А.И. ПУГАЧЕВ
Б.В. МАРТЕМЬЯНОВ

ТЕОРИЯ ПРОЕКТИРОВАНИЯ ЭВМ

Лабораторный практикум

Самара
Самарский государственный технический университет
2010

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

А.И. ПУГАЧЕВ
Б.В. МАРТЕМЬЯНОВ

ТЕОРИЯ ПРОЕКТИРОВАНИЯ ЭВМ

Лабораторный практикум

Самара
Самарский государственный технический университет
2010

Печатается по решению редакционно-издательского совета Самарского государственного технического университета

УДК 681.3
П 88

Пугачев А.И.

П 88 **Теория проектирования ЭВМ:** Лабораторный практикум / *А.И. Пугачев, Б.В. Мартемьянов.* – Самара: Самар. гос. техн. ун-т, 2010. – 46 с.: ил.

Дан теоретический материал для подготовки и проведения лабораторных работ по курсу. Изложен порядок выполнения лабораторных работ. Приведены типовые контрольные вопросы к отчету по работам.

Методические указания рассчитаны на студентов специальности 230101 «Вычислительные машины, комплексы, системы и сети».

Рецензент д-р техн. наук *П.К. Кузнецов*

УДК 681.3
П 88

ПРЕДИСЛОВИЕ

В технических текстах, особенно в текстах, предназначенных для учебного процесса, авторы часто используют терминологию, достаточно устоявшуюся при общении профессионалов, но носящую, по сути, сленговый характер. Так, при описании состояния сигналов часто можно прочитать авторские обороты вида «в момент начала сигнала», «окончание сигнала», «поступает сигнал», «передний фронт сигнала», «задний фронт сигнала» и т.п. Подобные термины неточны по своей сути, и эти неточности особенно нежелательны в условиях, когда электронные схемы изучаются на компьютерных моделях и обучаемые не имеют возможности «покопаться» в реальных схемах и посмотреть сигнал на экране реального осциллографа.

В чем же неточность подобных формулировок? Дело в том, что если электронная схема подключена к электропитанию и соединена своими входами с другими схемами, то на каждом подключенном входе сигнал присутствует постоянно. Со временем меняется состояние сигнала, его фаза. Закончиться и начаться может определенное состояние (фаза) сигнала, но не сам сигнал.

Применительно к схемам цифровой техники можно выделить следующие состояния сигналов на входах и выходах схем:

– *низкий уровень* напряжения (*арифметический* или *логический нуль*);

– *высокий уровень* напряжения (*арифметическая* или *логическая единица*);

– *положительный фронт* (переходный процесс, при котором состояние сигнала переключается от низкого уровня к высокому уровню);

– *отрицательный фронт* (переходный процесс, при котором состояние сигнала переключается от высокого уровня к низкому уровню).

В некоторых случаях состояния сигнала во время переходных процессов рассматриваются детальнее, и фаза фронта сигнала разлагается на более мелкие составляющие. Такой подход обеспечивает более адекватные результаты моделирования работы цифровых схем с синхронизацией.

Оборот «момент начала сигнала» следует заменить оборотом «момент начала такого-то состояния сигнала», например, «момент переключения сигнала в состояние высокого уровня» или «момент положительного фронта». Часто под термином «момент начала сигнала» подразумевается момент начала той фазы сигнала, которая переключает состояние элемента. Но разные операционные элементы могут переключаться (синхронизироваться) разными фазами сигнала на специальном входе операционного элемента, называемом входом синхронизации. В результате один и тот же термин, например, термин «момент начала сигнала» может указывать на совершенно разные фазы сигнала, что может вызывать неоднозначное толкование информации и приводить к формированию ошибочных «знаний».

Иногда нечеткость формулировок может приводить к принципиально неверному толкованию особенностей функционирования схем. Так, например, из-за необоснованного отождествления понятий «состояние триггера» и «сигнал, снимаемый с прямого плеча триггера», часто формулируют принципиально ошибочное утверждение, что при «запрещенной» комбинации сигналов на входах *RS*-триггера не определен сигнал, снимаемый с плеча триггера. На самом деле, в таком случае не определенной является только цифра, отображаемая состоянием триггера, а выходные сигналы, сформированные на плечах триггера, определены однозначно.

В данном пособии будет использована по возможности наиболее однозначная терминология. Далее по ходу изложения материала будут разъяснены некоторые особенности использованной в пособии терминологии.

1. ЭЛЕМЕНТЫ ТЕОРИИ ЭВМ

1.1. МОДЕЛИРОВАНИЕ РАБОТЫ ТРИГГЕРОВ

Триггер – это электронное устройство для хранения бита информации.

В цифровых схемах значение цифры двоичного кода и значение логической переменной отображается уровнями напряжения. При этом принято считать, что *высокому уровню* напряжения соответ-

ствуют *арифметическая единица* и *логическая единица*, а низкому уровню – *арифметический ноль* и *логический ноль*. В результате абсолютно разные по математическому смыслу константы: арифметическая единица и логическая, арифметический нуль и логический, на уровне представления электрическими сигналами оказываются неразличимыми. Это позволяет разработчику цифровых схем значение любого цифрового сигнала интерпретировать и как значение арифметическое, и как значение логическое. Поэтому далее обозначение конкретного значения сигнала в виде 0 или в виде 1 вовсе не предполагает именно арифметического смысла. Два символа – 0 и 1 – наиболее удобны для различения возможных значений сигнала.

В качестве самостоятельных структурных элементов чаще используются *RS*- и *D*-триггеры. Управление триггерами может быть *асинхронным* и *синхронным*. В соответствии со способом управления триггеры называются асинхронными или синхронными (синхронизируемыми). В асинхронных триггерах управляющие сигналы непосредственно воздействуют на состояние триггера. В синхронных – момент, когда триггер воспринимает действие управляющих сигналов, задается специальным *синхросигналом*, подаваемым на отдельный вход триггера. Такой вход называется входом синхронизации.

Простейший из триггеров – это *асинхронный RS-триггер*. Он является составной частью любого другого триггера. Триггеры разных по логике управления типов отличаются схемами, занимающими промежуточное положение между входами устройства и внутренними (недоступными пользователю) входами встроенного асинхронного *RS*-триггера.

Условное обозначение асинхронного *RS*-триггера, как структурного элемента, приведено на рис. 1.1. Через *R* и *S* обозначены *входные сигналы*, управляющие состояниями триггера.

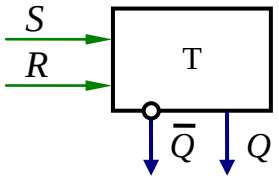

Рис. 1.1

Триггер имеет два выхода, которые часто называются *плечами* триггера. Различают *прямое плечо* и *инверсное плечо*. Сигнал, формируемый триггером на прямом плече,

принято обозначать символом Q. Сигнал на инверсном плече – символом $\overline{Q}$. Обратим внимание на то, что в асинхронном *RS*-триггере в некоторых особых случаях может иметь место равенство $Q = \overline{Q}$. Комбинация входных сигналов, приводящая к указанному равенству, называется *запрещенной комбинацией*.

В режиме хранения бита триггер может находиться в одном из двух устойчивых состояний, сопоставляемых хранимым значениям 0 и 1. Это соответствие принято задавать, как показано в табл. 1.1.

Таблица 1.1

Значение хранимого бита	Значение сигнала Q	Значение сигнала $\overline{Q}$
0	0	1
1	1	0
не определено	0	0
не определено	1	1

Применительно к *RS*-триггеру ошибочно отождествлять хранимое триггером значение со значением сигнала Q. Это связано с существованием запрещенной комбинации входных сигналов. На плечах триггера равные значения сигналов возможны только в течение времени удержания запрещенной комбинации входных сигналов. Такое состояние триггера является *неустойчивым*. В режиме хранения всегда имеет место неравенство: $Q \neq \overline{Q}$.

Отметим, что любой конкретный триггер реагирует на запрещенную комбинацию входных сигналов, формируя одну из указанных в табл. 1.1 пар равных значений Q и $\overline{Q}$. При этом другая пара равных значений не может быть установлена на выходах никакими сочетаниями входных сигналов. Триггер на основе элементов типа И-НЕ формирует пару $(Q, \overline{Q}) = (1, 1)$. Триггер на основе двух элементов типа ИЛИ-НЕ формирует пару $(Q, \overline{Q}) = (0, 0)$.

Запрещенная комбинация опасна тем, что при переключении входных сигналов триггера в состояния, задающие режим хранения, неизвестно какие значения примут сигналы Q и $\overline{Q}$.

Таким образом, хранимое *RS*-триггером значение представляется парой сигналов:

– значение 0 – парой (Q, $\overline{Q}$) = (0, 1);

– значение 1 – парой (Q, $\overline{Q}$) = (1, 0).

Микрооперация записи в триггер любого типа реализуется всегда в два этапа:

– формируется набор входных сигналов (входной набор), устанавливающих соответствующие значения выходных сигналов;

– входные сигналы переключаются в состояния, задающие для триггера режим хранения.

Например, для записи в асинхронный *RS*-триггер значения 1 необходимо сформировать на первом этапе входной набор, устанавливающий пару (Q, $\overline{Q}$) = (1, 0), а для записи значения 0 – сформировать входной набор, устанавливающий пару (Q, $\overline{Q}$) = (0, 1).

Асинхронный *RS*-триггер выполняет следующие микрооперации:

$$\begin{aligned} &\overline{S}R : T := 0 \text{ (запись 0)}; \\ &R\overline{S} : T := 1 \text{ (запись 1)}, \end{aligned} \qquad (1.1)$$

где T – переменная, представляющая *состояние триггера*;

$\overline{S}R$ и $\overline{R}S$ – входные наборы, инициализирующие выполнение соответствующих микроопераций.

Входные наборы в (1.1) описаны в виде конъюнкции входных сигналов. Это следует понимать так, что соответствующая микрооперация выполняется, когда указанная конъюнкция равна 1. Когда в триггер записана 1, говорят, что «*триггер установлен*». Когда в триггер записан 0, говорят, что «*триггер сброшен*». Эта терминология и определила обозначения соответствующих сигналов: S – сокращение от английского *Set* (установка); R – сокращение от *Reset* (сброс).

Заметим, что в виде микрооперации можно описать и режим хранения информации:

$$\overline{S}\,\overline{R} : T_{t+1} := T_t, \qquad (1.2)$$

где T_t и T_{t+1} два последовательных во времени состояния триггера.

Попытка инициировать выполнение одновременно двух команд: *Set* и *Reset* очевидно некорректна, поскольку означает попытку присвоить каждому из сигналов Q и $\overline{Q}$ одновременно два взаимоисключающих состояния.

