МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

А.В. ХЛУДЕНЕВ

ОПЕРАЦИОННОЕ УСТРОЙСТВО

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения высшего профессионального
образования «Оренбургский государственный университет»

Оренбург 2003

ББК 32.97 я 73

Х60
УДК 681.32 (075.8)

Рецензент д.т.н., доцент Булатов В.Н.

Х60 **Хлуденев А.В. Операционное устройство: Методические указания. – Оренбург: ГОУ ОГУ, 2003. - 48 с.**

Методические указания содержат рекомендации по выполнению и защите курсового проекта по дисциплине «Теория автоматов».

Методические указания предназначены для студентов заочной формы обучения специальности 220100.

ББК 32.97 я 73

Введение

Целью выполнения курсового проекта является закрепление теоретических знаний в области теории автоматов, установление междисциплинарных связей с теорией алгоритмов, получение практических навыков самостоятель-ного решения инженерных задач, связанных с построением автоматных моделей вычислительных устройств, развитие творческих способностей.

Над курсовым проектом студенты работают под руководством преподавателя в часы, отводимые расписанием для курсового проектирования, и самостоятельно в свободное от занятий время. Основная роль отводится самостоятельной работе студентов. Возникающие в процессе работы над проектом вопросы студенты могут решать в часы консультаций с руководителем проекта. Планируемое время выполнения проекта составляет 20 часов.

Для контроля за ходом курсового проектирования назначаются три контрольных срока, следующих с интервалом в четыре недели. В течение каждого интервала необходимо выполнить объем работ, относящихся к одному разделу проекта. Готовый проект студент сдает на проверку руководителю не менее чем за 3 дня до защиты. Без предварительной проверки проекты к защите не допускаются. Руководитель в течение 1-2 дней проверяет проект и возвращает его студенту с замечаниями, которые студент должен устранить, или подписанным, если проект допущен к защите.

При защите курсового проекта студент должен сделать короткий (3-5 минут) доклад по существу проекта, осветив наиболее важные и принципиальные его стороны, и ответить на вопросы. При оценке курсового проекта принимается во внимание:

- знания, умения и навыки студента в области теории автоматов;
- качество и достоверность полученных проектных решений;
- степень самостоятельности при выполнении проекта.

В случае получения неудовлетворительной оценки срок повторной защиты назначается с учетом необходимости дополнительной подготовки студента.

1 Задание на курсовой проект

Задание на курсовой проект содержит словесное описание функционального назначения объекта проектирования, перечень технических требований к его основным параметрам и перечень проектно-конструкторской документации, подлежащей разработке.

Темой курсового проекта является разработка автомата (операционного устройства В.М. Глушкова), реализующего алгоритм выполнения заданной вычислительной операции. Варианты заданий включают различные алгоритмы выполнения операций умножения и деления с фиксированной точкой. Исходные данные для выполнения проекта содержат тип операции, формат

представления данных, алгоритм операции, форму реализации управляющего автомата, тип элементов памяти.

Результаты выполненных проектных работ должны быть представлены в форме пояснительной записки и графической части, содержащей функциональные электрические схемы операционного и управляющего автоматов. Общие требования к оформлению пояснительной записки и графической части курсового проекта изложены в стандарте СТП 101-00 /1/. Краткое содержание требований стандартов к выполнению электрических схем можно найти в /2/. Ус-ловные обозначения элементов цифровой техники определяет ГОСТ 2.743-91.

Основные разделы проекта:

- разработка микропрограммы операции;
- построение схемы операционного автомата (ОА) на уровне регистровых передач;
- построение схемы управляющего автомата (УА) на вентильном уровне;

При проектировании возможны ошибки, поэтому рекомендуется проверять результаты синтеза, выполняя тестирование их имитационных моделей.

Методику выполнения курсового проекта будем рассматривать для задания, которое приведено в приложении А.

