А.Н. КАСЬЯНОВ

Micro Cap

В СХЕМОТЕХНИКЕ

Издательство ТГТУ

Учебное издание

КАСЬЯНОВ Александр Николаевич

Micro-Cap
В СХЕМОТЕХНИКЕ

Учебное пособие

Редактор З.Г. Чернова
Компьютерное макетирование М.А. Филатовой

Подписано в печать 19.03.04
Формат 60 × 84 / 16. Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Объем: 6,51 усл. печ. л.; 6,5 уч.-изд. л.
Тираж 150 экз. С. 114

Издательско-полиграфический центр
Тамбовского государственного технического университета,
392000, Тамбов, Советская, 106, к. 14

Министерство образования Российской Федерации
Тамбовский государственный технический университет

А.Н. КАСЬЯНОВ

Micro-Cap
В СХЕМОТЕХНИКЕ

Утверждено Ученым советом университета в качестве учебного пособия по дисциплине "Схемотехника" для студентов 3 курса дневного отделения специальности 220300

Тамбов
Издательство ТГТУ
2004

УДК 621.396.6(075)
ББК З 973.26-04я73
К28

Рецензенты:
Профессор, доктор технических наук
А.А. Арзамасцев
Профессор, доктор технических наук
А.А. Безбогов

Касьянов А.Н.

К28 Micro-Cap в схемотехнике: Учебное пособие. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2004. 112 с.

Настоящее пособие посвящено изучению основ схемотехники с использованием системы схемотехнического моделирования Micro-Cap V. Приведены основные интерфейсы, используемые для моделирования в среде Micro-Cap V, VI и VII. Рассмотрены синтез и методы проектирования комбинационных и последовательностных схем, а также временные анализы работы различных цифровых схем.

Предназначено для студентов 3 курса дневного отделения Тамбовского государственного технического университета, специальности 220300 – системы автоматизированного проектирования, а также может быть полезно для студентов и преподавателей вузов других специальностей.

УДК 621.396.6(075)
ББК З 973.26-04я73

ISBN 5-8265-0266-5

ВВЕДЕНИЕ

Все электронные изделия условно можно разделить на два типа. К первому типу относятся те, которые обрабатывают информацию, представленную в аналоговом виде, т.е. изменяющуюся непрерывно как по уровню, так и по времени. Ко второму типу относятся устройства перерабатывающие информацию, представленную в цифровой форме, которая по времени изменяется дискретно. Еще недавно представители первого типа доминировали перед цифровыми устройствами. Схемотехника цифровых устройств явилась огромным достижением науки конца XX века. Ее преимущество перед схемотехникой аналоговых устройств основано на следующих факторах:

1) информация в цифровых устройствах не подвержена воздействию окружающей среды (температуры, влажности, давления, напряжения питания и т.д.).

2) переработку цифровой информации возможно осуществлять с неограниченной точностью,

3) микросхемы цифровых устройств допускают большую степень интеграции, включающие десятки миллионов транзисторов, и каждый год это количество удваивается (Закон Мура).

Цифровая схемотехника развивалась по нескольким направлениям:

Проектирование и создание устройств переработки цифровой информации с помощью жесткой логики. Такие устройства предназначены для выполнения одной какой-либо операции. Создание устройств на жесткой логике привело к появлению универсальных микросхем, которые можно конфигурировать в соответствии с законом функционирования самим заказчиком. Эти микросхемы получили название – программируемые логические интегральные схемы (ПЛИС). Одна микросхема ПЛИС может заменить несколько сотен корпусов микросхем традиционной жесткой логики. Особенностями современных ПЛИС являются: низкая стоимость, высокое быстродействие (до 3,5 нс), широкие функциональные возможности, многократность перепрограммирования, низкое энергопотребление, гибкость архитектуры и др. Процесс проектирования устройства переработки цифровой информации с использованием ПЛИС заключается в описании его функционирования на входном языке используемого программного средства, выполнение автоматизированного синтеза, проведения моделирования и настройке выбранной ПЛИС с помощью программатора. Важной особенностью является тот факт, что время разработки, даже очень сложных схем, может составлять несколько часов. А для изменения алгоритма работы устройства достаточно перепрограммировать ПЛИС, причем, некоторые ПЛИС допускают перепрограммирование уже после установки их на плату.

Создание универсальных устройств, предназначенных для переработки разнородной информации. Типичным представителем является микропроцессор. Первоначально микропроцессорные устройства предназначались для управления работой дисплеев и принтеров. В дальнейшем развитие микропроцессорной техники позволило создать персональный компьютер, параметры которого постоянно совершенствуются. Универсальность применения микропроцессорных устройств обусловлено разнообразием программ, которые управляют работой устройства в целом.

Любое сложное цифровое устройство, будь то микропроцессор или ПЛИС, состоит из элементарных логических вентилей – устройств, реализующих какую-либо функцию алгебры логики. С помощью логических вентилей строятся как комбинационные схемы, так и последовательностные. К первому типу схем относятся – дешифраторы, шифраторы, мультиплексоры, двоичные сумматоры комбинационного типа, схемы сравнения и т.д.; ко второму типу схем относятся – триггеры, счетчики, регистры и др.

Современное схемотехническое проектирование любых схем невозможно без применения компьютерных методов расчета и проектирования электронных схем. В мире накоплен большой опыт по компьютерному проектированию электронных схем, разработано большое количество разнообразных программных средств. Значительных успехов достигла фирма *Spectrum Software*, создавшая целое семейство программ *Micro-Cap* (*Microcomputer Circuit Analisys Program*). На сайте фирмы можно бесплатно получить ознакомительные варианты программ *Micro-Cap V* (*MC5*), *Micro-Cap VI* (*MC6*) и *Micro-Cap VII* (*MC7*).

В данном пособии изложены принципы функционирования, синтеза и проектирования основных схемотехнических элементов, которые демонстрируются с использованием системы компьютерного схемотехнического проектирования и моделирования – *MC5*.

Схемотехника как учебная дисциплина для специальности 220300 имеет цель ознакомить студентов с принципами функционирования устройств цифровой электроники и развить навыки по их проектированию.

Изучению основ схемотехники и посвящено настоящее пособие, которое предназначено для студентов 3 курса дневного отделения Тамбовского государственного технического университета, специальности 220300 – системы автоматизированного проектирования, а также может быть полезно для студентов и преподавателей вузов других специальностей.

1 *MICRO-CAP* – КОМПЬЮТЕРНАЯ СИСТЕМА ПРОЕКТИРОВАНИЯ

1.1 КОМПЬЮТЕРНЫЕ МОДЕЛИ СИГНАЛОВ И ЭЛЕКТРОННЫХ КОМПОНЕНТ В *MICRO-CAP* (*MC*)

Меню *MC5* включает следующие разделы: ***File*** – работа с файлами, ***Edit*** – редактирование, ***Component*** – выбор компонент, ***Windows*** – работа с окнами, ***Analysis*** – анализ схем, ***Options*** – опции. В *MC6* и *MC7* добавлен пункт ***Design*** – синтез аналоговых пассивных и активных фильтров. Разделы меню – работа с файлами и их редактирование включают стандартные команды открытия файлов, их сохранения, печати, копирования, вставки и т.д. Главными разделами меню являются разделы **анализ схем и выбор компонент**. Раздел меню ***Component*** поддерживает:

- **аналоговые** компоненты, в том числе пассивные (резисторы, емкости, индуктивности, трансформаторы, диоды и др.), активные (биполярные, полевые транзисторы, ОУ), источники напряжения и тока (батарея, источник импульсных и гармонических колебаний, функциональный источник и др.), коннекторы, ключи и др.;
- **дискретные** компоненты, в том числе логические схемы (*AND*, *OR*, *NAND*, *NOR*, *INV*, *XOP* и др.), тристабильные компоненты, триггеры, программируемые логические матрицы, АЦП, ЦАП, линии задержки, генераторы двоичных сигналов и др.;
- **библиотеки** аналоговых и дискретных компонент, описываемые макросами (подсхемами).

Ниже приводятся описания **компьютерных моделей сигналов** и основных дискретных компонент, принятые в *MC5*, *MC6* и *MC7*.

Сигнал – это совокупность физического процесса (колебаний тока или напряжения) – носителя информации с наложенного на него путем модуляции сообщением – (носитель + сообщение). Носитель может иметь как гармоническую (синусоидальную), так и импульсную форму. Возможно задание другой произвольной формы сигнала.

Сигналы принято разделять на детерминированные и случайные, непрерывные и дискретные. Модель сигнала – это выбранный способ его математического описания. Далее рассмотрим источники сигналов, применяемые в цифровой схемотехнике.

Программируемый источник (импульсный) – двухполюсник, формирующий периодические или однократные импульсы напряжения с линейными или экспоненциально изменяющимися фронтами. Модель такого источника задается при вводе в окне настройки буквой *V* с указанием номера модели или полного имени. Модели и их параметры, задаваемые в *Split Text*, указаны ниже и в табл. 1.1.

.MODEL PULSE PUL (*VZERO*=0 *VONE*=0.1 *P*=0*u P*=.02*u P*=2.4*u P*=2.6*u P*=4.4*u*).

.MODEL TRIANGLE PUL (*VZERO*=0 *VONE*=2 *P*=0 *P*=500*N P*=1000*N*)

.MODEL IMPULSE PUL (*VZERO*=0 *VONE*=1 *P*=0*p P*=10000*p P*=10000*p P*=10000*p P*=100000*p*)

.MODEL SAWTOOTH PUL (*VZERO*=0 *VONE*=1 *P*=0 *P*=500*N P*=501*N*)

.MODEL SQUARE PUL (*VZERO*=0 *VONE*=1 *P*=0 *P*=0 *P*=500*N*)

1.1 Параметры модели программируемого источника

Параметр	Обо-	**ФИЗИЧЕСКИЙ СМЫСЛ**

	значение	
0: *Zero level voltage*	***Vzero***	Нулевой уровень напряжения
1: *One level voltage*	***Vone***	Единичный уровень напряжения
2: *Time delay to leading edge*	*P1*	Задержка от нулевого отсчета времени до начала нарастания
3: *Time delay to one level*	*P2*	Задержка до достижения единичного значения
4: *Time delay to falling edge*	*P3*	Задержка до начала спада
5: *Time delay to zero level*	*P4*	Задержка до достижения нулевого уровня
6: *Period of waveform*	*P5*	**Период повторения сигнала**

Случайный непрерывный сигнал формируется с помощью датчика случайных чисел, например, по следующему алгоритму: 3 + 0.5(*rnd* – 0.5), который читается так: если *rnd* (0…1) минус 0.5 больше 0, то 1, иначе 0.

Генераторы дискретных последовательностей (рис. 1.1 и табл. 1.2)

Формат схемотехнической модели (*SPICE*):
U<название> *STIM* (<ширина>,<массив форматов>)+<цифровой разъем питания> <цифровое заземление> +<вход> <ведущий>
<заземление> <конвертация>+<узел>*
+<имя модели ввода-вывода>
+[*IO_LEVEL*=<значение выбора интерфейса подсхемы>]
+[*TIMESTEP*=<размер временного шага>]
+<команды>*

Команда ***PART*:** <имя> – определяет имя элемента.

Примеры: *U1*
Uin

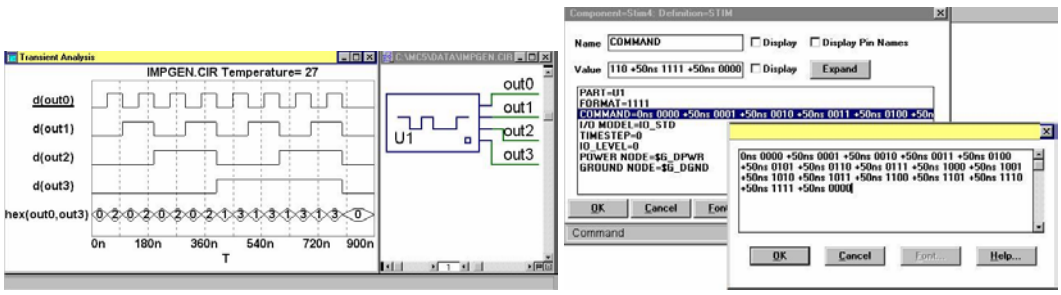


Рис. 1.1 Окна временного анализа и задания параметров модели генератора двоичной последовательности

1.2 Примеры моделей

U1 STIM(1,1) $*G_DPWR* – Пример 1 +$*G_DGND IN* +*IO_STD* +0*ns* 0 +*LABEL=START* +50*ns* 1 +50*ns* 0 +50*ns GOTO* *START* -1 *TIMES* *U2 STIM*(4,4) $*G_DPWR* – Пример 2 $*G_DGND* 4 3 2 1 +*IO_STD* +0*ns* 0 +*label=start* +25*ns INCR BY* 1 +50*ns GOTO START* *UNTIL GT B* +100*ns F*	*U1 STIM*(1,1) $*G_DPWR* – Пример 3 +$*G_DGND IN* +*IO_STD* +0*ns* 0 +*LABEL=START* +50*ns* 1 +100*ns* 0 +150*ns GOTO START* -1 *TIMES* Пример 4 *T D*(*IN8,IN7,IN6,IN5*, *IN4,IN3,IN2,IN1*) 0 ??101010 100*n* ??0100*FR* 200*n* ??010001 300*n* ??1011*XZ* 400*n* 00000000	Пример 5 *T* *D*(4,3,2,1) 0 0000 25*n* 0001 50*n* 0010 75*n* 0011 100*n* 0100 125*n* 0101 150*n* 0110 175*n* 0111 200*n* 1000 225*n* 1001 250*n* 1010 275*n* 1011 300*n* 1100 425*n* 1111

Команда ***FORMAT*:** <массив форматов> – определяет формат значений в *stim*-командах: 1 = двоичный, 3 = восьмеричный, 4 = шестнадцатеричный. Сумма этих цифр в параметре <*value*> команды должна быть равна количеству выходов двоичного генератора.

Примеры: <*value*> = 1311 – шесть выходов определенных тремя бинарными и одним восьмеричным значением.

<*value*> = 44 – восемь выходов определенных двумя шестнадцатеричными значениями.

<*value*> = 1 – один выход представленный одним двоичным значением.

Команда ***COMMAND*:** – определяет команды для генератора.

Имя генератора также может быть параметром, если его указать в выражении ***.define*** в текстовом окне (*Split text*).

Примеры: В строке ***COMMAND*:**

0*ns* 0 *label=start* 50*ns* 1 100*ns* 0 150*ns goto start* -1 *times*

0ns 42 +10*C A*3 +10*C* 0*F* 25*C* 79.

В текстовом файле *Split text* в выражении ***.define*:**

.define AIN +0*ns* 1011 +100*ns* 0011 +200*ns* 1010 +300*ns RND* +400*ns* 100?

Команда ***COMMAND*** <команда>* – описывают типы генерируемого выхода. Выход генерируется следующими *stim* – командами:

<*time*> <значение>,

<*LABEL*=<название метки>,

<*time*> *INCR BY* <значение>,
<*time*> *DECR BY* <значение>,
<*time*> *GOTO* <название метки> <*repeat*> *TIMES*,
<*time*> *GOTO* <название метки> *UNTIL GT* <значение>,
<*time*> *GOTO* <название метки> *UNTIL GE* <значение>,
<*time*> *GOTO* <название метки> *UNTIL LT* <значение>,
<t*ime*> *GOTO* <название метки> *UNTIL LE* <значение>.

Параметр <*time*> указывает время, когда генерируется новое <значение> команд *INCR*, *DECR*, или *GOTO*. <*Time*> может быть указано в секундах или циклах. Циклы указываются с помощью символа "*C*". Значение *TIMESTEP* будет умножаться на то, что стоит перед символом "*C*", чтобы определить значение в реальных секундах. Время может быть объявлено относительно предыдущего времени с помощью знака "+", например, +10c или +50*ns*.

Параметр <название метки> указывает начало цикла. *GOTO* <название метки> переведет исполнение программы к следующему необозначенному меткой оператору после выражения *LABEL*=<название той же метки>, а <*repeat*> указывает сколько раз повторять цикл. Значение –1 создает вечный цикл.

Время в командах *stim* может идти по возрастающей. Выражение *GOTO* должно обратиться к предыдущей метке <название метки>.

Параметр <значение> указывает значения для выходных зажи-мов генератора. Формат значений определяется командой *Format* 0, 1, *R* – фронт, *F* – спад, *X* – неопределенность, *Z* – высокий импеданс, *RND* – случайное число, ? – случайный символ, а также могут быть использованы бинарные, восьмеричные и шестнадцатеричные числа. Параметры *RND* и ? – оба случайно принимают значение от 0 до 1. Значение *RND* покрывает все символы в <значении> в то время, как команда ? повлияет лишь на один символ.

Команда ***I/O MODEL*:** <название модели ввода-вывода> – определяет имя модели для описания ввода-вывода.

Примеры: *IO_STD*; *IO_ACT*; *IO_HC*.

Команда ***TIMESTEP*:** <размер временного шага> – распространяется только на значения в *stim*-командах, которые имеют суффикс "*C*" и определяет количество секунд в одном временном шаге. Значения, указанные в секундах не будут затронуты.

Примеры: 5*n*
10*n*

Команда ***IO_LEVEL*:** <значение выбора интерфейса подсхемы> – выбирает один из четырех интерфейсов подсхем *AtoD* или *DtoA*. Это подсхема, которая будет вызываться, когда аналоговое устройство будет подключено к генератору. По умолчанию – 0.

0 = значение *DIGIOLVL* в Глобальных Установках.
1 = *AtoD1/DtoA1*
2 = *AtoD2/DtoA2*
3 = *AtoD3/DtoA3*
4 = *AtoD4/DtoA4*

Пример: 4

Команда ***POWER NODE*:** <цифровой разъем питания> – определяет цифровой разъем питания, который будет использоваться подсхемой в случае подключения аналогового устройства к генератору.

Пример: $*G_DPWR*

Команда ***GROUND NODE*:** <цифровой узел заземления> – определяет цифровой узел заземления, который будет использоваться подсхемами интерфейса, если аналоговое устройство связано с генератором. Пример: $*G_DGND*

Параметр <узел>* – для компонентов схемы *SPICE* определяет названия узлов выхода. Для схематического компонента эти данные автоматически соответствуют количеству выходов.

Параметр <размер шага> команды *TIMESTEP* определяет количество секунд за один цикл. Промежутки времени можно указывать в секундах и циклах размеров в шаг, с помощью знака "c". Время вычисляется умножением количества шагов на размер шага. По умолчанию шаг равен нулю.

Модели компонент имеют две составляющие: схемотехническую и функциональную. Схемотехническая (пространственная) модель представляется на экране в виде чертежа и настраивается в специальном **окне настройки**, которое открывается всякий раз при вызове на экран текущего компонента.

Функциональная модель скрыта от пользователя и представляется либо в окне, либо в специальном файле, **называемом *Split Text***, который вызывается при помощи одноименной команды в меню *Windows*.

Стандартные типы логических устройств

Стандартные типы логических устройств приведены на рис. 1.2 и в табл. 1.3, *а*.

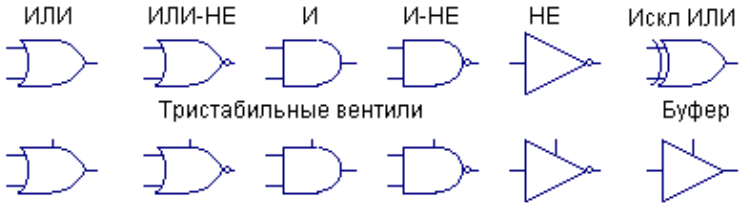


Рис. 1.2 Стандартные типы логических устройств в стандарте *ANSI*

1.3, *а* Типы логических схем

Тип	Параметры	Узлы	Описание
and	<количество входов>	*in*,out*	Вентиль И
buf		*in,out*	повторитель
inv		*in,out*	инвертор
nand	<количество входов>	*in*,out*	вентиль И-НЕ
nor	<количество входов>	*in*,out*	ИЛИ-НЕ
nxor		*in1,in2,out*	ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ
or	<количество входов>	*in*,out*	Вентиль ИЛИ
xor		*in1,in2,out*	ИСКЛЮЧАЮЩЕЕ ИЛИ Exclusive OR gate

Формат и список параметров схемотехнической модели, приводимой в окне настройки, для логических элементов приведен ниже. Знак + означает, что далее следует программная строка. Знак * в таблице и модели означает, что входов и выходов может быть более 1.

Формат: *U*<название> <тип вентиля> [(<параметры>)*]

+<питание> <земля>

+<вход>* <выход>*

+<название временной модели>

<название модели ввода/вывода>

+[*MNTYMXDLY*=<выбор значения задержки>]

+[*IO_LEVEL*=<выбор значения подсхемы интерфейса>]

Пример: Вентиль И-НЕ с тремя входами:

U1 NAND(3)

+$*G_DPWR* $*G_DGND*

+*IN1 IN2 IN3 OUT*

+*D0_GATE IO_STD*

+*MNTYMXDLY*=0

+*IO_LEVEL*=2

Команда ***PART*:** <название элемента> – определяет символьное название элемента. Примеры: *U1 Uor USELECT*

Команда ***TIMING MODEL*:** <название временной модели> – определяет название временной модели. Описание временной модели можно осуществить либо в текстовой области (файл *Split text*), либо в окне настройки схемы или в библиотеке.

Примеры: *D0_gate*
DLY1
GATEDLY

Команда ***I/O MODEL*:** <название модели ввода-вывода> – дает название блоку определения модели ввода-вывода. Описание этой модели можно также осуществить в окне схемы или в библиотеках.

Примеры: *IO_STD*
IO_ACT_OC
IO_HC

Команда ***MNTYMXDLY*:** <значение выбора задержки> – служит для выбора минимальной, типичной, или максимальной задержки для временной модели вентиля. По умолчанию значение 0.

Если *MNTYMXDLY* = 0 – берется значение *DIGMNTYMX* в Глобальных Установках (*Global Settings*) в меню *Options*. Если *MNTYMXDLY:*

=1 – берется минимальная задержка из временной модели,

=2 – берется типичная задержка,

=3 – берется максимальная задержка,

=4 – берется усредненная задержка (минимум/максимум).

Команда ***IO_LEVEL*:** <значение выбора подсхемы интерфейса> – определяет значение *IO_LEVEL* для выбора одного из четырех стандартных интерфейсов. В *MC5* переход от цифрового сигнала к аналоговому осуществляется через логические подсхемы. Этот параметр как раз и выбирает такую подсхему, которая будет вызываться каждый раз, когда аналоговое устройство будет подсоединяться к вентилю. По умолчанию значение равно 0.

IO_LEVEL = 0 – берется значение *DIGIOLVL* в Глобальных Установках (Global Settings) в меню Options.

=1 – берется *AtoD1/DtoA1*
=2 – берется *AtoD2/DtoA2*
=3 – берется *AtoD3/DtoA3*
=4 – берется *AtoD4/DtoA4*

Команда ***POWER NODE*:** <питание> – определяет узел питания, который будет использоваться подсхемой интерфейса, если аналоговое устройство подключено к вентилю.

Пример: $*G_DPWR*

Команда ***GROUND NODE*:** <земля> – определяет узел заземления, который будет использован подсхемой интерфейса в случае подключения аналогового устройства к вентилю.

Пример: $*G_DGND*

Формат описания временной модели, приводимой в файле *Split Text**:.model*** <название временной модели> *UGATE* ([параметры модели]) (табл. 1.3, *б*).

Пример: *.model DLY1 UGATE* (*tplhty*=10*ns tplhmx*=25*ns tphlt*y=12*ns tphlmx*=27*ns*)

1.3, *б* Таблица временных параметров стандартных вентилей

Название	**ЗАДЕРЖКА**	Единицы	По умолчанию
tplhmn	low to high, min	секунды	0
tplhty	**low to high, typ**	секунды	0
tplhmx	*low to high, max*	секунды	0
tphlmn	*high to low, min*	секунды	0
tphlty	high to low, typ	секунды	0
tphlmx	*high to low, max*	секунды	0

Delay low to high, min – минимальное (номинальное, максимальное) время задержки вход–выход при переключении от 0 к 1; *Delay high to low, min* – минимальное (номинальное, максимальное) время задержки вход–выход при переключении от 1 к 0.

Типы тристабильных ИС

Типы тристабильных схем приведены в табл. 1.4. Параметры зависят от выбранного типа и также приводятся в таблице. Значения параметров должны идти сразу же за указанием типа тристабильного вентиля.

1.4 Таблица тристабильных вентилей

ТИП ВЕНТИЛЯ	Параметры	Узлы	Описание
*and*3	<количество входов>	*in*,en,out*	Вентиль *and*
*buf*3		*in,en,out*	Буфер
*inv*3		*in,en,out*	Инвертор
*nand*3	<количество входов>	*in*,en,out*	Вентиль *nand*
*nor*3	<количество входов>	*in*,en,out*	Вентиль *nor*
*nxor*3		*In*1,*in*2,*en*,*out*	Исключающее *nor*
*or*3	<количество входов>	*In*,en,out*	Вентиль *or*
*xor*3		*In*1,*in*2,*en*,*out*	Исключающее *or*

Формат входов и выходов приведен в колонке узлы. Знак * обозначает один или больше узлов, а когда звездочки нет, это означает наличие одного контакта. Вход *enable* (управляющий) всего один.

Формат и параметры моделей тристабильной схемотехники аналогичны схемам обычной логики, за исключением наличия сигнала enable.

Триггеры (*flip-flop*). Программой *MC* поддерживается три типа триггеров: *RS*-триггер с раздельными входами, универсальный *JK*-триггер и *D*-триггер (рис. 1.5).

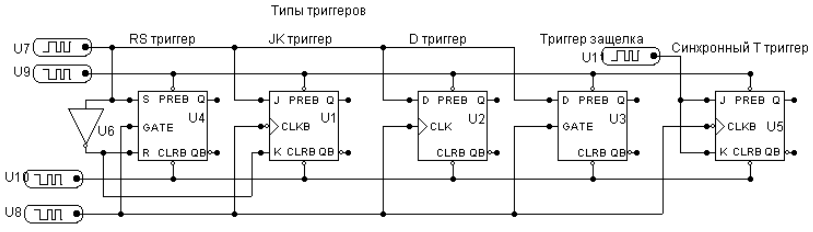


Рис. 1.5 Примитивы триггеров

Формат и параметры модели схемы триггеров *JK* и *D*, приводимых в окне настройки:

U<имя> *JKFF* (<количество триггеров>)

+<питание> <заземление>

+<контакт *presetbar*> <контакт clearbar> <контакт *clockbar*>

+<первый контакт *J*>...<последний контакт *J*>

+<первый контакт *K*>...<последний контакт *K*>

+<первый контакт *Q*>...<последний контакт *Q*>

+<первый выход *QBAR*>...<последний выход *QBAR*>

+<название временной модели> <модель ввода-вывода>

+[*MNTYMXDLY*=<значение параметра задержки>]
+[*IO_LEVEL*=<значение выбора подсхемы интерфейса>]
U<название> *DFF* (<количество триггеров>)
+<питание> <заземление>
+<контакт presetbar> <контакт clearbar> <контакт clock>
+<первый вход *D*>...<последний вход *D*>
+<первый выход *Q*>...<последний выход *Q*>
+<первый выход *QBAR*>...<последний выход *QBAR*>
+<название временной модели> <модель ввода-вывода>
+[*MNTYMXDLY*=<значение выбора задержки>]
+[*IO_LEVEL*=<значение выбора подсхемы интерфейса>]

Примеры:
```
U1 JKFF(2) $G_DPWR  $G_DGND
+PREBAR  CLRBAR  CLK
+J1 J2 K1 K2 Q1 Q2 Q1BAR Q2BAR
+D0_EFF IO_STD  IO_LEVEL=1
U4 DFF(1) $G_DPWR $G_DGND
+PREB CLRB CLKIN
+DIN Q QBAR  DLY_DFF  IO_ACT
```

Параметр <Количество триггеров> – определяет их количество в схеме, выводы preset – установить в 1, *clear* – очистить 0, и *clock* – такт – общие для всех триггеров. Выводы *preset* и *clear* служат для начальной установки триггера *JK* и *D*. Синхронизированные триггеры изменяют свое состояние по фронту или спаду тактирующего импульса *clock*, так *JK* – по спаду импульса и *D* – по фронту. Логика переключения триггеров приводится в таблицах истинности.

Команда ***PART*:** <название> – определяет название элемента.

Примеры:
```
U1
Uff
UJK
```

Команда ***TIMING MODEL*:** <название временной модели> – описывает название временной модели. Временная модель может быть определена в текстовой области *Split Text*, в окне настройки или в библиотеках.

Примеры:
```
D0_EFF
DLY4
JKDLY
```

Команда ***I/O MODEL*:** <название модели ввода-вывода> – определяет название модели ввода-вывода. Описание этой модели также возможно несколькими способами.

Примеры:
```
IO_STD_ST
IO_AC
IO_S
```

Команда ***MNTYMXDLY*:** <значение выбора задержки> – определяет значение *MNTYMXDLY* для выбора минимальной, типичной или максимальной задержки временной модели. Значение по умолчанию 0.

Если *MNTYMXDLY* = 0 – значение *DIGMNTYMX* берется в Глобальных Настройках. Если *MNTYMXDLY*:

= 1 – минимальная задержка

= 2 – типичная задержка

= 3 – максимальная задержка

= 4 – относительная задержка (минимум/максимум)

Команда ***IO_LEVEL*:** <значение выбора подсхемы интерфейса> – определяет параметр *IO_LEV*EL для выбора одной из необходимых четырех подсхем *AtoD* или *DtoA*, которая будет использоваться, когда аналоговое устройство подключено к триггеру. Значение по умолчанию 0.

0 = значение *DIGIOLVL* в Глобальных Настройках.

1 = *AtoD1/DtoA1*

2 = *AtoD2/DtoA2*

3 = *AtoD3/DtoA3*
4 = *AtoD4/DtoA4*

1.5 Некоторые временные параметры триггеров

Название	Задержка	Единицы	По умолчанию
tppcqlhmn	от *preb/clrb* до *q/qb* от *low* до *hi*, минимум	*Sec.*	0
tppcqlhty	от *preb/clrb* до *q/qb* от *low* до *hi*, номинал	*Sec.*	0
tppcqlhmx	от *preb/clrb* до *q/qb* от *low* до *hi*, максимум	*Sec.*	0
tppcqhlmn	от *preb/clrb* до *q/qb* от *hi* до *low*, минимум	*Sec.*	0
tppcqhlty	от *preb/cl*rb до *q/qb* от *hi* до *low*, номинал	*Sec.*	0
tppcqhlmx	от *preb/clrb* до *q/qb* от *hi* до *low*, номинал	*Sec.*	0
twpclmn	*min preb/clrb* шириной *low*, минимум	*Sec.*	0
twpclty	*min preb/clrb* шириной *low*, номинал	*Sec.*	0
twpclmx	*min preb/clrb* шириной *low*, максимум	*Sec.*	0
tpclkqlhmn	от фронта *clk/clkb* до *q/qb* от *low* до *hi*, минимум	*Sec.*	0
tpclkqlhty	от фронта *clk/clkb* до *q/qb* от *low* до *hi*, номинал	*Sec.*	0
tpclkqlhmx	от фронта *clk/clkb* до *q/qb* от *low* до *hi*, максимум	*Sec.*	0
tpclkqhlmn	от фронта *clk/clkb* до *q/qb* от *hi* до *low*, минимум	*Sec.*	0
tpclkqhlty	от фронта *clk/clkb* до *q/q*b от *hi* до *low*, номинал	*Sec.*	0
tpclkqhlmx	от фронта *clk/clkb* до *q/qb* от *hi* до *low*, максимум	*Sec.*	0

Команда ***POWER NODE*:** <питание> – определяет подсхему питания, которая будет задействована в случае подсоединения аналогового устройства к триггеру.

Пример: $*G_DPWR*

Команда ***GROUND NODE*:** <заземление> – определяет подсхему заземления, которая будет использоваться в случае подключения аналогового устройства к триггеру.

Пример: $*G_DGND*

Описание временной модели, приводимой в *Split Text*:

.model <название временной модели> *UEFF* ([параметры модели])

Пример: *.model JKDLY UEFF* (*tppcqlhty*=10*ns tppcqlhmx*=25*ns tpclkqlhty*=12*ns*+*twpclty*=15*ns tsudclkty*=4*ns*)

В табл. 1.5 показаны некоторые параметры триггеров.

Программируемые логические матрицы (ПЛМ) (рис. 1.6)

Формат и параметры схемной модели ПЛМ (*SPISE*), приводимой в окне настройки:

U<название> <тип матрицы> (<количество входов>,<количество выходов>)
+<цифровой разъем питания> <цифровое заземление>
+<входной узел>* <выходной узел>*
+<имя временной модели> <имя модели ввода-вывода>
+[*FILE*=<константа имени файла или выражение имени файла>]
+[*DATA*=<флаг>$<данные программы>$]
+[*MNTYMXDLY*=<значение выбора задержки>]
+[*IO_LEVEL*=<значение выбора интерфейса подсхемы>]

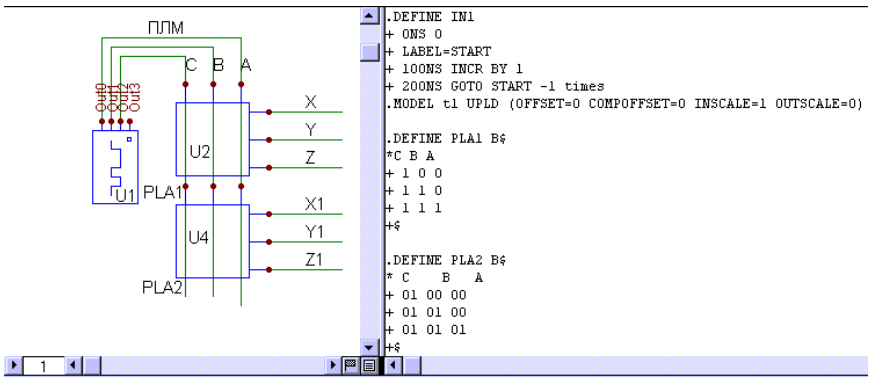


Рис. 1.6 Модели программируемых логических матриц (ПЛМ)

Примеры:

Первый пример логических выражений (~ , &, | – инверсия, конъюнкция, дизъюнкция):

Out1 = (*In1* & *In3*)
Out2 = (*In1* & *In2*)
Out3 = (*In2* & *In3*)

Модель ПЛМ:

U1 PLAND(3,3) $*G_DPWR* $*G_DGND*
+*I1 I2 I3 O1 O2 O3*
+*D0_PLA IO_STD*
+*DATA=B*$
In1 In2 In3
+1 0 1 ;*O1*
+1 1 0 ;*O2*
+0 1 1 ;*O3*
+ $

Второй пример логических выражений:

O1 = (~*I1* | ~*I2* | *I3*)
O2 = (~*I1* | *I2* | *I3*)
O3 = (*I1* | ~*I3*)
O4 = (~*I1* | ~*I2* | ~*I3*)

Модель ПЛМ для данных выражений:

U2 PLORC(3,4) $*G_DPWR* $*G_DGND*
+*I1 I2 I3 O1 O2 O3 O4*
+*DLYPLOR IO_LS*

+DATA=B$
I1 I2 I3
TF TF TF ; прямые и инверсные входы
+01 01 10 ;*O1*
+01 10 10 ;*O2*
+10 00 01 ;*O3*
+01 01 01 $;*O4*

Команда ***PART*:** <название> – определяет название элемента.