Универсальной формой описания функционирования *RS*-триггера являются его *характеристические уравнения*:

$$\begin{aligned} Q &:= S \vee \overline{R}\, Q; \\ \overline{Q} &:= R \vee \overline{S}\, \overline{Q}. \end{aligned} \qquad (1.3)$$

Характеристическое уравнение отражает все режимы триггера, в том числе, режим хранения информации. В отличие от введенных ранее микроопераций, каждое характеристическое уравнение определяет не состояние триггера, а состояние его выходного сигнала.

В (1.3) все переменные в правой части означают текущие значения, а в левой части обозначены новые, то есть будущие, значения переменных. Можно было бы отразить этот факт с помощью индексов, поставленных около каждой из переменных и отображающих фактор времени, как это сделано в (1.2). В (1.3) индексы не используются, чтобы не загружать выражения дополнительными символами.

Недостаток использования характеристических уравнений (1.3) состоит лишь в том, что они запрещенной комбинации входных сигналов (S=1 и R=1) сопоставляют конкретные значения выходных сигналов $Q = \overline{Q} = 1$, что не совсем точно, поскольку в зависимости от внутренней реализации триггера в таких случаях возможны два исхода: либо $Q = \overline{Q} = 1$, либо $Q = \overline{Q} = 0$. Причем, как отмечалось выше, конкретный триггер будет точно реализовывать один из этих возможных исходов.

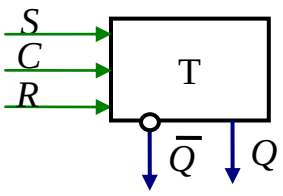

Рис. 1.2

Условное обозначение синхронизируемого (синхронного) *RS*-триггера приведено на рис. 1.2. Сигнал C – это сигнал синхронизации, подключенный к входу синхронизации триггера.

В триггере может использоваться *синхронизация уровнем* (*импульсная синхронизация*) или *фронтом* (*динамическая синхронизация*).

В *RS*-триггерах с импульсной синхронизацией набор сигналов *R* и *S* инициирует выполнение соответствующей микрооперации в течение всего времени, когда сигнал синхронизации имеет значение:

– 1 при *синхронизации высоким уровнем*;

– 0 при *синхронизации низким уровнем*.

Оборот «в течение всего времени» означает, что изменение в это время набора сигналов *R* и *S* может привести к изменению выполняемой микрооперации.

RS-триггер с импульсной синхронизацией выполняет микрооперации, описанные в (1.4) и (1.5).

При синхронизации высоким уровнем:

$$\begin{aligned} &\overline{S}RC : T := 0; \\ &\overline{R}SC : T := 1. \end{aligned} \qquad (1.4)$$

При синхронизации низким уровнем:

$$\begin{aligned} &\overline{S}R\overline{C} : T := 0; \\ &\overline{R}S\overline{C} : T := 1. \end{aligned} \qquad (1.5)$$

В (1.4) и в (1.5), как и в (1.1), условие инициирования выполнения микрооперации задано в виде конъюнкции входных сигналов.

В *RS*-триггерах с синхронизацией по фронту набор сигналов *R* и *S* инициирует выполнение соответствующей микрооперации в момент поступления соответствующего фронта (положительного либо отрицательного) сигнала *C*. Отсутствие соответствующего фронта независимо от набора сигналов *R* и *S* задает для триггера режим хранения.

Для синхронизируемого триггера можно записать характеристические уравнения, взяв в качестве основы уравнения (1.3):

$$\begin{aligned} &Q := C(S \vee \overline{R}\, Q) \vee \overline{C}\, Q; \\ &\overline{Q} := C(R \vee \overline{S}\, \overline{Q}) \vee \overline{C}\, \overline{Q}. \end{aligned} \qquad (1.6)$$

Понятно, что в операторах (1.6) выражения можно записать в ином эквивалентном и, может быть, более компактном виде. Этого не сделано, потому что приведенные выражения явно содержат в себе выражения из (1.3) и являются более простыми для интерпретации.

Заметим, что в (1.6), так же, как и в (1.3), опущены индексы при переменных.

Уравнения (1.6) описывают функционирование триггера с синхронизацией высоким уровнем сигнала *C*. Но возможно и более широкое толкование этих уравнений. Их можно распространить на триггеры с любым типом синхронизации, если принять следующее толкование символа *C*: под символом *C* будем понимать *активную фазу* сигнала синхронизации (*активный уровень*, *активный фронт*), то есть ту фазу, которая инициирует выполнение микроопераций записи информации на триггер. Так, например, для триггера с синхронизацией положительным фронтом символ *C* означает момент формирования положительного фронта на входе синхронизации триггера.

На структурных схемах такая детальная информация, как вид синхронизации, может отсутствовать. Если же это принципиально, то синхросигналы должны показываться в структурной схеме и отражаться в описании микропрограмм.

Запись на синхронизируемый фронтом *RS*-триггер конкретного значения описывается микрооперациями, как показано в (1.7) и (1.8).

При синхронизации положительным фронтом:

$$\begin{aligned} &\overline{S}RC^{+} : T := 0; \\ &\overline{R}SC^{+} : T := 1. \end{aligned} \tag{1.7}$$

При синхронизации отрицательным фронтом:

$$\begin{aligned} &\overline{S}RC^{-} : T := 0; \\ &\overline{R}SC^{-} : T := 1. \end{aligned} \tag{1.8}$$

Здесь и далее C^{+} обозначает положительный фронт синхросигнала, а C^{-} – отрицательный фронт.

На схемах электрических функциональных *RS*-триггеры асинхронного типа, *RS*-триггеры, синхронизируемые высоким уровнем сигнала синхронизации, и *RS*-триггеры, синхронизируемые положительным фронтом, изображаются так, как показано на рис. 1.3-1.5 соответственно. Изображение входа *C* между входами *R* и *S* предпочтительнее из следующих соображений:

– вход *C* располагается одинаково близко к входам *R* и *S*, подчеркивая равные условия взаимодействия *C* с *R* и *C* с *S*;

– входы *S* и *R* одинаково расположены относительно выходов *Q* и $\overline{Q}$ соответственно.

Отметим, что условные обозначения входов *R* и *S*, показанные на рис. 1.3-1.5, можно поменять местами, но более желательными вариантами является именно приведенные на рисунках изображения. На них входы *R* и *S* показаны против тех выходов, на которых сигналы *R* и *S* устанавливают значение 1. При перестановке обозначений входов *R* и *S* зависимость выходных сигналов от входных окажется перекрестной, а при данном изображении эта зависимость в чисто геометрическом смысле прямая, то есть более простая для визуального восприятия и поэтому более желательная.

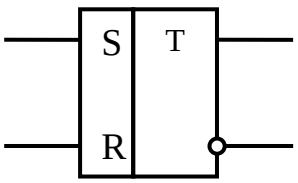

Рис. 1.3

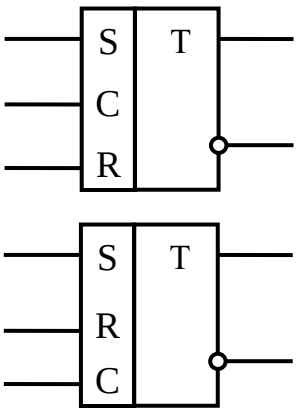

Рис. 1.4

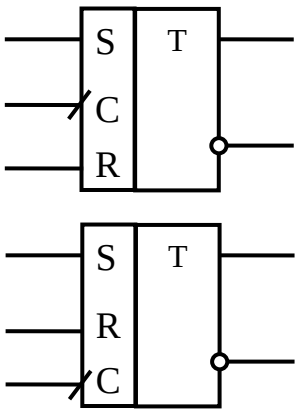

Рис. 1.5

На схемах электрических принципиальных приведенные обозначения триггеров дополняются номерами конкретных ножек микросхем, к которым подключены соответствующие входы и выходы элементов. Эти номера изображаются над соответствующими проводами, то есть за пределами нарисованных прямоугольников.

Второй из наиболее часто применяемых типов триггеров – это *D*-триггер. Условное обозначение *D*-триггера, как структурного элемента, дано на рис. 1.6.

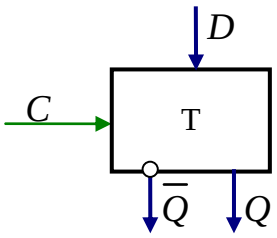

Рис. 1.6

В *D*-триггерах информация на входе *D* триггера задает состояние, в которое будет установлен триггер в результате действия импульса синхронизации. Поэтому сигнал, подаваемый на вход *D*, логично понимать не как управляющий сигнал, а как сигнал информационный, то есть как сигнал, несущий в себе информацию, которую необходимо записать на триггер. Поэтому на рис. 1.6 сигнал *D*, а значит и соответствующий вход триггера, изображен не рядом с сигналом *C*, как были изображены сигналы *R* и *S* на входах *RS*-триггера.

Условное изображение *D*-триггера на рис. 1.6 подчеркивает, что сигнал *D*, проходя через триггер (в соответствии с показанным стрелкой направлением), «превращается» в сигналы *Q* и $\overline{Q}$, то есть определяет значение выходных сигналов триггера. В отличие от *RS*-триггера, у *D*-триггера не существует запрещенной комбинации входных сигналов. В результате, у *D*-триггера сигналы *Q* (на прямом плече) и $\overline{Q}$ (на инверсном плече) всегда взаимно инверсны. По этой причине, *относительно D-триггера* (в отличие от *RS*-триггера) *отождествление сигнала Q с хранимым триггером значением является корректным.*

В принципе, сигнал *D* можно было бы рассматривать наряду с сигналом *C* как инициирующий выполнение микрооперации записи на триггер либо 0, либо 1. Описание соответствующих микроопераций выглядело бы так:

$$\begin{aligned} &\overline{D}C : T := 0; \\ &DC : T := 1. \end{aligned} \qquad (1.9)$$

При этом на условном изображении триггера сигнал *D* следовало бы изобразить рядом с сигналом *C*. Но анализ двух микроопераций в (1.9) позволяет заметить, что они могут быть обобщены и сведены к одной микрооперации:

$$C: T := D. \qquad (1.10)$$

Поэтому описание микроопераций в виде (1.9) не применяется.

Микрооперация (1.10) описывает будущее состояние *D*-триггера, синхронизируемого высоким уровнем сигнала *C*. В таком триггере в течение времени нахождения сигнала *C* в состоянии высокого уровня изменение значения сигнала *D* приводит к переустановке триггера в состояние, задаваемое новым значением сигнала *D*. По этой причине триггеры такого типа не следует использовать в цепях, охваченных обратной связью. В частности, *триггеры, синхронизируемые уровнем сигнала синхронизации, нельзя использовать в качестве элементов памяти управляющих автоматов*.

