переход сопровождается изменением значения разряда кода состояния с 0 на 1, то переменная подается на соответствующий вход без инверсии, при изменении с 1 на 0 — переменная, подаваемая на вход, инвертируется.

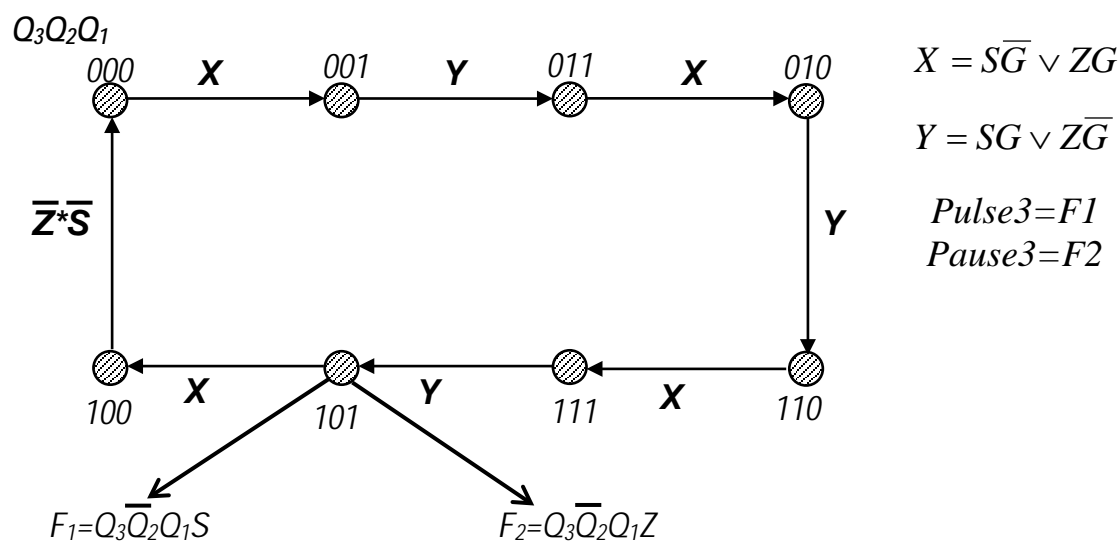


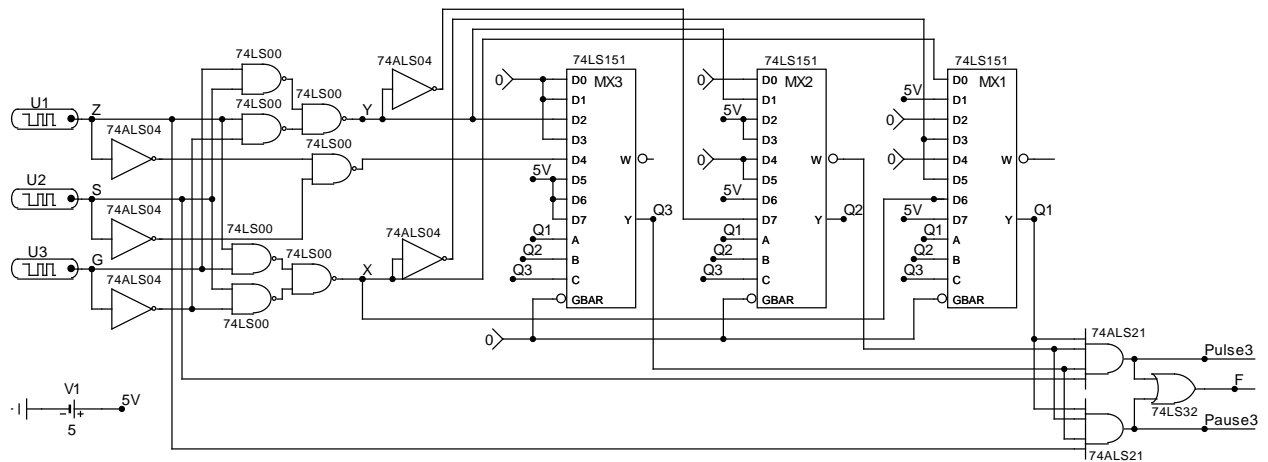
Рисунок 3.10 — Граф переходов асинхронного автомата МИЛИ