Примеры: *U4*
UAD

Команда ***TIMING MODEL*:** <название временной модели> – определяет имя для выражения временной модели, которую можно определить через текстовую область, в окне настройки схем или в библиотеках.

Примеры: *D*0*_ADC*
DLY_B
ADCDLY

Команда ***I/O MODEL*:** <название модели *I/O*> – имя модели ввода-вывода, которая также может быть определена тремя различными способами.

Примеры: *IO_STD*
IO_PLD

Команда ***FILE*:** <константа имени файла ИЛИ выражение имени файла> – определяет файл *JEDEC*, который содержит программу для матрицы.

Команда ***DATA*:**<<*data constant*> ИЛИ <<флаг>$<данные программы ПЛМ>$>> – определяет данные программы ПЛМ, а также их вид: флаг *B* – бинарный, флаг *O* – восьмеричный или флаг *X* – шестнадцатеричный. Имя, помещенное в <константа данных>, может быть названием программы ПЛМ, определенной в текстовом поле *Split Text* с директивой ***.define***.

Примеры: *B*$ 1 0 1 1 1 0 0 1 1 $ – бинарные данные;
O$ 2 5 4 7 3 0 1 2 $ – восьмеричные данные.

Шестнадцатеричные данные в подпрограмме *PLORDATA* с директивой *.define* показаны ниже. В данном случае подпрограмма *PLORDATA* помещается в файл *Split text*, а в атрибуте *DATA* указывается имя подпрограммы.

define PLORDATA
+X $
+A 2 7 5
+3 1 9 *C*
+D E 2 6 $

Команда ***MNTYMXDLY*:** <значение выбора задержки> – выбирает минимальную, типичную или максимальную задержку для временной модели. По умолчанию равно 0.

0 = значение ***DIGMNTYMX*** в Глобальных Установках
1 = минимальная задержка
2 = типичная задержка
3 = максимальная задержка
4 = наибольшая задержка (минимум/максимум)

Команда ***IO_LEVEL*:** <значение выбора интерфейса подсхемы> – выбирает один из четырех интерфейсов подсхем *AtoD* или *DtoA*, которая будет вызываться, когда аналоговое устройство будет подключено к ПЛМ. По умолчанию – 0.

0 = значение *DIGIOLVL* в Глобальных Установках
1 = *AtoD1/DtoA1*
2 = *AtoD2/DtoA2*

3 = *AtoD3/DtoA3*

4 = *AtoD4/DtoA4*

Команда ***POWER NODE*:** <цифровой разъем питания> – определяет цифровой разъем питания, который будет использоваться подсхемой в случае подключения аналогового устройства к ПЛМ.

Пример: $*G_DPWR*

Команда ***GROUND NODE*:** <цифровой узел заземления> определяет цифровой узел заземления.

Пример: $*G_DGND*

Имеются два способа программирования *PLA*. Первый способ состоит в том, чтобы обеспечить данные в файле формата *JEDEC*. Эти файлы обычно создаются специалистами и обеспечивают программируемость ПЛМ в процессе "изготовления". Второй метод состоит в том, чтобы ПЛМ программировались пользователем, что обеспечивается включением данных непосредственно в модель SPICE в команду *DATA*. Кроме того, команда *DATA* может содержать в качестве параметра ссылку на имя текстовой матрицы, помещаемой пользователем в *Split Text*. Текстовые строки матрицы содержат значения данных, которые программируют *PLA*. Если значение переменной столбца равно "0", то входной столбец не связан с вентилями. Если значение переменной "1", входной столбец связан с вентилями. Данные считываются из матрицы слева направо, т.е. начальные данные находятся в нулевом (левом) адресе. Матрица должна быть заключена в формат со знаком доллара $ в начале и в конце, перед первым знаком $ помещается флаг с указанием типа данных.

Формат описания временной модели ПЛМ, помещаемой в *Split Text*: *.mode*l <название модели> UPLD ([параметры]). Пример: *.model PLDMOD UPLD* (*tplhty*=10*ns tplhmx*=25*ns tphlty*=12*ns tphlmx*=27*ns*)

1.6 Таблица временных параметров ПЛМ

Название	Параметр	**ЕДИНИЦЫ**	**ПО УМОЛЧАНИЮ**
tplhmn	Задержка: *in* до *out*, *low* до *high*, минимум	Sec.	0
tplhty	Задержка: *in* до *out*, *low* до *high*, номинал	*Sec.*	0
tplhmx	Задержка: *in* до *out*, *low* до *high*, максимум	*Sec.*	0
tphlmn	Задержка: *in* до *out*, *high* до *low*, минимум	*Sec.*	0
tphlty	Задержка: *in* до *out*, *high* до *low*, номинал	*Sec.*	0
tphlmx	Задержка: *in* до *out*, *high* до *low*, максимум	*Sec.*	0
offset	Файл *JEDEC*: адрес первого входа и первой программы вентиля		0
compoffset	Файл *JEDEC*: адрес дополнения первого входа и первой программы вентиля		1

Обозначения наименования параметров задержек, используемых во временной модели ПЛМ, а также единицы измерения приведены в табл. 1.6.

Аналого-цифровые преобразователи (АЦП) (рис. 1.7)

Формат и параметры, схемотехнической модели (*SPICE*), приводимой в окне настройки:

U<название> *ADC* (<количество битов>)
+<цифровой разъем питания> <цифровое заземление>
+<аналог.вход> <питание> <заземление> <преобразование>
+<статус> <*over range node*>
+<выход *msb*> ... <выход *lsb*>
+<имя временной модели> <имя модели ввода-вывода>
+[*MNTYMXDLY*=<значение выбора задержки>]
+[*IO_LEVEL*=<значение выбора интерфейса подсхемы>]

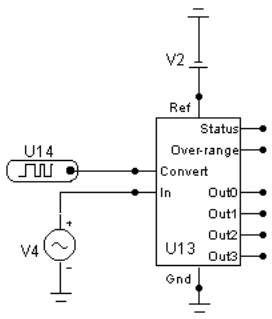


Рис. 1.7 Аналого-цифровой преобразователь

Примеры: *U1 ADC*(8) $*G_DPWR* $*G_DGND*
+ansig ref1 0 *conv1 stat1 over1*
+out7 out6 out5 out4 out3 out2 out1 out0
+D0_ADC IO_STD

Команда ***PART*:** <название> – определяет название элемента.

Примеры: *U4*
UAD

Команда ***TIMING MODEL*:** <название временной модели> – определяет тип временной модели, которую можно задать через текстовую область *Split Text*, схемотехнически или в библиотеках.

Примеры: *D0_ADC*
DLY_B
ADCDLY

Команда ***I/O MODEL*:** <название модели *I/O*> – модель ввода-вывода, также может быть определена тремя различными способами.

Примеры: *IO_STD*
IO_ACT
IO_HC

Команда ***MNTYMXDLY*:** <значение выбора задержки> – выбирает минимальную, типичную или максимальную задержку для временной модели. По умолчанию равно 0.

0 = значение *DIGMNTYMX* в Глобальных Установках

1 = минимальная задержка

2 = типичная задержка

3 = максимальная задержка

4 = наибольшая задержка (минимум/максимум)

Команда ***IO_LEVEL*:** <значение выбора интерфейса подсхемы> – выбирает один из четырех интерфейсов подсхем *AtoD* или *DtoA*. По умолчанию – 0.

0 = значение *DIGIOLVL* в Глобальных Установках

1 = *AtoD1/DtoA1*

2 = *AtoD2/DtoA2*

3 = *AtoD3/DtoA3*

4 = *AtoD4/DtoA4*

Команда ***POWER NODE*:** <цифровой разъем питания> – определяет цифровой разъем питания, который будет использоваться подсхемой в случае подключения аналогового устройства к АЦП.

Пример: $*G_DPWR*

Команда ***GROUND NODE*:** <цифровой узел заземления> – определяет цифровой узел заземления, который будет использоваться подсхемами интерфейса, если аналоговое устройство связано с АЦП.

Пример: $*G_DGND*

Выход АЦП будет эквивалентен ближайшему целому числу, рассчитанному по формуле $(V(in,gnd)/V(ref,gnd))*(2^{(\text{количество битов})})$.

Если число больше, чем $2^{(\text{количество битов})}$-1, то все выходы будут равны 1 и выход "*over*" будет установлен в 1. Если целое число меньше нуля, то все выходы будут равны 0 и выход "*over*" будет установлен в 1.

Вначале моделирования выходы *Out0*, *Out1*, ... *Out N* будут находиться в состоянии неопределенности "*X*", а выход *STATUS* станет равен "1" с задержкой *TPCS* секунд после фронта сигнала *CONVERT*, а *TPSD* секундами позже, *Out0*, *Out1*, ... *Out N* выходы станут равными истинным данным, а *TPDS* секундами позже, выход *STATUS* будет равен состоянию "0".

Синтаксис временной модели в *Split Text*:

.model <название временной модели> *UADC* ([параметры модели])

Пример:

.model ADCMOD UADC (*tpcsty*=10*ns tpsdty*=25*ns tpdsty* =12*ns*)

Обозначения наименования параметров задержек, используемых во временной модели АЦП, а также единицы измерения приведены в табл. 1.8.

1.8 Таблица временных параметров

Название	Задержка	Единицы	По умолчанию
tpcsmn	фронт конверт к фронту статуса, минимум	*Sec.*	0
tpcsty	фронт конверт к фронту статуса, номинал	*Sec.*	0
tpcsmx	фронт конверт к фронту статуса, максимум	*Sec.*	0
tpsdmn	фронт статуса к выходу и диапазонному сигналу, минимум	*Sec.*	0
tpsdty	фронт статуса к диапазонному сигналу, максимум	*Sec.*	0
tpsdmx	фронт статуса к выходу и диапазонному сигналу, максимум	*Sec.*	0
tpdsmn	выход и диапазонный сигнал к спаду статуса, минимум	*Sec.*	0
tpdsty	выход и диапазонный сигнал к спаду статуса, номинал	*Sec.*	0
tpdsmx	выход и диапазонный сигнал к спаду статуса, максимум	*Sec.*	0

Цифро-аналоговые преобразователи (ЦАП) по модели аналогичны АЦП и представляют собой следующую конструкцию:

Формат схемотехнической модели – *SPICE*:

U<название> *DAC* (<количество битов>)
+<цифровой разъем питания> <цифровое заземление>
+<выход> <питание> <заземление>
+<вход *msb*> ... <вход *lsb*>
+<имя временной модели>
+<имя модели ввода-вывода>
+*MNTYMXDLY*=<значение выбора задержки>
+*IO_LEVEL*=<значение выбора интерфейса подсхемы>.

Примеры: *U1 DAC*(8) $*G_DPWR* $*G_DGND*
+*outsig ref1* 0
+*in7 in6 in5 in4 in3 in2 in1 in0*
+*D0_DAC IO_STD*

1.2 СХЕМОТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ И АНАЛИЗ СРЕДСТВАМИ *MICRO-CAP*

На рис. 1.8 – 1.10 приведены центральные окна программ *MC5*, *MC6* и *MC7*. Учитывая, что меню команд у всех программ одинаковы, а строки инструментов отличаются, по большому счету незначительно, дальнейшее изложение будет ориентировано на *MC5*.

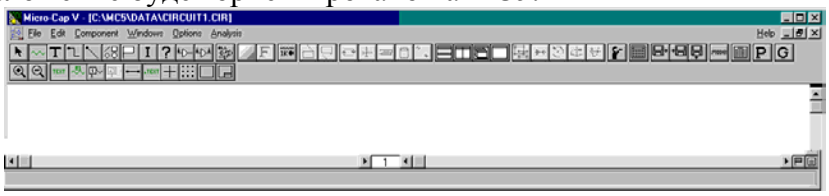


Рис. 1.8 Центральное окно программы *MC5*

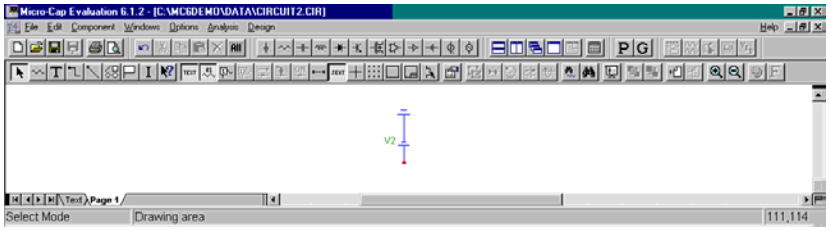


Рис. 1.9 Центральное окно программы *MC6*

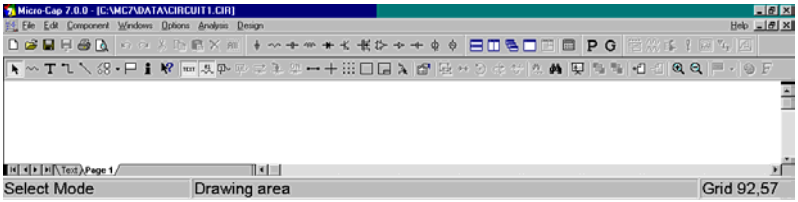


Рис. 1.10 Центральное окно программы *MC7*

Проектирование схем выполняется в центральном окне системы *Micro-Cap* путем выбора изображений необходимых компонентов из меню *Component* и размещения их в нужном месте экрана. При этом приходится выбирать компоненты из библиотек, перемещать изображения компонентов, поворачивать их вокруг базовой точки, указывать их имена и номиналы.

Например, в главном меню *Component => Digital Primitives => Standard Gates => Or Gates => Or2* с помощью наведенного указателя мыши на позицию *Or2* и щелчка левой клавиши (для *MC6* и *MC7* – следует удерживать ее после нажатия). Размещая указатель мыши на то место поля, где должен быть установлен вентиль и щелчка левой клавишей (для *MC6* и *MC7* – отпускания). На экране появится изображение двухвходового логического элемента ИЛИ. Оно сразу занимает правильное положение, но имеется возможность вращать его, если нажимать правую клавишу, **не отпуская левую**. Как только положение вентиля будет выбрано, отпускается левая клавиша.

В этот момент на экране появится окно настроек с перечислением типов модели вентиля – идеального или реального (*DO GATE*, *DLY_TTL*). С помощью курсора и щелчка левой клавиши мыши выбирается желаемый тип вентиля. Выбранный тип появится в меню. На этом ввод первого компонента завершается. Аналогичным образом вводятся другие компоненты схемы. Одновременно с этим в окне Split Text появится описание модели вентиля, например, в таком виде:

.MODEL DLY_TTL UGATE (*TPLHTY*=11*NS TPLHMX*=22*NS TPHLTY*=8*NS TPHLMX*=15*NS*)

При построении схемы придется неоднократно соединять компоненты отрезками проводов. Удобно делать это с помощью пиктограммы с изображением провода. Для этого необходимо установить указатель мыши на начало отрезка и нажать левую клавишу мыши, затем переместить курсор на нужную позицию и опустить левую кнопку мыши – будет построен нужный отрезок. Так можно "нарисовать" любые соединительные линии (рис. 1.11).

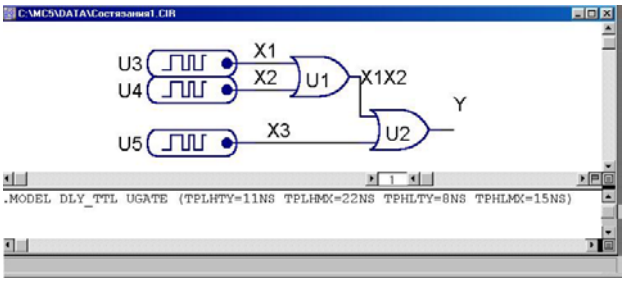


Рис. 1.11 Полностью собранная схема

Системы схемотехнического моделирования *Micro-Cap* выгодно отличаются от других систем (например, *PSPICE* или *Electronic Work bench*, *Serenade*) своим сервисом. Самый трудный этап проектирования (задание схемы и ее топологическое и математическое описание) в них реализован простым и наглядным графическим диалогом. Он напоминает сборку схем с помощью аппликативного конструктора, содержащего компоненты электронных схем, из которых пользователь собирает нужную схему.

Особенностью *MC* является то, что для задания схем и управления системой в ходе анализа "корректности" построения схем и их моделирования не требуется знания никаких входных языков. Результаты анализа получаются как в числовой (табличной форме), так и в виде графиков, напоминающих осциллограммы, получаемые при исследовании схемы с помощью электронного осциллографа, характериографа или измерителя частотных характеристик.

Малосигнальный анализ во временной области

После задания параметров всех компонент собранной схемы, она считается введенной в ЭВМ, после чего MC *автоматически формирует сложные системы дифференциальных уравнений состояния, описывающих работу схем. Расчет переходных процессов производится интегрированием уравнений состояния конечно-разностным методом с переменным шагом во времени (идет адаптация к скорости моделируемых процессов). Переменные состояния определяют коэффициенты или условия мате-*

матической модели, которая представляет схему в любой момент времени. Эти переменные должны быть установлены в начальные значения для запуска анализа.

MC выбирает начальные условия следующим образом: когда впервые выбирается временной анализ, все переменные состояния устанавливаются в ноль и все цифровые уровни – в "*X*". Это называется начальной инициализацией (*setup initialization*). Каждый раз, когда запускается новый процесс анализа, нажатием *F*2 или кнопки *Run*, выполняется текущая инициализация. Для того, чтобы решить что делать, *MC* читает опции переменных состояния из меню анализа. С помощью команд можно установить один из трех режимов инициализации:

Временной анализ: диалоговое окно управления

***Zero*:** переменные состояния, напряжения выводов, все токи приравниваются к 0. Цифровые уровни устанавливаются в "*X*", а для триггеров их выходы *Q* и *QB*, устанавливаются в "0", "1", или "*X*" в зависимости от глобального значения *DIGINITSTATE*. Это значение определено в Глобальных Установках.

***Read*:** *MC* читает переменные из файла *CIRCUITNAME.TOP*. Этот файл создается Редактором Переменных Состояния.

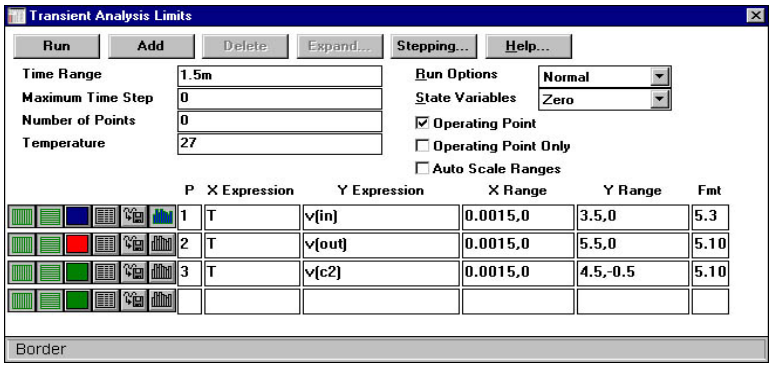
***Leave*:** *MC* ничего не делает с переменными состояния. Он просто не замечает их. При этом:

***First run*:** если переменные не редактировались с Редактором Переменных Состояния, они все же принимают нулевые значения начальной инициализации.

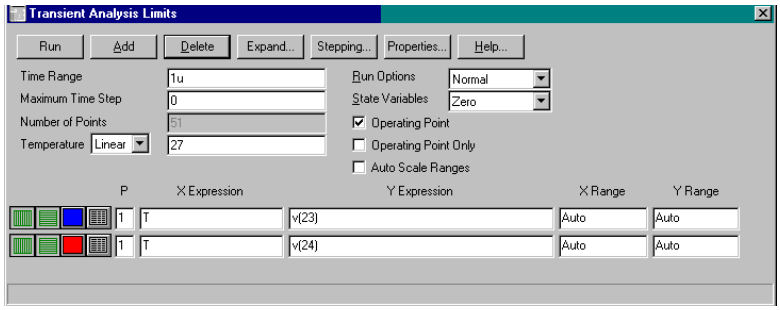
***Later run*:** если переменные не редактировались с Редактором Переменных Состояния, они принимают конечные значения последнего анализа.

***Edited*:** если переменные редактировались с Редактором Переменных Состояния, то их значения берутся из данных редактора. Все подключенные к генераторам последовательностей контакты меняются на их значение *T* = *tmin*.

Когда выполняется *Transient Analysis* (временной анализ) из меню *Analysis*, *MC* тестирует схему на топологическую правильность и правильность задания параметров модели. Если ошибки не найдены, на экран выводится диалоговое окно *TA*-анализа, которое называется *Analysis Limits* (рис. 1.12). Это диалоговое окно позволяет вводить данные, регулирующие режим выполнения анализа, выбирать временной и частотный диапазоны,



а)



б)

Рис. 1.12 Диалоговое окно временного анализа:

а – для *MC5*; *б* – для *MC6* и *MC7*

выбирать шкалу и выводить графики. Диалоговое окно *Analysis Limits* разделено на пять областей: Кнопки, Числовые пределы, Опции форм графика, Выражения и Опции.

Кнопки

***Run*:** запускает процесс анализа. Если нажать кнопку Run из меню *Tool*, или клавишу *F2* – тоже запустится анализ.

***Add*:** добавляет новую строку графика, после строки, где установлен курсор. Строка графика состоит из полей опций и полей выражений, характеризующих оси графика *X* и *Y*.

***Delete*:** удаляет строку графика, которая содержит курсор.

***Expand*:** расширяет текстовое поле, где находится курсор в большое диалоговое окно для редактирования и просмотра. Чтобы использовать эту кнопку, выделите мышью нужное текстовое поле и нажмите кнопку.

***Stepping*:** вызывает диалоговое окно пошагового анализа.

***Help*:** вызывает подсказку.

Поля числовых пределов

***Time Range*:** указывает временные границы анализа. Формат: *tmax*, *tmin*. Например, "3*u*,1*u*" указывает промежуток от 1 мкс до 3 мкс. По умолчанию *tmin* = 0.

***Maximum Time Step*:** обуславливает максимальный временной шаг для анализа. *MC* выбирает наибольший возможный шаг по алгоритму адаптивной временной дискретизации, учитывая параметр *RELTOL*, причем погрешность дискретизации определяется автоматически. Мелкий шаг создает большее количество точек графика, крупный – меньшее. Точность контролируется встроенным механизмом *LTE* (*Local Truncation Error*). По умолчанию значение шага (*tmax*-*tmin*)/50.

***Number of Points*:** указывает количество расчетных точек. По умолчанию – 51.

***Temperature*:** указывает температурные условия работы схемы, которые обычно указываются в градусах по Цельсию. Формат описания следующий: высшая точка, низшая точка, шаг изменения. Температура изменяется от низшей до высшей с указанным шагом. Один полный анализ проводится для каждого температурного изменения. Переменная текущей температуры называется *TEMP* и может использоваться в других выражениях.

Waveform Options* (опции форм графиков):** находятся ниже числовых пределов. Каждая опция свойств графика действует только на график в своей строке. Опции следующие: ***X Log/Linear Scale: линейный или экспоненциальный масштаб по оси *X*. Иконка слева – экспоненциальный масштаб, справа – линейный.

***Y Log/Linear Scale*:** аналогичная опция для оси *Y*.

Color: вызывает меню цветов. Есть 16 цветов для каждого конкретного графика.

***Numeric Output*:** кнопка выбирает график для цифрового вывода. Этот вывод производится в файл *CIRCUITNAME.ANO* и показывается в окне Numeric Output.

***User File*:** если кнопка нажата, тогда выражение *Y* в этой строке будет сохранно в пользовательский файл в табличном виде. Таблица позже могут быть использованы в других схемах пользовательских устройств, которые читают этот файл. Для дальнейшей информации обращайтесь к соответствующим разделам. Количество сохраняемых значений зависит от *RELTOL*: $= 2^{(6-lg(RELTOL))}$. Например, типичные значения *RELTOL Number of values* .001 512 .0001 1024 .00001 2048

***Monte Carlo*:** опция применяет к выходным параметрам алгоритмы Монте-Карло. Эти алгоритмы производят статистический анализ выбранного параметра, причем для анализа может быть выбран только один параметр.

***Plot Group*(*P*):** число от 1 до 9 в этом столбце делит графики на различные группы. Все графики с одним номером находятся в одной группе. Если поле – не заполнено, то это значит, что данный график не выводится на экран.

***Expressions*:** поля указывают диапазон значений шкалы (*X*) и (*Y*) для графиков и выражений. *MC* может обрабатывать большой набор переменных и выражений для каждой оси. Обычно, они легки для

понимания, как *F* (частота) или *V*(1) (напряжение в узле схемы № 1). Хотя, могут быть выражения и сложнее, например, *V*(2)*I(*V1*). Примеры:

D(*A*) – дискретный сигнал в узле *A*
V(*A*) – напряжение в узле *A*
V(*A*,*B*) – разность потенциалов в узлах *A* и *B*
V(*D*) – падение напряжения на элементе *D*
I(*D*) – ток через элемент *D*
I(*A*,*B*) – ток между точками *A* и *B*
IR(*Q*) – ток, например, коллектора транзистора *Q*
VRS(*Q*) – падение напряжения, напр., между базой и эмиттером *Q*
CRS(*Q*) – емкость, напр., между базой и эмиттером *Q*
R(*R*) – сопротивление резистора *R*
C(*X*) – емкость конденсатора или диода *X*
Q(*X*) – добротность конденсатора или диода *X*
L(*L*) – индуктивность катушки или дросселя *L*
X(*L*) – поток в катушке или дросселе *L*
B(*L*) – индукция *B* поля или дросселя *L*
H(*L*) – напряженность *H* поля дросселя *L*
T – время
F – частота
S – комплекс = 2**PI***F***j*
RND – датчик случайных чисел (0<= *RND* <=1)
ONOISE – напряжение помехи на выходном узле
INOISE – напряжение помехи на входе (*ONOISE*/коэф. усиления).

***X expression*:** поля в этом столбце используются для определения выражений оси *X*. Могут использоваться: *T* (время), иногда *H*(*K1*) (поле *H* элемента *K1*).

***Y expression*:** выражения для оси *Y*. Обычно, здесь записаны выражения напряжений между контактами, например, *V*(12,11), или источника тока *I*(*V1*), но иногда записываются и сложные выражения такие, как *V*(*VCC*)**I*(*VCC*) (мощность источника *VCC*).

***X range*:** определяет границы диапазона графиков по оси *X*. Формат: Правая граница, Левая граница. Например, чтобы указать промежуток от 1 до 10 мкс, нужно писать "10*u*,1*u*". Значение левой границы по умолчанию равно нулю. Ключевое слово '*AUTO*' также может быть использовано, тогда все границы будут определены автоматически.

***Y range*:** то же самое для оси *Y*.

***Options*:** опции *Transient Analysis* находятся справа. Эти опции контролируются либо выпадающим списком, либо кнопками. Опции *Run* устанавливаются выпадающими списками и могут быть доступны двумя путями.

Доступные опции

***Run Options. Normal*:** запускает моделирование без записи ее на диск.

Save: записывает результаты моделирования на диск в файл *CIRCUITNAME.TSA*.

***Retrieve*:** загружает сохраненные ранее данные о моделировании из файла *CIRCUITNAME.TSA*.

Опции возымеют действие только после повторного проведения анализа.

***Operating Point*:** кнопка включает режим построения переходного процесса, когда сначала рассчитываются токи и напряжения схемы по постоянному току для каждой точки, а затем по схемам замещения – амплитуды переменного тока, а на графике видно и то и другое.

***Operating Point Only*:** если кнопка включена, то расчет схемы ведется только в режиме переменного тока.

***Auto Scale Ranges*:** кнопка включает режим автоматического разбиения шкалы по осям *X* и *Y*.

Пример сеанса моделирования для схемы (рис. 1.11) приведен на рис. 1.13.

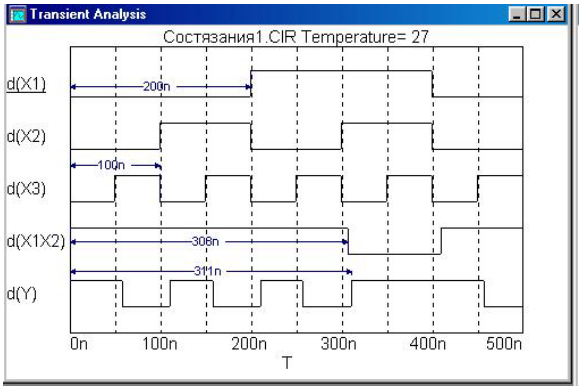


Рис. 1.13 Моделирование работы схемы при подаче на входы импульсных сигналов

Малосигнальный анализ в частотной области

(*AC*-анализ, рис. 1.14)

AC-анализ – частотный линейный анализ, осуществляемый *MC* по следующему алгоритму:

1 Рассчитывается режим схемы по постоянному току.

2 Создаются линейные эквиваленты моделей по переменному току для каждого компонента схемы.

3 Создается набор линейных дифференциальных уравнений.

4 Устанавливается частота входного сигнала fmin.

5 Решаются все уравнения для напряжений и токов во всех узлах схемы.

6 Строятся и выводятся запрошенные переменные.

7 Если частота равна fmax, то расчет заканчивается, если иначе – частота увеличивается на один шаг и переходят к пункту 5.

Частотный анализ используется при построении амплитудно-частотных и фазочастотных характеристик аналоговых схем, а также при расчете реактивной мощности. При этом следует указывать в строке графика выражение *V*(*C1*)**I*(*C1*), а чтобы построить реактивную мощность катушки индуктивности *L1*, указывайте *V*(*L1*)**I*(*L1*).

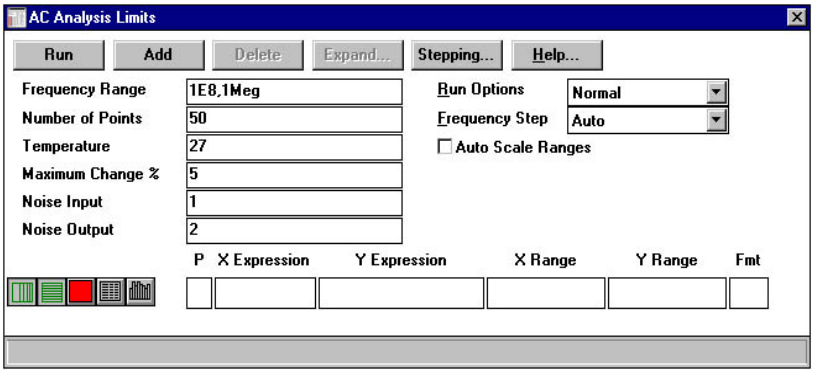


Рис. 1.14 Диалоговое окно *AC*-анализа

Когда выбирается *AC*-анализ из меню *Analysis*, *MC* тестирует схему и готовит внутренние структуры к анализу. Если ошибки не найдены, на экран выводится диалоговое окно *AC*-анализа. Называется оно *Analysis Limits*. Это диалоговое окно позволяет ввести данные, выбирать диапазон частоты, количе-

ство точек, графики. Диалоговое окно *Analysis Limits* разделено на пять областей: Кнопки, Числовые пределы, Опции формы графика, Выражения и Опции.

Область "кнопки" аналогична такой же области при временном анализе.

Числовые пределы

***Frequency range*:** поле определяет диапазон частот. Формат – *fmax*, *fmin*. Например, чтобы определить диапазон от 10 Гц до 100 кГц, печатайте его в виде "100K, 10". Анализ начинается подстановкой частоты fmin и идет до тех пор, пока частота не изменится до fmax. Ввод одиночного значения производит одиночное вычисление для заданного значения частоты.

***Number of points*:** в *AC*-анализе данные рассчитываются в следующих оцифрованных точках:

- для фиксированного линейного метода:

(*fmax-fmin*)/(*Number of Points* – 1)

- для фиксированного логарифмического метода:

(*fmax/fmin*)^(1/(*Number of points* – 1))

***Temperature*:** поле указывает температурные значения в градусах по Цельсию. Формат описания следующий: высшая точка, низшая точка, шаг изменения.

***Maximum change*:** указывает допуск изменения графика, что полезно для создания плавных кривых. Формат: *Node1*, *Node2*. Две точки разделяются запятой.

***Waveform Options*:** поля находятся ниже Числовых пределов. Каждая опция свойств графика действует только на график в своей строке. Опции аналогичны временному анализу.

***Options*:** опции *AC* находятся справа и опции контролируются либо выпадающим списком, либо кнопками. Опции Run и Frequency Step устанавливаются выпадающими списками и могут быть доступны двумя путями.

Доступные опции

Run Options. *Normal*: запускает симуляцию без записи ее на диск.

***Save*:** записывает симуляцию на диск в файл *CIRCUITNAME.ASA*.

***Retrieve*:** загружает сохраненные ранее данные о симуляции из файла *CIRCUITNAME.ASA*.

Frequency Step. *Auto*: метод использует свои алгоритмы для построения наиболее четкого и полезного графика, автоматически подбирая частоту.