Таким образом, в *D*-триггерах с синхронизацией высоким уровнем информация с *D*-входа окончательно фиксируется как состояние триггера в момент выхода сигнала *C* из состояния высокого уровня, то есть в момент формирования отрицательного фронта импульса синхронизации. Но этот факт не является основанием для классификации такого триггера как триггера, синхронизируемого отрицательным фронтом импульса синхронизации.

Запись информации на *D*-триггер с синхронизацией по положительному фронту описывается микрооперацией:

$$C^{+}: T := D. \qquad (1.11)$$

При отсутствии положительного фронта импульса синхронизации такой триггер находится в состоянии хранения записанной на него информации.

Функционирование *D*-триггера можно описать характеристическим уравнением (1.12), в котором, в отличие от приведенных выше характеристических уравнений, поставлены индексы, отражающие моменты времени. Индексы подчеркивают временную задержку распространения сигналов через триггер: состояние сигналов «сейчас» (в момент времени t) определяет состояние триггера в последующий момент времени t+1, то есть состояние «потом».

$$Q_{t+1} := C_t D_t \vee \overline{C}_t Q_t. \qquad (1.12)$$

На схемах электрических функциональных *D*-триггеры с синхронизацией высоким уровнем и положительным фронтом изображаются так, как показано на рис. 1.7 и рис. 1.8.

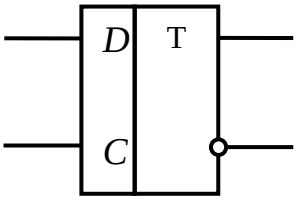

Рис. 1.7

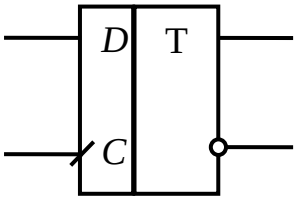

Рис. 1.8

Триггеры с синхронизацией уровнем более просты, и в них время распространения входного сигнала *D* до выходов триггера меньше, то есть такие триггеры отличаются более высоким быстродействием.

1.2. МОДЕЛИРОВАНИЕ РАБОТЫ СДВИГОВОГО РЕГИСТРА

Регистр по своему внутреннему устройству – это группа триггеров, у которых соединены между собой входы управляющих сигналов. В результате все триггеры регистра либо находятся в режиме хранения информации, либо одновременно выполняют одну и ту же микрооперацию. Чаще всего регистры строятся на основе *D*-триггеров.

Отдельный триггер регистра принято называть *разрядом регистра*. Термин «разряд» вызывает естественные ассоциации с разрядом некоторого числового кода. Применительно к регистрам такая ассоциация может быть как верной, так и ошибочной. Она будет ошибочной, когда отдельные разряды регистра будут отображать не значения разрядов некоторого двоичного кода, а, например, значения некоторых логических переменных или вычисленных логических функций.

Совокупность значений, хранимых всеми разрядами регистра, называют *кодом*, хранимым (записанным) на регистре. Учитывая отмеченное в предыдущем абзаце, не следует всегда ассоциировать этот код с кодом некоторого числа или его части.

Простейший тип регистров проектируется для выполнения только одной микрооперации: записи информации на регистр. У такого регистра предусматривается только один вход для управляющего сигнала: для сигнала синхронизации микрооперации записи. Часто у регистров предусматривают асинхронный вход, сигнал на котором «сбрасывает регистр»: в каждый из его разрядов записывает значение 0. Соответствующий вход и сигнал на нем обозначают символом *R* (*Reset*). Сигнал *Reset* всегда имеет более высокий приоритет, чем сигналы, инициирующие какую-либо иную микрооперацию. Это значит, что при инициировании микрооперации *Reset* одновременно с любыми другими микрооперациями на регистр будет записан код 0..0.

При увеличении количества входных управляющих сигналов удается в рамках одного регистра обеспечить возможность выполнения в данный момент любой из нескольких микроопераций. Типичный набор таких микроопераций следующий:

– *Сброс* (*Reset*) – установка всех разрядов регистра в состояние 0;

– *Загрузка* (*Load*) – запись на регистр параллельного кода, представленного сигналами на *D*-входах одновременно всех разрядов регистра;

– *Сдвиг* содержимого регистра *влево* или/и *вправо* на один разряд.

В режиме сдвига информация на *D*-входе отдельного триггера определяется содержимым соседнего с той или иной стороны разряда триггера.

Регистры, в которых предусмотрена микрооперация сдвига кода, называются *регистрами сдвига* или *сдвиговыми регистрами*.

Сдвиговый регистр называется *реверсивным*, когда предусмотрены микрооперации сдвига кода в двух направлениях. Если в регистре предусмотрено одно направление сдвига, его называют *однонаправленным сдвиговым регистром*.

Сдвиг кода в регистре всегда осуществляется как *сдвиг логический*. Поэтому в некоторых случаях сдвиг кода влево или вправо может быть использован для увеличения или, соответственно, уменьшения числового значения кода в два раза. Использованное ограничивающее словосочетание «в некоторых случаях» должно быть понятно

читателю, понимающему особенности представления числовой информации в прямом, обратном и дополнительном кодах.

Регистры, в которых предусмотрены микрооперации *Загрузка* и *Сдвиг*, могут использоваться в схемах преобразования параллельных кодов в коды последовательные и, наоборот, для преобразования последовательных кодов в коды параллельные. Такие преобразования необходимы, в частности, для обмена информацией через последовательные порты устройств ввода-вывода информации.

При сдвиге информации в регистре для крайнего слева разряда регистра (при сдвиге вправо), либо для крайнего справа разряда регистра (при сдвиге влево) в составе регистра не имеется соседнего разряда, информация с которого должна быть переписана на этот крайний разряд. Для заполнения таких крайних разрядов сдвиговый регистр имеет специальные информационные входы, часто обозначаемые как *DL* и *DR*. Сигнал на входе *DL* востребован при сдвиге влево, а на входе *DR* – при сдвиге вправо. В аббревиатурах *DL* и *DR* символ *D* обозначает информационный вход *D*-триггера, а символы *L* и *R* – направления сдвига информации: *L* (*Left*) – влево; *R* (*Right*) – вправо.

Один из возможных вариантов условного обозначения на структурных схемах *n*-разрядного однонаправленного сдвигового регистра показан на рис. 1.9, где использованы следующие обозначения:

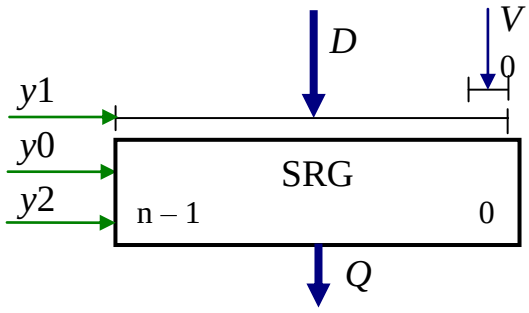

Рис. 1.9

D – совокупность проводов (*шина*) и одновременно код на этих проводах, подключенных к информационным входам регистра;

Q – код на регистре;

V – одноразрядный код на проводе, подключенном к информационному входу разряда регистра с номером 0 (номер указан над полочкой), востребованному только при выполнении микрооперации сдвига кода;

*y*0, *y*1, *y*2 – управляющие сигналы, инициирующие выполнение микроопераций.

Разрядность кодов $D = d_{n-1}\ldots d_1 d_0$ и $Q = q_{n-1}\ldots q_1 q_0$ одинакова. Содержимое разряда q_0 после выполнения микрооперации загрузки будет равно значению d_0, а после микрооперации сдвига – значению V.

В данном примере для инициирования выполнения каждой из микроопераций предусмотрен отдельный управляющий сигнал (1.13):

$$\begin{aligned} &y0\text{: } Q_{t+1} := 0 \text{ (сброс);} \\ &y1\text{: } Q_{t+1} := D_t \text{ (загрузка);} \\ &y2\text{: } Q_{t+1} := L1(Q_t, V_t) \text{ (сдвиг влево).} \end{aligned} \qquad (1.13)$$

Пара (Q, V) здесь обозначает операцию приклеивания кода V справа к коду Q. Таким образом, в сдвиге влево участвует код длиной в $n+1$ разряд.

На структурных схемах коды на выходах операционных элементов именуются не всегда. Пусть, например, на рис. 1.9 отсутствует обозначение Q кода на выходе регистра. В таких случаях микрооперации (1.13) могут быть заданы в следующем виде:

$$\begin{aligned} &y0\text{: } SRG_{t+1} := 0 \text{ (сброс);} \\ &y1\text{: } SRG_{t+1} := D_t \text{ (загрузка);} \\ &y2\text{: } SRG_{t+1} := L1(SRG_t, V_t) \text{ (сдвиг влево),} \end{aligned} \qquad (1.14)$$

где SRG обозначает код в сдвиговом (S) регистре (RG).

Логический смысл сигнала $y1$ на рис. 1.9 можно понять по положению его условного обозначения относительно полочки, к которой подведен код D. Подобное взаимное расположение стрелки с управляющим сигналом относительно полочки с кодом всегда обозначает, что управляющий сигнал регулирует прохождение кода через эту полочку. Подобные тройки: полочка, код и управляющий сигнал, обозначают *управляемую шину*.

Логический смысл сигналов $y0$ и $y2$ никак не следует из положения их условных обозначений относительно условного изображения регистра. Этот смысл определяется описанием соответствующих микроопераций.

На примере рис. 1.9 обратим внимание на некоторые принятые для структурных схем условности.

Если некоторый код (в нашем примере это код *Q*) помечает стрелку, исходящую непосредственно из условного изображения операционного элемента, то независимо от смещения этой стрелки относительно геометрической середины изображения операционного элемента она всегда отображает все выходы, то есть весь код, который может быть «снят», «считан» с этого элемента. Подобное замечание справедливо также относительно изображения информационных кодов, подключенных к информационным входам операционного элемента. При необходимости указания на часть входного (выходного) кода используются полочки, над которыми должны быть проставлены номера разрядов входного (выходного) кода. Над полочками указываются номера крайних разрядов кода. Принято использовать нумерацию, при которой номера разрядов кода возрастают справа налево. Геометрическое расположение полочек относительно условного изображения элемента, а также протяженность полочек сами по себе ничего не говорят о номерах соответствующих разрядов кода. Номера разрядов во всех случаях должны быть определены явно. На рис. 1.9 стрелка, отмеченная кодом *D*, подведена к полочке не с целью выделения части разрядов входного кода, а с целью указания особой роли управляющего сигнала *y*1. В подобных случаях отсутствие над полочкой номеров разрядов входного кода следует интерпретировать (по умолчанию) как указание на информационный код, длина которого совпадает с количеством информационных входов операционного элемента.