2 Методические указания по разработке микропрограммы операции

2.1 Алгоритмы умножения

Существуют четыре основные алгоритмы умножения, и им соответствуют показанные на рисунке 1 схемы. Каждая из схем состоит из трёх основных узлов: регистра сдвигателя множителя *B* (RgB), регистра или регистра-сдвигателя множимого *A* (RgA) и накапливающего сумматора или накапливающего сумматора-сдвигателя (SM). На рисунке показаны только значащие разряды перечисленных узлов (*1*- старший разряд, *n*- младший разряд).

На первой (рисунок 1, а) из четырёх схем умножение начинается с младших разрядов множителя. Процесс умножения имеет циклический характер. Во время *i*-го цикла (*i* =1, 2, ..., *n*) из регистра-сдвигателя RgB выдвигается очередная цифра множителя b_{n-i+1} , которая управляет передачей в сумматор-сдвигатель SM либо числа 0 (если цифра множителя равна 0), либо множимого (если цифра множителя равна 1). После прибавления очередного частного произведения производится одновременный сдвиг содержимого RgB и SM. Младший разряд SM вдвигается в освобождающийся разряд RgB, поскольку этот разряд в дальнейшем суммировании частичных произведений участвовать не будет. После проведения *n* циклов (сложений и сдвигов) в SM

будут находиться старшие *n* разрядов произведения $S = A \cdot B$, а в RgB - младшие *n* разрядов.

На второй схеме (рисунок 1, б) умножение тоже начинается с младших разрядов множителя, но сдвигается не сумма частичных произведений, а множимое.

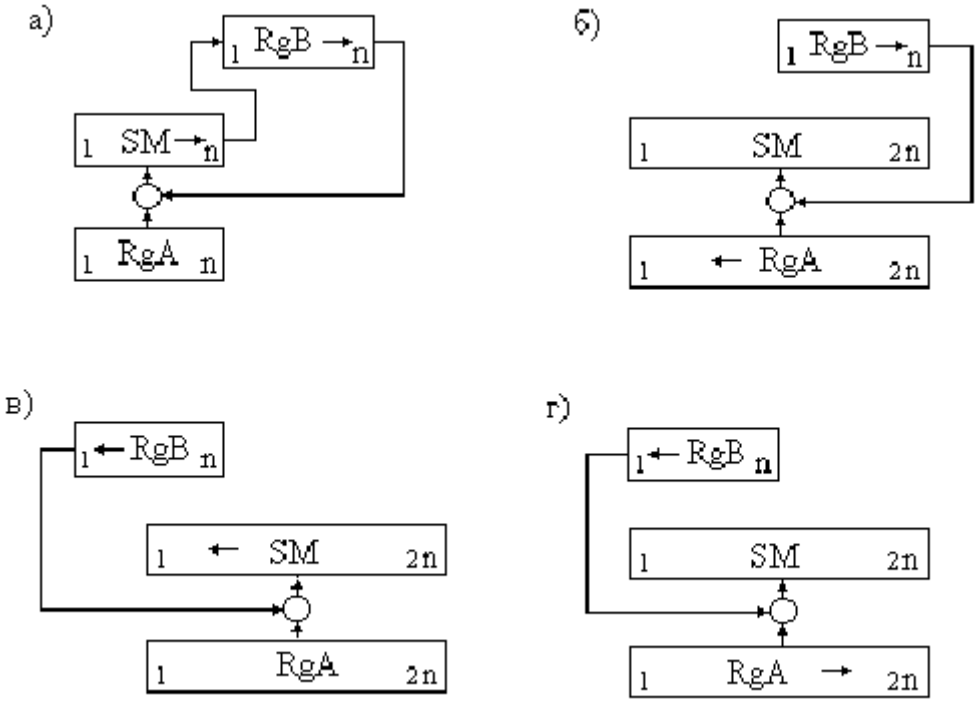


Рисунок 1 - Основные схемы умножения двоичных чисел

На третьей и четвёртой схемах (рисунок 1, в, г) соответственно умножение идёт со старших разрядов и поэтому множитель сдвигается влево.