Fixed Linear: устанавливает изменение частоты в линейный режим.

Fixed Log: устанавливает изменение частоты в логарифмический режим.

***Auto Scale Ranges*:** все границы по осям координат выбираются автоматически.

Расчет передаточных функций по постоянному току (*DC*-анализ)

Анализ по постоянному току предполагает расчет токов и напряжений во всех узлах схемы и дает возможность построить ее статическую характеристику. В ходе этого анализа формируется и решается система нелинейных алгебраических уравнений итерационным методом Ньютона-Рафсона и открывает окно управления (рис. 1.15).

Меню "кнопки" режима *DC* аналогично режиму временного анализа.

Числовые пределы

***Input 2 range*:** поле указывает начальное значение, конечное значение, и размер шага второго входа. Формат: конечное значение, начальное значение, шаг.

***Input 2*:** поле указывает название второго входа. Если второй вход не используется, следует ввести слово "*NONE*".

***Input 1 range*:** поле указывает начальное значение, конечное значение и величину шага первого входа. Формат: конечное значение, начальное значение, максимальный шаг.

***Input 1*:** название первого входа.

***Number of Points*:** поле указывает количество расчетных точек. По умолчанию 51, минимальное значение 5. Например, если вычислено 100 значений, а запрошены были 200 значений, то их значения вычисляются благодаря линейной интерполяции.

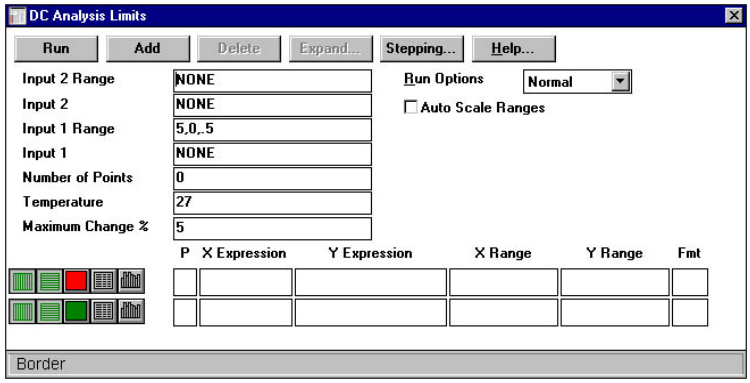


Рис. 1.15 Диалоговое окно *DC*-анализа

Temperature: поле указывает температурные значения.

Maximum change: указывает допуск изменения графика, что полезно для создания плавных кривых. Формат: *Node*1[, *Node*2]. Две точки разделяются запятой.

***Waveform Options*:** поля находятся ниже Числовых пределов. Каждая опция свойств графика действует только на график в своей строке. Опции аналогичны временному анализу.

***Options*:** опции *DC* находятся справа и контролируются либо выпадающим списком, либо кнопками.

***Auto Scale Ranges*:** все границы по осям координат выбираются автоматически.

Редактор переменных состояния используется для обзора или редактирование значений начальных условий для решения системы уравнений. Редактор показывает список напряжений, токов, цифровых уровней узлов. Прокрутка может быть использована для обзора тех значений, которые не видны. Все значения могут быть изменены.

Кнопки меню предназначены для следующих действий:

***Close*:** закрывает редактор.

***Clear*:** немедленно устанавливает все аналоговые значения в ноль, а дискретные в состояние "*X*".

***Read*:** немедленно читает данные из файла *CIRCUITNAME.TOP*. Его можно создать с помощью команды *Write*.

***Write*:** записывает все измененные значения в специальный файл на диске для последующего считывания либо редактором, либо при проведении анализов *CIRCUITNAME.TOP*.

***Print*:** копирует значения в файл и окно называемое *CIRCUITNAME.SVV*. Чтобы послать значения на принтер, нужно либо в файле, либо в окне с данными выполнить команду *print*.

***Help*:** вызывает подсказку по редактору переменных состояния. Очень важно запомнить, что редактор делает немедленные изменения, не дожидаясь конца анализа, в то время как команды *Zero* и *Read* в меню анализов действуют после начала следующего анализа.

Числовой вывод может быть осуществлен через выражения для осей *X* и *Y* каждого графика нажатием на кнопку *Numeric Output* в строке нужного параметра. Цифровые выводы могут быть нескольких типов.

Вывод сохраняется в следующие файлы в зависимости от типа произведенного анализа:

ТИП АНАЛИЗА	**ИМЯ ФАЙЛА**
временной	CIRCUITNAME.TNO
AC	*CIRCUITNAME.ANO*
DC	*CIRCUITNAME.DNO*

Результаты цифрового вывода печатаются в окне *Numeric Output*. Это окно доступно после прохождения анализа.

Пример проектирования логической схемы мультиплексора приведен на рис. 1.16. На схеме показаны три разновидности логических устройств, описываемых тремя типами моделей:

- элементы И, ИЛИ, НЕ, включающие простейшую модель, состоящую из пространственной и временной составляющих;
- генераторы двоичных последовательностей *U1*, *U6* и др., описываемые пространственной составляющей и программной конструкцией с директивой *.DEFINE*;
- дешифратор 74145, описываемый подсхемой – процедурой *.SUBCKT*.

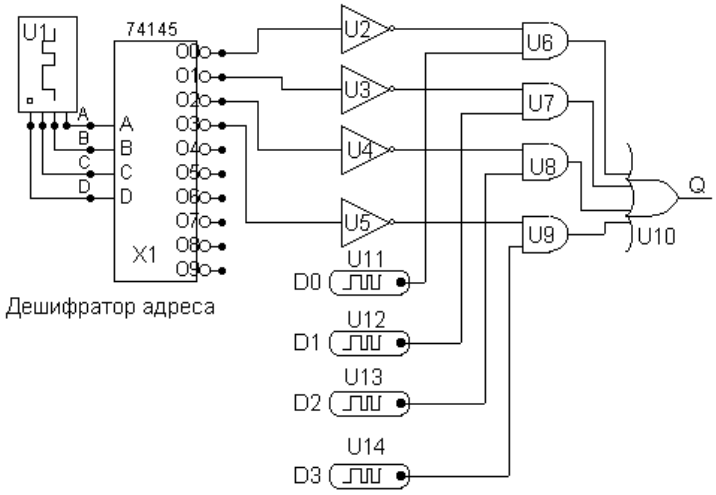


Рис. 1.16 Полностью введенная схема мультиплексора

Перечень моделей компонент мультиплексора приведен ниже:

.DEFINE IN – динамическая модель генератора двоичных сигналов *ABCD*

+0NS 0

+LABEL=START

+100*NS INCR BY* 1

+200*NS GOTO START* -1 *TIMES*

.MODEL D0_GATE UGATE () – идеальная (безынерционная) модель логических элементов И, ИЛИ, НЕ

.DEFINE a – статическая модель источника рабочего входа *A*+0*NS* 1

.DEFINE b – статическая модель источника рабочего входа *B*+0*NS* 0

.DEFINE c – статическая модель источника рабочего входа *C*+0*NS* 1

.DEFINE d – статическая модель источника рабочего входа *D*+0*NS* 1

Исследование дискретных схем производится в режиме временного анализа, который позволяет оценить работоспособность в соответствии с таблицей истинности или логическим уравнением. В частности, для схемы мультиплексора на рис. 1.17 показана временная диаграмма.

Как видно из диаграммы, параллельный рабочий код на входе мультиплексора 0101 (*d*(19), *d*(20), *d*(21), *d*(25)) преобразуется в последовательный на его выходе *d*(*Q*).

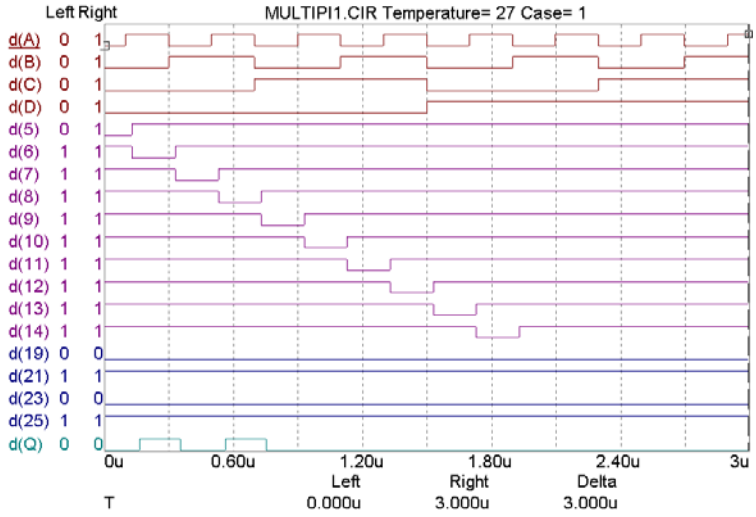


Рис. 1.17 Временная диаграмма функционирования мультиплексора

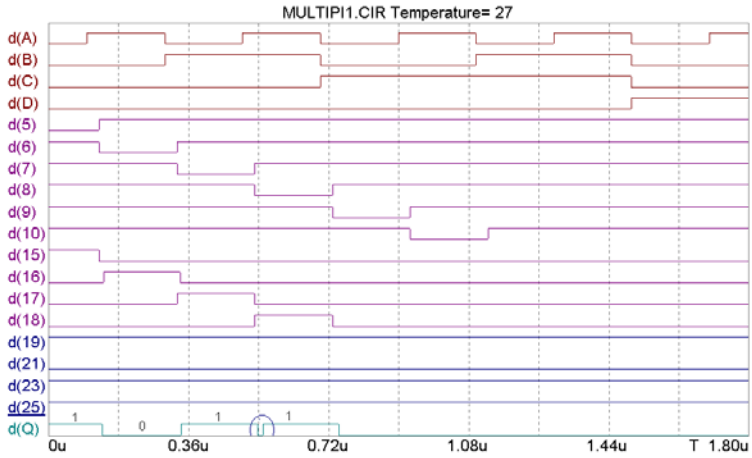


Рис. 1.18 Возникновение состязаний (гонок) в схеме мультиплексора

Если элементы схемы одинаковы, то все пути прохождения сигналов на выход имеют одинаковую задержку и, как видно из диаграммы, состязания сигналов отсутствуют, а соответственно, и помехи. Однако, при различной задержке сигналов на разных путях, в схеме могут иметь место состязания сигналов, что приведет, как показано на рис. 1.18, к возникновению "лишнего нуля" – помехи на выходе *d*(*Q*) типа 10101, при входном параллельном коде 1011.

В окне временного анализа могут также использоваться нижеприведенные операторы обработки дискретных сигналов:

HEX(*A,B,C,D*) – шестнадцатеричное представление переменных *A*, *B, C, D*
BIN(*A,B,C,D*) – бинарное представление *A, B, C, D*
DEC(*A,B,C,D*) – десятичное представление *A, B, C, D*
OCT(*A,B,C,D*) – восьмеричное представление *A, B, C, D*
AND – оператор И
NAND – оператор И-НЕ
NOR – оператор ИЛИ-НЕ
NOT – инверсия НЕ
OR – оператор ИЛИ
XOR – оператор ИСКЛЮЧАЮЩЕЕ ИЛИ.

2 АБСТРАКТНЫЙ СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ

2.1 ЭТАПЫ СИНТЕЗА КОМБИНАЦИОННЫХ СХЕМ

Абстрактным синтезом называют последовательность этапов, в результате которой получают функциональную схему комбинационного логического устройства на основе полученных в результате синтеза формальных представлений (таблицы истинности и логических уравнений).

Функциональными схемами называются схемы, в которых показаны связи между логическими элементами, а сами логические элементы представлены в виде условных обозначений.

Совершенной нормальной дизъюнктивной формой (СНДФ) называют запись логического уравнения в виде дизъюнкции (логической суммы) минтермов. Минтерм – это коньюнкция входных переменных, соответствующая конституенте единицы в таблице истинности.

Совершенной нормальной конъюктивной формой (СНКФ) называют запись логического уравнения в виде конъюнкции (логического произведения) макстермов. Макстерм – это дизьюнкция инверсий входных переменных, соответствующая конституенте нуля в таблице истинности.

В минтерм и макстерм должны входить все переменные как в прямом, так и в инверсном виде.

Под минимизацией логического уравнения понимается его упрощение с применением тождеств булевой алгебры или формальных методов минимизации типа карт Карно. Минтермы логического уравнения подвер гаются "склеиванию". "Склеивать" можно минтермы, отличающиеся только одной переменной, входящей в "склеиваемые" минтермы в прямой и инверсной форме. Идеальным упрощением считается такое, когда каждая переменная входит в логическое уравнение только один раз.

Функциональную схему логического устройства получают в результате абстрактного синтеза, который состоит из следующих этапов:

- словесная формулировка функций логического устройства;
- составление таблицы истинности по словесной формулировке;
- запись логического уравнения устройства в виде СНДФ или СНКФ;
- минимизация логического уравнения;
- выбор одного из логических базисов;
- преобразование логического уравнения с использованием правил де Моргана;
- построение функциональной схем ··ства.

Проектирование логических схем представим на конкретном примере.

1) Синтезировать логическое устройство на три входные переменные генерирующее сигнал "1" на выходе, если две рядом стоящие переменные из трех принимают значение "1".

2) Построить таблицу истинности (табл. 2.1).

2.1 Таблица истинности

A	***B***	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

3) Получить логическое уравнение в виде СНДФ, представляющее собой дизъюнкцию конъюнкций тех входных наборов, для которых $Y = 1$.

$$Y = \overline{A} \wedge B \wedge C \vee A \wedge B \wedge \overline{C} \vee A \wedge B \wedge C .$$

4) Минимизировать логическое уравнение путем применения тождеств булевой алгебры:

$$Y = \overline{A} \wedge B \wedge C \vee A \wedge B \wedge (C \vee \overline{C}) = \overline{A} \wedge B \wedge C \vee A \wedge B = \overline{A} \wedge B \wedge C \vee A \wedge B =$$

$$= B \wedge (\overline{A} \wedge C \vee A) = B \wedge (A \vee C) = B \wedge A \vee B \wedge C .$$

Очевидно, что заключительное уравнение значительно проще первоначального.

5) Принять для реализации схемы логического устройства в базисе И-НЕ.

6) Преобразовать минимизированное логическое уравнение по правилу де Моргана, а именно:

$$Y = \overline{\overline{Y}} = \overline{\overline{(B \wedge A \vee B \wedge C)}} = \overline{(\overline{B \wedge A}) \wedge \overline{B \wedge C})} .$$

7) С помощью выбранного базиса И-НЕ построить функциональную схему (рис. 2.1). При построении функциональных схем принят следующий порядок действий: сначала инвертируют (НЕ) те переменные, которые входят в логическое уравнение с инверсией, затем логически перемножают (И) переменные, входящие в конъюнкции переменных, последним этапом логически суммируют (ИЛИ) все конъюнкции, входящие в дизъюнктивную форму.

Для минимизации логических уравнений можно использовать и другой способ называемый методом карты (диаграммы) Карно. На рис. 2.2 представлена карта Карно для трех переменных.

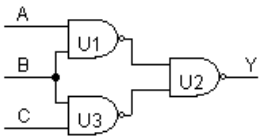


Рис. 2.1 Функциональная схема, полученная методом абстрактного синтеза

Каждая клетка диаграммы соответствует значению функции в строке таблицы истинности. Координаты клеток – значения переменных или их конъюнкции, соответствующие значениям переменных в таблице истинности. Порядок расположения координат клеток в карте Карно подчиняется последовательности в коде Грея.

Клетки, находящиеся на границах одной строки или одного столбца, считаются соседними. Для получения минимальных сумм произведений объединим в группы соседние клетки с единичными значения. По правилу: число единичных соседних клеток должно быть максимальным, а количество групп минимальным. Размерность каждой группы должно удовлетворять условию $2^n \times 2^m$, где 2^n – количество клеток по горизонтали (n = 0, 1, 2,…) а 2^m – количество клеток по вертикали (m = 0, 1, 2,…). Каждая группа представляет собой минимальное произведение переменных в прямом или инверсном виде. Если в пределах одной группы переменные меняют свое значение, то эти переменные не входят в произведение. Все полученные произведения объединяются операцией дизъюнкции. Для групп, изображенных на рис. 2.2, получим $Y = B \wedge A \vee B \wedge C$.

A \ BC	00	01	11	10
0	0	0	1	0
1	0	0	1	1

Рис. 2.2 Карта

Карту Карно для трех переменных можно рассматривать как развертку цилиндра. Поэтому, клетки, расположенные по границам таблицы можно и нужно считать соседними.

2.2 СИНТЕЗ ПРОСТЕЙШИХ ЛОГИЧЕСКИХ СХЕМ

К простейшим логическим схемам относят такие, которые реализуют элементарные булевы функции. Их условные обозначения, используемые в *MC*, показаны на рис. 1.2.

Полным базисом называют базис И, ИЛИ, НЕ, а дуальными базисами логических схем называют базисы И-НЕ, ИЛИ-НЕ. Если базис И-НЕ считать основным, то базис ИЛИ-НЕ будет дуальным и наоборот.

Таким образом, каждый логический элемент может быть представлен в трех вариантах: полном, основном и дуальном. Все варианты абсолютно равноправны, переход к дуальному базису осуществляется с помощью правил де Моргана.

В схемах константу нуля можно реализовать заземлением цепи, а единицы – подачей на рабочий вход напряжения питания. В реальных логических схемах могут использоваться также генераторы логического нуля или логической единицы, образованные из вентилей, входы которых заземлены или на них подан уровень напряжения питания.

Повторитель – это элемент, не нуждающийся в реализации за исключением случая, когда нагрузочной способности одного элемента не хватает для обслуживания всех потребителей сигнала. Этот элемент можно реализовать следующими способами:

- методом двойной инверсии $F = \overline{\overline{X}}$;
- логическим произведением аргумента с самим собой $F = X \wedge X = X$;
- логической суммой аргумента с самим собой $F = X \vee X = X$;

Инвертор может быть реализован:

- инверсией переменной $F = \overline{X}$;
- инверсией логического произведения аргумента с самим собой $F = \overline{X \wedge X} = \overline{X}$;
- инверсией логической суммы аргумента с самим собой $F = \overline{X \vee X} = \overline{X}$.

Реализация повторителя и инвертора показана на рис. 2.3.

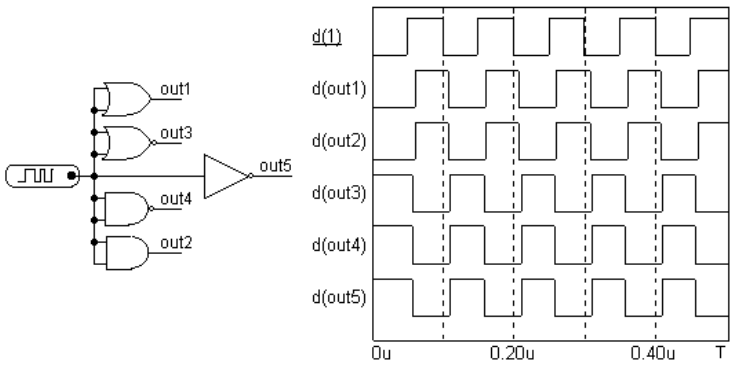


Рис. 2.3 Функциональные схемы и диаграммы повторителей и инверторов

Синтез схемы И (вентиля) (табл. 2.2)

2.2 Таблица истинности логической функции И для двух переменных

X2	*X1*	*Y*
0	0	0
1	0	0
0	1	0
1	1	1

Логические уравнения:
$Y = X1 \wedge X2$ – базис И, ИЛИ, НЕ
$Y = \overline{X1 \wedge X2}$ - базис И-НЕ
$Y = \overline{\overline{X1 \wedge X2}} = \overline{\overline{X1} \vee \overline{X2}}$ – базис ИЛИ-НЕ

Функциональные схемы вентилей на дуальных базисах показаны на рис. 2.4.

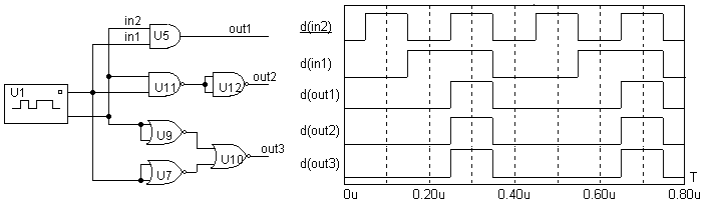


Рис. 2.4 Функциональные схемы и диаграммы вентилей И

Синтез элемента штрих Шеффера (И-НЕ) (табл. 2.3, рис. 2.5)

2.3 Таблица истинности логической функции И-НЕ

X2	*X1*	*Y*
0	0	1
1	0	1
0	1	1
1	1	0

Логические уравнения:
$Y = \overline{X1 \wedge X2}$ – базис И, НЕ
$Y = \overline{X1 \wedge X2}$ – базис И-НЕ
$Y = \overline{\overline{X1 \wedge X2}} = \overline{\overline{X1} \vee \overline{X2}}$ – базис ИЛИ-НЕ

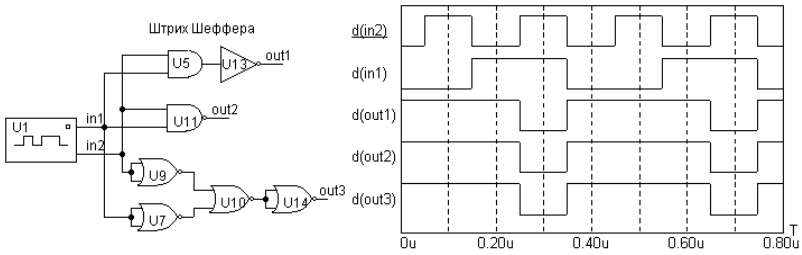


Рис. 2.5 Функциональные схемы и диаграммы И-НЕ

Синтез схемы ИЛИ (табл. 2.4, рис. 2.6)

2.4 Таблица истинности логической функции ИЛИ

X2	*X1*	*Y*
0	0	0
1	0	1
0	1	1
1	1	1

Логические уравнения:
$Y = X1 \vee X2$ – базис ИЛИ;
$Y = \overline{\overline{X1 \vee X2}} = \overline{\overline{X1} \wedge \overline{X2}}$ - базис И-НЕ;
$Y = \overline{\overline{X1 \vee X2}}$ – базис ИЛИ-НЕ.

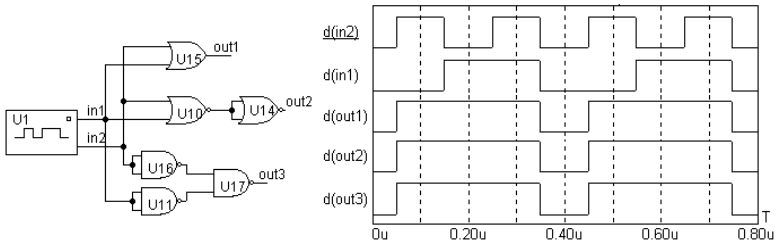


Рис. 2.6 Функциональные схемы и диаграммы ИЛИ

Синтез стрелки Пирса (ИЛИ-НЕ) (табл. 2.5, рис. 2.7)

2.5 Таблица истинности логической функции ИЛИ-НЕ (стрелка Пирса)

X2	*X1*	*Y*
0	0	1
1	0	0
0	1	0
1	1	0

Логические уравнения:
$Y = \overline{X1 \vee X2}$ – базис ИЛИ, НЕ;
$Y = \overline{\overline{X1 \vee X2}} = \overline{\overline{X1} \wedge \overline{X2}}$ - базис И-НЕ;
$Y = \overline{X1 \vee X2}$ – базис ИЛИ-НЕ.

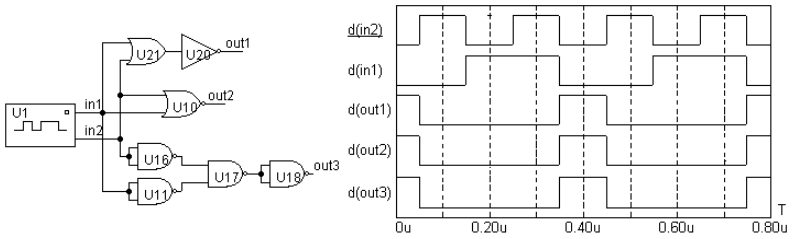


Рис. 2.7 Функциональные схемы и диаграммы ИЛИ-НЕ

Синтез импликатора и схемы запрета (табл. 2.6, рис. 2.8)

2.6 Таблица истинности функций ИМПЛИКАЦИЯ и ЗАПРЕТ

X2	*X1*	*Y*им	*Y*зп
0	0	1	0
1	0	1	0
0	1	0	1
1	1	1	0

Логические уравнения:

$Y\text{зп}=\overline{X2}\wedge X1 = \overline{\overline{X1\wedge\overline{X2}}}=\overline{\overline{X2\vee X1}}$;

$Y\text{им}=\overline{X1\wedge\overline{X2}}$.

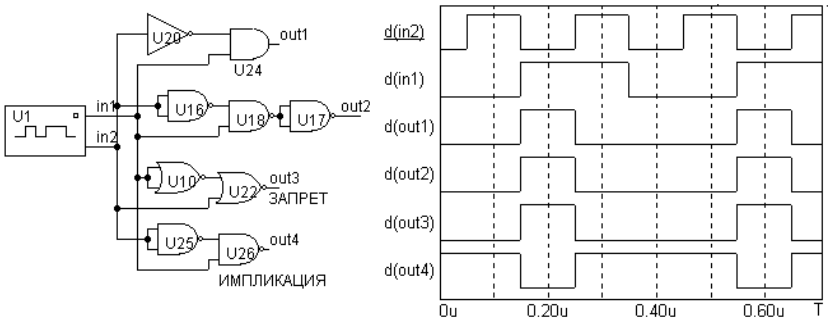


Рис. 2.8 Функциональные схемы и диаграммы импликатора и запрета

Синтез "исключающее ИЛИ" – сумма по модулю 2 (М2)
(табл. 2.7, рис. 2.9)

2.7 Таблица истинности (М2) и символьного компаратора

X2	*X1*	*Yм 2*	*Yк*
0	0	0	1
1	0	1	0
0	1	1	0
1	1	0	1

Логические уравнения:

$Ym2=\overline{X1}\wedge X2\vee X1\wedge\overline{X2} = \overline{\overline{\overline{X1}\wedge X2}\wedge\overline{X1\wedge\overline{X2}}}$;

$Yк=\overline{X1}\wedge\overline{X2}\wedge X1\wedge X2 = \overline{\overline{X2\wedge\overline{X2}}\wedge\overline{X1\wedge X2}}$.

Одним их распространенных базисов в интегральной схемотехнике является базис И–ИЛИ–НЕ. Этот элемент позволяет строить более экономичные схемы. На рис. 2.10 приведен пример использования микросхемы 7450, аналогом которой является МС К555ЛР1, для реализации двух логических уравнений типа $Y = \overline{A \wedge B \vee C \wedge D}$.

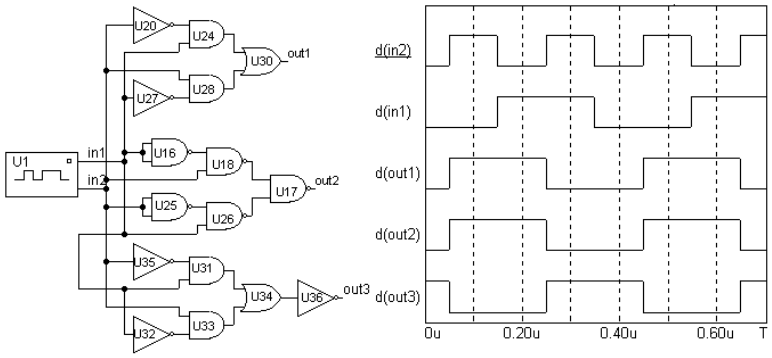


Рис. 2.9 Функциональные схемы и диаграммы "исключающее ИЛИ" и "эквивалентность"

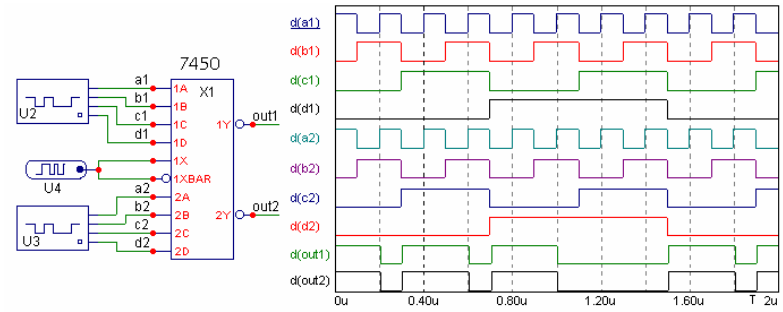


Рис. 2.10 Функциональная схема и диаграммы 2–2И–ИЛИ–НЕ

2.3 СОСТЯЗАНИЯ (ГОНКИ) В ЛОГИЧЕСКИХ СХЕМАХ

Состязаниями или гонками называют кратковременную неоднозначность выходного сигнала комбинационной логической схемы (КЛС) при изменении сигнала на каком-либо из входов, вызванную конечным значением времени прохождения сигнала через логические элементы (ЛЭ).

В КЛС встречаются участки, где сигнал разветвляется, и получившиеся два сигнала распространяются по двум независимым направлениям, а потом оба сигнала встречаются на входах одного элемента.

Например, на КЛС константы 1, (рис. 2.11) в тракте *A* четное число вентилей, в тракте *B* – нечетное. Анализ подобной схемы методами алгебры логики без учета задержки ЛЭ показывает, что выход КЛС будет равен 1 при любом значении входа.

Если же учесть суммарную задержку ЛЭ при прохождении сигнала по обоим трактам, то может оказаться, что задержка по тракту *A* не равна задержке по тракту *B*. Это обстоятельство вызовет на выходе ложное срабатывание. В этом можно убедиться, анализируя временную диаграмму, приведенную на рис. 2.11.

Возможность появления ложного срабатывания, не предусмотренного алгоритмом работы схемы, называется **риском сбоя**.

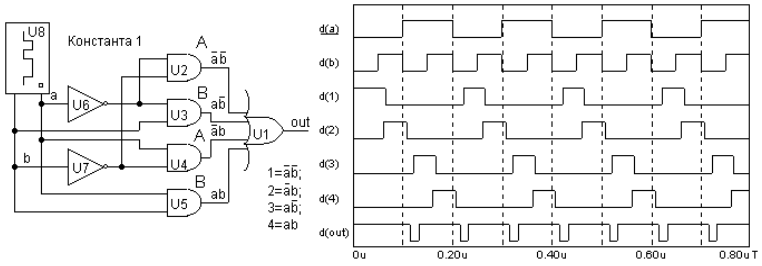


Рис. 2.11 Функциональная схема КЛС и диаграммы гонок

Различают следующие виды рисков сбоя: статические; динамические; логические; функциональные.

Статическими называются риски сбоя в случае, если $y(X1) = y(X2)$, где y – булева функция; $X1$, $X2$ – наборы входных сигналов, до и после события изменения перехода, соответственно.

Риск сбоя называется статическим в нуле S_0, если $y(X1) = y(X2) = 0$.

Риск сбоя называется статическим в единице S_1, если $y(X1) = y(X2) = 1$.

Риск сбоя называется динамическим, если $y(X1) \neq y\ (X2)$.

Риск сбоя называется динамическим D_+ при переходе на выходе с низкого уровня на высокий (**01**), если $y(X1) = 0$, а $y(X2) = 1$.

Риск сбоя называется динамическим $D\_$ при переходе на выходе с высокого уровня на низкий (**10**), если $y(X1) = 1$, а $y(X2) = 0$.

Логическим риском сбоя называется статический риск сбоя, проявляющийся при соседней смене наборов и имеющий возможность устранения изменением логической структуры, реализующей булеву функцию.

Функциональным называется риск сбоя, проявляющийся при многоместной смене наборов и определяется характером самой функции. Такие риски не устраняются изменением логической структуры, реализующей булеву функцию.

2.4 ТРИСТАБИЛЬНАЯ СХЕМОТЕХНИКА

2.8 Таблица истинности повторителя для тристабильного вентиля

ТУС	Вход	Выход
1	0	0
1	1	1
0	1	**Z**
0	1	*Z*

Тристабильные логические схемы отличаются от обычных тем, что позволяют совмещать обычный режим работы логических схем и режим высокого импеданса. Этот режим характерен тем, что логические схемы вводятся в «разомкнутое» (*Z*-состояние) состояние сигналом тристабильного управления (ТУС), не пропускающее сигналы с входа на выход (табл. 2.8).

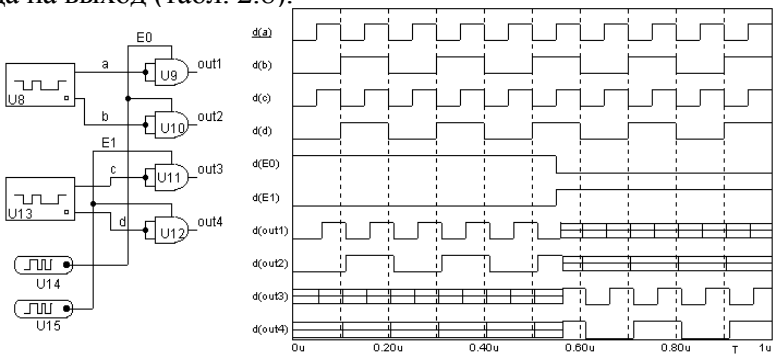


Рис. 2.12 Функциональная схема и диаграммы тристабильной шины

Как видно из таблицы выходной сигнал логической схемы может принимать три состояния 0, 1, *Z*.

На рис. 2.12. приведен пример тристабильной буферной шины, пропускающей или не пропускающей сигналы с входа шины на выход.

Как видно из диаграмм, шина может функционировать в двух режимах, ТУС = 1 и ТУС = 0. Тристабильная шина используется, для реализации монтажной логики, например, в компьютере при подключении к шине нескольких источников и нескольких приемников информации.