В соответствии с вышеприведенными правилами таблица настройки мультиплексоров имеет вид:

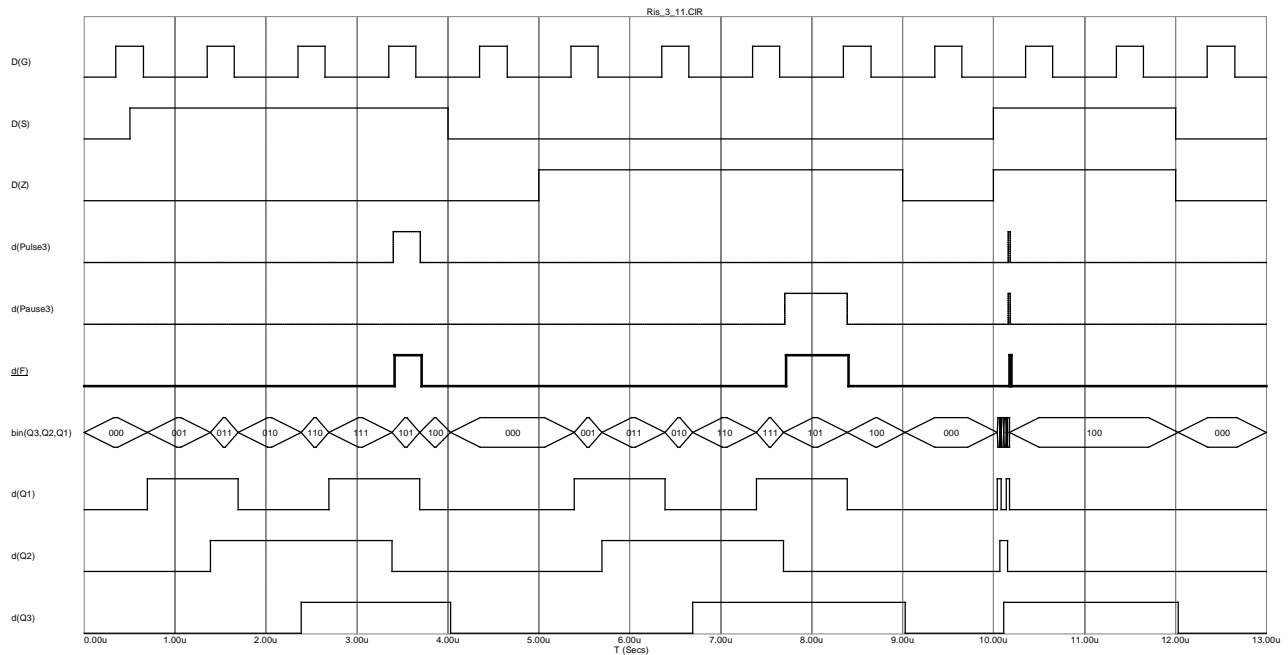
Таблица 3.1 — Программирование мультиплексоров асинхронного автомата Мили

i (соседние состояния)	0	1	3	2	6	7	5	4
$Q_3 Q_2 Q_1$	000	001	011	010	110	111	101	100
$D_3^i D_2^i D_1^i$	00X	0Y1	01 \bar{X}	Y10	11X	1 \bar{Y} 1	10 \bar{X}	$\bar{\bar{Z}}\bar{\bar{S}}$ 00

Принципиальная схема синтезированного в соответствии с табл. 3.1 асинхронного автомата Мили и результаты его анализа в среде пакета программ Micro-CAP приведены на рис. 3.11, а, б соответственно.



а



б

Рисунок 3.11 — Асинхронный автомат Мили (выделения 3-го импульса, 3-ей паузы):
а — схема; б — временные диаграммы

3.2.2 Пример 2. Асинхронный автомат Мура

Пример 2. Асинхронный автомат Мура, выделяющий первый полный импульс из последовательности U_{II} , если приходит сигнал запуска U_S ; и вторую полную паузу U_{PI} , если приходит сигнал запуска U_Z (см. рис. 3.5).