Сомножители со знаком могут быть представлены в прямом коде (ПК), дополнительном коде (ДК) или обратном коде (ОК). Частные произведения также могут находиться в ПК, ДК или ОК, что требует соответствующей организации сумматора.

При выполнении операции умножения в ПК, суммирование частичных произведений проводится без учёта знаковых разрядов сомножителей. Знак произведения формируется отдельно как сумма по модулю 2 знаковых разрядов сомножителей: $SgS = SgA \wedge SgB$.

Сумматор, складывающий частичные произведения, должен быть *(n+1)*-разрядным, так как при суммировании может возникнуть единица переноса из старшего разряда, которую на следующем шаге необходимо вдвинуть в регистр-накопитель. Произведение прямых кодов сомножителей - есть прямой код результата.

Условимся обозначать *A(m : n)* - слово *A*, или его подслово (поле), где *m* - номер старшего разряда слова или поля, *n* - номер младшего разряда слова или поля; *A(i)* - *i*-й разряд слова *A*;

Пример 1 - Умножить в ПК числа $IA_{ПК} = 0.1101$ и $IB_{ПК} = 1.1011$.

Составим спецификацию слов:

- *IA(0:4)* – множимое (входное слово);
- *IB(0:4)* – множитель (входное слово);
- *OS(0:8)* – произведение (выходное слово);
- *A(0:4)* – содержимое RgA (внутреннее слово);
- *B(0:4)* – содержимое RgB (внутреннее слово);
- *S(0:4)* – содержимое SM (внутреннее слово).

Последовательность действий в процессе выполнения операции умножения представлена в таблице 1. Условные обозначение действий:

- присваивание « := »;
- конкатенация (соединение) слов « . »;
- сложение « + »;
- сдвиг слова на один разряд вправо R1;
- сложение по модулю 2 « ^ » .

Сумматор прямых кодов и регистр-накопитель RgS имеют пять разрядов, так как сложение выполняется без учета знака множимого.

После выполнения четырех сдвигов знак множителя сместится в разряд *B(4),* поэтому знаковый разряд выходного слова *OS(0):=A(0)^B(4)),* старшие значащие разряды выходного слова формируются из поля слова *S(1:4),* а младшие разряды из *B(0:3).* Ответ *OS*пк=1.10001111.

Таблица 1

S	B	Комментарий
0.0000	11011	S := 0; B := IB; A := IA;
+ 0.1101		
0.1101		B(4) = 1; S := S + A; B := R1(S(4).B); S := R1(0.S)
0.0110	→1 1101	B(4)=1;
+ 0.1101		
1.0011		S := S + A; B := R1(S(4).B); S := R1(0.S)
0.1001	→11 110	B(4)=0; B := R1(S(4).B); S := R1(0.S)
0.0100	→111 11	B(4)=1;
+ 0.1101		S := S + A; B := R1(S(4).B); S := R1(0.S)
1.0001		
0.1000	→1111 1	OS := (A(0)^ B(4)).S(1:4).B(0:3)
		$OS_{ПК}$=1.10001111

При выполнении умножения в ДК произведение ДК сомножителей равно ДК результата только в случае положительного множителя. Умножение на сумматоре ДК заключается в анализе разрядов множителя и при $b_i = 1$ в прибавлении ДК множимого к содержимому сумматора накопителя.

Внутренние слова *A* и *S* должны быть представлены в модифицированном коде, а сумматор должен быть *(n+2)*-разрядным. Также можно использовать *(n+1)*- разрядный сумматор с флагом переноса.

При умножении в ДК и ОК необходимо выполнять модифицированные сдвиги (таблица 2).