3 КОМБИНАЦИОННАЯ СХЕМОТЕХНИКА ЦИФРОВЫХ СИСТЕМ

3.1 СИНТЕЗ ШИФРАТОРОВ И ДЕШИФРАТОРОВ

Дешифратор (декодер) – комбинационная логическая структура, преобразующая код числа, поступающий на входы в управляющий сигнал на одном единственном выходе. По причине того, что управляющий сигнал формируется только на одном выходе, а на остальных в это время отсутствует, дешифратор называется избирательной схемой.

В условных обозначениях дешифраторов используются буквы *DC* (от английского слова *decoder*).

При дешифрации *M*-разрядного двоичного кода число выходных линий, при условии реализации всех комбинаций этого кода (полный дешифратор), определяется по формуле (3.1):

$$N = 2^M, \tag{3.1}$$

где N – число выходов. В противном случае, если число выходных линий меньше 2^M, то такой дешифратор называется неполным.

Входной код может быть однофазным, при наличии только прямых входов, и парафазным, при наличии пар входов: прямых и инверсных. Число входов $m = M$ при однофазном коде и $m = 2M$ при парафазном.

Дешифраторы могут выполняться на одноступенчатой (линейный дешифратор) и многоступенчатой (прямоугольный и пирамидальный дешифраторы) схеме дешифрации.

Линейный дешифратор выполняется прямой схемной реализацией системы (3.2):

$$\begin{cases} F_0 = \overline{X}_m \wedge \overline{X}_{m-1} \wedge \ldots \wedge \overline{X}_2 \wedge \overline{X}_1; \\ F_1 = \overline{X}_m \wedge \overline{X}_{m-1} \wedge \ldots \wedge \overline{X}_2 \wedge X_1; \\ \ldots \\ F_n = X_m \wedge X_{m-1} \wedge \ldots \wedge X_2 \wedge X_1, \end{cases} \tag{3.2}$$

где $X_m, X_{m-1}, \ldots, X_2, X_1$ – сигналы на входах; $F_n, F_{n-1}, \ldots, F_2, F_1$ – сигналы на выходах дешифратора. Никаких логических преобразований не производится, за счет чего достигается высокое быстродействие:

$$\tau = \tau_{ср}, \tag{3.3}$$

где τ – время работы дешифратора; $\tau_{ср}$ – время работы одного вентиля.

Дешифратор с разрешением по входу называется дешифратором-демультиплексором.

Линейный (одноступенчатый) дешифратор имеет наибольшее быстродействие, но его реализация при значительной разрядности входного слова требует применения логических элементов с большим числом входов и, кроме того, сопровождается большой нагрузкой на источники входных сигналов. Следовательно, при наличии микросхем дешифраторов с ограниченным числом разрядов, любой необходимый дешифратор может быть построен по многоступенчатой схеме. При комбинировании по пирамидальной схеме входное слово делится на поля, разрядность которых соответствует числу входов базовых дешифраторов.

Дешифратор применяется для выбора ячейки памяти в запоминающих устройствах, а также в различных устройствах, например, для визуальной индикации десятичных цифр на световых табло газоразрядных индикаторов.

В микропроцессорных системах с помощью дешифраторов, помимо выборки необходимых ячеек памяти, осуществляется расшифровка кодов операций с выдачей соответствующих управляющих сигналов, выбор направлений потоков информации и т.д.

Построение линейных дешифраторов, не ограничивается простой реализацией системы уравнений (3.2). Если проинвертировать левую и правую часть каждого уравнения, затем по теореме де Моргана заменить конъюнкции дизъюнкциями, то получим систему уравнений линейного дешифратора, выполненного полностью на дизъюнкторах.

В таких дешифраторах активное выходное значение представляется низким уровнем.

На рис. 3.1 приведена схема и временная диаграмма полного линейного дешифратора с однофазными входами, построенная по системе логических уравнений в конъюнктивной форме. Сначала получаем инверсии всех переменных при помощи инверторов *U3*, *U4*, *U5*, а затем составляем конъюнкцию, например, для $Y0 = \overline{A} \wedge \overline{B} \wedge \overline{C}$, согласно уравнению (3.2), затем для *Y*1 и т.д.

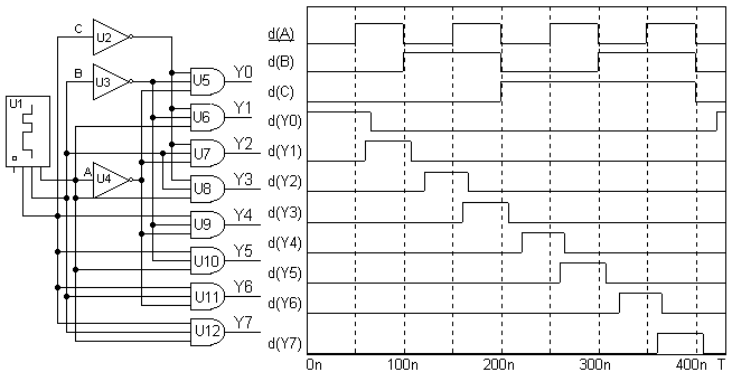


Рис. 3.1 Функциональная схема и временная диаграмма дешифратора

Линейный (одноступенчатый) дешифратор имеет наибольшее быстродействие, но его реализация при повышенной разрядности входного слова требует применения логических элементов с большим числом входов, что приводит к увеличению нагрузки на источники входных сигналов. Кроме того, повышенная разрядность конъюнкторов снижает быстродействие последних. Следовательно, при наличии микросхем малоразрядных дешифраторов, любой необходимый дешифратор может быть построен по многоступенчатой схеме.

3.1.1 Многоступенчатые дешифраторы

Среди многоступенчатых дешифраторов можно выделить два типа: прямоугольный (матричный) и пирамидальный. Для реализации прямоугольного дешифратора необходимо входное слово разбить на группы. Оптимальным считается разбиение пополам. При этом младшая часть входного слова дешифрируется линейным дешифратором, выходы которого составляют строки матрицы. А столбцами матрицы являются выходы дешифратора старшей части входного слова. В пересечении строк и столбцов устанавливаются двухвходовые конъюнкторы, количество которых определяется как $2^m \times 2^n$, где m – количество разрядов младшей части входного слова; n – количество разрядов старшей части входного слова.

Если число групп больше двух, то количество ступеней возрастает. При этом, первая ступень состоит из нескольких линейных дешифраторов, количество которых определяется количеством групп, на которое разбивается входное слово. Во второй ступени осуществляется конъюнкция выходных сигналов по матричной схеме строк и столбцов линейных дешифраторов с помощью двухвходовых вентилей. При нечетном количестве дешифраторов в первой ступени, оставшиеся без пары выходы собирают на третьей ступени с помощью двухвходовых конъюнкторов с выходами второй ступени.

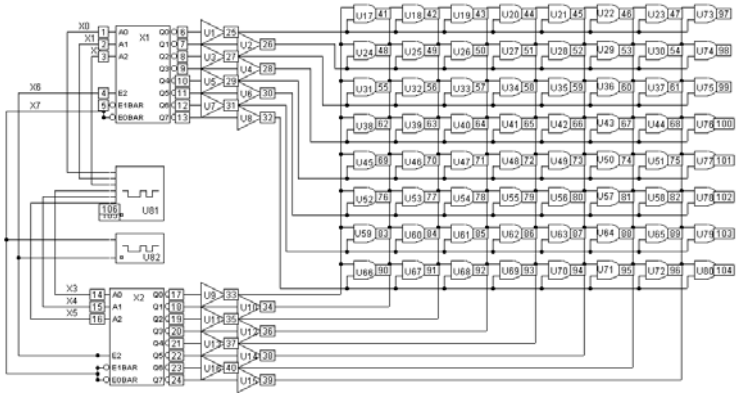


Рис. 3.2 Матричный дешифратор на 64 выхода

На рис. 3.2 и 3.3 изображены схема матричного дешифратора шестиразрядного входного слова и временная диаграмма его работы. Дешифратор, построен на основе двух микросхем первой ступени 74*F*138 (К555ИД7), *X1* является дешифратором строк, а *X2* – дешифратором столбцов. Во второй ступени, узлах матричной сетки расположены конъюнкторы, с которых снимаются выходные сигналы.

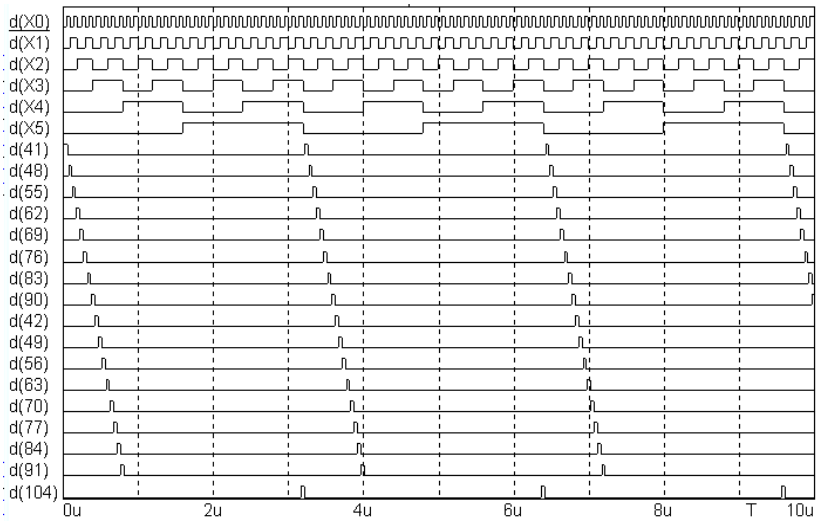


Рис. 3.3 Временная диаграмма работы матричного дешифратора

3.1.2 Пирамидальные дешифраторы

Пирамидальные дешифраторы так же как прямоугольные, относятся к разряду многоступенчатых дешифраторов, особенностью которых является применение во всех ступенях дешифрации двухвходовых вентилей с обязательным подключением выходов элемента *i*-ой ступени ко входам только двух элементов (*i*+1)-ой ступени. Число ступеней (*k*) в пирамидальном дешифраторе на единицу меньше разрядности дешифрируемого числа $k = M–1$, а число вентилей в каждой из ступеней определяется из выражения

$$B_i = 2^{i+1}, \tag{3.4}$$

где *i* – номер ступени пирамидального дешифратора.

Общее количество вентилей на дешифратор определяется из выражения

$$\Sigma B = \sum_{i=1}^{k} 2^{i+1}, \qquad i = 1, 2, 3, \ldots . \tag{3.5}$$

Принцип построения пирамидального дешифратора наглядно виден из примера построения такого дешифратора на восемь выходов (рис. 3.4). Сигналы на каждом из восьми выходов дешифратора формируются с помощью двух ступеней, поскольку разрядность дешифрируемого кода $M = 3$. Вентили *U3 – U6* первой ступени формируют четыре сигнала, являющиеся различными конъюнкциями двух входных переменных X_1, X_0 и их отрицаний $\overline{X_1}$, $\overline{X_0}$. Аналогичным образом формируются сигналы на выходах второй ступени, с той лишь разницей, что одной из переменных каждого из вентилей второй ступени являются сигналы X_2 и $\overline{X_2}$, а второй переменной – выходные сигналы первой ступени дешифрации. В результате на второй ступени формируются уже восемь выходных сигналов пирамидального дешифратора ($Y_0 – Y_7$):

$$Y_0 = \overline{X_2} \wedge \overline{X_1} \wedge \overline{X_0}; \quad Y_1 = \overline{X_2} \wedge \overline{X_1} \wedge X_0; \quad Y_2 = \overline{X_2} \wedge X_1 \wedge \overline{X_0};$$
$$Y_3 = \overline{X_2} \wedge X_1 \wedge X_0; \quad Y_4 = X_2 \wedge \overline{X_1} \wedge \overline{X_0}; \quad Y_5 = X_2 \wedge \overline{X_1} \wedge X_0;$$
$$Y_6 = X_2 \wedge X_1 \wedge \overline{X_0}; \quad Y_7 = X_2 \wedge X_1 \wedge X_0. \tag{3.6}$$

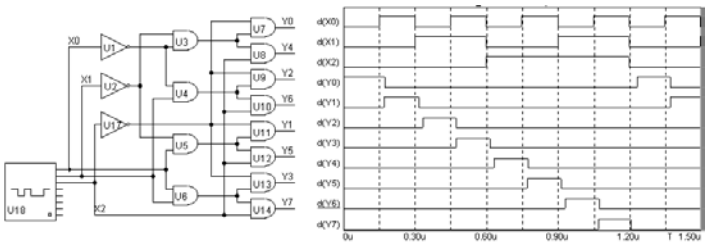


Рис. 3.4 Схема и временная диаграмма пирамидального дешифратора на восемь выходов

Учитывая, что первая ступень дешифратора всегда содержит 2^2 вентилей, а в последующих ступенях число вентилей всегда удваивается, можно записать выражение (3.2) как функцию от разрядности числа *M*:

$$\Sigma B = 2^2(2^{M-1}-1) \tag{3.7}$$

Недостатком пирамидальных дешифраторов следует считать большое число ступеней (*M*-1), снижающих быстродействие дешифратора. При реализации пирамидального дешифратора на элементах И, его быстродействие определяется как

$$\tau_д = \tau_{ср}(M - 1). \tag{3.8}$$

3.1.3 Шифратор

Шифратор (кодер) – комбинационная схема, преобразующая унитарный код в некоторый позиционный. Если выходной код является двоично-позиционным, то шифратор называется двоичным. Шифратор представляет собой устройство, выполняющее функцию, обратную по отношению к дешифратору, т.е. формирование двоичного кода на выходах при появлении сигнала на одном из входов.

У двоичного шифратора существует связь между числом входных и выходных линий, где количество выходных линий *m* определяется по формуле

$$m = \log_2 n, \tag{3.9}$$

где *n* – количество входных линий.

Выведем систему булевых функций работы шифратора с восемью входными линиями. Для этого составим таблицу истинности (табл. 3.1).

3.1 Таблица истинности для шифратора с восемью входными линиями

Код унарный	Код двоичный	Y_2	Y_1	Y_0
0	000	0	0	0
1	001	0	0	1
2	010	0	1	0
3	011	0	1	1
4	100	1	0	0
5	101	1	0	1
6	110	1	1	0
7	111	1	1	1

Из приведенной таблицы, выберем конституенты единицы для каждого выходного двоичного разряда и запишем систему булевых функций (3.10), которая описывает работу рассматриваемого шифратора:

$$\begin{cases} Y_0 = X_1 \wedge X_3 \wedge X_5 \wedge X_7; \\ Y_1 = X_2 \wedge X_3 \wedge X_5 \wedge X_7; \\ Y_2 = X_4 \wedge X_5 \wedge X_6 \wedge X_7. \end{cases} \tag{3.10}$$

Возможно также построение шифратора с линией разрешения. В этом случае система булевых функций (3.10) примет вид

$$\begin{cases} Y_0 = (X_1 \wedge X_3 \wedge X_5 \wedge X_7) \wedge C; \\ Y_1 = (X_2 \wedge X_3 \wedge X_5 \wedge X_7) \wedge C; \\ Y_2 = (X_4 \wedge X_5 \wedge X_6 \wedge X_7) \wedge C. \end{cases} \tag{3.11}$$

Кодер (*CD*) нашел применение в различных устройствах. Например, в постоянных запоминающих устройствах, матрицу ИЛИ можно рассматривать как кодер. Шифратор широко используется в программируемой матричной логике. Шифраторы находят применение при проектировании различных типов клавиатур, в которых для устранения явления дребезга контактов от клавиши в некоторых случаях

применяют шифратор с линией разрешения.

3.2 МУЛЬТИПЛЕКСОРЫ И ДЕМУЛЬТИПЛЕКСОРЫ

Мультиплексор (*multiplexor*) – комбинационное устройство, обеспечивающее коммутацию одного из рабочих входов на общий выход под управлением сигналов на адресных входах.

Логическое уравнение мультиплексора

$$Y = D0 \wedge \overline{A1} \wedge \overline{A0} \vee D1 \wedge \overline{A1} \wedge A0 \vee D2 \wedge A1 \wedge \overline{A0} \vee D3 \wedge A1 \wedge A0 . \quad (3.12)$$

Анализируя логическое уравнение (3.12), легко убедиться, что мультиплексор содержит две независимые части: адресный дешифратор и выходные вентили (табл. 3.2).

Каждый выход дешифратора активизирует свой вентиль, на который поступает информационный сигнал. Все вентили рабочих входов объединяются дизъюнктором. Схема мультиплексора и его рабочая диаграмма, построенные средствами *MC5*, приведена на рис. 1.17 и 1.18.

3.2 Таблица истинности мультиплексора

Вход	Адресные входы		Выход *Y*
	*A*1	*A*0	
***D*0**	0	0	***D*0**
*D*1	0	1	*D*1
*D*2	1	0	*D*2
*D*3	1	1	*D*3

Демультиплексором (***demultiplexer***) называют комбинационное устройство, передающее сигналы с общего входа на рабочие выходы в соответствии с управляющими сигналами, действующими на адресных входах. ДМ выполняет микрооперацию обратную мультиплексору.

3.3 Таблица истинности ДМ на четыре выхода

Вход	Адресные входы		Выходы			
Pp	*A*1	*A*0	*Y*1	*Y*2	*Y*3	*Y*4
***DAT*0**	0	0	***DAT*0**	0	0	0
*Dat*1	0	1	0	***DAT*1**	0	0
*Dat*2	1	0	0	0	***DAT*2**	0
*Dat*3	1	1	0	0	0	***DAT*3**

Логические уравнения, вытекающие из табл. 3.3:

$$Y1 = \overline{A1} \wedge \overline{A0} \wedge Dat0 ; \quad Y2 = \overline{A1} \wedge A0 \vee Dat1 ;$$

$$Y3 = A1 \wedge \overline{A0} \wedge Dat2 ; \quad Y4 = A1 \wedge A0 \wedge Dat3 . \quad (3.13)$$

Как видно из уравнений (3.13), функциональная схема демультиплексора (см. рис. 3.5), имеет много общего с линейным дешифратором и отличается от него только структурой входных сигналов.

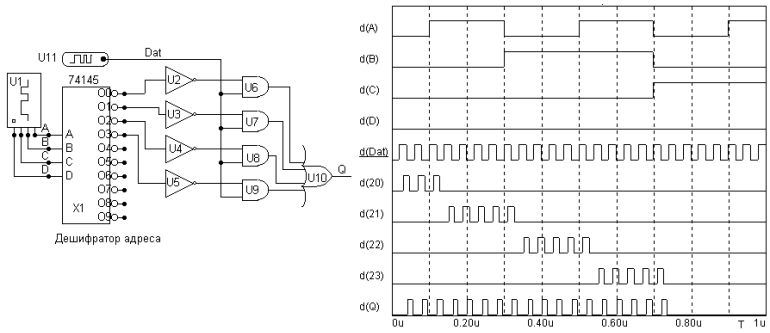
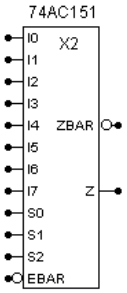


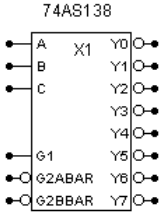
Рис. 3.5 Схема и временная диаграмма демультиплексора



Мультиплексор 74*AC*151 (К555КП7)
*I*0 – *I*7 – информационные входы
*S*0 – *S*2 – адресные входы
EBAR – управляющий вход

Z – *ZBAR* – прямой и инверсный выходы

Рис. 3.6 Примеры микросхем мультиплексора и демультиплексора



Демультиплексор 74*AS*138 (К555ИД7)
A,*B*,*C* – адресные входы
*E*1 – информационный вход
*E*0*BAR*, *E*1*BAR* – управляющие входы
*I*0 – *I*7 – инверсные выходы

Рис. 3.6 (Продолжение)

3.3 УЗЛЫ ДВОИЧНОЙ АРИФМЕТИКИ

Сумматор комбинационного типа (*summator*) – это узел цифровой системы, выполняющий арифметическое суммирование кодов слагаемых. В зависимости от системы счисления различают:

- двоичные сумматоры;
- двоично-десятичные сумматоры (в общем случае двоично-кодированные);
- десятичные сумматоры;
- прочие (например, амплитудные).

В компьютере под знак числа отводится бит в старшем разряде слова: если этот разряд установлен в "1", то число отрицательно, "0" – число положительно. Максимальное число со знаком, которое можно представить с помощью n битов равно $2^{n-1}-1$. Отрицательные числа в компьютере представляются двумя типами кода: обратным и дополнительным. Обратным кодом числа называют код, в котором все разряды слова инвертированы. Дополнительным кодом называют код, формируемый из двоичного путем инвертирования всех разрядов слова и сложением инверсного кода с единицей. Указанные коды имеют другое название – коды с неполным и полным дополнением.

Правила двоичного сложения:

$0+0=0$
$0+1=1$
$1+0=0$
$1+1=0$ перенос 1 в старший разряд

$$\begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011 \\ \text{перенос} \end{array} \qquad \begin{array}{r} 5 \\ +\ 6 \\ \hline 11 \end{array}$$

Правила двоичного вычитания:

0 - 0 = 0
0 - 1 = 1 заем 1 из старшего раряда
1 - 0 = 1
1 - 1 = 0

заем
$\overset{\frown}{1}$
0110 – б
0101 – 5
0001 – 1

Замена вычитания сложением:

1) при записи отрицательных чисел обратным кодом можно инвертировать вычитаемое и прибавить его к уменьшаемому. Если при сложении отрицательных чисел в знаковом разряде возникает перенос, то бит переноса необходимо прибавить к младшему разряду результата сложения;

2) при записи отрицательных чисел в дополнительном коде необходимо постоянно прибавлять 1 к младшему разряду суммы.

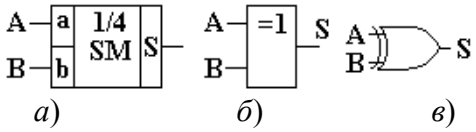
Правила сложения: сложение двоичных чисел производится поразрядно от младшего разряда к старшему; в младшем разряде вычисляется сумма младших разрядов слагаемых *A* и *B*. Эта сумма может быть записана либо в виде одноразрядного числа *S*, либо двухразрядного числа *SC*, где *S* – сумма; *C* – перенос; во всех последующих разрядах сумма вычисляется путем сложения разрядов слагаемых *A* и *B* и переноса *C*. Сумма может записана либо в виде одноразрядного числа *S* или двухразрядного числа *SC*.

3.4 Таблица истинности "ИСКЛ.ИЛИ"

A	*B*	*S*
0	0	0
0	1	1
1	0	1
1	1	0

Простейшим двоичным суммирующим элементом является четвертьсумматор. Такое название этот элемент получил из-за того, что он имеет в два раза меньше выходов и в два раза меньше строк в таблице истинности по сравнению с полным двоичным одноразрядным сумматором. Это устройство нам известно как элемент "сложение по модулю 2", "исключающее ИЛИ", "неэквивалентность". Схема (рис. 3.7, *а*) имеет два входа *A* и *B* для двух слагаемых и один выход *S* для суммы. Ее работу отражает таблица истинности (табл. 3.4), а соответствующее уравнение имеет вид

$$S = \overline{A} \wedge B \vee A \wedge \overline{B} = A \oplus B\,. \qquad (3.14)$$



а) *б*) *в*)

Рис. 3.7 Условные графические обозначения четвертьсумматора

Реализуем четвертьсумматор в базисах И-НЕ, ИЛИ-НЕ и с использованием только одного инвертора, для чего преобразуем уравнение (3.14):

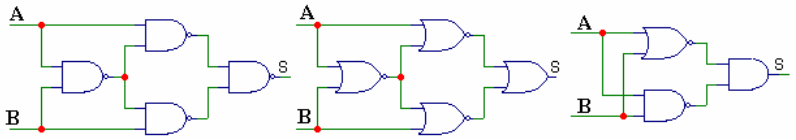
С целью снижения громоздкости и увеличения наглядности, условно заменим знаки операций логических соотношений знаками арифметических операций, где произведение соответствует конъюнкции, а сложение – дизъюнкции.

$$S = \overline{A}B + A\overline{B} = \overline{A}A + \overline{A}B + \overline{B}B + A\overline{B} = A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B}) = \\ = A\overline{AB} + B\overline{AB} = \overline{\overline{A\overline{AB}} \cdot \overline{B\overline{AB}}}; \qquad (3.15)$$

$$S = \overline{A}B + A\overline{B} = \overline{A}A + \overline{A}B + \overline{B}B + A\overline{B} = \overline{\overline{A(A+B)}} + \overline{\overline{B(A+B)}} = \\ = \overline{A + \overline{A+B}} = \overline{B + \overline{A+B}}; \qquad (3.16)$$

$$S = \overline{A}B + A\overline{B} = \overline{A}A + \overline{A}B + \overline{B}B + A\overline{B} = \overline{A}(A+B) + \overline{B}(A+B) = \\ = (A+B)(\overline{A} + \overline{B}) = (A+B)\overline{AB}; \qquad (3.17)$$

На рис. 3.8 приведены схемы, реализующие уравнения (3.15) – (3.17).



а) *б*) *в*)

Рис. 3.8 Варианты четвертьсумматоров в базисах И-НЕ, ИЛИ-НЕ, ИЛИ

Полусумматорами называются устройства с двумя входами и двумя выходами, на которых вырабатываются сигналы суммы двух одноразрядных двоичных чисел и переноса.

Для синтеза полусумматора воспользуемся таблицей сложения двоичных чисел, на основании которой построим таблицу истинности (табл. 3.5).

На основании таблицы истинности, выписав сумму минтермов, построим переключательные функции в СДНФ для результата сложения S и переноса C_{i+1} и минимизируем их.

$$S_i = \overline{B_i} A_i + B_i \overline{A_i} = B_i \oplus A_i;$$
$$C_{i+1} = B_i A_i. \qquad (3.18)$$

3.6 Таблица истинности полного сумматора

C_{i-1}	A_i	B_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3.5 Таблица истинности полусумматора

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

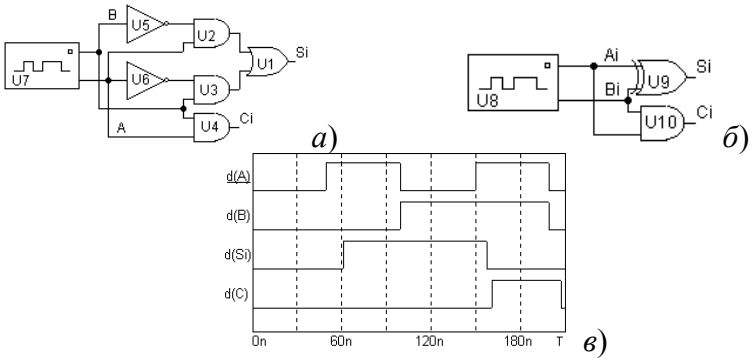


Рис. 3.9: *а* – полусумматор на вентилях в базисе И, ИЛИ, НЕ; *б* – использующий вентиль "Исключающее ИЛИ" и вентиль И; *в* – временная диаграмма работы полусумматора

Полусумматор реализует лишь часть задачи суммирования, так как не учитывает еще одной входной величины – переноса из соседнего младшего разряда в данных. По этой причине он осуществляет сложение только в разряде единиц многоразрядного двоичного слова. На рис. 3.9, *а* показана реализация характеристического уравнения (3.18), а на рис. 3.9, *б* используются вентили, реализующие бинарную функцию "Исключающее ИЛИ" и вентиль "И". На рис. 3.9, *в* представлена временная диаграмма работы полусумматора.

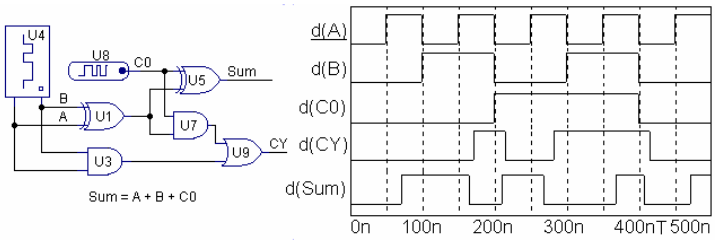
Одноразрядный двоичный сумматор, его еще называют полным сумматором, состоит из двух комбинационных схем: одна формирует результат сложения Sum_i, вторая – бит переноса CY_i. (см. рис. 3.10).

Одноразрядные полные сумматоры имеют три входа, которые обеспечивают сложение разрядов слагаемых и разряд переноса из предыдущего разряда по правилу $C_{i-1}+A_i+B_i$ *(см. табл. 3.6).*

Для полного сумматора минимизированные переключательные функции для S_i и C_{i+1} будут иметь вид:

$$
\begin{aligned}
S_i &= \overline{C_{i-1}}\cdot\overline{A_i}\cdot B_i + \overline{C_{i-1}}\cdot A_i\cdot\overline{B_i} + C_{i-1}\cdot\overline{A_i}\cdot\overline{B_i} + C_{i-1}\cdot A_i\cdot B_i = \\
&= \overline{C_{i-1}}\cdot(\overline{A_i}\cdot B_i + A_i\cdot\overline{B_i}) + C_{i-1}\cdot(\overline{A_i}\cdot\overline{B_i} + A_i\cdot B_i) = \\
&= \overline{C_{i-1}}\cdot(A_i\oplus B_i) + C_{i-1}\cdot(\overline{A_i\oplus B_i}) = C_{i-1}\oplus(A_i\oplus B_i); \\
C_{i+1} &= \overline{C_{i-1}}\cdot B_i\cdot A_i + C_{i-1}\cdot\overline{B_i}\cdot A_i + C_{i-1}\cdot B_i\cdot\overline{A_i} + C_{i-1}\cdot B_i\cdot A_i = \\
&= C_{i-1}\cdot(\overline{B_i}\cdot A_i + B_i\cdot\overline{A_i}) + B_i\cdot A_i = C_{i-1}\cdot(A_i\oplus B_i) + B_i\cdot A_i
\end{aligned}
\qquad (3.19)
$$

Используя уравнения (3.19), построим полный сумматор в *MC5* (см. рис. 3.10).



а) *б*)

Рис. 3.10 Схема полного сумматора (*а*) и временная диаграмма (*б*)

Для выполнения операции сложения над многоразрядными словами одноразрядные сумматоры объединяют в группы, где каждый одноразрядный сумматор суммирует одноименные разряды слагаемых.

В зависимости от характера ввода-вывода кодов и организации переносов комбинационные многоразрядные сумматоры бывают с последовательным переносом и параллельным (ускоренным) переносом.

В сумматоре с последовательным переносом сложение кодов осуществляется поразрядно начиная с младшего разряда с помощью комбинационного сумматора на три входа. Образующийся в данном разряде перенос C_{j+1} задерживается на время $t_{зд}$ и поступает на вход C_j сумматора в момент поступления следующего разряда слагаемых. Таким образом, последовательно разряд за разрядом производится сложение кодов чисел. Достоинством последовательного сумматора является простота аппаратурной реализации, а недостатком – большое время суммирования (см. рис. 3.11).

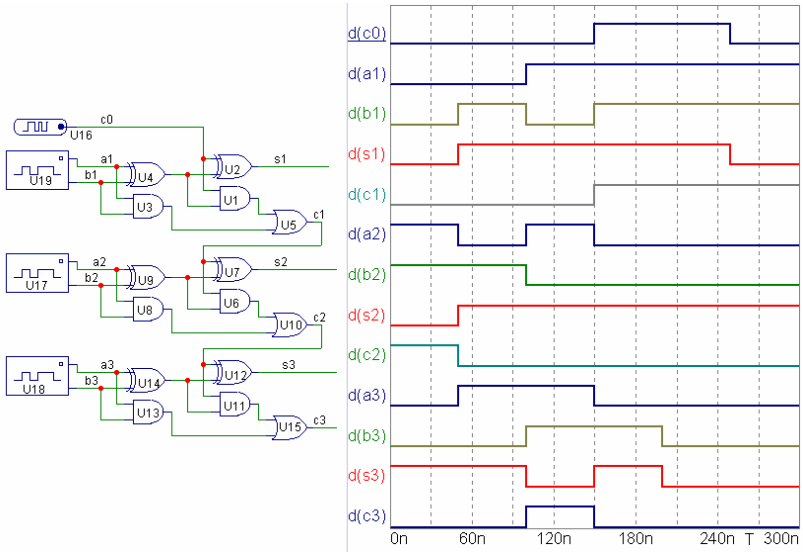


Рис. 3.11 Схема трехразрядного сумматора с последовательным переносом и временная диаграмма, работы трехразрядного сумматора с последовательным переносом

В сумматоре с параллельным (ускоренным) переносом достигается более высокое быстродействие. Параллельная схема каскадирования использует параллельный групповой или ускоренный перенос, причем схема сумматора значительно усложняется по сравнению с сумматором с последовательным

переносом.

Суммируемые коды поступают на входы сумматора одновременно по всем разрядам. Значение окончательного переноса формируется специальной схемой, называемой схемой ускоренного переноса. С целью повышения быстродействия сумматоры в интегральном исполнении изготавливают с малой разрядностью обрабатываемых слов, чаще всего, четырехразрядными.