Автоматы Мура описываются функциями переходов и выходов:

$$a_{t+1} = f(a_t, x_t), \quad z_t = \varphi(a_t),$$

т.е. каждое новое состояние обусловлено предшествующим состоянием и входным сигналом, а выход в каждый момент однозначно определяется состоянием

автомата. Таким образом, выходы однозначно определяются состояниями автомата и поэтому могут быть указаны в вершинах графа.

Т.к. необходимо выделить либо первый импульс, либо вторую паузу *полной* длительности, то при запуске автомата привязку необходимо осуществлять к фронту (спаду) импульса периодической последовательности (т.е. из начального состояния возникнет переход в одно состояние по конъюнкции сигналов $S \cdot \bar{G}$, и переход в другое состояние — по конъюнкции сигналов $Z \cdot G$). Следовательно, по одной из этих двух ветвей будет осуществляться начальный переход в графе переходов автомата Мура.

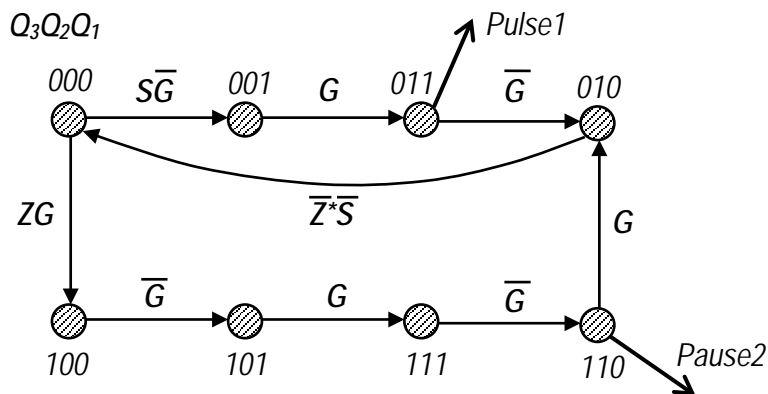


Рисунок 3.12 — Граф переходов асинхронного автомата Мура

Построим граф переходов автомата, используя противогоночное (в коде Грея) кодирование состояний (см. рис. 3.12).

При этом выходной сигнал (соответствующий 2-ой полной паузе — *Pause2*) является логическим произведением соответствующих разрядов кода состояния автомата $Q_3 \cdot Q_2 \cdot \bar{Q}_1$, а выход-

ной сигнал (соответствующий 1-му полному импульсу) является логическим произведением разрядов кода состояния автомата $\bar{Q}_3 \cdot Q_2 \cdot Q_1$. Таким образом, выражение для выходной функции F записывается в следующем виде:

$$F = Q_3 \cdot \bar{Q}_2 \cdot Q_1 \vee \bar{Q}_3 \cdot Q_2 \cdot Q_1.$$

В соответствии с графом переходов ($8=2^3$ устойчивых состояний) для синтеза устройства понадобится 3 мультиплексора из 8 в 1.

Составим таблицу программирования мультиплексоров по следующим правилам:

В первой строке таблицы указываются коды состояний автомата $Q_3Q_2Q_1$, которые соответствуют кодам на селектирующих входах мультиплексоров.

Во второй строке таблицы указываются значения сигналов на информационных входах всех 3-х мультиплексоров $D_0 \dots D_7$, которые формируются по следующим правилам:

а) в столбце соответствующем i -ому ($i=0 \dots 7$) состоянию автомата значения сигналов D_j^i ($j=1 \dots 3$) равны значениям Q_j для тех разрядов, которые остаются неизменными при переходе в следующее (соседнее) состояние. Отметим, что в случае если из данного состояния возможно несколько переходов, D_j^i копирует те разряды Q_j ($Q_3Q_2Q_1$), которые остаются неизменными при всех возможных переходах из данного состояния.