Таблица 2

Сдвинутая влево на один разряд	Исходная комбинация	Сдвинутая вправо на один разряд
$0\ a_1.\ a_2\ a_3 \dots a_n\ 0$	$0\ 0.\ a_1\ a_2\ a_3 \dots a_n$	$0\ 0.\ 0\ a_1\ a_2\ a_3 \dots a_{n-1}$
$1\ a_1.\ a_2\ a_3 \dots a_n\ 0$	$0\ 1.\ a_1\ a_2\ a_3 \dots a_n$	$0\ 0.\ 1\ a_1\ a_2\ a_3 \dots a_{n-1}$
$0\ a_1.\ a_2\ a_3 \dots a_n\ \alpha$	$1\ 0.\ a_1\ a_2\ a_3 \dots a_n$	$1\ 1.\ 0\ a_1\ a_2\ a_3 \dots a_{n-1}$
$1\ a_1.\ a_2\ a_3 \dots a_n\ \alpha$	$1\ 1.\ a_1\ a_2\ a_3 \dots a_n$	$1\ 1.\ 1\ a_1\ a_2\ a_3 \dots a_{n-1}$

Значение α выбирается по правилу

$$\alpha = \begin{cases} 0, \text{ если информация в ДК;} \\ 1, \text{ если информация в ОК.} \end{cases} \qquad (1)$$

Поскольку на ДК складываются числа со знаком, то знак произведения получается автоматически после завершения операции умножения.

Пример 2 - Умножить в ДК числа $IA_{ДК} = 11.0011$ и $IB_{ДК} = 00.1011$.

Составим спецификацию слов:

- *IA(0:5)* – множимое (входное слово);
- *IB(0:5)* – множитель (входное слово);
- *OS(0:9)* – произведение (выходное слово);
- *A(0:5)* – содержимое RgA (внутреннее слово);
- *B(0:5)* – содержимое RgB (внутреннее слово);
- *S(0:5)* – содержимое SM (внутреннее слово).

Последовательность действий над числами представлена в таблице 3. Ответ: $OS_{ДК} = 11.01110001$; $OS_{ПК} = 11.10001111$.

Таблица 3

S	B	Комментарий
00.0000	001011	S := 0; B := IB; A := IA;
+ 11.0011		
11.0011		B(5)=1; S := S+A; B:= R1(S(5).B); S := R1(0.S)
11.1001	→1 00101	B(5)=1;
+ 11.0011		
10.1100		S := S+A; B:= R1(S(5).B); S := R1(0.S)
11.0110	→01 0010	B(5)=0;
11.1011	→001 001	B := R1(S(5).B); S := R1(0.S)
+ 11.0011		
10.1110		B(5)=1; S := S+A; B:= R1(S(5).B); S := R1(0.S)
11.0111	→0001 00	OS := S(0:5).B(0:3)
		$OS_{ДК}$ = 11.01110001

Если множитель отрицательный, то произведение чисел в ДК получается прибавлением поправки к *n* старшим разрядам произведения дополнительных кодов сомножителей. Поправка получается инвертированием вместе со знаковыми разрядами множимого, представленного в ДК, плюс единица к младшему разряду.

Пример 3 - Умножить в ДК числа $IA_{ДК}$ = 11.0011 и $IB_{ДК}$ = 11.0101.

Последовательность действий над числами представлена в таблице 4.

Ответ: $OS_{ДК} = OS_{ПК}$ = 00.10001111.

Таблица 2.4

S	B	Комментарий
00.0000	110101	S :=0; B :=IB; A :=IA;
+ 11.0011		
11.0011		B(5)=1; S: = S+A; B:= R1(S(5).B); S:= R1(0.S)
11.1001	→1 11010	B(5)=0;
11.1100	→11 1101	B := R1(S(5).B); S := R1(0.S)
+ 11.0011		B(5)=1;
10.1111		S :=S+A; B := R1(S(5).B); S:= R1(0.S)
11.0111	→111 110	B(5)=0;
11.1011	→1111 11	B := R1(S(5).B); S := R1(0.S)
+ 00.1101		B(5)=1; S := S+ $\overline{A}$ +1;
00.1000	1111	OS := S(0:5).B(0:3)
		OS = 00.10001111

При выполнении операции умножения в ОК произведение ОК сомножителей равно ОК результата только в случае положительного множителя. Умножение в ОК также заключается в анализе разрядов множителя. Если оказывается, что очередной разряд множителя равен единице, то к содержимому сумматора-накопителя добавляется ОК множимого.