Отметим, что управляющие цепи и сигналы могут подводиться к условному изображению операционного элемента как слева, так и справа.

Заметим, что приведенные здесь условные обозначения *y*0, *y*1 и *y*2 сигналов, инициирующих выполнение микроопераций, не являются обязательными. У второго экземпляра точно такого же сдвигового регистра вместо использованного здесь обозначения, например, *y*0, может быть задано обозначение, например, *y*15.

В (1.13) сигнал *y*2 определен как сигнал, инициирующий выполнение микрооперации «сдвиг влево». Следует понимать, что указанное направление сдвига верно относительно взаимного перемещения содержимого разрядов кода *Q*. Но оно может быть неверным относительно разрядов кода *I*, который был источником кода *D*, записанного на регистр. Все зависит от того, как разряды кода *I* подключены к шине *D*, следовательно, и к входам регистра. Рассмотрим примеры таких подключений.

Вариант 1. Пусть $D = d_{n-1}\ldots d_1 d_0 = i_{n-1}\ldots i_1 i_0 = I$. Тогда, с помощью сдвига влево кода *Q* осуществляется сдвиг в том же направлении предварительно записанного на регистр кода *I*.

Вариант 2. Пусть $d_{n-1}\ldots d_1 d_0 = i_0 i_1 \ldots i_{n-1}$. Тогда с помощью сдвига влево кода *Q* осуществляется сдвиг вправо предварительно записанного на регистр кода *I*.

Вариант 3. Для этого случая с целью упрощения примера примем, что регистр 8-разрядный, и пусть $d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 = i_{14} i_{12} i_{10} i_8 i_6 i_4 i_2 i_0$. Тогда с помощью сдвига влево кода *Q* осуществляется сдвиг в том же направлении, но на два разряда, предварительно записанного на регистр кода, составленного из разрядов кода *I* с четными номерами. Чтобы этот пример не казался абсурдным, предлагаем читателю представить, что на второй экземпляр такого же регистра были записаны разряды кода *I* с нечетными номерами, и оба экземпляра регистра соединены между собой по входам управляющих сигналов. Тогда два таких 8-разрядных регистра, выполнив одновременно микрооперацию сдвига кодов на один разряд влево, осуществят сдвиг кода *I* влево на два разряда.

При большом числе выполняемых микроопераций управление регистром удобнее осуществлять путем предварительной его настройки на нужную микрооперацию с помощью специальных управляющих сигналов. В частности, так реализовано управление большинства реверсивных сдвиговых регистров. Пример такого регистра дан на рис. 1.10.

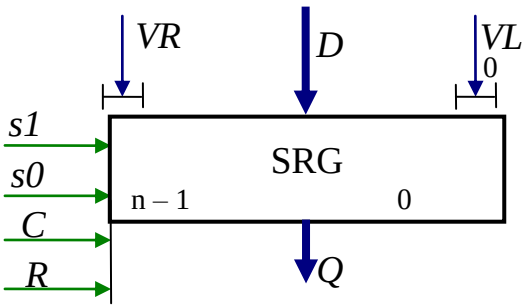

Рис. 1.10

Сдвиговый регистр (см. рис. 1.10) имеет два дополнительных информационных входа:

– вход *VL* используется в микрооперации сдвига кода на регистре влево для заполнения младшего (крайнего правого) разряда регистра;

– вход *VR* используется в микрооперации сдвига кода на регистре вправо для заполнения старшего (крайнего левого) разряда регистра.

Входные сигналы *s*1 и *s*0 задают код, настраивающий внутренние логические схемы на выполнение одной из микроопераций, инициирование которой осуществит сигнал синхронизации *C*. Можно сказать, что пара (*s*1, *s*0) управляющих сигналов задает режим работы регистра.

Отметим, что правильное управление регистром предполагает некоторое запаздывание активной фазы сигнала синхронизации *C* относительно управляющих сигналов *s*1 и *s*0, а также относительно момента изменения входного кода *D*.

Микрооперация «Сброс» устанавливает на регистре код Q = 0..0 и инициируется непосредственно управляющим сигналом, обозначенным как *R* (*Reset*), асинхронно по отношению к синхросигналу *C*, то есть независимо от сигнала *C* (1.15):

$$R\text{: } Q_{t+1} := 0. \qquad (1.15)$$

В регистрах сигнал сброса всегда имеет наибольший приоритет. Достигается это тем, что внутри операционного элемента сигнал *R* воздействует на входы *R* асинхронных *RS*-триггеров, входящих в состав регистра, в обход логических цепей, реализующих остальную логику управления регистром.

Для выполнения остальных микроопераций необходимо с помощью управляющих сигналов *s*1, *s*0 настроить регистр на соответствующий режим работы. Непосредственное исполнение заданной микрооперации инициируется активной фазой сигнала синхронизации *C*.

Напомним, что при использовании синхронизации импульсом в течение активной фазы сигнала синхронизации каждое изменение информации на входе триггера может приводить к переключениям состояния триггера. По этой причине в цепях сдвига информации нельзя использовать триггеры, синхронизируемые испульсом. Следовательно, любой сдвиговый регистр должен синхронизироваться каким-либо фронтом импульса синхронизации. При этом для гарантированного сдвига информации под управлением одного активного фронта ровно на один разряд предъявляются особые требования к крутизне фронта сигнала, то есть к скорости изменения уровня напряжения: *длительность активного фронта импульса синхронизации должна быть меньше временной задержки переключения отдельного триггера*. Только при выполнении этого условия к моменту переключения разрядов сдвигового регистра в новые состояния активный фронт импульса синхронизации оказывается завершенным, что делает невозможным продвижение информации в заданном направлении еще на один разряд.

В табл. 1.2 для рассматриваемого реверсивного сдвигового регистра приведены двоичные коды режимов работы и соответствующие им микрооперации. Еще раз подчеркнем, что инициирование этих микроопераций сигналом синхронизации возможно только вне режима «Сброс».

Таблица 1.2

$s1$ $s0$	Микрооперация
0 0	Хранение
0 1	$Q_{t+1} := R1(VR_t, Q_t)$ – сдвиг кода вправо
1 0	$Q_{t+1} := L1(Q_t, VL_t)$ – сдвиг кода влево
1 1	$Q_{t+1} := D_t$ – запись (загрузка параллельного кода)

Заметим, что по аналогии с переходом от описания микроопераций в виде (1.13) к описанию в виде (1.14) можно в табл. 1.1 обозначение Q кода на регистре заменить обозначением *SRG*, понимая под *SRG* не сам регистр, а код, устанавливаемый на регистре и формируемый на его выходе.

На рис. 1.11 приведено условное графическое обозначение на принципиальных электрических схемах реального 8-разрядного реверсивного сдвигового регистра, реализованного в виде микросхемы К155ИР13.

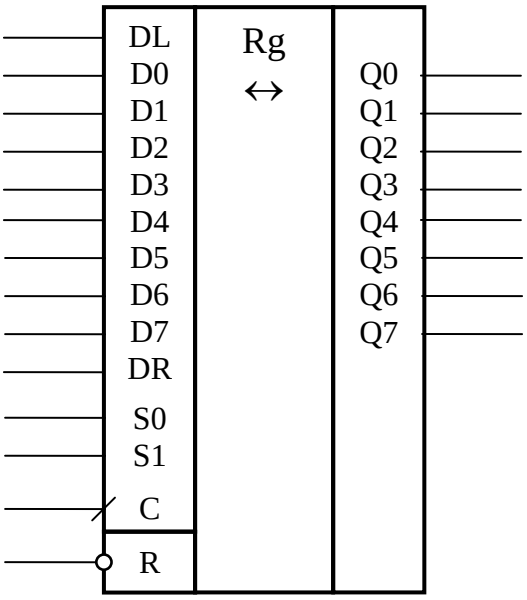

Рис. 1.11

Назначение входов и выходов микросхемы приведено в табл. 1.3.

Таблица 1.3

Выводы	Назначение
DL	Информационный вход для режима «Сдвиг влево»
*D*7 .. *D*0	Информационные входы, используемые для загрузки параллельного кода
*Q*7 .. *Q*0	Информационные выходы
DR	Информационный вход для режима «Сдвиг вправо»
*S*1, *S*0	Входы задания режима работы регистра
C	Вход синхронизации микрооперации, заданной кодом (*S*1, *S*0)
R	Вход инициализации режима «Сброс»

Полная таблица зависимости выходов микросхемы от ее управляющих и информационных входов приведена в табл. 1.4.

К управляющим входам регистра относятся входы *R*, *S*0, *S*1 и *C*. Высоким уровнем на входе *R* (*Reset*) производится асинхронный сброс кода на регистре в значение 0. Остальные режимы работы задаются кодом (*S*1, *S*0), а выполнение соответствующих микроопераций синхронизируется положительным фронтом сигнала на

входе *C*. Сигналы на управляющих входах *S*0 и *S*1 задают режим работы регистра в соответствии с содержимым табл. 1.2.

Таблица 1.4

Входы $_t$							Выходы $_{t+1}$			Коммен-тарии
управляющие				информационные						
R	*S*1	*S*0	*C*	*DR*	*DL*	*D*7…*D*0	*Q*7	*Q*6 … *Q*1	*Q*0	
0	*	*	*	*	*	*	0	0 …0	0	Сброс
1	1	1	↑	*	*	*A* ... *H*	*A*	*B* ... *G*	*H*	Запись
1	0	1	↑	1	*	*	1	$Q7_t$ … $Q2_t$	$Q1_t$	Сдвиг вправо
1	0	1	↑	0	*	*	0	$Q7_t$ … $Q2_t$	$Q1_t$	Сдвиг вправо
1	1	0	↑	*	1	*	$Q6_t$	$Q5_t$ … $Q0_t$	1	Сдвиг влево
1	1	0	↑	*	0	*	$Q6_t$	$Q5_t$ … $Q0_t$	0	Сдвиг влево
1	*	*	не↑	*	*	*	$Q7_t$	$Q6_t$ … $Q1_t$	$Q0_t$	Хранение
1	0	0	*	*	*	*	$Q7_t$	$Q6_t$ … $Q1_t$	$Q0_t$	Хранение

Переменные *A* .. *H* обозначают разряды двоичного кода на входах *D*7.. *D*0.

Входные сигналы сдвигового регистра должны формироваться с учетом следующего простого правила: к моменту поступления положительного фронта импульса синхронизации (*C* в состоянии ↑) на остальных входах регистра сигналы должны быть установлены, то есть переходные процессы, связанные с переключениями этих сигналов, должны быть завершены.