б) остальным информационным сигналам D_j^i , подаваемым на i -ые входы j -ых мультиплексоров, присваиваются значения переменных, вызывающих данный переход по следующим правилам. Если переход сопровождается изме-

3.3 Синтез синхронных автоматов

3.3.1 Пример 3. Синтез счетчика с изменяемым коэффициентом пересчёта

Пример 3. Синтезировать и протестировать цифровой синхронный автомат (программно-управляемый счетчик), обеспечивающий следующие последовательности счета:

A=1; B=0

↪ 0, 1, 2, 3 ↻

A=0; B=1

↪ 0, 1, 6, 7 ↻

В остальных случаях

↪ 0, 1, 2, 3, 4, 5, 6, 7 ↻

В соответствии с совокупностью сигналов, управляющих последовательностью переходов синхронного автомата, сформируем новые сигналы, управляющие работой синхронного автомата.

$$X_1 = A \cdot \bar{B}, \quad X_2 = \bar{A} \cdot B.$$

Данные сигналы просто могут быть получены с помощью полного двоичного дешифратора 2X4.

Обобщенная схема синхронного автомата приведена на рис. 3.14. В соответствии с этой структурой проведем синтез синхронного автомата примера.

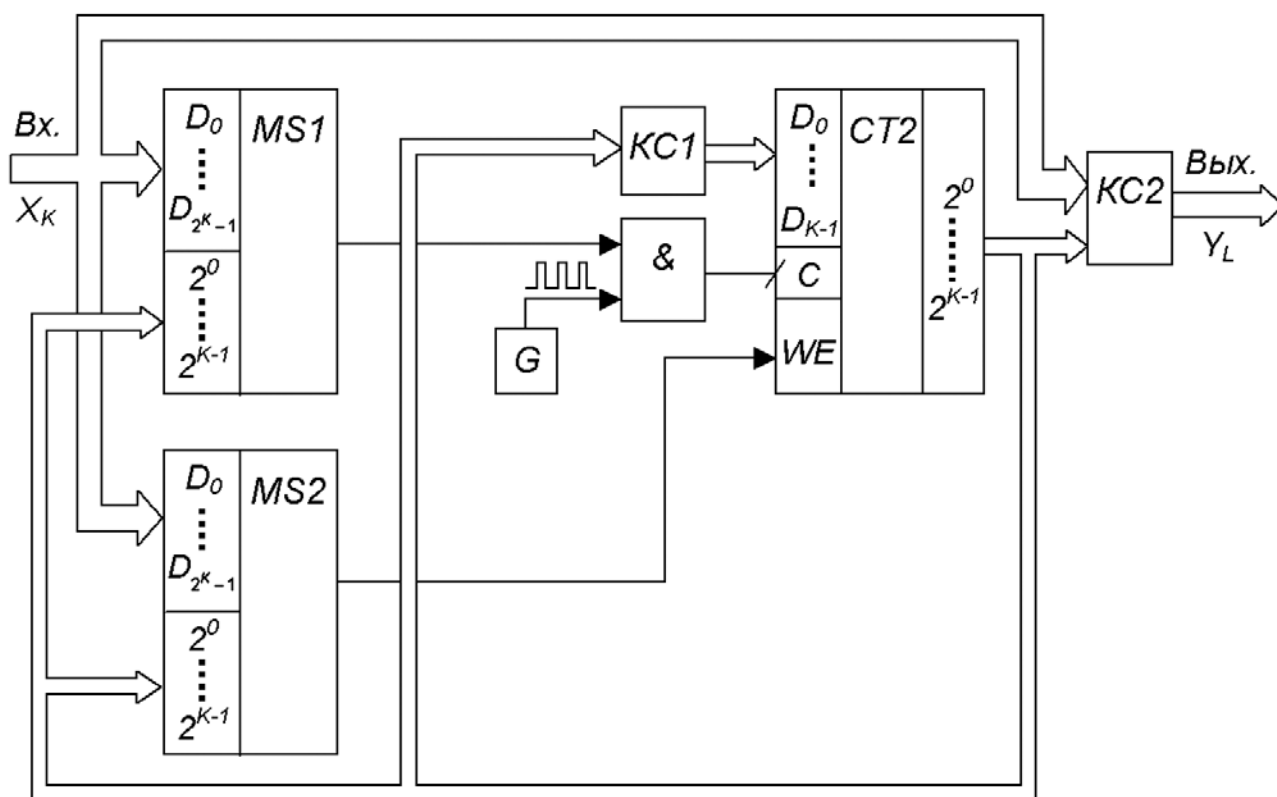


Рисунок 3.14 — Обобщенная структурная схема синхронного автомата

Составим граф переходов синтезируемого синхронного автомата (рис. 3.15). Отметим, что поскольку переход из состояния в состояние осуществляется по синхросигналу, в графе переходов нет необходимости использовать про-

тивогоничное кодирование состояний, их достаточно закодировать позиционным двоичным кодом.