Первая схема (рисунок 1, а) является самой неудобной для умножения в ОК. До начала операции умножения регистр-накопитель необходимо обнулить. Однако при представлении чисел в ОК имеется двузначность нуля, и для операции умножения, проводимой в ОК по этой схеме, оказывается небезразличным, какой из нулей, +0 или -0 записывается в регистр-накопитель до начала операции. Правило выбора нуля следующее:

- если $SgA \oplus SgB = 0$, то в регистр-накопитель записывается +0;
- если $SgA \oplus SgB = 1$, то в регистр-накопитель записывается -0.

Несоблюдение этого правила может привести к возникновению положительной погрешности, которая будет находиться в n младших разрядах полного произведения.

Пример 4 – Умножить в ОК числа $IA_{ОК}$ = 11.0010 и $IB_{ОК}$ = 00.1011. Последовательность действий, производимых над числами, представлена в таблице 5. Ответ: $OS_{ОК}$ = 11.01110000; $OS_{ПК}$ = 11.10001111.

Таблица 5

S	B	Комментарий
11.1111 + 11.0010 11.0001 + 1 11.0010	001011	S :="- 0"; B=IB; A :=IA; B(5)=1; S := S + A;
11.1001 + 11.0010 10.1011 + 1 10.1100	→0 00101	B:= R1(S(5).B); S := R1(0.S) B(5)=1; S := S + A;
11.0110	→00 0010	B:= R1(S(5).B); S := R1(0.S) B(5)=0;
11.1011 + 11.0010 10.1101 + 1 10.1110	→000 001	B:= R1(S(5).B); S := R1(0.S) B(5)=1; S := S + A; B:= R1(S(5).B); S := R1(0.S)
11.0111	→0000 00	OS := S(0:5).B(0:3) OS = 11.01110000

Если множитель отрицательный, то произведение чисел в ОК получается прибавлением поправок *П1*, *П2* и *П3* к произведению ненулевых ОК сомножителей.

Поправка *П1* представляет собой множимое *A*, представленное в ОК и сдвинутое на *n* разрядов вправо. Иными словами, поправка *П1* прибавляется к *n* младшим разрядам полного произведения:

$$П1 = A_{ОК} \cdot 2^{-n}. \qquad (2)$$

Поправка *П2* представляет собой число $\pm (2^n - 1) \cdot 2^{-n}$. Ее необходимо учитывать только для схемы (рисунок 1, а). Это число должно быть представлено в ОК и сдвинуто на *n* разрядов вправо, т.е. поправка *П2* также прибавляется к младшим разрядам произведения. Знак поправки *П2* выбирается в зависимости от знака множимого *A* в соответствии со следующим правилом

$$П2 = \begin{cases} 00.11...1 \cdot 2^{-n}, & если \quad A > 0, \\ 11.00...0 \cdot 2^{-n}, & если \quad A < 0. \end{cases} \qquad (3)$$

Поправка *П3* представляет собой множимое $A_{ОК}$, у которого проинвертированы все разряды, включая знаковые:

$$П3 = \overline{A}_{ОК}. \qquad (4)$$

Поправка *П3* прибавляется к *n* старшим разрядам произведения.

Для схем (рисунок 1, а и б) поправку *П1* и поправку *П2* (если она необходима) удобнее прибавлять к содержимому регистра-накопителя до начала выполнения анализа разрядов множителя и сдвигов. После проведения *n* сдвигов поправки *П1* и *П2* окажутся в младших разрядах произведения. Поправку *П3* удобнее прибавить после завершения операции умножения - после проверки *n* сложений и *n* сдвигов. Для схем (рисунок 1, в и г) последовательность прибавления поправок должна быть противоположная.