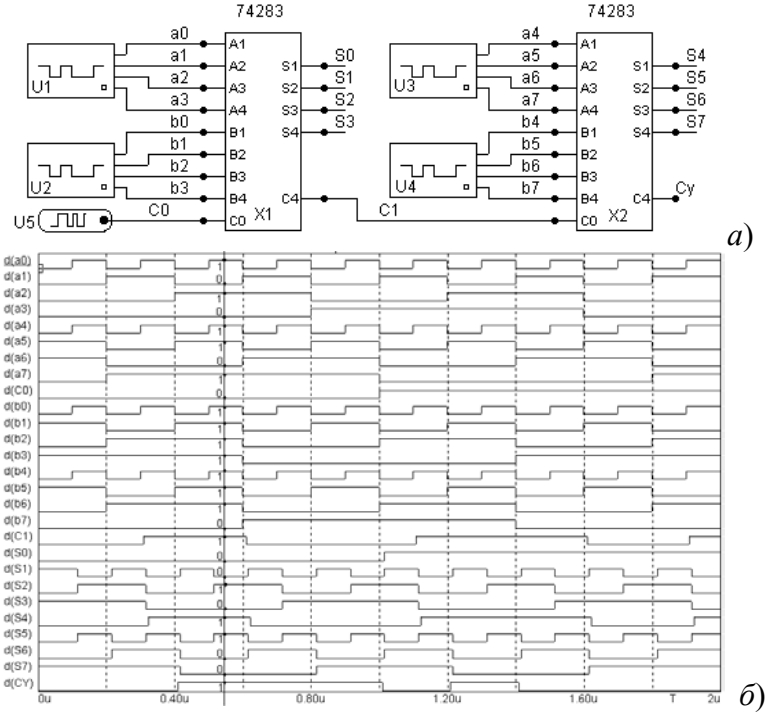


Рис. 3.12 *а*, *б* – схема и временные диаграммы восьмиразрядного параллельного сумматора

Схему ускоренного переноса четырехразрядного сумматора можно синтезировать на основе комбинационной таблицы, в которой в качестве переменных выступают слагаемые и бит переноса. Следовательно, такая таблица должна содержать $2^9 = 512$ строк (4 разряда первое слагаемое + 4 разряда второе слагаемое +1 разряд переноса). В качестве функции выступает бит ускоренного переноса. Составив такую таблицу и получив СДНФ, которую необходимо минимизировать, можно приступить к составлению принципиальной схемы. Читателю предоставляется уникальная возможность самостоятельно спроектировать схему ускоренного переноса. Результат можно сравнить, например, со схемой сумматора, на МС К555ИМ3 (7483), которая снабжена схемой ускоренного переноса.

В параллельном сумматоре обычно применяются различные способы ускорения переноса (параллельный перенос, групповой и т.п.)

Схема каскадирования сумматоров 74283 с целью получения параллельного восьмиразрядного сумматора с последовательным переносом и диаграмма его работы приведена на рис. 3.12, *а*, *б*).

Вычитатели (*subtractor*) и компараторы (*comparator*) – комбинационные устройства, осуществляющее вычитание и сравнение двоичных чисел ($A - B = 0$, $A = B$; $A - B > 0$, $A > B$; $A - B < 0$ $A < B$) (табл. 3.7 и рис. 3.13).

3.7 Таблица истинности вычитателя:

A	*C*	*D*	*Br*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Логические уравнения полувычитателя: $D = \overline{A} \wedge C \vee A \wedge \overline{C} = A \oplus C$; $Br = \overline{A} \wedge C$, где A – уменьшаемое; C – вычитаемое; D – разность (от английского *Difference* – разность); *Br* – заем (от английского *Borrow* – заем).

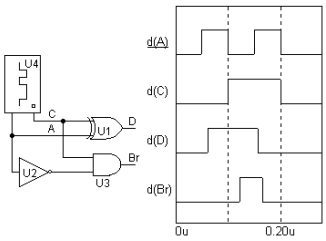


Рис. 3.13 Полувычитатель и временная диаграмма его работы

Закон функционирования полного вычитателя, выполняющего действие *A-C-Brin*, приведен в табл. 3.8.

Логические уравнения полного сумматора:

$$D = \overline{A} \wedge \overline{C} \wedge Brin \vee \overline{A} \wedge C \wedge \overline{Brin} \vee$$
$$\vee A \wedge \overline{C} \wedge \overline{Brin} \vee A \wedge C \wedge Brin;$$

3.8 Таблица истинности полного вычитателя

A	*C*	*Brin*	*D*	*Brout*
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

для заема:

$$Brout = \overline{A} \wedge \overline{C} \wedge Brin \vee \overline{A} \wedge C \wedge \overline{Brin} \vee$$
$$\vee \overline{A} \wedge C \wedge Brin \vee A \wedge C \wedge Brin.$$

Минимизация логического уравнения:

$$D = \overline{A} \wedge \overline{C} \wedge Brin \vee \overline{A} \wedge C \wedge \overline{Brin} \vee A \wedge \overline{C} \wedge \overline{Brin} \vee A \wedge C \wedge Brin =$$
$$= \overline{Brin} \wedge (\overline{A} \wedge C \vee A \wedge \overline{C}) \vee Brin \wedge (\overline{A} \wedge \overline{C} \vee A \wedge C) =$$
$$= \overline{Brin} \wedge (A \oplus C) \vee Brin \wedge \overline{(A \oplus C)} = Brin \oplus (A \oplus C);$$
$$Brout = \overline{A} \wedge \overline{C} \wedge Brin \vee \overline{A} \wedge C \wedge \overline{Brin} \vee \overline{A} \wedge C \wedge Brin \vee A \wedge C \wedge Brin =$$
$$= \overline{A} \wedge C \wedge (Brin \vee \overline{Brin}) \vee Brin \wedge (\overline{A} \wedge \overline{C} \vee A \wedge C) = \overline{A} \wedge C \vee Brin \wedge \overline{(A \oplus C)}.$$

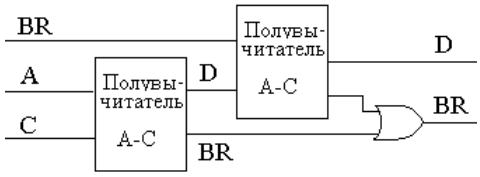


Рис. 3.14 Структурная схема полного вычитателя с использованием двух полувычитателей

На рис. 3.14 изображена структурная схема полного вычитателя с использованием двух полувычитателей, а на рис. 3.15 приведена схема и временной анализ работы полного вычитателя. На выходе вентиля *U6* получается результат операции "эквивалентность" переменных *A* и *C* ($\overline{A \oplus C}$).На выходе вентиля *U7* осуществляется конъюнкция переменных Brin и $\overline{A \oplus C}$. *Brout* является результатом дизъюнкции вентилем *U9* переменных $\overline{A} \wedge C$ и результата вентиля *U7*.

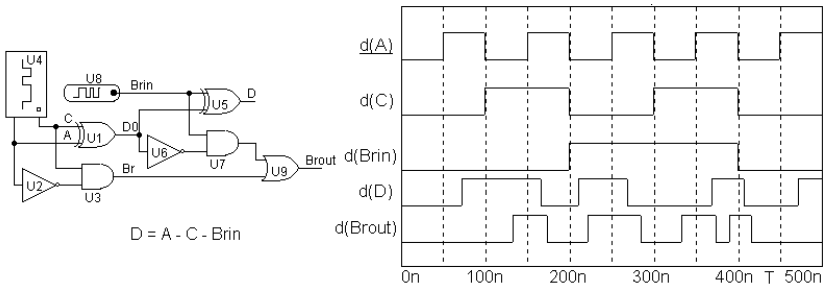


Рис. 3.15 Схема и временные диаграммы работы полного вычитателя

В двоичной арифметике вычитание можно заменить сложением, используя следующие приемы:

- при записи отрицательных чисел в обратном коде можно инвертировать вычитаемое и прибавить его к уменьшаемому. Если при сложении отрицательных чисел в старшем разряде возникает перенос, то бит переноса необходимо прибавить к младшему биту суммы.
- при записи отрицательных чисел в дополнительном коде необходимо постоянно прибавлять 1 к младшему разряду суммы.

Рассмотрим изложенное на примере вычитания двух десятичных чисел и их двоичным эквивалентом: 10 – 6 = 4; 1010 – 0110 (рис. 3.16).

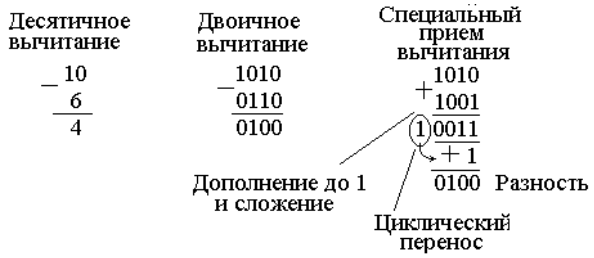


Рис. 3.16 **Вычитание двоичных чисел способом дополнения до** 1 **с циклическим переносом**

Исходя из приведенного примера, можно спроектировать параллельный универсальный сумматор/вычитатель. Функциональная схема и временной анализ универсального сумматора/вычитателя приведены на рис. 3.17. В схеме использован четырехразрядный сумматор 74283 и инверторы выполненные на базе четырех четвертьсумматоров (*U11–U14*), а также вентиль И, обеспечивающий циклический перенос из старшего разряда сумматора в старший, в режиме вычитания.

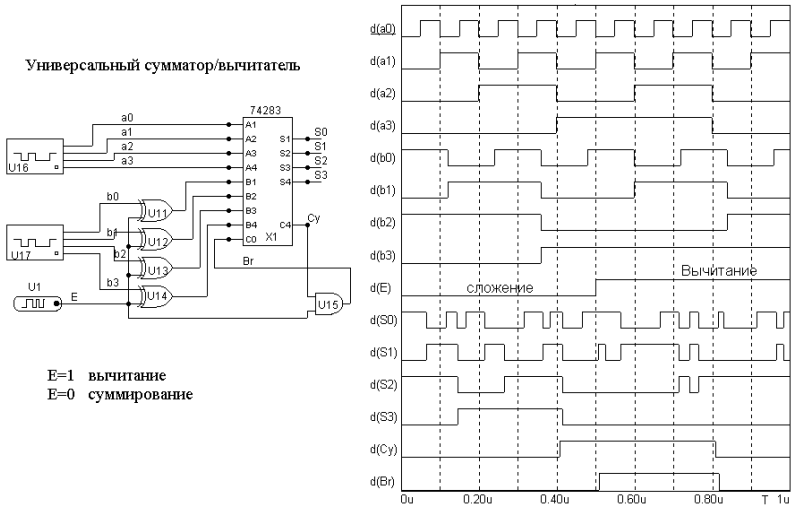


Рис. 3.17 Схема и диаграмма четырех битного универсального сумматора/вычитателя с представлением отрицательного числа дополнительным кодом

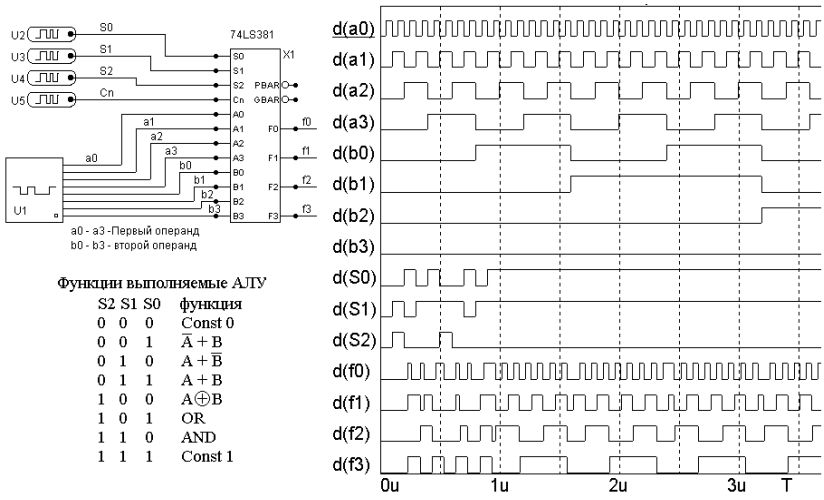
3.3.1 Арифметико-логическое устройство

Арифметико-логическим устройством (АЛУ) называют операционный блок процессора, выполняющий арифметические и логические операции над входными операндами (числами). Основным устройством АЛУ является сумматор, предназначенный для сложения или вычитания двоичных чисел. Кроме того, АЛУ выполняет и другие функции, например, инверсию, суммирование по модулю 2, константу 0 или 1, логические операции конъюнкции, дизъюнкции и др.

Микросхема АЛУ 74381 выполняет восемь элементарных функций, в зависимости от команды поданной на управляющие входы *s0 s1 s2* с помощью генераторов команд (см. рис. 3.18).

Входные операнды (числа) *A* и *B* генерируются двоичным источником, а результирующая функция

снимаются с выходов *f0 f1 f2 f3*. Например, на рис. 3.18, *а*, *б*) показано выполнение операции сложения без знака по команде *s0*=1 *s1*=1 *s2*=0.



Функции выполняемые АЛУ

S2	S1	S0	функция
0	0	0	Const 0
0	0	1	$\overline{A}+B$
0	1	0	$A+\overline{B}$
0	1	1	$A+B$
1	0	0	$A \oplus B$
1	0	1	OR
1	1	0	AND
1	1	1	Const 1

Рис. 3.18 Схема и диаграмма четырехбитного АЛУ в режиме беззнакового суммирования

Цифровые схемы сравнения формируют на выходе *F*=1 при равенстве подаваемых на вход двух двоичных чисел *A* (поразрядно записываем *a* и *b*) и *B* (*c* и *d*). Цифровая схема сравнения это цифровой аналог компаратора (см. рис. 3.19), являющегося одним из важнейших устройств импульсной техники. На основе таблицы истинности для компаратора составим уравнения (для *A*>*B*, *A*<*B*, *A*=*B*), минимизируем их, используя законы алгебры логики (см. табл. 3.7).

Логические уравнения:

$$A > B = \overline{a} \cdot b \cdot \overline{c} \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{d} + a \cdot \overline{b} \cdot \overline{c} \cdot d + a \cdot b \cdot \overline{c} \cdot \overline{d} + a \cdot b \cdot \overline{c} \cdot d + a \cdot b \cdot c \cdot \overline{d}$$

$$A < B = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot d + \overline{a} \cdot \overline{b} \cdot c \cdot \overline{d} + \overline{a} \cdot \overline{b} \cdot c \cdot d + \overline{a} \cdot b \cdot c \cdot \overline{d} + \overline{a} \cdot b \cdot c \cdot d + a \cdot \overline{b} \cdot c \cdot d \quad (A = B) = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d} + \overline{a} \cdot b \cdot \overline{c} \cdot d + a \cdot \overline{b} \cdot c \cdot \overline{d} + a \cdot b \cdot c \cdot d .$$

3.7 Таблица истинности цифровой схемы сравнения

A		*B*		*A*>*B*	*A*<*B*	*A*=*B*
a	*b*	*c*	*d*			
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

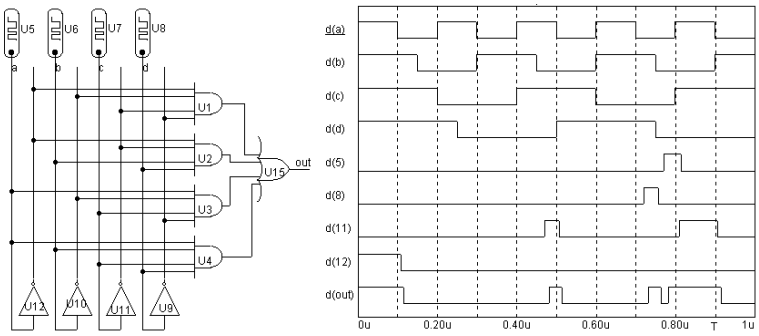


Рис. 3.19 Схема сравнения двух двухразрядных чисел на равенство и временной анализ ее работы

Компаратор, построенный на базе микросхемы 7485 из библиотеки *MC* состоит из полного многоразрядного вычитателя и логики сравнения (см. рис. 3.20), причем режим работы микросхемы зависит от команды управления, поданной на ее вход: *A*0 – *A*3 число *A*, *B*0 – *B*3 – число *B*, *ALTBI*, *AEQBI*, *AGTBI* – команды больше, меньше, равно.

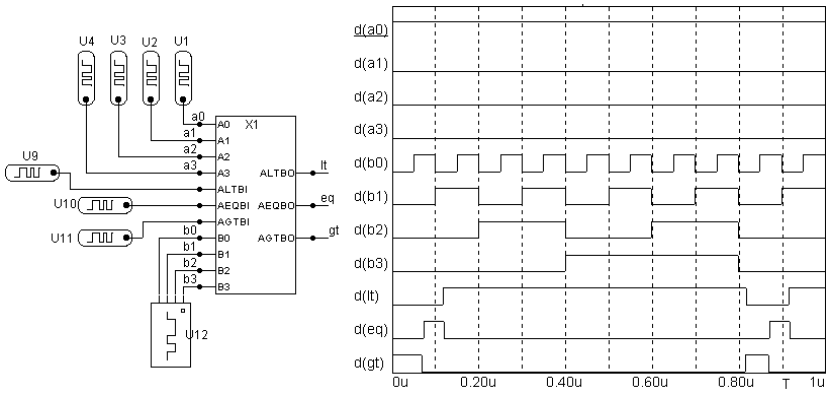


Рис. 3.20 Схемы и временные диаграммы компаратора

Все рассмотренные в данной главе комбинационные устройства могут быть реализованы на матричной логике – программируемых логических матрицах (ПЛМ) или ПМЛ. При этом проектирование функционального назначения ИМС сводится к составлению булева выражения для каждого выхода многовыходного устройства. Например, проектирование четырехразрядного сумматора можно свести к двум последовательно соединенным матрицам – матрице "И" и матрице "ИЛИ", где матрица "И" выполняет роль дешифратора, а матрица "ИЛИ" роль шифратора. Входами матрицы И являются слова составленные из слагаемых и бита переноса. Количество выходов дешифратора составляет 512 (2^{8+1}). Каждый выход представляет собой унитарный код, который с помощью шифратора преобразуется в пятиразрядный код результата сложения. Из них один разряд – бит переноса. В таком сумматоре результат и бит переноса формируются одновременно, что положительно сказывается на быстродействии.

В современных микропроцессорах практически все функциональные узлы комбинационного типа проектируются на основе матричной логики.

4 ПОСЛЕДОВАТЕЛЬНОСТНАЯ СХЕМОТЕХНИКА

Последовательностными логическими схемами называют полные цифровые автоматы, выходные сигналы которых зависят не только от состояния входных сигналов в текущий момент времени, но и от

состояния схемы в предыдущий (предыдущие) моменты времени, т.е. от последовательности входных сигналов, следовательно, данные схемы обладают памятью.

Простейшими типами последовательностных схем являются триггеры.

4.1 ТРИГГЕРЫ

Триггером (**flip-flop**) называют простейшую последовательностную логическую схему, имеющую два устойчивых состояния, обозначаемые как "1" и "0", и сохраняющую эти состояния сколь угодно долго.

Большинство триггеров имеют два выхода прямой Q и инверсный $\overline{Q}$, т.е. $Q = 1$, $\overline{Q} = 0$ или наоборот $Q = 0$, $\overline{Q} = 1$. Состояние триггера определяют по значению выхода Q – нулевое, когда $Q = 0$ и единичное, если $Q = 1$.

Триггер изменяет свое состояние при некоторых сочетаниях входных сигналов (режим переключения) и сохраняет свое состояние при действии других сочетаний сигналов (режим хранения), т.е. обладает памятью. Существует большое количество триггеров разного типа, построенных на элементах И-НЕ, ИЛИ-НЕ, которые синтезируются как комбинационные логические схемы, а также триггеры в виде интегральной микросхемы. По способу функционирования различают триггеры: *RS* – триггеры с раздельной установкой; *D* – триггеры задержки; *T* – счетные триггеры; *JK* – универсальные триггеры.

Название триггеров определяются первыми буквами английских слов: *set* – установить, *reset* – сбросить, *toggle* – релаксатор, *delay* – задержка, *jerk* – резко включить, *kill* – резко выключить. По способу синхронизации триггеры разделяются на асинхронные и синхронные или тактируемые.

Micro-Cap предоставляет пользователю примитивы следующих видов триггеров *RS*, *D*, и *JK*.

Рассмотрим пример синтеза отмеченных выше триггеров и временные диаграммы прямого и инверсного выходов.

Учитывая, что состояние последовательностной схемы зависит не только от состояния входных сигналов, но и от состояния выходов схемы, в качестве входной переменной в таблице истинности необходимо использовать значение выхода наряду с входными сигналами R и S сменить регистр для *RS*-триггера. Составим табл. 4.1 переключения асинхронного *RS*-триггера.

4.1 Таблица для асинхронного *RS*-триггера

Q_t	S_t	R_t	Q_{t+1}	Действие
0	0	0	0	Хранение 0
0	0	1	0	Подтверждение 0
0	1	0	1	Запись 1
0	1	1	×	Запрещено
1	0	0	1	Хранение 1
1	0	1	0	Запись 1
1	1	0	1	Подтверждение 1
1	1	1	×	Запрещено

Q_t – предыдущее состояние триггера; Q_{t+1} – последующее состояние триггера; × – запрещенное состояние неопределенности триггера

Запрещенной является комбинация входных сигналов, вызывающая неопределенное состояние триггера. Эта комбинация может быть выражена условием $R_tS_t = 0$, т.е. нельзя одновременно выполнить две противоречивые команды.

С целью получения минимальных сумм произведений минтермов, необходимо, чтобы количество конституент единицы в таблице истинности было максимальным. Поэтому, карта Карно составляется для $Q_{(t+1)}$, с доопределенными единичными значениями $Q_{(t+1)}$, соответствующими запрещенным комбинациям R_t и S_t. Если в карте Карно объединить клетки с единичными значениями в группы (рис. 4.1, *а*), получается минимальная сумма произведений минтермов, представляющая закон функционирования

RS-триггера, называемый ***характеристическим уравнением*** *RS*-триггера:

$$Q_{(t+1)} = S_t \vee \overline{R_t} Q_t \,.\; R_tS_t = 0. \qquad (4.1)$$

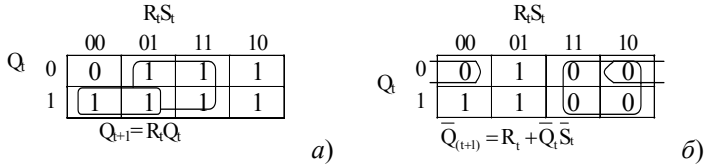


Рис. 4.1 Минимизирующие карты Карно для асинхронного *RS*-триггера с инверсными *а* и прямыми *б* входами

Выбрав в качестве элементной базы базис Шеффера (И-НЕ), функция (4.1), используя закон отрицания (правило де Моргана), преобразовывается к виду

$$Q_{(t+1)} = \overline{\overline{S_t}\,\overline{R_t Q_t}}\,, \quad \overline{R_t} \vee \overline{S_t} = 1\,. \qquad (4.2)$$

На рис. 4.2, *а* приведена схема

асинхронного *RS*-триггера, реализующая выражение (4.2). Она построена на двух логических элементах И-НЕ (*U1*, *U2*), связанных выходом каждого элемента И-НЕ, которые подключены к одному из входов другого, что образует закольцованное соединение двух вентилей. Такое соединение элементов в схеме обеспечивает два устойчивых состояния.

На рис. 4.2, *б* показано условное графическое обозначение *RS*-триггера с инверсными входами.

Из временной диаграммы (рис. 4.2, *в*) видно, что для данного триггера комбинация входных сигналов $R_t = 0$ и $S_t = 0$ является запрещенной, а комбинация $R_t = 1$ и $S_t = 1$ не меняет его предыдущего состояния и переводит триггер в режим хранения 1 бита информации. На участке от 0.6 мкс до 0.7 мкс, временной диаграммы, запрещенному состоянию соответствует высокий уровень выходного напряжения на обоих выходах триггера, что является недопустимым состоянием для *RS*-триггера.

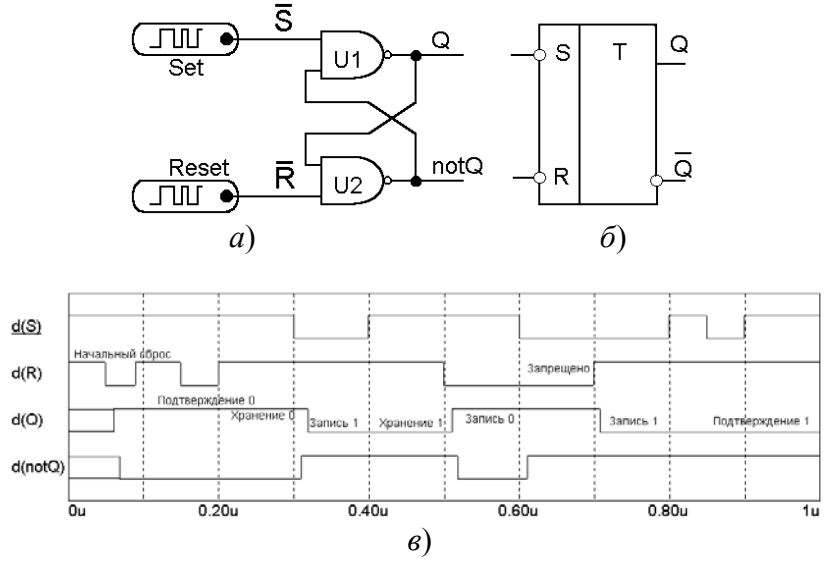


Рис. 4.2 Асинхронный *RS*-триггер с инверсными входами:
а – функциональная схема; *б* – условное графическое обозначение;
в – временная диаграмма

Триггер изменяет свое состояние под воздействием входных сигналов низкого уровня – логического нуля и называется ***RS*-триггером с инверсными входами**.

Из карты Карно можно получить минимальное произведение сумм, если доопределить в табл. 4.2 значения Q_{t+1}, соответствующие запрещенным комбинациям, нулями (см. рис. 4.2). Тогда, объединив в карте Карно нулевые значения в группы, можно получить

$$\overline{Q}_{(t+1)} = R_t + \overline{Q}_t \overline{S}_t, \quad \overline{R}_t + \overline{S}_t = 1. \qquad (4.3)$$

Выбрав в качестве элементной базы базис Пирса (ИЛИ-НЕ) и, используя закон отрицания, выражение (4.3) преобразовывается к виду

$$\overline{\overline{Q}}_{(t+1)} = \overline{\overline{R_t + \overline{Q}_t \overline{S}_t}} = \overline{Q}_{(t+1)} = \overline{\overline{R_t + \overline{Q}_t} + S_t}. \qquad (4.4)$$

На рис. 3, *а* приведена схема асинхронного *RS*-триггера на двух логических элементах ИЛИ-НЕ, (*U1*,*U2*) реализующая выражение (4.4). Последовательное закольцованное соединение элементов в схеме обеспечивает также два устойчивых состояния.

Комбинация входных сигналов $R_t = 1$ и $S_t = 1$ для данного триггера является запрещенной, а комбинация $R_t = 0$ и $S_t = 0$ переводит триггер в режим хранения. Для такого триггера активным уровнем входных сигналов является высокое значение, то есть логической единицы, при которых он меняет свое состояние. Поэтому такой триггер называется ***RS-триггером с прямыми входами***.

В режиме хранения информации необходимо поддерживать на входах *R* и *S* триггеров потенциалы, соответствующие логической 1, для триггера с инверсными входами (см. рис. 4.2, *а*) и логическому 0, для триггера с прямыми входами (рис. 4.3, *а*).

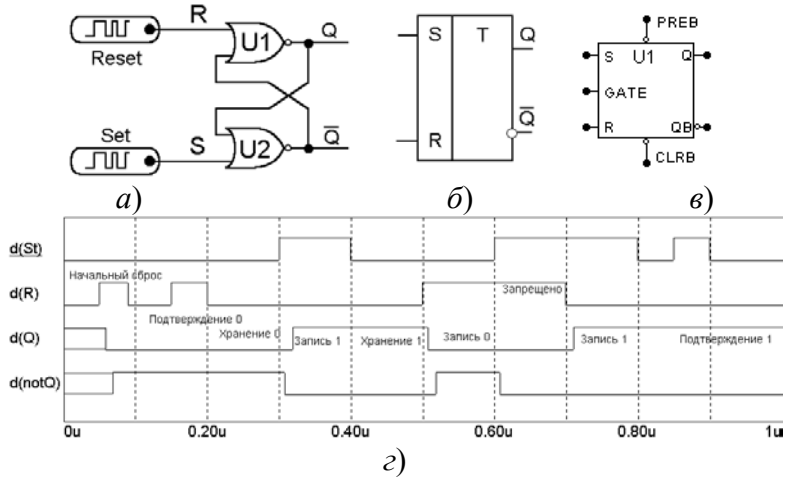


Рис. 4.3 Асинхронный *RS*-триггер с прямыми входами:
а – функциональная схема; *б*, *в* – условное графическое обозначение;
г – временная диаграмма

При переключении асинхронного *RS*-триггера из одного состояния в другое его элементы последовательно переключаются (на рис. 4.2, *в* и рис. 4.3, *в* видно, что первым переключается тот вентиль на который воздействует активное значение входного сигнала) и время переключения $t_{\text{пер}}$ равно удвоенному среднему времени задержки распространения сигнала в логическом элементе:

$$t_{\text{пер}} = 2t_{\text{И-НЕ}} \quad \text{и} \quad t_{\text{пер}} = 2t_{\text{ИЛИ-НЕ}},$$

где $t_{\text{ИЛИ-НЕ}}$ – время задержки сигнала на элементе ИЛИ-НЕ; $t_{\text{И-НЕ}}$ – время задержки сигнала на элементе И-НЕ.

Очевидно, чем меньше $t_{\text{пер}}$ тем большее число переключений триггера можно произвести в единицу времени, а следовательно, будет выше допустимая частота переключений, т.е. быстродействие триггера.

4.1.1 Синхронные *RS*-триггеры

Работа цифровых систем сопровождается негативным явлением: **гонками** или **состязаниями**. Это происходит оттого, что на входы логического элемента (ЛЭ) сигналы не всегда поступают одновременно, так как перед этим они могут проходить через цепи, обладающие различной задержкой. В результате таких состязаний новые значения одних сигналов будут сочетаться с предыдущими значениями других, что может привести к ложному срабатыванию ЛЭ (устройства).

Одним из способов устранения гонок является применение временного стробирования – когда информационные сигналы воздействуют на схему только при подаче тактирующих (синхронизирующих) импульсов, к моменту прихода которых информационные сигналы успевают установиться на входах.

Основное условие правильной работы логических каскадов на *RS*-триггерах и управляемых ими логических схем – отсутствие одновременного действия сигналов *Rt* или *St*, переключающих триггер.

Синхронные *RS*-триггеры, кроме информационных входов *R* и *S*, имеют вход синхронизации *C*. Работает такой триггер так: если на синхронизирующем входе действует логический уровень $C_t = 0$, то триггер сохраняет свое состояние, а если $C_t = 1$, то он работает в режиме асинхронного *RS*-триггера.

По табл. 4.2 для синхронного *RS*-триггера, тактируемого уровнем логической 1 ($C_t = 1$), составим карту Карно (рис. 4.4) для Q_{t+1} предварительно доопределив значения $Q_{(t+i)}$, соответствующие запрещенным комбинациям R_t и S_t, единицами. Объединим единичные клетки в группы. Тогда на основании карты Карно характеристическое уравнение синхронного *RS*-триггера будет иметь вид

$Q_{(t+1)} = S_t C_t \vee \overline{R}_t Q_t \vee Q_t \overline{C}_t$, (4.5)

Таблица 4.2 Таблица для синхронного *RS*-триггера

Номер набора *i*	S_t	R_t	Qt	$Q_{(t+1)}$ при $C_t = 0$	$Q_{(t+1)}$ при $C_t = 1$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	1
6	1	1	0	0	X
7	1	1	1	1	X

из которого следует, что при $C_t = 0$, $Q_{(t+1)} = Q_t$ т.е. триггер сохраняет свое состояние, а при $C_t = 1$, $Q_{(t+1)} = S_t \vee \overline{R}_t Q_t$, т.е. получаем выражение (4.1) – характеристическое уравнение асинхронного *RS*-триггера.

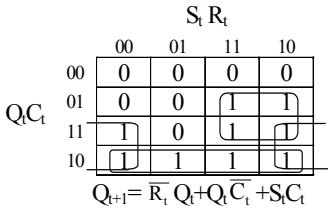
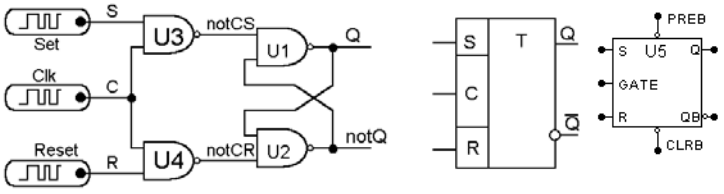


Рис. 4.4 Минимизирующая карта Карно для синхронного *RS*-триггера

На рис. 4.5, *а* приведена функциональная схема, соответствующая характеристическому уравнению (4.5) синхронного *RS*-триггера со статическими входами, тактируемого уровнем логической 1.



а) *б*) *в*)

Рис. 4.5 Синхронный *RS*-триггер, тактируемый уровнем логической 1: *а* – функциональная схема; *б* – условное графическое обозначение, принятая в отечественной схемотехнике; *в* – условное графическое обозначение в *MC*

Элементы И-НЕ (*U3*) и И-НЕ (*U4*) принимают и передают переключающую логическую 1 с информационного входа *S* или *R* на соответствующие входы асинхронного *RS*-триггера с инверсными входами (элементы И-НЕ (*U1*) и И–НЕ (*U2*)) только при наличии на синхронном входе *C* уровня логической 1.

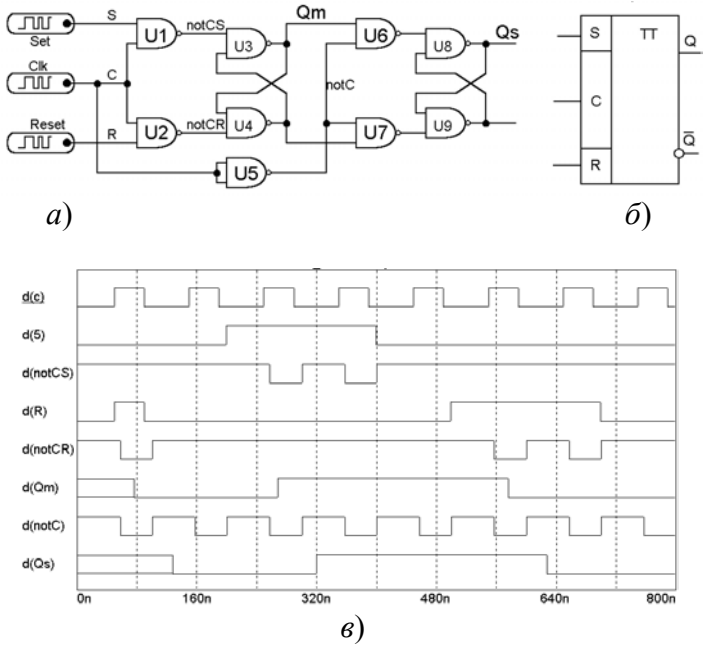
Синтез функциональной схемы одноступенчатого синхронного *RS*-триггера, тактируемого уровнем логического 0, может быть произведен аналогичным образом, если доопределить запрещенные состояния в карте Карно нулями.