1.3. МОДЕЛИРОВАНИЕ РАБОТЫ РЕВЕРСИВНОГО СЧЕТЧИКА

Возможны два варианта реализации счетчика:

– в виде комбинационной схемы, на вход которой подается код числа, а на выходе формируется код следующего или предыдущего числа;

– в виде регистра хранения информации, состав микроопераций которого дополнен микрооперацией счета.

Счетчик, в котором реализованы два варианта микрооперации счета: счет на сложение и счет на вычитание, называется *реверсивным счетчиком*.

В этом подразделе будут рассмотрены счетчики на основе регистров, допускающие параллельную загрузку на встроенный регистр кода, поданного на информационные входы счетчика.

Отметим, что относительно входных сигналов счетчика правильнее говорить об информационных входах именно счетчика, а не встроенного в счетчик регистра. Разряды встроенного регистра естественно имеют информационные входы, но они не имеют прямой электрической связи с входами счетчика, так как отделены от входов счетчика логическими схемами, реализующими всю логику выполнения микроопераций.

Пример условного обозначения n-разрядного реверсивного счетчика на структурных схемах показан на рис. 1.12:

D – совокупность проводов (*шина*) и одновременно код на этих проводах, подключенных к n информационным входам счетчика;

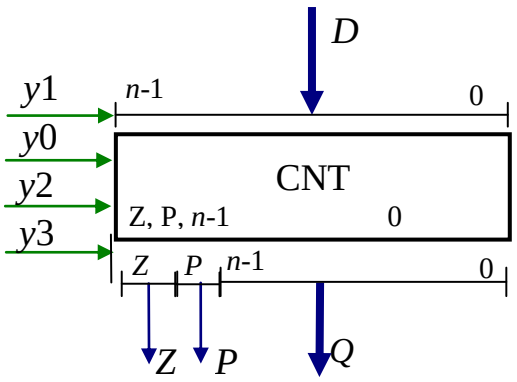

Рис. 1.12

Q – n-разрядный код на регистре счетчика (код на счетчике);

$y0$, $y1$, $y2$, $y3$ – управляющие сигналы, инициирующие выполнение микроопераций;

P – выход сигнала переноса (переполнения) счетчика при сложении;

Z – выход сигнала заема единицы при вычитании.

Таким образом, у реверсивного счетчика количество информационных выходов оказывается на два больше количества информационных входов.

Выходы P и Z могут использоваться для последовательного соединения счетчиков и в качестве сигналов логических условий, передаваемых в управляющий автомат.

Реверсивный счетчик выполняет следующий набор микроопераций:

$$y0: Q_{t+1} := 0 \text{ (сброс)};$$

$$y1: Q_{t+1} := D_t \text{ (загрузка)};$$

$$y2: Q_{t+1} := Q_t + 1 \text{ (счет на сложение)}; \qquad (1.16)$$

$$y3: Q_{t+1} := Q_t - 1 \text{ (счет на вычитание)}.$$

В (1.16) обозначение Q кода на счетчике можно заменить обозначением, совпадающим с именем операционного элемента, то есть именем *CNT* (count).

Микрооперации сброса и загрузки кода на счетчик инициируются уровнями сигналов синхронизации соответствующих микроопераций. При этом режим сброса кода на счетчике в ноль имеет наивысший приоритет. Микрооперации счета инициируются только фронтами сигналов синхронизации. Причина использования только фронтов такая же, как у сдвиговых регистров.

Особенности формирования сигналов переноса и заема будут рассмотрены на примере конкретного реверсивного 4-разрядного счетчика К155ИЕ7, условное графическое обозначение которого на принципиальных электрических схемах приведено на рис. 1.13.

На рис. 1.13 номера разрядов входного и выходного кодов задают веса соответствующих разрядов двоичных кодов. При такой нумерации разрядов удобнее интерпретировать коды на входах и выходах счетчика.

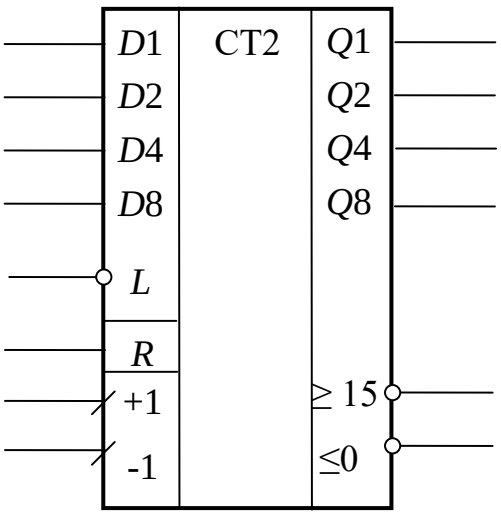

Рис. 1.13

Назначение выводов микросхемы К155ИЕ7 приведено в табл. 1.5.

Таблица зависимости выходных сигналов микросхемы от ее управляющих и информационных входов приведена в табл. 1.6.

Таблица 1.5

Выводы	Назначение
*D*8, *D*4, *D*2, *D*1	Информационные входы: используются для параллельной загрузки кода на счетчик
*Q*8, *Q*4, *Q*2, *Q*1	Информационные выходы
L	Вход синхронизации записи (загрузки) кода на счетчик
R	Вход сброса счетчика в нулевое состояние
+1	Вход синхронизации положительным фронтом микрооперации счета на сложение
-1	Вход синхронизации положительным фронтом микрооперации счета на вычитание
≥ 15	Выход переноса из старшего разряда при счете на сложение
≤ 0	Выход заема из старшего разряда при счете на вычитание

Заметим, что выполнение микроопераций счета возможно только при высоком уровне сигнала на другом счетном входе и отключенных режимов сброса и загрузки кода.

Таблица 1.6

Входы $_t$					Выходы $_{t+1}$			Комментарий
управляющие				информационные				
R	¬*L*	+1	-1	*D*8 … *D*1	*Q*8 … *Q*1	≥ 15	≤ 0	
1	*	*	0	*	0000	1	0	Сброс
1	*	*	1	*	0000	1	1	Сброс
0	0	*	0	0000	0000	1	0	Запись кода
0	0	*	1	0000	0000	1	1	Запись кода
0	0	*	*	*ABCD*	*ABCD*	1	1	Запись кода
0	0	0	*	1111	1111	0	1	Запись кода
0	0	1	*	1111	1111	1	1	Запись кода
0	1	↑	1	*	$Q_t + 1$	1	1	Счет на сложение
0	1	1	↑	*	$Q_t - 1$	1	1	Счет на вычитание

Обратим внимание на то, что входной сигнал, инициирующий режим загрузки (*Load*) кода, после инвертирования задает значение переменной *L*. Поэтому этот сигнал на входе счетчика имеет значение $\overline{L} \equiv \neg L$, что и отражено в табл. 1.6.

В течение интервала времени удержания на счетчике кода 1111 сигнал переноса «≥ 15» повторяет сигнал на счетном входе «+1».

В течение интервала времени удержания на счетчике кода 0000 сигнал заема «≤ 0» повторяет сигнал на счетном входе «-1».

1.4. МОДЕЛИРОВАНИЕ РАБОТЫ КОМБИНАЦИОННЫХ ОПЕРАЦИОННЫХ ЭЛЕМЕНТОВ

1.4.1. КОМБИНАЦИОННЫЙ СУММАТОР

Комбинационные операционные элементы способны выполнять только преобразование и передачу данных, но не их хранение, так как в них отсутствуют элементы памяти, то есть триггеры.

Комбинационный сумматор предназначен для вычисления арифметической суммы S двоичных кодов:

$$S := A + B + C, \qquad (1.17)$$

где A, B – двоичные коды слагаемых;

C – бит переноса.

При суммировании бит переноса складывается с младшими разрядами кодов слагаемых A и B.

Условное обозначение n-разрядного комбинационного сумматора на структурных схемах показано на рис. 1.14, где P – выходной бит переноса из старшего разряда суммы, то есть из разряда с номером n–1.

На основе k экземпляров сумматоров, показанных на рис. 1.14, можно построить kn-разрядный сумматор. Тогда отдельный экземпляр сумматора будет суммировать группу разрядов кодов слагаемых. При этом P задает бит переноса из группы разрядов суммы и называется *групповым переносом*. Выход группового переноса из i-той группы разрядов соединяется с входом переноса C соседней старшей, то есть (i+1)-й группы разрядов. Групповой перенос из самой старшей (k-той) группы разрядов суммы задает перенос из всей суммы кодов слагаемых. Отметим, что этот перенос нельзя отождествлять с признаком переполнения суммы.

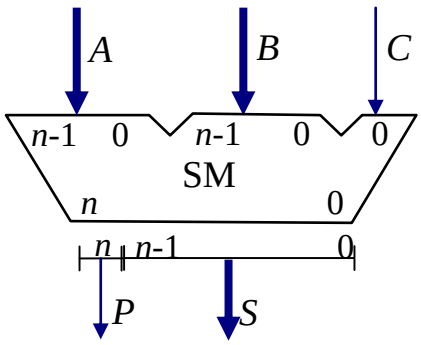

Рис. 1.14

Сигнал P переноса может использоваться в качестве сигнала логического условия. Обычно этот сигнал запоминается на триггере в виде значения *флага переноса*, и логическое условие считывается с этого триггера.

При сложении дополнительных кодов знаки чисел располагаются в старшем разряде разрядной сетки и участвуют в суммировании наряду с числовыми разрядами. В результате, на выходе, соответствующем старшему разряду суммы, формируется знак результата.

Напомним, что при использовании модифицированных кодов знаки слагаемых и суммы занимают два старших разряда разрядной сетки.

Комбинационный сумматор выполняет только одну микрооперацию и поэтому для его работы не требуется управляющих сигналов. С другой стороны, вход *C* можно использовать как управляющий, задающий выполнение одной из двух различных микроопераций:
S:= *A* + *B* + 1 либо *S*:= *A* + *B*.

1.4.2. КОМБИНАЦИОННЫЙ СДВИГАТЕЛЬ КОДОВ

Комбинационные *сдвигатели* кодов – это многофункциональные элементы, служащие для передачи кодов с входа *D* на выход *U* без сдвига, либо со сдвигом влево, либо со сдвигом вправо на 1, 2 или большее число разрядов. Микрооперации, выполняемые сдвигателем, необходимы, в частности, в микропрограммах арифметических команд умножения и деления.