Кроме этого, для схемы (рисунок 1, а) при отрицательных ($SgA = SgB = 1$) сомножителях, имеющих такое абсолютное значение, что все значащие разряды произведения располагаются в *n* младших разрядах, нужно учитывать, что если после прибавления *П3* в старших разрядах получается -0, его необходимо преобразовать в +0. Иначе результат умножения будет ошибочным. Например, если $IA_{ОК}$ = 11**.**1101; $IB_{ОК}$ = 11**.**1101 и операция преобразования - 0 в +0 не была проведена, будем иметь следующее ошибочное произведение: $OS^{*}_{ОК}$ = 11**.**11110100. Правильный результат: $OS_{ОК}$ = 00**.**00000100.

При выполнении всех перечисленных правил произведение ОК сомножителей даст ОК результата.

Пример 5 - Умножить в ОК числа $IA_{ОК}$ = 11**.**0010 и $IB_{ОК}$= 11**.**0100. Последовательность действий, производимых над числами, показана в таблице 6. Ответ: $OS_{ОК} = OS_{ПК}$ = 00**.**10001111.

2.2 Алгоритмы деления

К наиболее распространённым методам деления, для которых время выполнения операции не зависит от величины исходных чисел *A* и *B*, а определяется только лишь разрядностью *n*, относятся *деление с восстановлением остатка* и *деление без восстановления остатка*. Для этих методов *Q* содержит старшие *n* разрядов точного частного *A / B*.

Операция деления, как и операция умножения, выполняется на сумматорах, и для вычисления разности *A - B* требуется операция вычитания (сложение *A* и *B*, имеющих разные знаки), поэтому в основе ОА должен лежать двоичный сумматор обратного кода или сумматор дополнительного кода.

Таблица 6

S	B	Комментарий
00.0000	110100	S:="+0"; B:=IB; A:=IA(0).IA;
+ 11.0010		S:= S + П1;
11.0010		
+ 11.0000		S:= S + П2;
10.0010		
+ 1		
10.0011	110100	
		B(5)=0;
11.0001	→1 11010	B:= R1(S(5).B); S := R1(0.S)
		B(5)=0;
11.1000	→11 1101	B:= R1(S(5).B); S := R1(0.S)
+ 11.0010		B(5)=1;
10.1010		S:= S + A;
+ 1		
10.1011		
11.0101	→111 110	B:= R1(S(5).B); S := R1(0.S)
11.1010	→1111 11	B(5)=0;
+ 00.1101		B:= R1(S(5).B); S := R1(0.S)
00.0111		S := S + П3;
+ 1		
00.1000	1111	OS := S(0:5).B(0:3) OS = 00.10001111

Если операнды представлены в ПК, то знак частного Q определяется сложением по модулю 2 знаков операндов, которые в определении цифровых разрядов частного не используются. То есть, A и B в процессе деления считаются положительными числами.

Если операнды представлены в ОК или ДК, то их можно предварительно преобразовать в ПК или выполнять деление в исходном коде, учитывая при определении разрядов частного знаки делимого и делителя.

Для операции деления важно точно определить положение точки в формате чисел, так как правила выравнивания делимого и делителя для целых чисел и правильных дробей отличаются. Здесь будем считать, что операнды являются правильными дробями. Тогда необходимым условием возможности выполнения операции деления является выполнение неравенства:

$$|A| < |B|. \qquad (5)$$

Если $|A| \geq |B|$, то есть $|A/B| \geq 1$, то должен вырабатываться признак переполнения.

Пронумеруем разряды частного $Q = q_0 , q_1 , q_2 , ... q_n$. Само деление представляет собой циклический процесс; на *i*-м цикле определяется очередная двоичная цифра частного q_i. Общее количество циклов деления равно $n+1$. Начальный (нулевой) цикл отличается от других, здесь выполненяется проверка неравенства (5) путем вычисления разности $|A| - |B|$. Инверсию знака полученной разности используют для формирования:

- признака (флага) переполнения *OVFL*, при делении чисел в ПК;
- значения разряда переполнения при делении чисел в ДК или ОК, представленных модифицированными кодами. На последующих циклах определяются значения цифровых разрядов. Реализация алгоритмов деления будет более простая, если результат деления, состоящий из признака или разряда переполнения и цифровых разрядов частного, будет формироваться микрооперацией сдвига слево частного $Q:=L1(Q.q_i), i = 0, ..., n$.