Двоичные переходы на структурной схеме осуществляет мультиплексор MX1, имеющий выход Y1. Сигнал на счетном входе (приращения) счетчика:

$$SYN = Y1 \cdot G.$$

Мультиплексор MX2 — мультиплексор недвоичных переходов. Его выход Y2 подается на вход параллельной загрузки счетчика WE (LOADBAR, имеющий активный низкий уровень, на схеме примера см. рис. 3.16). В соответствии с этим таблица программирования мультиплексора MX2 рассматриваемого примера следующая:

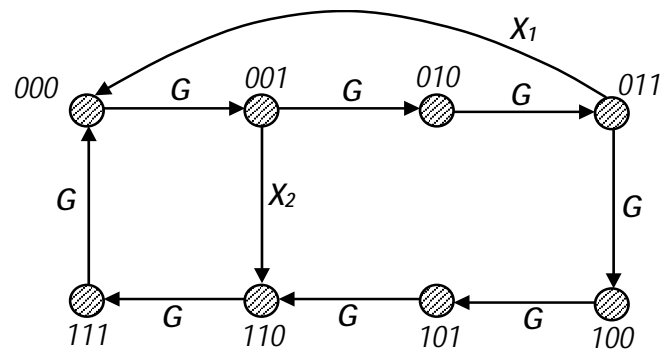


Рисунок 3.15 — Граф переходов синхронного автомата (счетчика с изменяемой последовательностью счета)

$$D2 = \bar{X}_2; \quad D4 = \bar{X}_1.$$

На остальных информационных входах MX2 — постоянный уровень логической «1».

Инверсия сигналов для входов недвоичных переходов и уровень лог. «1» для входов двоичных переходов используются, поскольку вход параллельной загрузки счетчика имеет активный низкий уровень.

Перейдем теперь к синтезу комбинационной схемы преобразователя кодов (КС1 на рис. 3.14), которая подает на входы параллельной загрузки счетчика код следующего состояния при осуществлении недвоичного перехода синхронного автомата.

Таблица недвоичных переходов синхронного автомата:

$Q_2 Q_1 Q_0(t_i)$	$Q_2 Q_1 Q_0(t_{i+1})$
011	000
001	110

Минимизируем логическую функцию для каждого входа параллельной загрузки Q_0 (P_0), Q_1 (P_1), Q_3 (P_2) с помощью карт Карно:

$Q_1 \backslash Q_0$	00	01	11	10
Q_2				
0	X	0	0	X
1	X	X	X	X

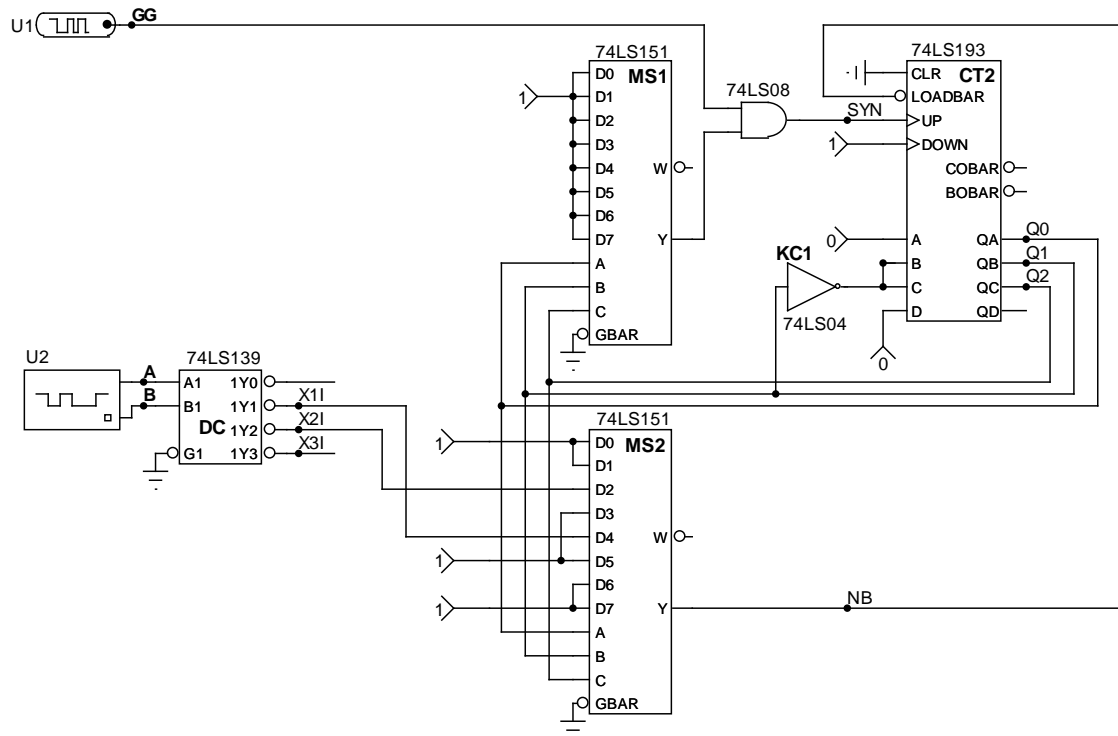
$$Q_0 \quad (P_0) \quad P_0 = 0$$

$Q_1 \backslash Q_0$	00	01	11	10
Q_2				
0	X	1	0	X
1	X	X	X	X

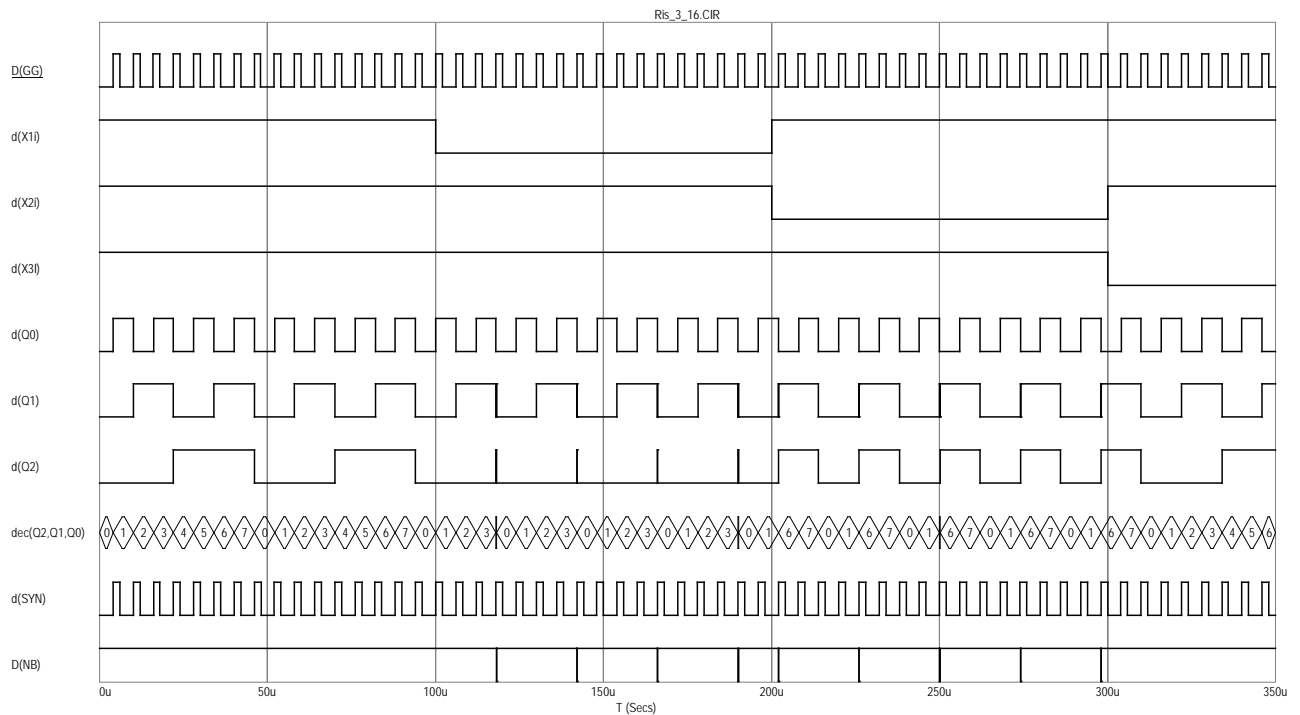
$$Q_1 \quad (P_1) \quad P_1 = \bar{Q}_1$$