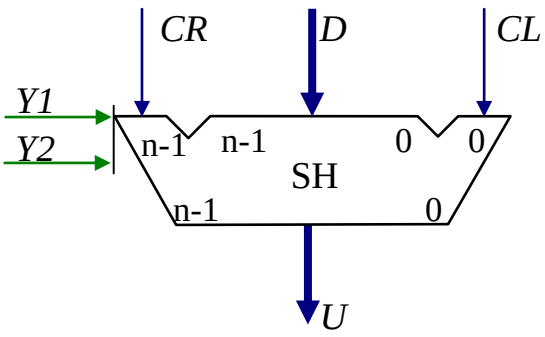

Рис. 1.15

Настройка таких многофункциональных элементов, как сдвигатель, на выполнение заданной микрооперации осуществляется двоичным управляющим кодом от управляющего автомата.

На рис. 1.15 показано условное обозначение сдвигателя на структурных схемах, а его функционирование описано в табл. 1.7.

Таблица 1.7

*Y*2, *Y*1	Микрооперация
0 0	Не используется
0 1	*U* := *R*1(*CR*, *D*);
1 0	*U* := *L*1(*D*, *CL*);
1 1	*U* := *D*

Входы CL и CR служат для подачи бита в освобождающийся разряд разрядной сетки и востребованы только при соответствующих микрооперациях «косой» передачи кодов.

Отметим, что рассматриваемый операционный элемент не имеет специальных входов синхронизации начала выполнения микрооперации. В любой момент времени, за исключением моментов переключения сигналов Y2 и Y1 (в эти моменты микрооперация не определена) на управляющих входах, выполняется одна из микроопераций, указанных в табл. 1.7. Любые изменения входных информационных и/или управляющих кодов вызывают изменение выходного кода.

Комбинация управляющих сигналов, обозначенная в табл. 1.7 как неиспользуемая, может быть использована для расширения функциональных возможностей операционного элемента. Можно, например, перевести выходы элемента в третье состояние: *состояние высокого импеданса*. Можно задать выходному коду значение 0..0. Можно для снятия неопределенности выполнить микрооперацию прямой передачи кода: *U* := *D*.

1.4.3. КОМБИНАЦИОННЫЙ ФОРМИРОВАТЕЛЬ КОДОВ

Формирователи (преобразователи) двоичных кодов предназначены для преобразования двоичных кодов из одного формата в другой.

Для арифметических операционных устройств одной из типичных является операция смены знака числа. Для чисел, представленных в обратном коде и в дополнительном коде, операция смены знака числа требует инвертирования всех разрядов кода. Инвертирование разрядов кода необходимо также при преобразованиях прямого кода отрицательного числа в обратный код и в дополнительный код, а также при обратных преобразованиях: из обратного и дополнительного кодов в прямой код. Напомним, что для положительных чисел их прямой, обратный и дополнительный коды совпадают.

Из изложенного следует, что для операционных устройств типичным является операционный элемент, осуществляющий управляемую передачу кода с входа на выход либо в неизменном виде, либо в

поразрядно инвертированном виде. Назовем такое устройство *формирователем инверсного кода* (ФИК). Условное обозначение сдвигателя на структурных схемах показано на рис. 1.16, а описание его функционирования – в табл. 1.8.

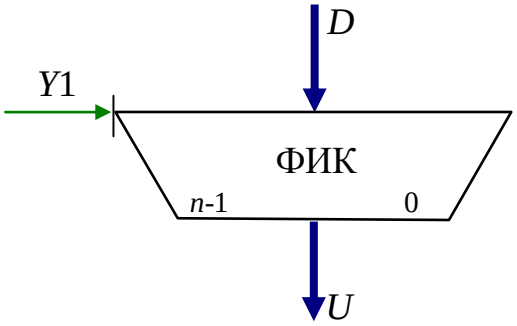

Рис. 1.16

Таблица 1.8

*Y*1	Микрооперация
0	*U* := *D*
1	*U* := ¬*D*

1.4.4. МУЛЬТИПЛЕКСОРЫ

Мультиплексоры предназначены для передачи информации с любой из входных шин *D*0, *D*1, …, *Dk* на выходную шину *U*.

Условное обозначение мультиплексора с 4 входными шинами приведено на рис. 1.17, а микрооперации, выполняемые рассматриваемым мультиплексором, – в табл. 1.9.

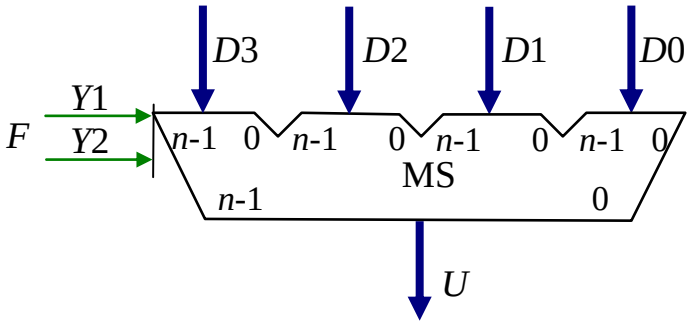

Рис. 1.17

Таблица 1.9

*Y*2, *Y*1	Микрооперация
0 0	*U* := *D0*
0 1	*U* := *D1*
1 0	*U* := *D2*
1 1	*U* := *D3*

Выбор входной шины (D3, …, D0) зависит от двоичного кода F, задаваемого управляющими сигналами *Y*1, *Y*2.

Мультиплексор, изображенный на рис. 1.17, структурно состоит из *n* мультиплексоров типа MS 4-1, у которых соединены между собой одноименные управляющие входы. Отдельный экземпляр MS 4-1 имеет 4 входные одноразрядные шины и выходную одноразрядную шину.

При использовании двоичного кода F длиной k разрядов можно построить мультиплексор с 2^k входными шинами.

1.4.5. ШИННЫЕ ФОРМИРОВАТЕЛИ

Шинные формирователи предназначены для управляемого подключения к общим шинам выходов операционных элементов – источников информации. Условные обозначения шинных формирователей приведены на рис. 1.18 и рис. 1.19.

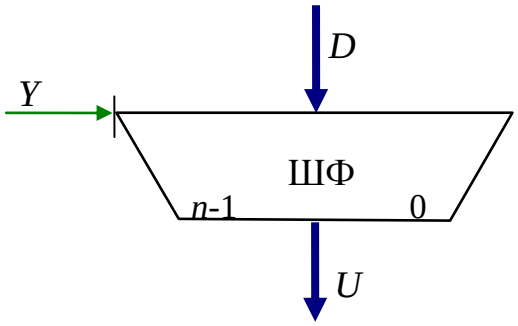

Рис. 1.18

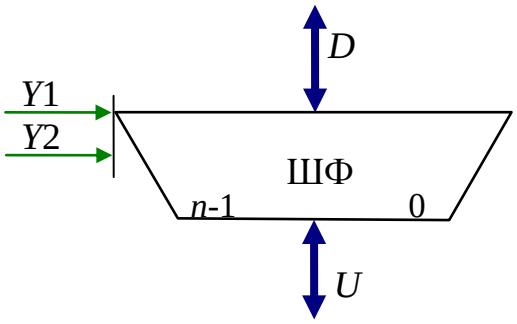

Рис. 1.19

Простейший n-разрядный шинный формирователь можно построить на основе n конъюнкторов с выходами типа *открытый коллектор*. При этом напряжение питания выходных транзисторов шинных формирователей подается с общей шины. При отключении всех формирователей от общей шины на проводах этой шины будет удерживаться высокое напряжение. Логика управления формирователями должна быть такой, чтобы не более чем один формирователь переключал состояния своих выходных транзисторов. Остальные формирователи должны удерживать свои выходные транзисторы в закрытом состоянии.

Второй из простейших вариантов реализации шинного формирователя состоит в применении конъюнкторов, выходы которых можно устанавливать в одно из трех состояний: низкое напряжение, высокое напряжение, состояние высокого выходного сопротивления (высокого импеданса). Как и в первом случае, логика управления формирователями должна допускать наличие не более одного источника информации на общую шину. Остальные формирователи должны удерживать свои выходы в состоянии высокого импеданса. При отключении

всех формирователей от общей шины на проводах этой шины будет отсутствовать как низкое, так и высокое напряжение, то есть провода общей шины оказываются изолированными от источника питания. Поэтому второй вариант реализации шинных формирователей схемотехнически лучше первого. Именно в рамках второго варианта можно построить формирователь, изображенный на рис. 1.18. Формирователь, показанный на рис. 1.19, имеет более сложную внутреннюю структуру, чем описано в двух рассмотренных вариантах.

В табл. 1.10 приведены микрооперации, выполняемые шинным формирователем, представленным на рис. 1.18.

Таблица 1.10

Y	Микрооперация
0	Высокое выходное сопротивление
1	*U* := *D*

В состоянии высокого выходного сопротивления выходы формирователя фактически изолируются от общей шины. В этом случае другой шинный формирователь, подключенный к общей шине, может быть переключен в режим передачи информации на шину.

На рис. 1.19 изображен двунаправленный шинный формирователь, обеспечивающий, в зависимости от управляющих сигналов, передачу данных как в прямом направлении от шины *D* к шине *U*, так и в обратном.

Режимы функционирования двунаправленного шинного формирователя приведены в табл. 1.11.

Таблица 1.11

*Y*2, *Y*1	Микрооперация
x 1	Высокое выходное сопротивление в обоих направлениях
1 0	*U* := *D*
0 0	*D* := *U*

При высоком уровне сигнала *Y*1 устанавливается высокое выходное сопротивление в обоих направлениях. Шинный формирователь оказывается электрически изолированным как от шины *U*, так и от

шины *D*. Низкий уровень на *Y*1 разрешает передачу данных через ШФ. Направление передачи определяется уровнем сигнала на *Y*2.

Двунаправленный шинный формирователь можно синтезировать из двух однонаправленных шинных формирователей за счет их параллельного встречного соединения.

1.5. МОДЕЛИРОВАНИЕ РАБОТЫ ОПЕРАЦИОННОГО АВТОМАТА

В данном разделе приведен пример операционного автомата (ОА), способного выполнять несколько команд арифметической обработки кодов операндов, загруженных на регистры ОА. На рис. 1.20 приведена его структурная схема, в которой использованы следующие обозначения:

ШД – 16-ти разрядная шина данных;

ШФ – шинный формирователь;

УА – управляющий автомат;

F – двоичный код, указывающий на выполняемую микропрограмму;

RGA, RGB – регистры для размещения операндов;

ФИК – формирователь инверсного кода;

A, B, R – имена кодов на выходах некоторых операционных элементов;

SM – сумматор;

RGR – регистр результатов;

RGF – 2-х разрядный регистр флагов (признаков результата);

*C*1, *C*2 – синхросигналы;

y_i – сигналы для управления элементами хранения;

Y_i – сигналы для управления комбинационными элементами;

*x*0 – признак совпадения знаков операндов;

*x*1 – флаг *CF* переноса из 15-го разряда суммы;

*x*2 – флаг *SF* знака результата, формируемый на выходе старшего 15-го (знакового) разряда сумматора;

OF – Overflow – переполнение разрядной сетки;

RESULT – признак завершения выполнения микропрограммы.