4.1.2 Двухступенчатый синхронный *RS*-триггер

Для выполнения условий надежной работы логических каскадов на каждый двоичный разряд, хранящий 1 бит информации, используют два триггера, которые управляются двумя сдвинутыми во времени тактирующими импульсами. Это дает возможность использовать информацию, снимаемую с выходов триггера, для управления сигналами на его входах, что необходимо при построении более сложных триггерных схем. Второй тактирующий импульс можно получить инвертированием входного тактирующего сигнала. Тактирование триггера фронтом импульса или перепадом потенциала можно обеспечить, выполняя его двухступенчатым.

На рис. 4.6, *а* приведена функциональная схема двухступенчатого синхронного *RS*-триггера, каждая ступень которого представляет синхронный *RS*-триггер.

Если на синхронизирующий вход подается сигнал $C = 1$, то входная информация, определяемая сигналами на *R*- и *S*-входах, принимается в основной – ведущий триггер *M*, образованный вентилями *U3* и *U4*.



а) *б*)

в)

Рис. 4.6 Двухступенчатый синхронный *RS*-триггер:
а – функциональная схема; *б* – условное графическое обозначение;
в – временная диаграмма работы

При этом состояние ведомого триггера сохраняется соответствующим моменту времени *t*, запись в триггер *S* блокируется уровнем логического 0 с выхода элемента НЕ (*U5*).

Двухступенчатые синхронные триггеры называют *MS*-триггерами. Название происходит от начальных букв английских слов *Master* (хозяин) – *Slave* (раб).

Как только импульс синхронизации примет значение $C = 0$, триггер *M*, образованный вентилями *U8* и *U9*, будет переведен в режим хранения информации, а с инвертора НЕ (вентиль *U5*) уровень логической 1 запишет информационное состояние триггера *M* в триггер *S*.

Временная диаграмма работы двухступенчатого синхронного *RS*-триггера со статическим управлением и его условное графическое обозначение приведены на, рис. 4.6, *б*, *в*, соответственно.

Синхронный триггер обычно имеет дополнительные асинхронные входы *R* и *S*, по которым он независимо от сигнала на синхронизирующем входе С переключается в состояние 0 или 1.

Триггер (рис. 4.6, *а*) работает как обычный двухступенчатый синхронный триггер при наличии на входах *R* и *S* уровня логической 1.

Подчеркнем разницу между одноступенчатым и двухступенчатым синхронными *RS*-триггерами со статическим управлением. Одноступенчатый синхронный триггер (см. рис. 4.5, *а*) можно переключить, если при $C = 1$ изменить комбинацию на установочных входах с прежней (например, $R = 0$, $S = 1$) на новую ($R = 1$, $S = 0$) или наоборот. В двухступенчатом синхронном триггере (рис. 4.6, *а*) при $C = 1$ вторая ступень отключена от первой, а при $C = 0$ первая ступень не принимает информацию с входов *R* и *S*. Лишь при изменении сигнала на синхронизирующем *C* входе с высокого на низкий уровень, информация из первой ступени переписывается во вторую ступень и состояния выходов *Q*, и $\overline{Q}$, изменяются.

4.1.3 *D*-триггер

4.3 Комбинации *D*-триггера

Q_t	D_t	$Q_{(t+1)}$
0	0	0
0	1	1
1	0	0
1	1	1

Триггеры этого типа имеют один информационный вход *D* и реализуют функцию временной задержки.

В соответствии с таблицей переходов (табл. 4.3) закон функционирования *D*-триггера описывается характеристическим уравнением:

$$Q_{t+1} = D_t. \qquad (4.6)$$

Такой триггер не обладает памятью. В связи с этим асинхронные *D*-триггеры

не применяются, так как его выход будет повторять входной сигнал с некоторой задержкой во времени.

Синхронные *D*-триггеры строятся на базе одноступенчатых и двухступенчатых синхронных *RS*-триггеров.

Синхронный *D*-триггер функционирует в соответствии с таблицей переходов (табл. 4.4) Из карты Карно (рис. 4.7) вытекает характеристическое уравнение для *D*-триггера:

$$Q_{(t+1)} = D_t C_t \vee Q_t \overline{C_t} \qquad (4.7)$$

Из уравнения (4.7) следует, что при $C_t = 0$, $Q_{t+1} = Q_t$, а при $C_t = 1$, $Q_{t+1} = D_t$.

Одноступенчатый *D*-триггер, реализующий характеристическое уравнение (4.7), может быть построен из одноступенчатого синхронного *RS*-триггера и элемента И-НЕ (*U1*) (рис. 4.8, *а*), объединяющего инверсные входы *D*-триггера в один информационный вход *D*.

При *C* = 0 синхронный *RS*-триггер на рис. 4.8, *а*, заблокирован уровнем логической 1 с выходов элементов И-НЕ (*U2*) и И-НЕ (*U3*). При *C* = 1 уровень, поданный на информационный вход *D*, создает уровень логического 0 либо на входе *S* (при *D* = 1), либо на входе *R* (при *D* = 0) асинхронного триггера *T* и триггер *T* устанавливается в состояние, соответствующее логическому уровню на входе *D* (рис. 4.8, *б*). Как видно из временной диаграммы работы (рис. 4.8, *б*), одноступенчатый *D*-триггер задерживает распространение входного сигнала на время паузы между синхронизирующими сигналами.

4.4 Комбинации синхронного *D*-триггера

Q_t	D_t	C_t	$Q_{(t+1)}$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

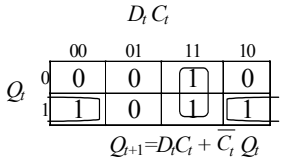


Рис. 4.7 Минимизирующая карта Карно для синхронного *D*-триггера

В *MC* (рис. 4.8, *в*) модель *D* – триггера имеет один информационный вход – *D*, другой тактирующий – *CLK* и устанавливается в состояние соответствующее входу *D* при действии тактирующего сигнала *CLK*, либо по окончанию его, т.е. с задержкой, либо без нее. Кроме того, имеются два асинхронных входа, *prebar* и *clrbar*, соответствующие входам *S* и *R*, которые активны при низком уровне входных сигналов. Таблица истинности модели такого *D*-триггера приведена ниже (см. табл. 4.5).

Двухступенчатый синхронный *D*-триггер обеспечивает задержку входного сигнала на период (на один такт) следования синхронизирующих сигналов.

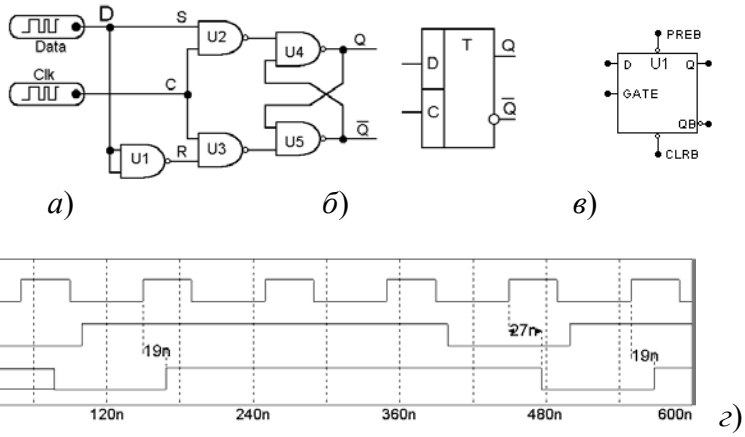


Рис. 4.8 Одноступенчатый синхронный *D*-триггер:
а – функциональная схема; *б* – условное графическое обозначение; *г* – условное графическое обозначение *D*-триггера в *MC*; *в* – временная диаграмма работы

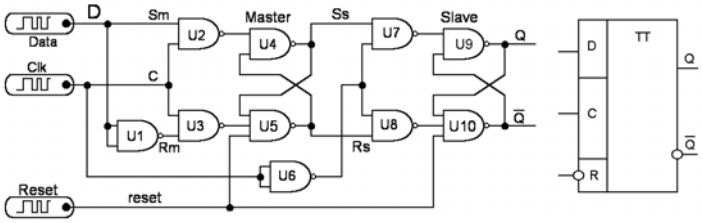
На рис. 4.9, *а* показан один из вариантов построения двухступенчатого *D*-триггера. Он состоит из одноступенчатых синхронных *D*- и *RS*-триггеров, тактируемых уровнем логической 1, и инвертора НЕ (*U6*).

При *C* = 0 информация со входа *D* не принимается в триггер *Master*. Этот уровень через инвертор НЕ подается на синхронизирующий вход триггера *Slave* и состояние триггера *Master* передается триггеру *Slave*.

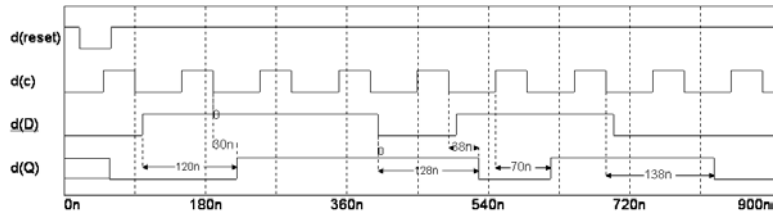
4.5 Таблица истинности моделей D-*триггера*

Входы				Выходы	
D	*CLK*	*PREBAR*	*CLRBAR*	*Q*	*QBAR*
×	×	1	0	0	1
×	×	0	1	1	0
×	×	0	0	1	0
×	0	1	1	*Q*'	*Q*B'
×	1	1	1	*Q*'	*QB*'
0	**RE**	1	1	0	1
1	**RE**	1	1	1	0
D – информационный вход; *CLK* – синхросигналы; *preb*, *clrb* – начальная установка; *Q*', *Qb*' – предыдущие состояния выхода; × – безразличное состояние; *RE* – фронт синхросигнала					

При подаче на вход *C* уровня логической 1 (*C* = 1) на синхронизирующий вход триггера *Slave* подается уровень логического 0 и связь между триггерами разрывается, а триггер *Master* сигналом *C* = 1 устанавливается в состояние, соответствующее уровню на входе *D.* После окончания действия сигнала на входе *C* (*C* = 0) производится передача состояния триггера *Master.* триггеру *Slave*.



а) *б*)



в)

Рис. 4.9 Двухступенчатый синхронный *D*-триггер:
а – функциональная схема; *б* – условное графическое обозначение;
в – временная диаграмма работы

4.1.4 *JK*-триггер

Отличие *JK* триггера от синхронного *RS*-триггера заключено в том, что при значениях входной информации, запрещенной для *RS*-триггера, он инвертирует хранимую в нем информацию.

Функционирование *JK*-триггера описывается таблицей переходов (табл. 4.6). Используя карту Карно (рис. 4.10), получим характеристическое уравнение для *JK*-триггера

$$Q_{(t+1)} = \overline{K}_t Q_t \vee J_t \overline{Q}_t .$$

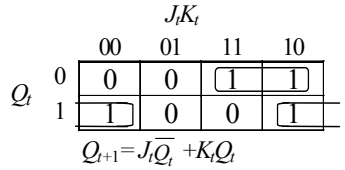


Рис. 4.10 Минимизирующая карта Карно для *JK*-триггера

(4.8)

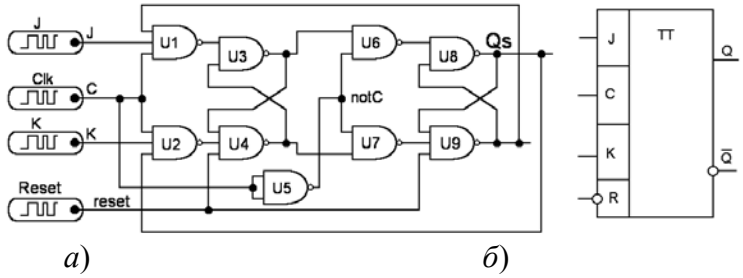
Как видно из табл. 4.6, состояние *JK*-триггера определяется не только уровнями на информационных входах *J* и *K*, но и состоянием Q_t, в котором ранее находился *JK*-триггер. Это дает возможность строить функциональные схемы *JK*-триггеров на двухступенчатых *RS*-триггерах. *JK*-триггеры могут быть асинхронными и синхронными. Интегральные *JK*-триггеры обычно выполняются синхронными.

4.6 Таблица переходов *JK*-триггера

Q_t	J_t	K_t	$Q_{(t+l)}$	Примечание
0	0	0	0	Хранение 0
0	1	0	1	Установка 1
0	0	1	0	Подтверждение 0
0	1	1	1	Инвертирование
1	0	0	1	Хранение 1
1	1	0	1	Подтверждение 1
1	0	1	0	Установка 0
1	1	1	0	Инвертирование

Для получения *JK*-триггера из двухступенчатого синхронного *RS*-триггера необходимо ввести обратные связи с выходов двухступенчатого *RS*-триггера на входы логических элементов его первой ступени.

На рис. 4.11 представлена функциональная схема двухступенчатого *JK*-триггера, состоящего из триггера T_M (*U3*, *U4*) и триггера T_S (*U8*, *U9*). *J* и *K* информационные входы.



а) *б*)

Рис. 4.11 а – функциональная схема *JK*-триггера;
***б* – условное графическое обозначение *JK*-триггера**

Рассмотрим работу *JK*-триггера. Если $J = K = 0$, то на выходах элементов И-НЕ (*U1*) и И-НЕ (*U2*) устанавливается уровень логической 1, триггер T_M и, следовательно, *JK*-триггер сохраняют прежнее состояние. Пусть *JK*-триггер находится в состоянии 0 ($Q = 0$, $\overline{Q} = 1$).

Тогда при подаче сигналов $J = 1$ и $C = 1$ на выходе элемента И-НЕ (*U1*) (рис. 4.11) установится уровень логического 0, который запишет 1 в триггер T_M, а после окончания действия синхронизирующего сигнала
($C = 0$) состояние триггера T_M уровнем логического 0 с выхода элемента И-НЕ (*U6*) передается в триггер T*s*, т.е. *JK*-триггер переключается в состояние 1 ($Q = 1$, $\overline{Q} = 0$).

Если теперь на *JK*-триггер подать сигналы $K = 1$ и $C = 1$, то с выхода элемента И-НЕ (*U2*) логический уровень 0 установит триггер T_M в состояние 0, а после окончания действия синхронизирующего сигнала (C = 0) – состояние триггера T_M уровнем логического 0 с выхода элемента И-НЕ (*U7*) будет передано в триггер *Ts*, т.е. *JK*-триггер переключается в состояние 0 ($Q = 0$, $\overline{Q} = 1$).

Таким образом сочетания сигналов $J = 1$, $K = 0$ и $J = 0$, $K = 1$ дают возможность сигналом $C = 1$ переключать *JK*-триггер в состояние 1 и 0. Нетрудно убедиться в том, что при $J = K = 1$ и $C=1$ *JK*-триггер изменяет свое состояние на противоположное. Следовательно, при подаче на вход *C* серии синхронизирующих сигналов ($J = K = 1$) *JK*-триггер работает в счетном режиме.

На базе синхронного *JK*-триггера можно реализовать асинхронный (рис. 4.12, *а*) и синхронный (рис. 4.12, *в*) *T*-триггер, *D*-триггер (рис. 4.12, *б*) и синхронный *RS*-триггер (рис. 4.12, *г*). Переключения *JK*-триггера, представленного в *MC*, показаны в табл. 4.7.

При проектировании сложных логических схем необходимы триггеры различных типов. Поэтому выгоднее изготовить универсальный тип триггера, который можно использовать в разных режимах работы и модификациях. В интегральной схемотехнике наибольшее распространение получили синхронные *D*- и *JK*-триггеры.

4.7 Таблица истинности моделей *JK*-триггеров

Входы					Выходы	
J	*K*	*CLK*	*PREBAR*	*CLRBAR*	*Q*	*QBAR*
×	×	×	1	0	0	1
×	×	×	0	1	1	0
×	×	×	0	0	1	0
×	×	0	1	1	*Q*'	*QB*'
×	×	1	1	1	*Q*'	*QB*'
0	0	**FE**	1	1	*Q*'	*QB*'
0	1	**FE**	1	1	0	1
1	0	*FE*	1	1	1	0
1	1	*FE*	1	1	*QB*'	*Q*'
Q', *QB*' – предыдущее состояние; × – безразличное состояние; *FE* – управление спадом синхросигнала						

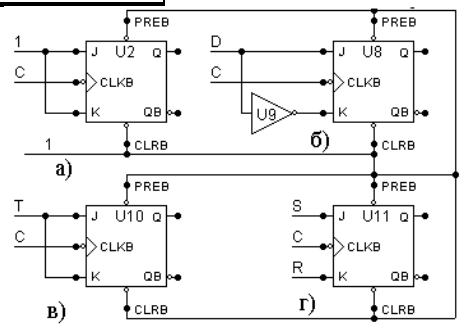


Рис. 4.12 Варианты применения *JK*-триггера: *а* – асинхронный *T*-триггер; *б* – *D*-триггер; *в* – синхронный *T*-триггер; *г* – синхронный *RS*-триггер

В цифровых системах широко используют *JK*-триггеры с групповыми *J* и *K*, и дополнительными асинхронными *R* и *S* входами. Каждая группа входов объединена конъюнкцией (рис. 4.13), что позволяет расширить логические возможности триггера.

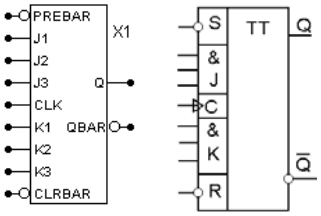


Рис. 4.13 Условное графическое обозначение универсального *JK*-триггера 74*LS*72 и его аналога К555ТВ1

4.1.5 *T*-триггер

Триггер этого типа имеет только один информационный вход *T*, называемый *счетным входом,* и меняет свое состояние на противоположное после прихода на счетный вход *T* каждого управляющего (счетного) сигнала.

В соответствии с таблицей переходов (табл. 4.8) *T*-триггера закон его функционирования описывается характеристическим уравнением

$$Q_{(t+1)} = T_t\overline{Q_t} \vee \overline{T}_t Q_t , \qquad (4.9)$$

из которого следует, что при $T_t = 0$, $Q_{(t+1)} = Q_t$, т.е. триггер сохраняет свое состояние, а при $T_t = 1$, $Q_{(t+1)} = \overline{Q_t}$ (он изменяет свое состояние на противоположное).

4.8 Закон функционирования Т-триггера

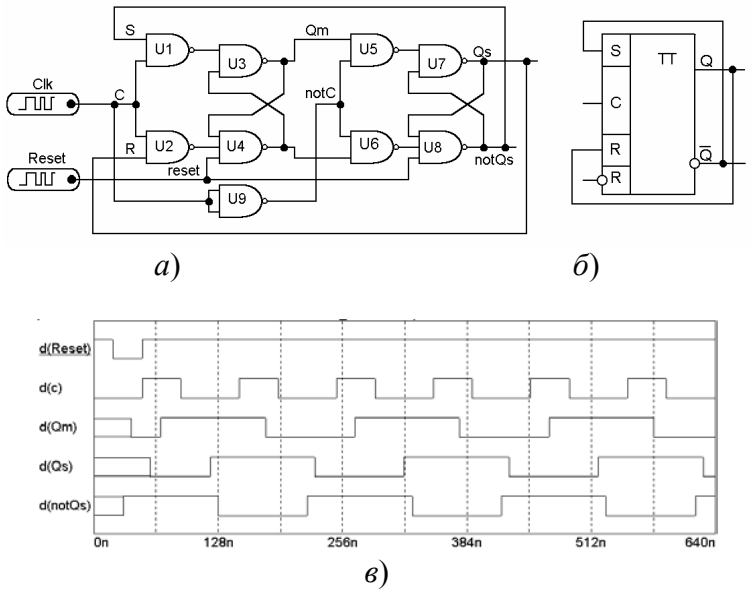
Q_t	T_t	$Q_{(t+1)}$
0	0	0
0	1	1
1	0	1
1	1	0

Как видно из табл. 4.8, *T*-триггер реализует операцию сложения по модулю, 2, что и обусловило название *T*-триггера счетным триггером, а вход *T* – счетным входом. Сигнал на его выходе *Q* появляется в два раза реже, чем на входе *T*, т.е. *T*-триггер может использоваться как делитель частоты.

Асинхронный *T*-триггер может быть построен на базе двухступенчатого синхронного *RS*-триггера с дополнительными связями: выход триггера *Q* соединяется с информационным входом *R,* а $\overline{Q}$ – информационным входом *S* (рис. 4.8, *б*). Информационным входом *T* является синхронизирующий вход *C*.

Рассмотрим работу *T*-триггера. Предварительно, сигналом *Reset* низкого уровня через асинхронный вход *R*, обе ступени триггера устанавливаются в нулевое состояние. Далее, при *T* = 0 происходит постоянное копирование состояния триггера *M* триггером *S*, так как элемент И-НЕ (*U9*) выдает уровень логической 1 на входы элементов И-НЕ (*U5*) и И-НЕ (*U6*). Если *T*-триггер находится в состоянии 0 (*Q* = 0, $\overline{Q}$ = 1), то тогда на входах *R* и *S* будут действовать уровни логического 0 и 1 соответственно (рис. 4.14).

При поступлении на вход *T* первого счетного сигнала (*T* = 1) в триггер *M* запишется 1 уровнем логического 0 с выхода элемента И-НЕ (*U1*).



а) *б*)

в)

Рис. 4.14 Двухступенчатый асинхронный *T*-триггер:
а – функциональная схема; *б* – условное графическое обозначение;
в – временная диаграмма работы

Состояние триггера *Slave* при этом не изменится, так как уровень логического 0 с выхода элемента И-НЕ (*U9*) будет блокировать его состояние. После окончания действия счетного сигнала на входе *T* (*T* = 0) триггер *Slave* установится в состояние 1 уровнем логического 0 с выхода элемента И-НЕ (*U5*) и произойдет изменение потенциалов на выходах *T*-триггера (*Q* = l, $\overline{Q}$ = 0), а также на *R* и *S* входах триггера *M*.

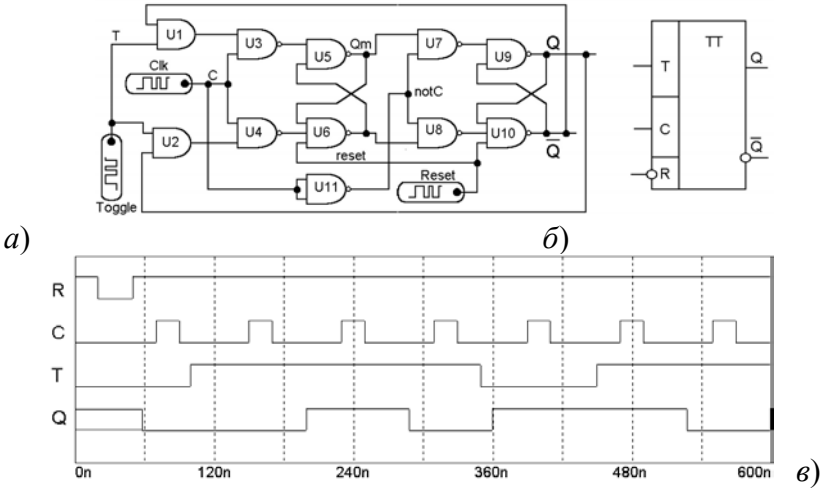
При поступлении на вход *T* второго счетного сигнала (*T* = 1) в триггер *M* запишется 0 уровнем логического 0 с выхода элемента И-НЕ (*U2*), а после окончания действия второго счетного сигнала на входе *T* (*T* = 0) в триггер *S* запишется 0 уровнем логического 0 с выхода элемента И-НЕ (*U6*). На выходах *T*-триггера произойдет изменение потенциалов (*Q* = 0, $\overline{Q}$ = l). а также на *R* и *S* входах триггера *Master*.

Таким образом, каждый счетный сигнал на входе *T* переводит триггер *M* в противоположное состояние триггера *S*, а после окончания действия сигнала на входе *T*, триггер *S* переключается в состояние, определяемое триггером *M*.

Как видно из временной диаграммы (рис. 4.14, *в*), частота сигналов, снимаемых с выходов *Qs* и $\overline{Q}$ триггера (рис. 4.14, *а*), в два раза меньше частоты входных сигналов *T*.

Синхронный *T*-триггер (рис. 4.15, *а*) используется в случае необходимости представлять потенциалом последовательность 1 на входе *T*-триггера. С помощью двухступенчатого синхронного *RS*-триггера и входной логики на вентилях И-

НЕ реализуется характеристическое уравнение (4.9) *T*-триггера.



а) *б*) *в*)

Рис. 4.15 Синхронный *T*-триггер:
а – функциональная схема; *б* – условное графическое обозначение;
в – временная диаграмма работы

Пусть *T*-триггер находится в состоянии 0 ($Q = 0$, $\overline{Q} = 1$). При $T = 0$ наличие синхронизирующего сигнала на входе *C* ($C = 1$) не сможет вызвать переключение *T*-триггера, так как входы *R* и *S* двухступенчатого синхронного *RS*-триггера заблокированы уровнем логического 0 от выходов вентилей И (*U1*) и И (*U2*) (рис. 4.14, *б*). При наличии на входе *T* сигнала высокого уровня ($T = 1$) каждый синхронизирующий сигнал на входе *C* ($C = 1$) будет вызывать переключение триггера из одного состояния в другое, причем смена состояния, как всегда в двухступенчатых синхронных *RS*-триггерах, происходит после окончания действия синхронизирующего сигнала на входе *C* ($C = 0$).

4.2 РЕГИСТРЫ

В составе любого микропроцессора, микропроцессорного комплекта или чипсета содержатся регистры, которые являются основными узлами, с помощью которых производится переработка информации, представленной в виде машинных слов.

Регистры – это устройства цифровых систем, выполняющие функции приема, хранения и передачи информации в виде машинных слов. Кроме того, с помощью регистров можно выполнять некоторые логические преобразования над машинными словами.

Регистры представляют собой полные цифровые автоматы, выполненные на триггерах того или иного типа со схемами управления входными и выходными сигналами.

С помощью регистра можно выполнить следующие операции:

- установка всех разрядов в 0;
- установка всех разрядов в 1;
- прием и хранение в регистре кода *n*-разрядного числа;
- сдвиг хранимого двоичного кода вправо и влево на заданное число разрядов;
- преобразование параллельного кода в последовательный и, наоборот, – при приеме и выдаче информации;
- поразрядные логические операции;

По способу представления информации различают параллельные и последовательные регистры.

4.2.1 Параллельные регистры

Параллельные регистры или регистры памяти применяются для ввода, хранения и вывода двоичной информации в параллельном коде. Они могут быть образованы из асинхронных и синхронных, одноступенчатых и двухступенчатых триггеров.

Однофазный параллельный регистр (рис. 4.16) построен на одноступенчатых асинхронных *RS*-триггерах. Так как на кодовые шины слова (КШС) параллельного регистра подается двоичное слово *D1D2D3D4* в прямом коде (однофазный код), то КШС с помощью элементов И-НЕ

(*U7 – U10*) подключены к инверсным асинхронным входам *S* установки в 1 триггеров регистра. Прием двоичного слова *D1D2D3D4* в регистр осуществляется в два такта. По первому такту сигналом *CLR* на асинхронном входе *R*, регистр устанавливается в состояние "0000". По второму такту сигналом прием слова (*WR*) в регистр записывается параллельный код двоичного слова *D1D2D3D4*. При этом в соответствии с обратным кодом двоичного слова *D1D2D3D4* каждый из триггеров *X1–X4* регистра будет либо переключен в состояние 1, либо останется в состоянии 0 (см. временную диаграмму на рис. 4.16, *б*).

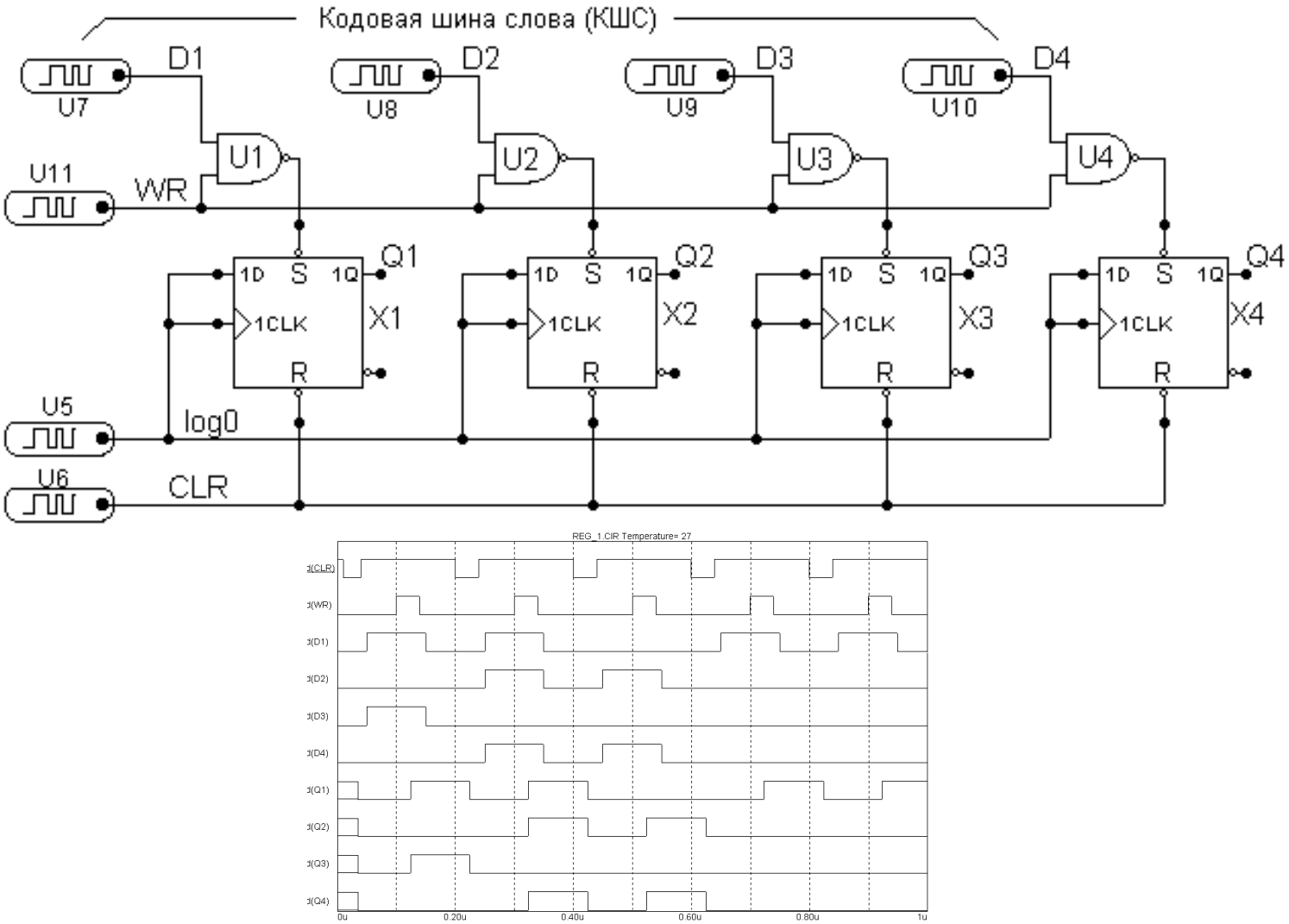


Рис. 4.16 Однофазный параллельный регистр и временная диаграмма его работы

Записанный в регистр код двоичного слова может храниться до тех пор, пока регистр не будет установлен сигналом *CLR* в состояние "0000".

Сигнал *WR* обычно импульсный, так как кратковременное подключение регистра к КШС уменьшает вероятность занесения в регистр ошибочной информации.

Записанный в регистр код двоичного слова может храниться до тех пор, пока регистр не будет установлен сигналом *CL*R в состояние "0000".

Сигнал *WR* обычно импульсный, так как кратковременное подключение регистра к КШС уменьшает вероятность занесения в регистр ошибочной информации.

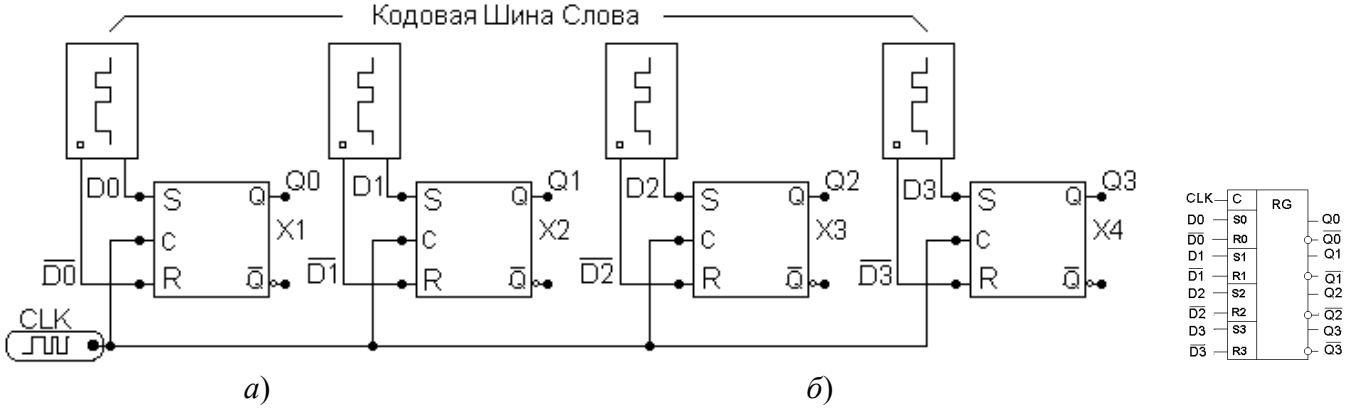
Таким образом, при записи двоичного слова в регистр в худшем случае каждый его триггер будет переключаться дважды: один раз при установке регистра в нулевое состояние сигналом *CLR* и второй раз при приеме 1 в данный триггер по сигналу *WR*.