$Q_1 Q_0$	00	01	11	10
Q_2				
0	X	1	0	X
1	X	X	X	X

$$Q_2 (P_2) \quad P_2 = \overline{Q_1}$$



а



б

Рисунок 3.16 — Синхронный автомат: а — схема для моделирования; синхронного автомата; б — временные диаграммы

Принципиальная схема синхронного автомата для анализа с помощью программы Micro-CAP приведена на рис. 3.16, а, а временные диаграммы работы — на рис. 3.16, б.

Отметим, что верхний мультиплексор (мультиплексор двоичных переходов) для данной задачи может отсутствовать, поскольку счетчик начинает счет в двоичном режиме независимо от значения управляющих сигналов. Т.е., для рассматриваемой задачи можно подавать импульсы тактового генератора GG непосредственно на счетный вход счетчика.

ЛИТЕРАТУРА

1. **Джон Ф. Уэйкерли** Проектирование цифровых устройств том 1, 2 Пер. с англ. Е.В. Воронова, А.Л. Ларина, Постмаркет, Москва, 2002
2. **Угрюмов Е.П.** Цифровая схемотехника: Учеб. Пособие для вузов. — СПб.: БХВ-Петербург, 2004. — 528 с.: ил.
3. **Уилкинсон Барри** Основы проектирования цифровых схем. : Пер. с англ. — М.: Издательский дом «Вильямс», 2004. — 320 с.: ил.
4. **Новиков Ю. В.** Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. — М.: Мир, 2001. — 379 с, ил. — (Современная схемотехника)
5. **В.Л. Шило** Популярныe цифровые микросхемы: Справочник. — М.: Радио и связь, 1987. — 352 с.: ил.
6. **В.Л. Шило** Популярныe микросхемы КМОП. Справочник. — М.: Издательство «Ягуар», 1993. — 64 с.: ил.
7. **М.А. Амелина, Ю.В Троицкий** Синтез комбинационных и последовательностных логических устройств. Лабораторный практикум по курсу «Электронные промышленные устройства». — М.: МЭИ, 1999. — 48 с.
8. **Амелина М.А., Амелин С.А.** Программа схемотехнического моделирования Micro-Cap 8. — М.: Горячая линия-Телеком, 2007. — 464 с.: ил.
9. Проектирование схем управления на базе микросхем повышенной степени интеграции. Троицкий Ю.В. /Под ред. В.А. Циганкова. — М.: Моск. Энерг. ин-т, 1989. — 80 с.
10. **Хоуп Г.** Проектирование цифровых вычислительных устройств на интегральных схемах: Пер. с англ. — М.: Мир, 1984. — 400 с.
11. **Опадчий Ю.Ф., Глудкин О.П., Гуров А.И.** Аналоговая и цифровая электроника (полный курс): Учебник для вузов. Под ред. О.П. Глудкина. — М.: Горячая линия –Телеком, 2003. — 768 с.: ил.
12. **Марченко А.Л.** Основы электроники. Учебное пособие для вузов / Марченко. — М.: ДМК Пресс, 2008. — 296 с., ил.
13. **Брей Б.** Применение микроконтроллеров PIC18. Архитектура, программирование и построение интерфейсов с применением С и ассемблера. Пер. с англ. — К.: «МК-пресс», СПб: «КОРОНА-ВЕК», 2008. — 576 с., ил.