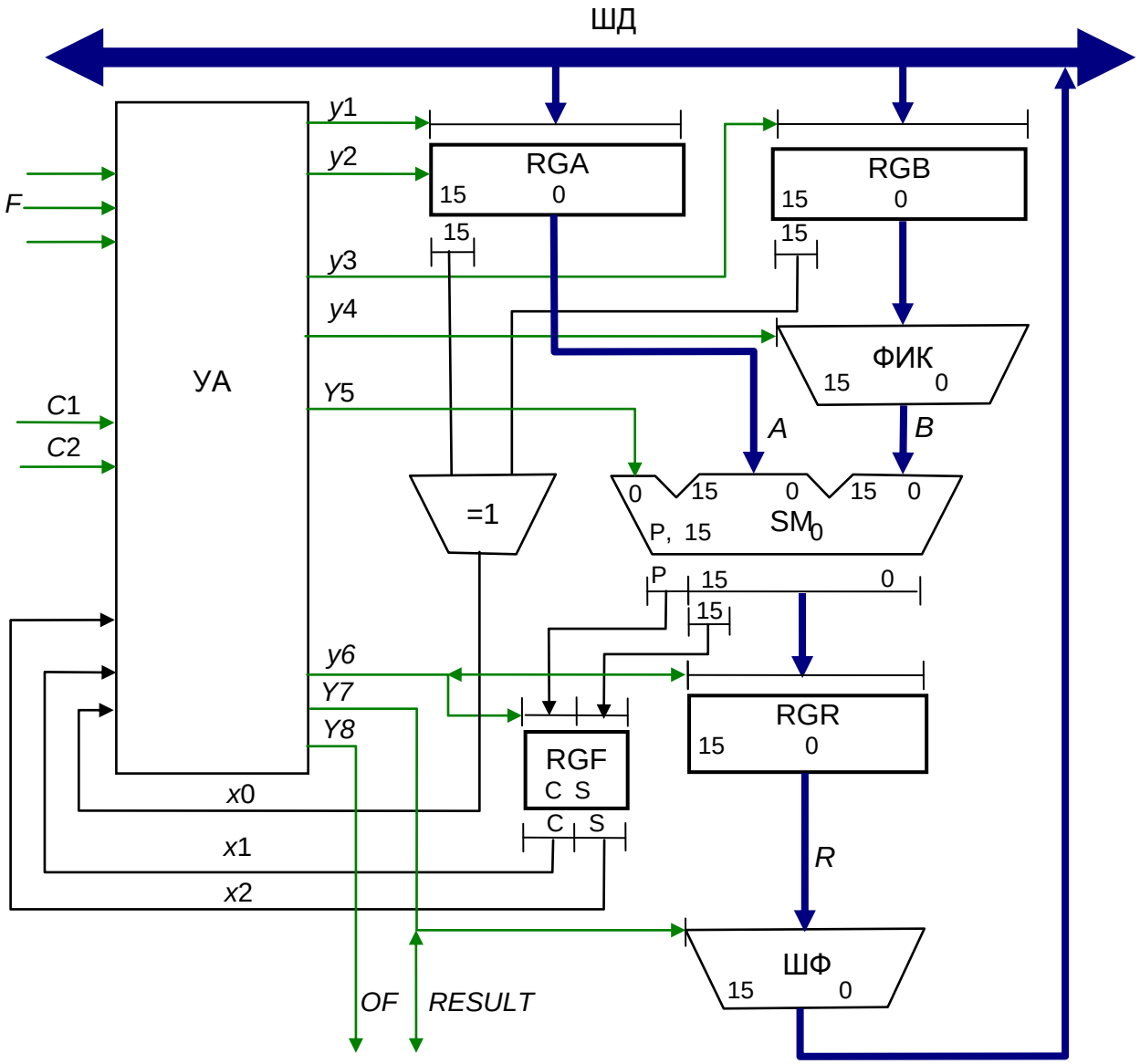

Рис. 1.20

В ОА предусмотрены следующие микрооперации:

$y1$: $RGA := ШД$;

$y2$: $RGA := 0$;

$y3$: $RGB := ШД$; (1.18)

$Y4$: $B := \overline{RGB}$;

$Y5$: слагаемое на входе переноса сумматора;

$y6$: $RGR := SM[15..0]$; $RGF[S] := SM[15]$; $RGF[C] := SM[P]$;

$Y7$: $ШД := RGR$; $RESULT := 1$.

Сигнал *Y*5 используется в качестве слагаемого, подаваемого на вход переноса в младший разряд сумматора.

Сигналы управления *Y*4, *Y*5, *Y*7, *Y*8 комбинационными схемами имеют длительности, равные длительности такта работы управляющего автомата: они устанавливаются в состояния «1» в момент перехода УА в соответствующее состояние и удерживаются в состоянии «1» до момента перехода УА в очередное состояние. Сигнал *y*2 – это сигнал асинхронного сброса регистра, поэтому он может быть сформирован так же, как сигналы управления комбинационными схемами. Остальные сигналы управления элементами хранения: *y*1, *y*3, *y*6 – это сигналы динамической синхронизации регистров хранения. Активный фронт этих сигналов должен быть достаточно задержан относительно сигналов управления комбинационными схемами для того, чтобы к моменту появления активного фронта коды операндов успели пройти через управляемые комбинационные схемы. Например, для выполнения микрооперации *RGR*:= *A*+*RGA*+1 необходимо установить в состояние «1» сигналы *Y*4 и *Y*5. При этом активный фронт сигнала *y*6 необходимо задержать относительно начала соответствующего такта на время, достаточное для последовательного переключения *ФИК* и *SM*.

Функционирование операционного устройства (ОУ), изображенного на рис. 1.20, начинается подачей от внешнего устройства на шину *F* кодов микропрограмм УА конкретного кода микропрограммы. УА дешифрирует этот код и начинает выполнять соответствующую микропрограмму.

Наиболее просто в данном ОУ выполняются следующие микропрограммы.

1. Сложение дополнительных кодов **ADD**: $R^{дк}$:= $A^{дк}$+$B^{дк}$.
2. Вычитание дополнительных кодов **SUB**: $R^{дк}$:= $A^{дк}$-$B^{дк}$.
3. Прибавление 1 **INC**: *R*:= *B*+*1*.
4. Смена знака числа **ISG**: *R*:= -*B*.
5. Инвертирование (поразрядная операция) **NOT**: *R*:= NOT(*B*).

При выполнении всех микропрограмм исходные операнды поступают с *ШД*. Результат формируется на регистре результата *RGR*, а

в регистре *RGF* фиксируются признаки результата. Микрооперация *Y7* выводит содержимое регистра результатов через шинный формирователь *ШФ* на шину данных *ШД*.

В процессе выполнения арифметических команд управляющий автомат может сформировать управляющий сигнал *OF* (Overflow – переполнение). В последнем такте любой микропрограммы УА формирует сигнал *RESULT*. Сигналы *OF* и *RESULT* предназначены для внешних устройств.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

2.1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Целью проведения лабораторных работ является закрепление теоретического материала на основе программного моделирования работы основных видов операционных элементов, а также законченного операционного автомата.

Программы, разрабатываемые на лабораторных занятиях должны обладать общепринятыми элементами интерфейса: системным и контекстным меню, другими элементами диалога.

Для моделирования работы операционных элементов рекомендуется использовать объектно-ориентированный подход. В программе операционный элемент должен быть представлен как объект, имеющий две категории свойств и методов. Первая категория должна представлять элемент на внешнем уровне, необходимом для взаимодействия с пользователем. Здесь свойства и методы объекта должны включать условное графическое обозначение элемента, диалоговые элементы для ввода исходной информации и управляющих сигналов. Вторая категория свойств и методов должна использоваться для моделирования работы элемента. В этой части свойства должны наиболее близко соответствовать полям и сигналам моделируемого элемента, а методы должны соответствовать микрооперациям, выполняемым элементом.

Моделирование операционных элементов можно строить с использованием процедурного подхода. В этом случае функциональная модель элемента должна представляться самостоятельной процедурой с параметрами, которые соответствуют сигналам на его выводах.

В любом случае при разработке концепции моделирующей программы необходимо учитывать, что основной задачей является разработка модели операционного элемента. Модель внешней среды и взаимодействие с ней как пользователя, так и модели элемента являются вспомогательными средствами, необходимыми для демонстрации результатов решения основной задачи. Поэтому приоритетной следует считать задачу построения эффективной модели элемента: модель должна быть по возможности наиболее простой и быстродействующей и, конечно, адекватной моделируемому объекту. Структуру данных следует прорабатывать исходя из таких приоритетов. Неудачно выбранная структура данных приводит к построению недостаточно эффективной модели. Если наиболее разумная с точки зрения эффективности модели структура данных приводит к усложнению задачи программирования взаимодействия с моделью внешней среды, то следует пойти на реализацию таких усложненных алгоритмов: главное эффективно решить основную задачу.

За основу концептуальной модели любого элемента целесообразно принять следующее положение: только модель элемента должна «принимать решение» о том, как следует реагировать на любое изменение любого входного сигнала. Поэтому при любом действии пользователя, изменяющем любой из входных сигналов элемента, необходимо вызывать модель элемента: даже в тех случаях, когда программист знает, что при данном действии пользователя состояние элемента не должно как-либо измениться.

Моделирование работы операционного автомата должно строиться с использованием моделей отдельных операционных элементов, разработанных в первой части лабораторного практикума.

При сдаче лабораторной работы студент должен продемонстрировать преподавателю работу действующей программы, произвести

ее тестирование, дать необходимые пояснения теоретического характера. Затем требуется прокомментировать разработку текста программы в той части, которая касается моделирования изучаемого объекта. При этом должны приводиться теоретические обоснования, подтверждающие правильность моделирования.

2.2. ЛАБОРАТОРНАЯ РАБОТА №1

Тема: «Моделирование работы триггеров».

Цель работы – закрепление знаний по теме «Триггеры» на основе разработки программы моделирования работы *RS*– и *D*-триггеров с различными видами синхронизации.

Задание на работу

Используя материал подраздела 1.1, разработать программу моделирования функций следующих видов триггеров:

– асинхронного *RS*-триггера;
– синхронного *RS*-триггера с синхронизацией уровнем;
– синхронного *RS*-триггера с синхронизацией фронтом;
– *D*-триггера с синхронизацией уровнем;
– *D*-триггера с синхронизацией фронтом.

Порядок проведения работ

В интерфейсной части программы необходимо разработать условные графические обозначения исследуемых элементов, диалоговые элементы для ручного ввода значений управляющих сигналов, элементы индикации состояния выходов элементов.