Быстродействие регистра – время записи в регистр $T_{\text{рг зап}}$ двоичной информации определяется минимально допустимым временем между поступлениями очередных кодов на входах регистра: $T_{\text{рг. зап}} = 2t_{\text{Т}} + t_{\text{И}}$, где $t_{\text{Т}}$ – время задержки сигнала триггером; $t_{\text{И}}$ – время задержки сигнала на элементе И.

Прямой и обратный код двоичного слова снимается соответственно с выходов $Q1Q2Q3Q4$ или $\overline{Q}1\overline{Q}2\overline{Q}3\overline{Q}4$.

4.2.2 Парафазный параллельный регистр

Парафазный параллельный регистр (рис. 4.17) построен на одноступенчатых синхронных *RS*-триггерах. При наличии на кодовых шинах слова (КШС) парафазного кода на одном из входов каждого триггера регистра обязательно присутствует 1, которая по сигналу *WR1* и установит триггер в требуемое состояние независимо от той информации, которая в нем хранилась.



а) *б*)

Рис. 4.17 Парафазный параллельный регистр:
а – функциональная схема; *б* – условное графическое обозначение

Использование парафазного кода позволяет ускорить запись в регистр $T_{\text{рг. зап}}$ информации за счет исключения такта предварительной установки регистра в нулевое состояние: $T_{\text{Рг. зап}} = t_{\text{И}}$

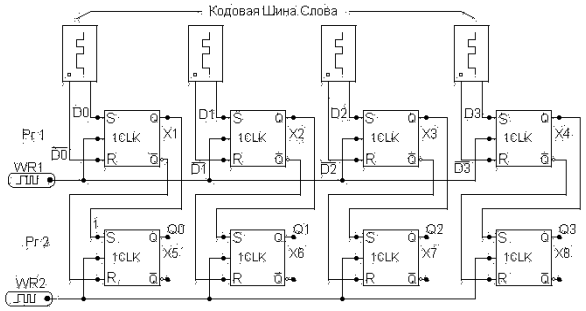


Рис. 4.18 Схема передачи слова из одного параллельного регистра в другой

При выполнении арифметических и логических операций над двоичными словами возникает необходимость в их передаче с одного регистра на другой. Это действие называют операцией передачи слова. На рис. 4.18 приведена функциональная схема передачи двоичного слова из одного параллельного регистра в другой. Двоичное слово, представленное в парафазном коде, записывается в регистр Рг1 управляющим сигналом *WR1* (прием в регистр Рг1); сигналом *WR2* (прием в регистр Рг2) двоичное слово передается из Рг1 в Рг2, при этом состояние Рг1 не изменяется.

Сдвигающие регистры обеспечивают запись и считывание при последовательном обмене информацией с жесткими и гибкими дисками, *CD*, осуществляют связь между устройствами через порты последовательного обмена, такие как *COM*, *USB*, модем, монитор и др.

4.2.3 Сдвигающие регистры

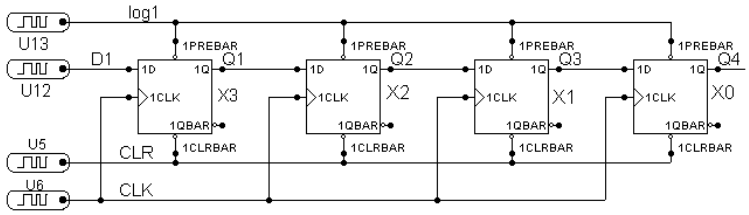
Сдвигающие, **или последовательные, регистры** – это регистры выполняющие сдвиг двоичной информации вправо или влево, в зависимости от управляющих сигналов.

Сдвиг вправо заключается в том, что значение, хранящееся в i-ом разряде, переходит в (i+1)-й разряд; из (i+1)-ого в (i+2)-ой и т.д. Из закона функционирования сдвигающего регистра следует, что в каждом разряде регистра одновременно с хранением цифры, имевшейся до сдвига и предназначенной для передачи в следующий разряд, необходимо предусматривать возможность приема новой цифры из предыдущего разряда. Эти условия выполняются при построении сдвигающих регистров на двухступенчатых синхронных триггерах, работающих в режиме D, или D-триггерах с динамическим входом синхронизации.

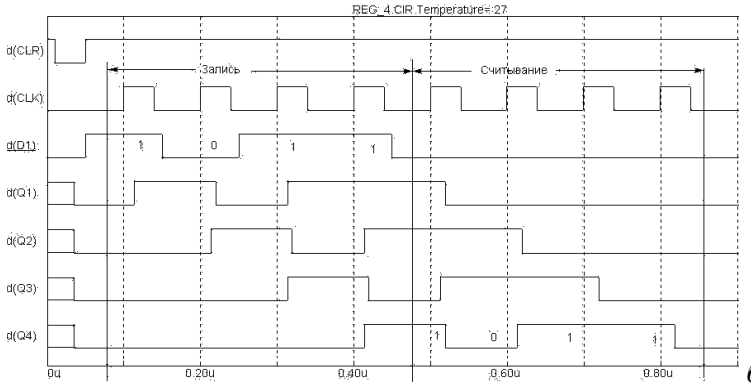
При построении сложных логических схем используются регистры, которые могут выполнять сдвиг кода как вправо, так и влево. Такие регистры называются **реверсивными**.

Сдвигающие регистры, в которых ввод и вывод двоичного слова производится в последовательном коде, называют последовательными регистрами.

На рис. 4.19, *а* сдвигающий (последовательный) регистр построен на D-триггерах с динамическим синхронизирующим входом, которые соединены последовательно таким образом, что сигнал (Q_i) с выхода триггера старшего разряда регистра поступает на информационный вход D_{i-1} триггера последующего младшего разряда регистра. Все триггеры управляются



а)



б)

Рис. 4.19 Сдвигающий регистр на *D*-триггерах:
а – функциональная схема; *б* – временная диаграмма работы

общим входом синхронизации. Такое соединение D-триггеров обеспечивает сдвиг в регистре информации вправо, если управление сдвигом осуществлять синхронизирующими сигналами, называемыми управляющими сигналами сдвига вправо – *CLK*. Установка регистра в состояние "0000" производится управляющим сигналом *CLR*.

Информационный вход D используют для подачи двоичного слова последовательно разряд за разрядом (первым в регистр записывается старший разряд).

Запись в регистр последовательного двоичного кода, например *D1D2D3D4*=1101, производится через информационный вход D следующим образом.

Пусть регистр находится в состоянии "0000". Тогда при $D = D0 = 1$ первый сигнал сдвига *CLK* установит триггер *X3* в состояние 1, остальные *X2*, *X1* и *X0* не изменят своего состояния, т.е. в регистре установится двоичный код "1000" (рис. 4.5, *б*). При $D = X1 = 0$ второй сигнал сдвига *CLK* установит *X3* в

состояние 0, а его информационное состояние *X0* = 1 будет передано в *X2*; *X1* и *X0* не изменят своих состояний, т.е. в регистре установится двоичный код "0100" (рис. 4.20, *б*).

При *D* = *D2* = 1 третий сигнал сдвига *CLK* установит *X*3 в состояние 1, а его информационное состояние *X1*= 0 будет передано триггеру *X2*; с *X2* информационное состояние *D0* = 1 в *X1*; *X0* не изменит своего состояния, т.е. в регистре установится двоичный код "1010".

При *D* = *D3* = 1 четвертый сигнал сдвига *CLK* не изменит состояния триггера *X3*, содержимое *X3* передается триггеру *X2*, и т.д., т.е. в регистре установится код двоичного слова "1101".

Последовательный двоичный код *D3D2D1D0* = 1101 из регистра может быть выдан сигналами сдвига *CLK* в виде высокого (логической 1) и низкого (логического 0) уровней на выходе регистра (рис. 4.19, *б*).

Как видно из временной диаграммы работы регистра, при считывании информация в регистре теряется. Если организовать обратную связь путем замыкания выхода *Q4* с входом *D*, то при считывании организуется цепь связи младшего разряда регистра со старшим и тогда одновременно будет происходить его перезапись. Такие регистры принято называть кольцевыми регистрами.

Двоичный код, хранимый в регистре (рис. 4.19, *а*), может быть снят в виде параллельного кода с выходов *Q* (прямой код) или $\overline{Q}$ (обратный код) всех триггеров регистра.

Сдвигающие регистры с цепями приема двоичной информации в последовательном коде и выдачи – в параллельном коде называют **последовательно-параллельными регистрами, а также преобразователями последовательного двоичного кода в параллельный двоичный код.**

Сдвигающие регистры с цепями приема двоичной информации в параллельном коде и выдачи – в последовательном коде называют **параллельно-последовательными регистрами или преобразователями параллельного двоичного кода в последовательный двоичный код.**

В качестве примера на рис. 4.20 приведена функциональная схема трехразрядного параллельно-последовательного регистра на синхронных двухступенчатых *JK*-триггерах.

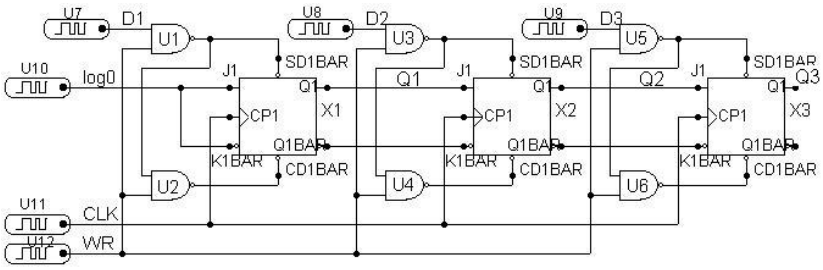


Рис. 4.20 Параллельно-последовательный регистр на *JK*-триггерах

Параллельный двоичный код *D1D2D3* преобразуется в последовательный следующим образом. По сигналу *WR* параллельный двоичный код *D1D2D3* с *U7 – U8* проходит через логические схемы И (схему управления *U1 – U6*) и поступает в парафазном коде на входы сдвигающего регистра (Рг) выполненного на триггерах *X1 – X3*; в Рг записывается двоичный код *D1D2D3*.

Выдача из Рг, хранимого двоичного кода *D1D2D3* в последовательном коде производится сигналами сдвига вправо (*CLK*). При этом последовательный двоичный код может быть снят старшим разрядом вперед, т.е. *D3D2D1* с выхода регистра *Q3* в виде логических уровней.

В цифровых системах при построении сложных логических схем используются также реверсивные сдвигающие регистры. – регистры, двоичная информация в которых может сдвигаться как вправо, так и влево по регистру.

На рис. 4.21 приведена функциональная схема трехразрядного реверсивного сдвигающего регистра на *D*-триггерах с динамическим синхронизирующим входом *CLK*.

Сдвиг двоичного кода в регистре вправо или влево задается разрешающим уровнем логической 1 или на входе *ER* (сдвиг вправо), или на входе *EL* (сдвиг влево) и производится под воздействием сигнала сдвига (*CLK*), подаваемого на синхронизирующий вход всех *D*-триггеров регистра. Под воздействием каждого *CLK* происходит сдвиг на один разряд вправо или влево.

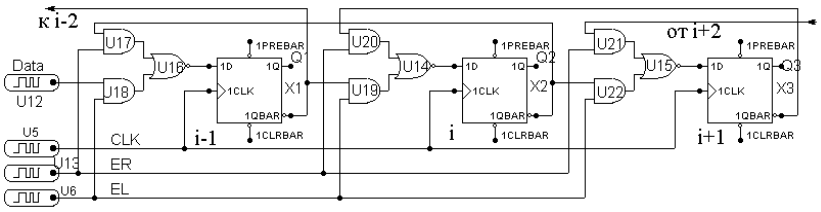
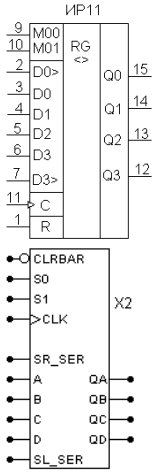


Рис. 4.21 Реверсивный сдвигающий регистр на *D*-триггерах



Режимы работы ИР11

Выводы		Режимы
M0	M01	
0	0	Хранение
0	1	Сдвиг влево
1	0	Сдвиг вправо
1	1	Параллельная запись

Рис. 4.22 Условное графическое обозначение и режимы работы универсального регистра К555ИР11 и его зарубежного аналога 74*LS*194

В интегральной технике получили распространение регистры универсального типа с логикой, обеспечивающей возможность организации регистра любого типа. На рис. 4.22 приведено условное графическое обозначение универсального регистра К555ИР11 и его зарубежного аналога 74*LS*194 с обозначениями входов и выходов:
R (*CLRBAR*) – вход для установки регистра в нулевое состояние;
D0 > (*SR_SER*) – вход для последовательного ввода данных в регистр при сдвиге вправо;
D3 > (*SL_SER*) – вход для последовательного ввода данных в регистр при сдвиге влево;
D1, *D2*, *D3*, *D4* (*A*, *B*, *C*, *D*) – входы для параллельного ввода данных в регистр;
M00, *M01* (*S0*, *S1*) – входы, определяющие режимы работы регистра;
C (*CLK*) – вход для сигналов сдвига.
Q0, *Q1*, *Q2*, *Q3* (*QA*, *QB*, *QC*, *QD*) – выходные разряды регистра.

4.3 СЧЕТЧИКИ

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

В качестве входных сигналов понимаются как перепады уровня напряжения или тока, так и импульсы.

Счетчики, как и сдвиговые регистры, 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.

4.3.1 Основные параметры и классификация счетчиков

Основным статическим параметрами счетчика являются:

- **модуль счета М**, или коэффициент пересчета *K*, который характеризует максимальное число импульсов, после прихода которых счетчик устанавливается в исходное состояние.
- **информационная емкость** – максимальное число сигналов, которое может быть подсчитано счетчиком. Максимальное число *N*, которое может быть записано в счетчике равно $(2^n - 1)$, где *n* – число разрядов счетчика. Каждый разряд счетчика включает в себя триггер.

Основными динамическими параметрами, определяющими быстродействие счетчика, являются:

- **время установления выходного кода** – t_k, характеризующее временной интервал между моментом подачи входного сигнала и моментом установления нового кода на выходе;
- **разрешающая способность** – это минимальное время между двумя последовательно поступающими сигналами, которые надежно фиксируются счетчиком;
- **максимальное быстродействие счетчика** – величина, обратная разрешающей способности и равная числу сигналов, фиксируемых счетчиком в единицу времени.

Счетчики различаются назначением, типом и количеством используемых триггеров, режимами работы, порядком изменения состояния, организацией связей между триггерами счетчика и другими особенностями его структуры.

Счетчики могут быть одноразрядные, многоразрядные, двоичные, десятичные, а также с любым иным целым по значению коэффициентом пересчета *K*.

По порядку изменения состояния счетчики делятся на счетчики с естественным и на счетчики с произвольным (принудительным) порядком изменения состояния.

В **счетчиках с естественным порядком изменения состояния** значение кода каждого последующего состояния счетчика отличается на единицу от кода предыдущего состояния. В **счетчиках с произвольным порядком изменения состояния** значения кодов соседних состояний могут отличаться более чем на единицу.

Счетчики с естественным порядком изменения состояний подразделяются на простые (суммирующие и вычитающие) и реверсивные, которые в зависимости от управляющих сигналов могут работать как в режиме сложения, так и в режиме вычитания.

По способу организации счета счетчики делятся на **асинхронные** и **синхронные**. В **асинхронных счетчиках** переключение триггеров происходит последовательно во времени, в **синхронных счетчиках** – параллельно (одновременно) во времени.

По значению модуля счета счетчики подразделяют на:

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

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 микрооперацию инкремента или декремента

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 ис-

пользуются различные комбинации способов переноса.

Одноразрядные двоичные счетчики строятся на основе *T*-триггеров, осуществляющих сложение по модулю 2, т.е. счет и хранение не более двух сигналов в соответствии с характеристическим уравнением

$$Q_{t+1} = T_t \wedge \overline{Q_t} \vee \overline{T_t} \wedge Q_t . \qquad (4.11)$$

В общем случае *n*-разрядный двоичный счетчик осуществляет сложение по модулю 2.

4.3.2 Суммирующие двоичные счетчики

Суммирующий счетчик работает по принципу суммирования сигналов, поступающих на его вход. В суммирующих двоичных счетчиках счетный вход каждого последующего триггера соединен с выходом предыдущего таким образом, что при переходе триггера младшего разряда из состояния 1 в состояние 0 в цепи переноса между триггерами появляется сигнал переноса, под воздействием которого триггер следующего разряда изменяет свое состояние на противоположное. В зависимости от способа организации цепей переноса различают двоичные счетчики с последовательным, сквозным, параллельным и групповым переносами.

4.3.3 Двоичные счетчики с последовательным переносом

Двоичные счетчики с последовательным переносом строятся на основе асинхронных *T*-триггеров.

Рассмотрим синтез и работу трехразрядного суммирующего двоичного счетчика с естественным порядком изменения состояний, закон функционирования которого задан таблицей переходов (табл. 4.10).

4.10 Таблица переходов трехразрядного двоичного счетчика

Номер входного сигнала T_0	T_0	Q_2	Q_1	Q_0	Номер входного сигнала T_0	T_0	Q_2	Q_1	Q_0
1	1	0	0	0	5	1	1	0	0
	0	0	0	1		0	1	0	1
2	1	0	0	1	6	1	1	0	1
	0	0	1	0		0	1	1	0
3	1	0	1	0	7	1	1	1	0
	0	0	1	1		0	1	1	1
4	1	0	1	1	8	1	1	1	1
	0	1	0	0		0	0	0	0

Из табл. 4.10 следует, что изменение младшего разряда Q_0 связано с изменением единичного значения сигнала счета T_0 на нулевое, а изменение состояния каждого последующего разряда Q_l связано с изменением единичного состояния на нулевое предыдущего Q_{i-1} разряда. Таким образом в счетчике сигналы переноса распространяются последовательно от младшего разряда к старшему.

Так как каждый триггер счетчика осуществляет сложение по модулю 2, то закон функционирования трехразрядного суммирующего двоичного счетчика может быть представлен характеристическими уравнениями (4.12).

$$\left.\begin{aligned} Q_{0(t+1)} &= \overline{Q_{0t}} T_{0t} \vee Q_{0t} \overline{T_{0t}}; \\ Q_{1(t+1)} &= \overline{Q_{1t}} T_{1t} \vee Q_{1t} \overline{T_{1t}}; \\ Q_{2(t+1)} &= \overline{Q_{2t}} T_{2t} \vee Q_{2t} \overline{T_{2t}}. \end{aligned}\right\} \qquad (4.12)$$

На рис. 4.24, *а* приведена принципиальная схема асинхронного суммирующего двоичного счетчика на *T*-триггерах, которые получены из универсальных *JK*-триггеров. Генератор *Log*1 является генератором высокого логического уровня.

Перед началом счета сигналом *CLR* счетчик устанавливается в состояние "000". Из временной диа-

граммы работы счетчика (рис. 4.24, *б*) видно, что после прихода 7-го входного сигнала на вход *CLK* показание счетчика будет "111". При поступлении 8-го входного сигнала *CLK* счетчик переходит в исходное состояние "000". При этом на выходе счетчика *Q2* в результате перехода триггера *X3* в состояние 0 возникает сигнал переноса, который называют **сигналом переполнения счетчика**.

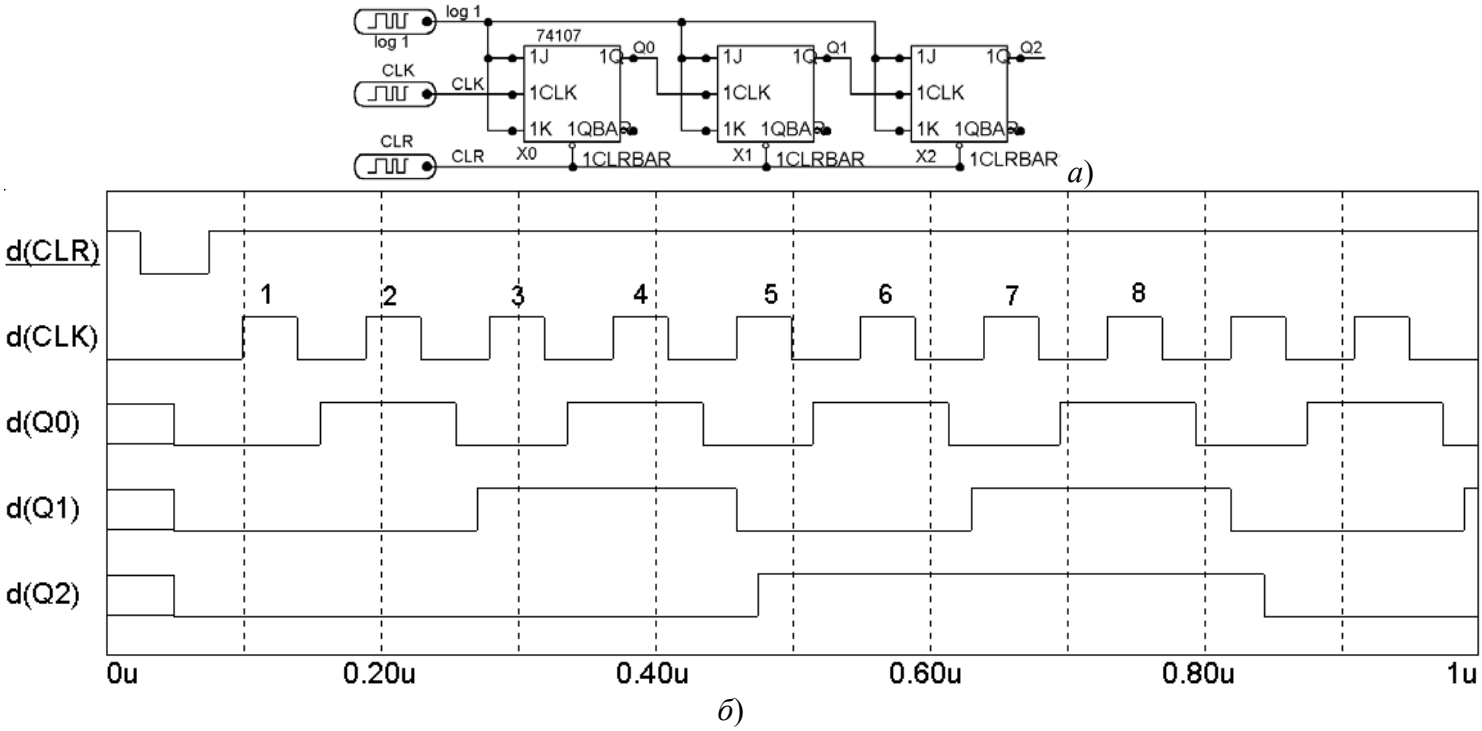


Рис. 4.24 Асинхронный суммирующий двоичный счетчик с цепями последовательного переноса на *JK*-триггерах:

а – функциональная схема; *б* – временная диаграмма работы

Быстродействие двоичного счетчика с последовательным переносом зависит то быстродействия триггера младшего разряда, так как каждый последующий триггер уменьшает частоту следования сигналов, поступающих на его вход, и определяется временем $T_{сч}$ распространения сигналов переноса.

Максимальное время $T_{сч.\ макс}$ установления кода в счетчике с последовательным переносом равно:

$$T_{сч.\ макс} = nt_T, \qquad (4.13)$$

где n – число разрядов счетчика; t_T – время задержки сигнала в одном разряде счетчика, т.е. триггере.

Как видно из выражения (4.13), с увеличением разрядности счетчика удлиняется время задержки сигнала в счетчике, а следовательно, понижается предельная частота его работы. Счетчики с последовательным переносом при их построении требуют минимальное количество элементов и межэлементных связей.

Асинхронный суммирующий двоичный счетчик может быть реализован на *D*- и *JK*-триггерах, работающих в режиме счетного *T*-триггера. В этом случае он описывается характеристическим уравнением (4.12).

Следует иметь в виду, что, счетчики на *JK*-триггерах по сравнению со счетчиками на *D*-триггерах имеют большие аппаратурные затраты и меньшее быстродействие.

4.3.4 Двоичные счетчики со сквозным переносом

Для ускорения работы счетчика необходимо, чтобы изменение состояний отдельных разрядов в счетчике происходило не последовательно, а непосредственно вслед за приходом очередного счетного сигнала.

Анализируя табл. 4.10 можно заметить, что переключение каждого триггера возможно, если на его счетном входе *T* будет 1. Переключение каждого последующего триггера произойдет только в том случае, если все предшествующие триггеры младших разрядов находятся в состоянии 1. Так, если *n*-разрядный счетчик находится в состоянии 00 . . . . . 1111, то следующее его состояние 00 10000,

причем старшие разряды счетчика не изменяют своего состояния. Отсюда следует, что *i*-й разряд *n*-разрядного счетчика может переключаться в следующее состояние, если

$$Q_{i\text{-}1} \wedge Q_{i\text{-}2} \wedge \ \dots \ \wedge \ Q_1 \wedge \ Q_0 = 1, \qquad (4.14)$$

где $i = 0, 1, 2, ..., (n–1)$.

Если условие (4.14) не выполняется, то разряд сохраняет предыдущее состояние. На основе указанного правила формирования переносов, переносы из младших разрядов счетчика в старшие разряды могут быть организованы по следующим логическим выражениям:

$$\left.\begin{aligned} T_1 &= Q_0 \wedge T_0 \\ T_2 &= Q_1 \wedge T_1 \\ &\dots\dots \\ T_i &= Q_{i-1} \wedge T_{i-1} \end{aligned}\right\} \qquad (4.15)$$

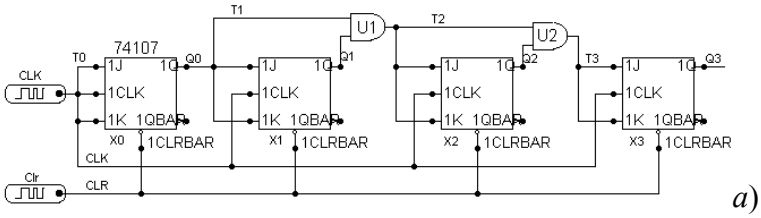
или

$$T_i = Q_{i-1} \wedge Q_{i-2} \wedge \cdots \wedge Q_1 . \qquad (4.16)$$

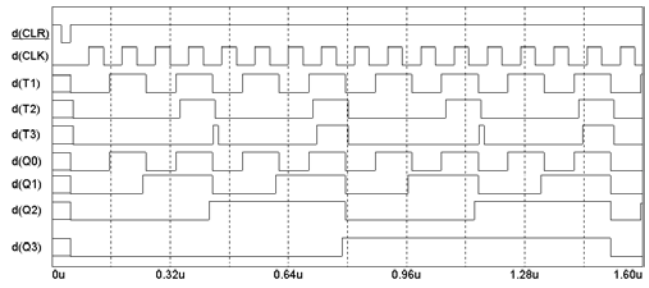
Счетчик, разряды которого построены в соответствии с уравнениями (4.15), носит название **счетчика со сквозным переносом**.

Для построения двоичных счетчиков со сквозным переносом используют синхронные *T*-триггеры, выполненные из универсальных *JK*-триггеров.

На рис. 4.25, *а* представлена принципиальная схема синхронного четырехразрядного суммирующего двоичного счетчика со сквозным переносом. Цепь сквозного переноса сформирована на элементах *U1* и *U2*. Входной счетный сигнал *CLK* подается одновременно на синхронизирующие входы *CLK* триггеров *X0*, *X1*, *X2*, *X3*, счетчика, а на другие счетные входы – сигнал T_i определяемый выражением (4.15). Изменение старших разрядов счетчика имеет место тогда и только тогда, когда предшествующие триггеры младших разрядов находятся в состоянии 1. Формирование сигнала T_I в цепи сквозного переноса обычно заканчивается к моменту прихода счетного сигнала *CLK*, поэтому практически все триггеры переключаются



а)



б)

Рис. 4.25 Схема счетчика со сквозным переносом (*а*); временная диаграмма (*б*)

одновременно. Так, например, после седьмого счетного сигнала *CLK* на счетных входах триггеров *X1*, и *X2* установится уровень логической 1. Поэтому восьмой счетный сигнал *CLK* вызовет переключение триггеров *X0*, *X1*, и *X2* из состояния 1 в состояние 0, а триггера *X3* – из состояния 0 в состояние 1. На рис. 4.25, *б* видно, что появление на выходе вентиля *U2* ложного сигнала *T3*, вызванного гоночными явлениями, не приводит к ложному переключению триггера *X3*, так как сигнал синхронизации *CLK* в это время не активен.

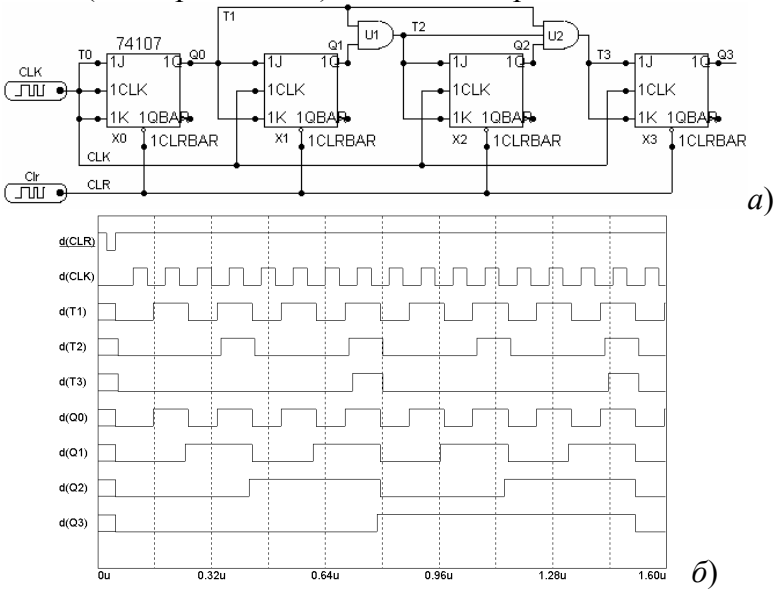
Максимальное время $T_{\text{сч. макс}}$ установления кода в двоичном счетчике со сквозным переносом определяется временем распространения сигналов переноса и равно;

$$T_{\text{сч.макс}} = t_{\text{Т}} + (n-2)t_{\text{И}},$$

где $t_И$ – время задержки сигнала на элементе И.

4.3.5 Двоичные счетчики с параллельным (одновременным) переносом

Как видно из рис. 4.25, *б* цепь последовательного сквозного переноса вызывает задержку в асинхронном установлении уровней сигналов T_i. Этот недостаток можно устранить, если в счетчике сформировать цепь параллельного (одновременного) сквозного переноса.



а)

б)

Рис. 4.26 Суммирующий двоичный счетчик с параллельным переносом:
а – принципиальная схема; *б* – временная диаграмма

Такой счетчик получил название **двоичного счетчика с параллельным переносом** (рис. 4.26, *а*) и строится в соответствии с уравнением (4.16). Для его построения используется синхронный *Т*-триггер. Цепь параллельного переноса построена на элементах И. Очевидно, что по мере увеличения разрядности счетчика требуются схемы И с количеством входов равным n+1, где n – порядковый номер вентиля в счетчике.

Формирование сигналов T_i в цепи параллельного переноса происходит одновременно и заканчивается к приходу следующего очередного счетного сигнала *CLK*. Поэтому все триггеры в счетчике как с параллельным, так и со сквозным переносом переключаются практически одновременно. На временной диаграмме (Рис. 4.26, *б*) видно, что каждый последующий триггер переключается несколько раньше, чем предыдущие.

Поэтому введение цепи параллельного переноса позволяет сократить время распространения сигналов переноса.

Максимальное время $T_{\text{сч.макс}}$ установления кода в таком счетчике равно: $T_{\text{сч.макс}} = t_Т + t_И$.

4.3.6 Синтез суммирующего трехразрядного двоичного счетчика на универсальных *JK*-триггерах

Двоичные счетчики с параллельным переносом, построенные на универсальных *JK*-триггерах со встроенными логическими элементами И, по структурной организации получаются очень простыми. Чтобы показать это, рассмотрим синтез суммирующего трехразрядного двоичного счетчика на *JK*-триггерах с цепью параллельного переноса, функционирующего по закону заданному табл. 4.11.

4.11 Таблица переходов трехразрядного двоичного счетчика

Номер входного сигнала T_0	Q_{2t}	Q_{1t}	Q_{0t}	$Q_{2(t+1)}$	$Q_{1(t+1)}$	$Q_{0(t+l)}$
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	1	0	1
6	1	0	1	1	1	0
7	1	1	0	1	1	1
8	1	1	1	0	0	0

Для перевода *JK*-триггера в требуемые состояния необходимо на его входах *J* и *K* иметь определенные логические уровни (табл. 4.12).

4.12 Соответствие уровня на входе виду перехода

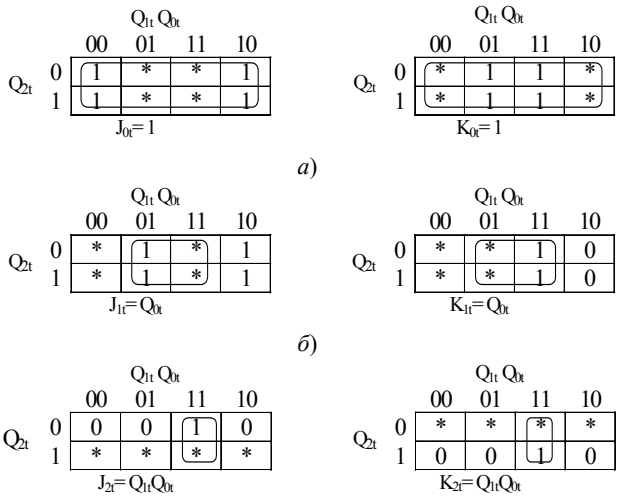
Вид перехода *JK*-триггера	Логические уровни на входах	
	J	*K*
0 → 0	0	*
0 → 1	1	*
1 → 0	*	1
1 → 1	*	0

данного перехода логический извольным. Используя табл. 4.12 и таблицу переходов (табл. 4.11), строятся карты Карно для *J* и *K*, каждого триггера (рис. 4.27, *а*, *б*, *в*).