Так как операция деления относится к операциям, не всегда дающим точный результат, признаком окончания операции может быть:

- достижение заданной точности;
- заполнение разрядной сетки частного (признаком окончания операции деления является число сдвигов очередного частного остатка, равное количеству разрядов частного);
- получение в процессе деления остатка $A_i = 0$. В этом случае операция может быть остановлена и в оставшиеся разряды частного записывается нуль.

2.2.1 Деление с восстановлением остатка

На рисунке 2.1 показан алгоритм одного *(i+1)*-го цикла деления положительных чисел с восстановлением остатка.

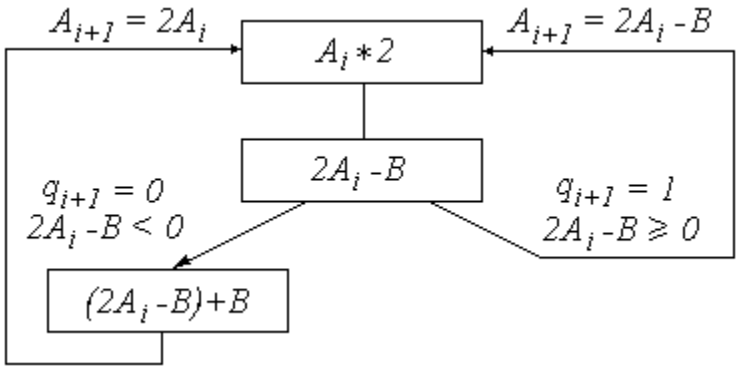


Рисунок 2 - Алгоритм выполнения *(i+1)*-го цикла деления с восстановлением остатка

Здесь *(i+1)*-й цикл начинается со сдвига на один разряд влево (удвоения) остатка A_i , полученного во время предыдущего *i*-го цикла. После этого вычисляется разность между удвоенным остатком и делителем. Если она неотрицательна, то в очередной разряд частного записывается цифра q_{i+1} =1, а

эта разность является новым остатком A_{i+1}. Если разность между $2A_i$ и делителем отрицательна, то, как видно из рисунка, в частное записывается q_{i+1} = 0, а остаток $(2A_i - B)$ восстанавливается $(2A_i - B)+B$ и воспринимается как новый остаток A_{i+1}. Таким образом, во время одного цикла производится одна или две операции сложения-вычитания.

Если используется сумматор ОК, то при определении значений разрядов частного необходимо учитывать неоднозначность представления нуля (машинное представление нуля может иметь во всех разрядах или нули, или единицы).

Пример 6 - На сумматоре ОК разделить IA = -9/16 на IB = -12/16 представленные в ПК. Разрядность операндов n = 4. Для этого и всех последующих примеров будем считать, что входные слова сохраняются в ячейках памяти операционного автомата в виде внутренних слов A и B.

Машинные изображения делимого и делителя: $A_{ПК}$ = 1.1001; $B_{ПК}$ = 1.1100. Определим знак частного: $SgQ = SgA \wedge SgB = 1 \oplus 1 = 0$. В процессе выполнения деления A и B считаем положительными, их представление в прямом и обратном кодах совпадает $A_{ОК}$ = 0.1001; $B_{ОК}$ = 0.1100.

Для выполнения операции вычитания потребуется отрицательное значение делителя B: $-B_{ОК}$ = 1.0011.