Программная модель каждого триггера должна быть представлена в виде процедуры с параметрами. В качестве параметров должны выступать переменные логического типа, соответствующие выводам триггера, то есть его входам и выходам. Например, в процедуре моделирования асинхронного *RS*-триггера параметрами должны быть переменные, соответствующие входам *R*, *S* и *C*, а также выходам *Q* и $\overline{Q}$.

При запуске программы должны быть определены состояния всех входных и выходных сигналов всех триггеров, изображенных на

форме. При этом выходные сигналы должны соответствовать входным и должны быть сформированы программной моделью. Дальнейшее поведение элементов должно полностью определяться состояниями входных сигналов.

Программа считается неразработанной, если она не протестирована на всех возможных вариантах переключения управляющих сигналов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются между собой синхронизация уровнем и по фронту?
2. К каким сигналам и к каким типам триггеров относится термин «запрещенная комбинация» и в чем состоит особенность поведения триггера в таких случаях?
3. В каких случаях возникают неустойчивые состояния выходов *RS*-триггера?
4. Какими наборами входных сигналов можно задать режим хранения для синхронного *RS*-триггера?
5. Как влияет изменение сигнала на *D*-входе *D*-триггера, если в это время на *C*-входе установлен активный уровень сигнала?
6. Какие типы триггеров допустимо и недопустимо использовать в качестве элементов памяти управляющих автоматов?

2.3. ЛАБОРАТОРНАЯ РАБОТА №2

Тема: «Моделирование работы реверсивного сдвигового регистра»

Цель работы – закрепление знаний по теме «Регистры» на основе разработки программы моделирования функционирования реверсивного сдвигового регистра.

Задание на работу

Используя материал подраздела 1.2, разработать программу моделирования функций сдвигового регистра КР155ИР13.

Порядок проведения работы

В интерфейсной части программы необходимо разработать условное графическое обозначение универсального сдвигового регистра, диалоговые элементы для ручного ввода значений управляющих сигналов, элементы индикации состояния выходов регистра.

После завершения разработки программы ее необходимо протестировать на всех возможных наборах управляющих и информацион-

ных сигналов, приведенных в табл. 1.4. При тестировании следует обращать внимание на правильность выполнения всех программируемых функций.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются операционные элементы хранения от комбинационных элементов?
2. Могут ли операционные элементы хранения выполнять микрооперации по обработке данных?
3. Как заполняются крайние освобождающиеся разряды разрядной сетки в микрооперациях сдвига?
4. Почему при сдвигах значения разрядов кода передаются только соседним разрядам и не распространяются на несколько разрядов?
5. Какой вид синхронизации допустимо (недопустимо) использовать в микрооперациях сдвига кодов и почему?

2.4. ЛАБОРАТОРНАЯ РАБОТА №3

Тема: «Моделирование работы реверсивного счетчика»

Цель работы – закрепление знаний по теме «Счетчики» на основе разработки программы для моделирования работы двоичного реверсивного счетчика К155ИЕ7.

Задание на работу

Используя материал подраздела 1.3, разработать программу моделирования функций двоичного реверсивного счетчика К155ИЕ7.

Порядок проведения работы

В интерфейсной части программы необходимо разработать условное графическое обозначение счетчика, диалоговые элементы для ручного ввода значений управляющих сигналов, элементы индикации состояния выходов счетчика.

После завершения разработки программы ее необходимо протестировать на всех возможных наборах управляющих сигналов, приведенных в табл. 1.6. Особенно тщательно следует экспериментально убедиться в правильности формирования выходных сигналов переноса и заема.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для каких целей могут использоваться выходы переполнения счетчика?
2. Каковы взаимные приоритеты управляющих сигналов счетчика?
3. Нарисовать на бумаге и продемонстрировать на модели временные диаграммы, объясняющие особенности переключения сигналов на выходах заема и переноса.
4. Как из нескольких экземпляров счетчика построить счетчик большей разрядности?
5. Имеется ли взаимная зависимость сигналов синхронизации микроопераций счета на сложение и счета на вычитание?
6. Какую последовательность управляющих сигналов надо подать на счетчик, чтобы произвести реверс, то есть изменить операцию счета на «противоположную»?
7. Какой код установится на счетчике, если в состоянии *CNT* = 0 инициировать микрооперацию счета на вычитание?
8. Может ли в счетчике переключиться сигнал на выходе переноса (заема) при загрузке кода? Если может, объяснить: в каких случаях это возможно, и в каком направлении может переключиться этот сигнал?

2.5. ЛАБОРАТОРНАЯ РАБОТА №4

Тема: «Моделирование работы комбинационных операционных элементов»

Цель работы – закрепление знаний по теме «Операционные элементы комбинационного типа» на основе разработки программы для моделирования работы комбинационных операционных элементов.

Варианты заданий по лабораторной работе №4 представлены в табл. 2.1.

Таблица 2.1

Вариант	Номера бригад	Комбинационные операционные элементы
1	1, 6, 11	8-разрядный сумматор
2	2, 7, 12	8-разрядный реверсивный сдвигатель на один разряд
3	3, 8	8-разрядный формирователь инверсного кода
4	4, 9	Мультиплексор двух 8-разрядных кодов
5	5, 10	8-разрядный шинный формирователь

Задание на работу

Используя материал разд. 1.4, разработать программу моделирования функций заданного комбинационного операционного элемента.

Порядок проведения работы

В интерфейсной части программы необходимо разработать условное графическое обозначение элемента, диалоговые формы для ручного ввода значений управляющих сигналов и формы элементов индикации состояния выходов.

После завершения разработки программы ее необходимо протестировать на правильность выполнения всех функций заданного элемента.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем принципиальное отличие комбинационных операционных элементов от элементов хранения?
2. Чем отличается сдвигатель от сдвигового регистра?
3. Для чего используются управляющие сигналы в комбинационных операционных элементах?
4. Чем отличаются выходы шинных формирователей от выходов других элементов как комбинационного, так и некомбинационного типов?
5. Можно ли сигнал переноса из старшего разряда сумматора отождествлять с сигналом переполнения разрядной сетки? Ответ проиллюстрировать примерами.
6. Как из нескольких экземпляров сумматоров построить сумматор большей разрядности? Как изменится быстродействие составного сумматора по сравнению с исходным сумматором?
7. Как из нескольких сдвигателей построить сдвигатель большей разрядности? Как изменится быстродействие составного сдвигателя по сравнению с исходным сдвигателем?
8. Нарисовать структуру мультиплексора двух 4-разрядных кодов.
9. Нарисовать структуру простейшего шинного формирователя.

2.6. ЛАБОРАТОРНАЯ РАБОТА №5

Тема: «Моделирование работы операционного автомата».

Цель работы – закрепление знаний по теме «Операционные автоматы» на основе моделирования работы операционного автомата по заданной структурной схеме собственными программными средствами.

Варианты заданий по лабораторной работе №5 представлены в табл. 2.2.

Таблица 2.2

Вариант	Команда
1	Сложение. На ЩД данные представляются в прямом коде
2	Сложение. На ЩД данные представляются в обратном коде
3	Вычитание. На ЩД данные представляются в прямом коде
4	Вычитание. На ЩД данные представляются в обратном коде
5	Прибавление 1. На ЩД данные представляются в прямом коде
6	Прибавление 1. На ЩД данные представляются в обратном коде
7	Вычитание 1. На ЩД данные представляются в прямом коде
8	Вычитание 1. На ЩД данные представляются в обратном коде
9	Смена знака числа. На ЩД данные представляются в прямом коде
10	Смена знака числа. На ЩД данные представляются в обратном коде
11	Инвертирование (поразрядная логическая операция)

Задание на работу

Используя материал из подраздела 1.5, разработать микропрограмму работы операционного автомата для заданной команды, а также программу для моделирования работы операционного автомата, выполняющего эту микропрограмму.

Порядок проведения работы

В интерфейсной части программы на графической форме необходимо подготовить изображение структурной схемы ОА. На форме следует также разместить диалоговые элементы для ручного ввода двоичных кодов на ЩД, а также элементы индикации состояния управляющих сигналов и кодов на всех внутренних шинах ОА.

Считать, что исходные данные попадают на ШД из оперативной памяти. Поэтому при необходимости запоминания промежуточных кодов принять, что код, выведенный на ШД из ОА, запоминается в оперативной памяти. Этот факт в процессе выполнения микропрограммы фиксируется путем записи сформированного кода на листке бумаги. В дальнейшем этот записанный код задается на ШД, имитируя считывание кода из оперативной памяти.

В программе каждый операционный элемент структурной схемы должен быть представлен своей программной моделью.

Для выполнения моделирования необходимо разработать микропрограмму выполнения заданной команды на рассматриваемом ОА. Затем микропрограмму (управляющие сигналы и логические условия) следует закодировать и записать в текстовый файл. При запуске созданная программа должна последовательно считывать строки микропрограммы из файла и интерпретировать их как управляющие сигналы.

После завершения разработки программы необходимо исследовать выполнение микропрограммы заданной команды в пошаговом и автоматическом режимах с использованием заранее подготовленных тестовых примеров.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются микрокоманды от микроопераций?
2. Как производится загрузка операндов выполняемой команды в исследуемом автомате?
3. Как реализуется операция вычитания чисел с помощью сумматора? Ответ проиллюстрировать примерами.
4. Как формируется признак переполнения разрядной сетки? Ответ проиллюстрировать примерами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Таненбаум Э. Архитектура компьютера. 5-е изд. – СПб.: Питер, 2007.
2. Догадин Н.Б. Архитектура компьютера: Учеб. пособие. – М.: Бином, 2008.
3. Колдаев В.Д., Лупин С.А. Архитектура ЭВМ: Учеб. пособие. – М.: ИНФРА-М, 2009.
4. Буза М. Архитектура компьютера. – Минск: НОВОЕ ЗНАНИЕ, 2006.
5. Тихонов В.А., Баранов А В. Организация ЭВМ и систем. – М.: София, 2008.

СОДЕРЖАНИЕ

Учебное издание

Теория проектирования ЭВМ

ПУГАЧЕВ Анатолий Иванович
МАРТЕМЬЯНОВ Борис Викторович

Редактор *Ю.А. Петропольская*
Верстка *И.О. Миняева*
Выпускающий редактор *Н.В. Беганова*

Подписано в печать 16.02.10.
Формат 60×84 1/16. Бумага офсетная.
Усл. п. л. 2,79. Уч.-изд. л. 2,76.
Тираж 50 экз. Рег. № 400/09.

Государственное образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»
443100, г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
443100, г. Самара, ул. Молодогвардейская, 244. Корпус №8