В результате склеивания карт Карно получим:

минтермов и минимизации с помощью

$$\left.\begin{aligned} J_{0t} &= K_{0t} = 1; \\ J_{it} &= K_{it} = Q_{0t}; \\ J_{2t} &= K_{2t} = Q_{1t}Q_{0t}. \end{aligned}\right\} \qquad (4.17)$$



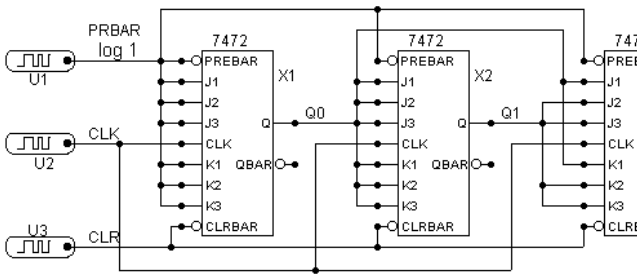
а)

б)

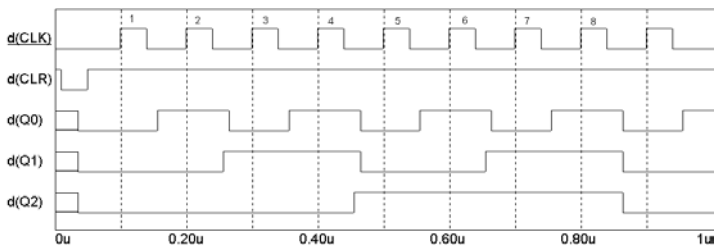
в)

Рис. 4.27 Минимизирующие карты Карно для суммирующего двоичного счетчика с параллельным переносом на *JK*-триггерах:
а – для J_{0t} и K_{0t}; *б* – для J_{1t} и K_{1t}; *в* – для J_{2t} и K_{2t}

Выражения (4.17) определяют объединение *JK*-триггеров в суммирующем трехразрядном двоичном счетчике с параллельным переносом (рис. 4.28). Для *X1* не требуются управляющие сигналы, и поэтому на его входы J_0 и K_0 подается уровень логической 1. Входы J_i и K_i триггера *X*2 соединяются с прямым *Q* выходом *X*1, а на входы J_2 и K_2 триггера *X*3 подается сигнал, являющийся конъюнкцией Q_0 и Q_1 выходов *X*1 и *X*2, соответственно.



а)



б)

Рис. 4.28 Суммирующий двоичный счетчик с параллельным переносом на *JK*-триггерах (*а*), временная диаграмма (*б*)

4.3.7 Принцип формирования группового переноса

В тех случаях, когда число разрядов счетчика превышает число входов логических элементов И данной серии, счетчик разбивают на группы. Внутри каждой группы счетчика организуют параллельный перенос, а перенос между группами может быть последовательный, сквозной и параллельный.

Рассмотрим синтез построения цепей группового переноса со сквозным переносом между группами на примере 12-разрядного счетчика. Счетчик разделяется на три группы, каждая из которых представляет собой двоичный четырехразрядный счетчик СТ2 с параллельным переносом. Закон формирования сигналов T_i между группами в соответствии с выражением (4.16) может быть представлен в общем виде:

$$T_j = T_{j-1} Q_k Q_{k+1} \dots Q_{k+m-1}, \tag{4.18}$$

где j – номер группы; k – номер младшего разряда в группе; m – число разрядов в группе.

Для 12-разрядного счетчика (рис. 4.29) j = 0, 1, 2, 3; m = 4; k = 0, 4, 8. Логические выражения (4.8) для сигналов T_j имеют вид:

$$T_1 = T_0 Q_0 Q_1 Q_2 Q_3;$$
$$T_2 = T_1 Q_4 Q_5 Q_6 Q_7;$$
$$T_3 = T_2 Q_8 Q_9 Q_{10} Q_{11}.$$

На рис. 4.29 приведена функциональная схема счетчика со сквозным переносом между группами.

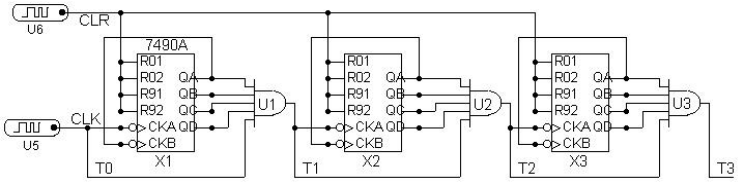


Рис. 4.29 Суммирующий двоичный счетчик со сквозным переносом между группами

Двоичные счетчики с групповым переносом, у которых перенос внутри каждой группы и между группами организуется параллельно, называют **счетчиками параллельного группового переноса**.

Закон формирования сигналов T_j между группами такого счетчика может быть представлен в общем виде логическим выражением:

$$T_j = T_0 Q_0 Q_1 \dots Q_{jm-1}. \tag{4.19}$$

Для 12-разрядного двоичного счетчика (рис. 4.30) j = 0, 1, 2, 3; m = 4. Логические выражения (4.19) для сигналов T_J с параллельным переносом между группами будут иметь вид:

$$\begin{aligned} T_1 &= T_0 Q_0 Q_1 Q_2 Q_3; \\ T_2 &= T_0 Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7; \\ T_3 &= T_0 Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 (Q_8 Q_9 Q_{10} Q_{11}). \end{aligned} \tag{4.20}$$

Как видно из рис. 4.30, увеличение разрядности двоичного счетчика с параллельным переносом требует увеличения либо числа групп, либо числа разрядов в группе. И то, и другое требуют схем И с большим числом входов или несколько последовательно включенных вентилей И.

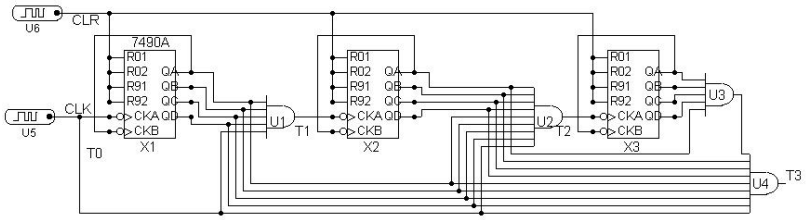


Рис. 4.30 Суммирующий двоичный счетчик с параллельным переносом между группами

Максимальное время $T_{\text{сч.макс}}$ установления кода в двоичном счетчике с групповым переносом равно

$$T_{\text{сч.макс}} = t_{\text{Т}} + \frac{n}{m} t_{\text{И}} .$$

4.3.8 Вычитающие двоичные счетчики

Счетчики, реализующие микрооперацию вычитания, строятся аналогично суммирующим двоичным счетчикам с последовательным, сквозным, параллельным и групповым переносом.

Рассмотрим синтез и работу трехразрядного вычитающего двоичного счетчика с последовательным переносом и естественным порядком изменения состояний, закон функционирования которого задан таблицей переходов (табл. 4.13).

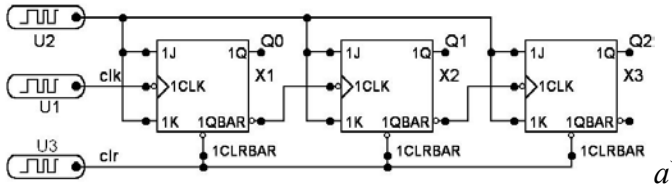
Из табл. 4.13 следует, что изменение младшего разряда связано с изменением единичного значения сигнала счета T_0 на нулевое, а изменение состояния каждого последующего разряда Q_i связано с изменением нулевого состояния на единичное предыдущего Q_{i-1} разряда.

4.13 Закон функционирования двоичного вычитающего счетчика

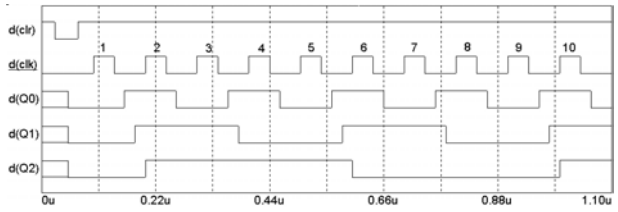
Номер входного сигнала T_0	T_0	Q_2	Q_1	Q_0	Номер входного сигнала T_0	T_0	Q_2	Q_1	Q_0
1	1	1	1	1	5	1	0	1	1
	0	1	1	0		0	0	1	0
2	1	1	1	0	6	1	0	1	0
	0	1	0	1		0	0	0	1
3	1	1	0	1	7	1	0	0	1
	0	1	0	0		0	0	0	0
4	1	1	0	0	8	1	0	0	0
	0	0	1	1		0	1	1	1

Каждый триггер вычитающего двоичного счетчика также как и суммирующего, осуществляет сложение по модулю 2. А поэтому закон функционирования трехразрядного вычитающего двоичного счетчика может быть также представлен характеристическими уравнениями (4.12).

На рис. 4.31, *а* приведена функциональная схема асинхронного вычитающего двоичного счетчика. Счетчик построен на асинхронных *T*-триггерах. Перед началом счета счетчик во все разряды сигналом *PRBAR* записывает двоичный код "111" (табл. 4.13). После прихода 7-го входного сигнала *CLK* показание счетчика будет "000", а 8-й входной сигнал *CLK* установит счетчик в исходное состояние "111" (рис. 4.31, *б*).



а)



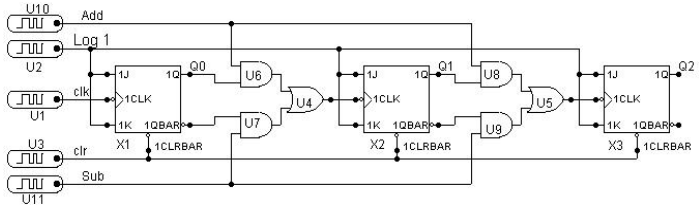
б)

Рис. 4.31 Вычитающий двоичный счетчик с цепями последовательного переноса:
а – функциональная схема; *б* – временная диаграмма работы

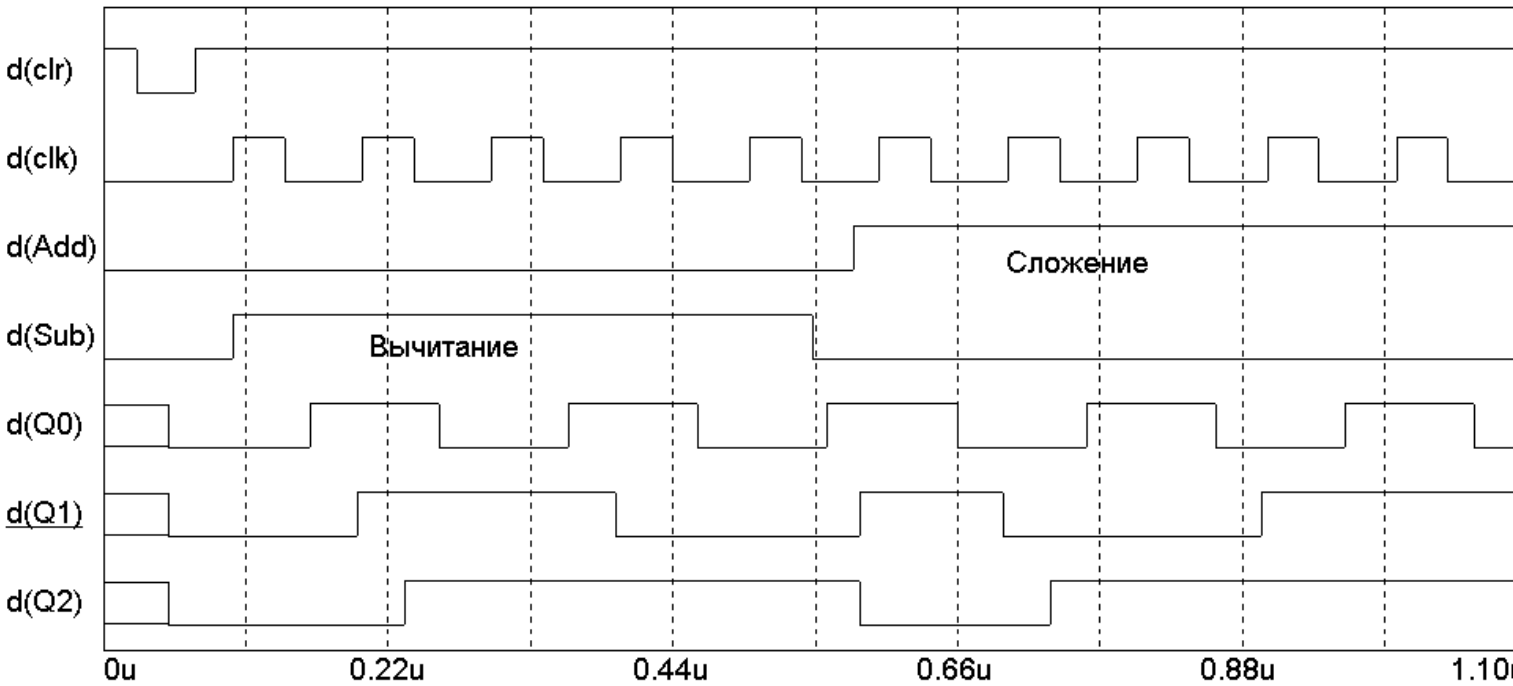
Также как и суммирующие, вычитающие двоичные счетчики строятся на *D*-, *JK*-триггерах, работающих в режиме *T*.

4.3.9 Реверсивные двоичные счетчики

Вычитающие двоичные счетчики отдельно используются редко. В реверсивных счетчиках в операции вычитания и сложения участвуют одни и те же триггеры. На рис. 4.32, *а* приведена функциональная схема трехразрядного реверсивного двоичного счетчика на *JK*-триггерах с последовательным переносом. Временная диаграмма работы реверсивного счетчика показана на рис. 4.32, *б*. Межразрядные связи выполнены на элементах И и ИЛИ. Реверсирование достигается тем, что в цепях межразрядных связей производятся передачи сигналов переноса либо с прямых выходов Q триггеров, либо с инверсных $\overline{Q}$ выходов. В связи с этим реверсивный счетчик помимо информационного входа T_0 имеет управляющие режимом работы счетчика входы: *Log*1, разрешающий счет при *Log*1 = 1 (*JK*-триггер в режиме *D*-триггера); *Add*, разрешающий операцию "Сложение", и *Sub*, разрешающий операцию "Вычитание".



а)



б)

Рис. 4.32 Реверсивный двоичный счетчик

Двоичные счетчики любого типа имеют коэффициент пересчета $K = \frac{f_{вх}}{f_{вых}} = 2^n$, где $f_{вх}$ – частота входных сигналов, $f_{вых}$ – частота выходных сигналов.

4.3.10 Синтез пересчетных схем с параллельным переносом на универсальных *JK*-триггерах

Пересчетные схемы с параллельным переносом удобно строить на *JK*-триггерах со встроенными логическими элементами.

В качестве примера произведем синтез пересчетной схемы с коэффициентом пересчета $K = 10$ на универсальных *JK*-триггерах. Из соотношения $2^{n-1} < K < 2^n$ получим необходимую разрядность пересчетной схемы $n = 4$, а число запрещенных состояний, возникающих в схеме, определим как: $M = 2^n - K = 16 - 10 = 6$. Пусть требуется построить пересчетную схему с параллельным переносом и естественным порядком изменения состояний, закон функционирования которой задан табл. 4.14.

4.14 Закон функционирования пересчетной схемы с $K = 10$

Номер входного сигнала T_0	Q_{3t}	Q_{2t}	Q_{1t}	Q_{0t}	$Q_{3(t+1)}$	$Q_{2(t+1)}$	$Q_{1(t+1)}$	$Q_{0(t+1)}$
1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	0	1	1	0	1	0	0
5	0	1	0	0	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	1	0	0	1	1	1
8	0	1	1	1	1	0	0	0
9	1	0	0	0	1	0	0	1
10	1	0	0	1	0	0	0	0

Из табл. 4.12 выберем значения для *J* и *K*, при которых триггер переключается из одного состояния в другое, и по табл. 4.14 составим карты Карно для входов *J* и *K*, каждого триггера пересчетной схемы. Например, при $T_0 = 1$, Q_{0t} переключается в $Q_{0(t+1)}$, т.е. из 0 в 1. По табл. 4.12 определяем логический уровень на входе $J_0 = 1$, а на входе $K_0 = *$, соответствующие этому виду перехода. В картах Карно (рис. 4.33) знак "×" означает запрещенные для десятичного счетчика состояния, у которого входы *J* и *K* всех триггеров могут принимать произвольное значение (либо 0, либо 1); знак "*" означает произвольные значения *J* и *K*.

После склеивания и минимизации с помощью карт Карно получаем ДНФ для входов *J* и *K* каждого триггера пересчетной схемы с параллельным переносом для $K = 10$, т.е.

$$J_{0t} = 1;\ J_{1t} = Q_{0t}\overline{Q_{3t}};\ J_{2t} = Q_{0t}Q_{1t};\ J_{3t} = Q_{0t}Q_{1t}Q_{2t};$$
$$K_{0t} = 1;\ K_{1t} = Q_{0t}Q_{1t};\ K_{2t} = Q_{0t}Q_{1t};\ K_{3t} = Q_{0t} \qquad (4.21)$$

На рис. 4.34, *а* представлена принципиальная схема двоичного счетчика с $K = 10$, реализующая переключательные функции (4.21), а на рис. 4.34, *б* временная диаграмма работы пересчетной схемы.

Рассмотрим работу схемы. Предположим, что после поступления семи входных сигналов T_0 на схему показание счетчика достигло "0111" (табл. 4.14). Тогда на входах *K* триггеров *X2*, *X3*, и *X4* устано-

вится уровень логической 1. Тогда, восьмой входной сигнал T_0 вызовет переключение всех триггеров счетчика, т.е. в нем будет записан код "1000". А девятый входной сигнал *CLK* вызовет переключение только триггера *X1*, так как триггеры *X2*, *X3* и *X4* будут заблокированы по входу *J* уровнем логическо-

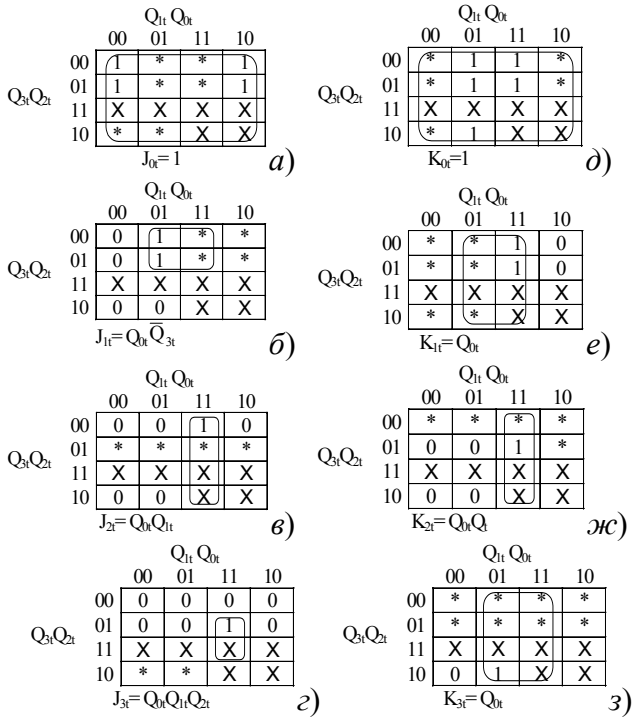


Рис. 4.33 Минимизирующие карты Карно для пересчетной схемы (*K* = 10) на *JK*-триггерах

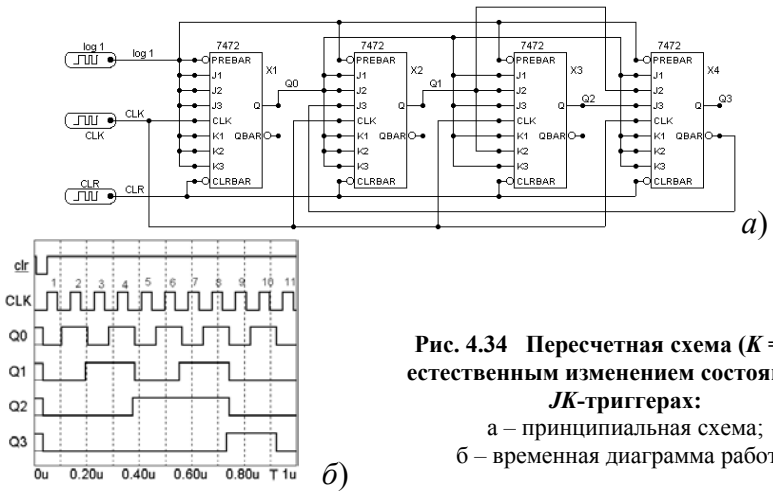


Рис. 4.34 Пересчетная схема (*K* = 10) с естественным изменением состояний на *JK*-триггерах:
а – принципиальная схема;
б – временная диаграмма работы

го 0 с выходов триггеров *X2*, *X3* и *X4*, соответственно. Показание счетчика будет "1001". 10-й входной сигнал *CLK* вызовет переключение только триггеров *X1* и *X4*, а *X2* и *X3* будут заблокированы по входу *J* уровнем логического 0 с выходов соответствующих триггеров.

Счетчик зафиксирует двоичный код "0000", т.е. установится в исходное состояние.

Пересчетные схемы с *K* = 10 называют **декадными счетчиками**.

Используя рассмотренный выше способ, можно синтезировать пересчетную схему с любым целым значением коэффициента пересчета *K*.

4.3.11 Общий метод введения обратных связей

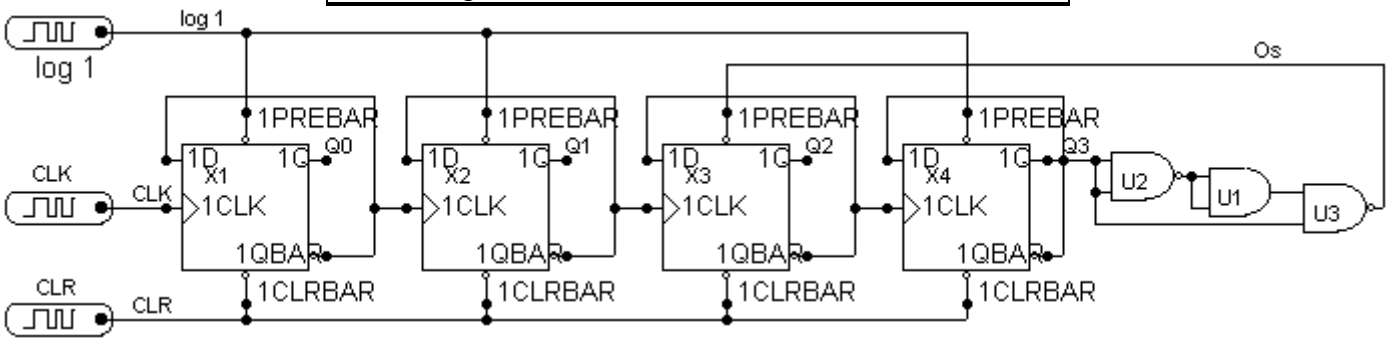
Уменьшение числа устойчивых состояний в счетчике прямого счета путем введения обратных связей обеспечивает поступление дополнительных сигналов с какого-либо старшего разряда в младшие, обеспечивая при этом изменение естественной последовательности двоичных чисел при подсчете входных сигналов.

На триггерах, работающих в счетном режиме *T* и имеющих дополнительные входы *R* и *S* для синтеза пересчетных схем с последовательным переносом, используют так называемый **общий метод введения обратных связей**.

В качестве примера рассмотрим построение пересчетной схемы с обратными связями для $K = 12$ на синхронных *D*-триггерах с динамическим управлением, работающих в режиме *T* (рис. 4.35, *а*). В соответствии с: $2^{n-1} \le K \le 2^n$, разрядность пересчетной схемы n = 4; число запрещенных состояний $M = 2^n - K = 16 - 12 = 4_{10} = 0100_2$. Закон функционирования такой пересчетной схемы задан таблицей переходов (табл. 4.15), а ее работа иллюстрируется временной диаграммой, приведенной на рис. 4.35, *б*.

4.15 Закон функционирования пересчетной схемы с $K = 12$

Номер входного сигнала T_0	Q_3	Q_2	Q_1	Q_0	Номер входного сигнала T_0	Q_3	Q_2.	Q_1	Q_0
1	0	0	0	1	8	1	0	0	0
2	0	0	1	0	ОС*	1	1	0	0
3	0	0	1	1	9	1	1	0	1
4	0	1	6	0	10	1	1	1	0
5	0	1	0	1	11	1	1	1	1
6	0	1	1	0	12	0	0	0	0
7	0	1	1	1					
* ОС – обратная связь									



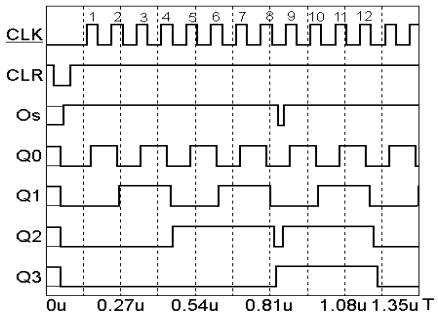


Рис. 4.35 Пересчетная схема ($K = 12$) с обратными связями и временная диаграмма

При построении схем счетчика с обратными связями в *MC5*, следует учитывать тот факт, что в цепи

обратной связи должен возникать импульс малой длительности, достаточный для установки триггера в единичное состояние. Это достигается путем введения дополнительной последовательности вентилей, которые обеспечивают формирование короткого, импульса *Os*, отрицательной полярности, в цепи обратной связи.

Период следования входных сигналов T_0 не может быть меньше $2t_T$, где t_T – время задержки сигнала триггером. В противном случае 8-й входной сигнал T_0 будет воздействовать на триггер T_2 до прихода сигнала на его вход *S* по цепи обратной связи, что может привести к ошибке в подсчете входных сигналов. Частота выходных сигналов равна $f = f_{вх}/K$.

Многоразрядные десятичные счетчики представляют собой "*n*" декадных счетчиков (*n* – число десятичных разрядов счетчика), соединенных между собой цепями переноса.

На рис. 4.36 представлена функциональная схема *n*-разрядного суммирующего десятичного счетчика с последовательным переносом между декадными счетчиками, построенных из двоичных счетчиков.

Сигнал переноса T_j (j = 1, 2, 3, ..., n–1) формируется с помощью логических элементов И как логическое произведение:

$$T_j = T_{j-1}(Q_0 Q_3)_{j-1}.$$

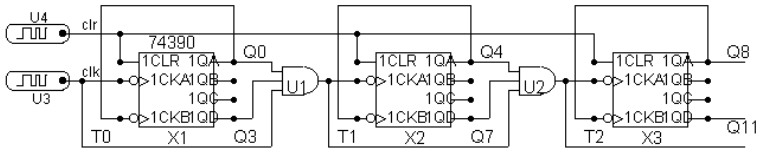


Рис. 4.36 Многоразрядный десятичный счетчик

В библиотеке примитивов *MC* отсутствуют регистры и счетчики. Поэтому для проверки работы последних, необходимо собирать их из отдельных триггеров. Библиотека цифровых элементов представляет широкий выбор моделей микросхем триггеров, регистров, счетчиков различных технологических исполнений.

ЗАКЛЮЧЕНИЕ

Использование персонального компьютера создает приемлемую альтернативу учебной лаборатории – виртуальную лабораторию, которая является по существу программой численного расчета схем с интерфейсом, имитирующим деятельность исследователя в реальной лаборатории. С помощью численных методов расчета при высоком быстродействии и большом объеме памяти современных персональных компьютеров можно исследовать настолько сложные модели, что по точности результаты приближаются к экспериментальным исследованиям на реальных объектах.

В учебном пособии схемотехническому моделированию отводится роль инструмента, с помощью которого изучаются схемотехнические решения цифровых схем. Такой подход позволяет организовать обучение не только непосредственно в аудитории, но и при самостоятельном, дистанционном изучении предмета. При этом нужно осознавать, что основная функция моделирующей программы состоит в получении численных значений тех или иных переменных, определяемых по достаточно жестким алгоритмам. Интеллектуальная оценка схемы и происходящих в ней процессов лежит на разработчике. Он должен или непосредственно устанавливать параметры элементов схемы, или изменять, или определять целевую функцию для параметрического синтеза.

Задачи проектирования цифровых схем можно успешно решать только при глубоком понимании работы цифровых устройств. Необходимым условием является знание основных количественных соотношений, характеризующих электрическую схему и являющихся основой для выбора начальных значений параметров элементов, а также направления и степени изменения этих параметров.

Автор с благодарностью примет все замечания и пожелания, возникшие при прочтении пособия, пришедшие по электронному адресу: kasal@ mail.sapr.tstu.ru.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1 Угрюмов Е.П. Цифровая схемотехника. СПб.: БХВ – Санкт-Петербург, 2000.

2 Схемотехника ЭВМ / Под ред. проф. Соловьева. Л., 1988.

3 Панфилов Д.И., Иванов В.С., Чепурин И.Н. и др. Электротехника и Электроника в экспериментах и упражнениях: Практикум на *Electronics Workbench*. В 2 т. / Под общ. ред. проф. Д.И. Панфилова. М.: Додека, 2000. 600 с.

4 Лачин В.И., Савелов Н.С. Электроника: Учеб. пособие: 3-е изд., перераб. и доп. Ростов н/Д: Изд-во Феникс, 2002. 576 с.

5 Букреев И.Н., Горячев В.И., Мансуров Б.М. Микроэлектронные схемы цифровых устройств. М., Советское радио, 1975.

6 Алексеенко А.Г. Основы микросхемотехники. М.: Советское радио, 1977.

7 Потемкин И.С. Функциональные узлы цифровой автоматики. М.: Энергоатомиздат, 1988.

8 Голсуорт Б. Проектирование цифровых логических устройств / Пер. с англ.; Под ред. Ю.И. Топчеева. М., Машиностроение, 1985.

9 Хоровиц П., Хилл У. Искусство схемотехники: В 2 т. / Пер. с англ. М., Радио и связь, 1983.

10 Разевиг В.Д. Система схемотехнического моделирования Micro-Cap V. М.: "СОЛОН", 1997.

11 Разевиг В.Д. Система схемотехнического моделирования Micro-Cap 6. М.: "Горячая линия-Телеком", 2001.

12 Воробьев Н.В. Сумматоры // Chip News. 2000. № 2. С. 38 – 41.

13 Воробьев Н.В. Сумматоры: определения, классификация, уравнения, структуры и применение // Chip News. 2000. № 3. С. 37 – 40.

14 Системы автоматизированного проектирования в радиоэлектронике: Справочник / Е.В. Авдеев, А.Т. Еремин, И.П. Норенков, М.И. Песков; Под ред. И.П. Норенкова, М., Радио и связь, 1986.

15 Micro-CAP and Micro-LOGIC/ Byte. 1986. Vol. 11. № 6.

16 Micro-CAP. Andrew V. Thompson. Spectrum Software. 1983.

17 Laurenze W. SPICE2: A Computer Program to Simulate Semicondactor Circuits. Memorandum No. ERL-M520.

18 Пухальский Г.И., Новосельцева Т.Я. Цифровые устройства: Учебное пособие для втузов. СПб.: Политехника. 1996.

19 The TTL Data Book. Volume 1. Texas Instruments. 1989.

20 Цифровые интегральные микросхемы: Справочник/ П.П. Мальцев, Н.С. Долидзе, М.И. Критенко и др. М: Радио и связь. 1994.

21 Шило В.Л. Популярные цифровые микросхемы: Справочник. М.: Радио и связь. 1987.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ
……..

1 *MICRO-CAP* – КОМПЬЮТЕРНАЯ СИСТЕМА ПРОЕКТИРОВАНИЯ
……

1.1 Компьютерные модели сигналов и электронных компонент в *Micro-Cap*
…….

1.2 Схемотехническое проектирование и анализ средствами *Micro-Cap* …….
2 АБСТРАКТНЫЙ СИНТЕЗ КОМБИНАЦИОННЫХ СХЕМ ……..
2.1 Этапы синтеза комбинационных схем ……...
2.2 Синтез простейших логических схем …….
2.3 Состязания (гонки) в логических схемах ……...
2.4 Тристабильная схемотехника …….
3 КОМБИНАЦИОННАЯ СХЕМОТЕХНИКА ЦИФРОВЫХ СИСТЕМ …….
3.1 Синтез шифраторов и дешифраторов ……
3.1.1 Многоступенчатые дешифраторы ……...
3.1.2 Пирамидальные дешифраторы ……
3.1.3 Шифратор ……..
3.2 Мультиплексоры и демультиплексоры …….
3.3 Узлы двоичной арифметики ……
3.3.1 Арифметико-логическое устройство …….
4 ПОСЛЕДОВАТЕЛЬНОСТНАЯ СХЕМОТЕХНИКА ……..
4.1 Триггеры ……
4.1.1 Синхронные *RS*-триггеры ……
4.1.2 Двухступенчатый синхронный *RS*-триггер ……
4.1.3 *D*-триггер ……...
4.1.4 *JK*-триггер ……..
4.1.5 *T*-триггер ……
4.2 Регистры

……
4.2.1 Параллельные регистры ……...
4.2.2 Парафазный параллельный регистр ……
4.2.3 Сдвигающие регистры …….
4.3 Счетчики ……..
4.3.1 Основные параметры и классификация счетчиков ...
4.3.2 Суммирующие двоичные счетчики ……
4.3.3 Двоичные счетчики с последовательным переносом
4.3.4 Двоичные счетчики со сквозным переносом …….
4.3.5 Двоичные счетчики с параллельным (одновременным) переносом ……...
4.3.6 Синтез суммирующего трехразрядного двоичного счетчика на универсальных *JK*-триггерах ……
4.3.7 Принцип формирования группового переноса ……..
4.3.8 Вычитающие двоичные счетчики ……
4.3.9 Реверсивные двоичные счетчики ……...
4.3.10 Синтез пересчетных схем с параллельным переносом на универсальных *JK*-триггерах …….
4.3.11 Общий метод введения обратных связей …….
ЗАКЛЮЧЕНИЕ ……
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ ……..