Проверим, выполняется ли условие (5) и затем определим цифру за цифрой разряды частного:

$A_{ОК}$ = 0.1001
+
$-B_{ОК}$ = 1.0011

$S_{ОК}$ = 1.1100 $|A| < |B|$, $q_0 = 0$ – переполнения нет ($OVFL=0$)
+
$B_{ОК}$ = 0.1100 Восстановление

$A_{ОК}$ = 0.1001
$2A_{ОК}$ = 1.0010
+
$-B_{ОК}$ = 1.0011

$A_{1ОК}$ = 0.0110 $q_1 = 1$
$2A_{1ОК}$ = 0.1100
+
$-B_{ОК}$ = 1.0011

$A_{2ОК}$ = 1.1111 $q_2 = 1$, так как $A_2=0$
$2A_{2ОК}$ = 1.1111
+
$-B_{ОК}$ = 1.0011

$A_{3ОК}$ = 1.0011 $q_3 = 0$

+

$B_{ОК}$ = 0.1100 Восстановление

$A_{3ОК}$ = 1.1111

$2A_{3ОК}$ = 1.1111

+

$-B_{ОК}$ = 1.0011

$A_{4ОК}$ = 1.0011 $q_4 = 0$

+

$B_{ОК}$ = 0.1100 Восстановление

$A_{3ОК}$ = 1.1111

Ответ формируем из значений SgQ и q_1, q_2, q_3, q_4 : $Q_{ПК}$ = 0.1100, Q=3/4.

Следует отметить, что хотя операция деления, выполняемая согласно алгоритму (рисунок 2), проводится либо на сумматоре ОК, либо на сумматоре ДК, частное получается в ПК.

Если делимое и делитель представлены в ОК или ДК, то деление можно выполнять с учетом их знаков. Правила выполнения алгоритма изменяются следующим образом:

- на каждом шаге выполняется проверка условия $|2A_i| - |B| \geq 0$, для этого при отрицательном частном ($SgQ=1$) вместо вычитания делителя выполняется сложение, соответственно восстановление остатка выполняется вычитанием делителя;

- при $|2A_i| - |B| \geq 0$ очередная цифра частного $q_i = \bar{S}gQ$, при $|2A_i| - |B| < 0$ $q_i = SgQ$.

Проверка условия $|2A_i| - |B| \geq 0$ отличается от проверки условия $2A_i - B \geq 0$. Здесь уже недостаточно анализировать знаковый разряд разности (а в случае ОК значение разности «минус 0»). Очевидно, что $|2A_i| - |B| > 0$, если знак разности совпадает со знаком делимого, то есть $Sg(2A_i - B) \wedge SgA = 0$. Значение делимого в процессе выполнения операции, как правило, не сохраняется, поэтому придется воспользоваться значениями SgQ и SgB следующим образом

$$SgQ \wedge SgB = SgA \wedge SgB \wedge SgB = SgA . \quad (6)$$

Условию $|2A_i| - |B| \geq 0$ соответствует логическая функция

$$\overline{(Sg(2A_i - B) \wedge SgQ \wedge SgB) \vee (2A_i - B = 0)}. \quad (7)$$

Конечно, можно выполнять проверку условия $|2A_i| - |B| < 0$, которому соответствует логическая функция

$$(Sg(2A_i - B) \wedge SgQ \wedge SgB) \,\&\, (2A_i - B \neq 0). \quad (8)$$

Результат деления получается в ОК. Если требуется частное, представленное в ДК, то к младшему разряду регистра, содержащего отрицательное частное, нужно прибавить единицу. Результат операции в модифицированном ОК или ДК проще всего формировать путем записи в младший разряд регистра частного значения *SgQ* с последующим выполнением *n+1* микроопераций сдвига влево $Q:=L1(Q.q_i)$ на каждом шаге алгоритма.

Пример 7 - На сумматоре ОК разделить *IA* = 9/16 на *IB* = -12/16 представленные в ОК. Разрядность операндов *n* = 4.

Машинные изображения делимого и делителя: $A_{ОК}$ = 00.1001; $B_{ОК}$ = 11.0011 . Для выполнения операции вычитания потребуется отрицательное значение делителя *B*: $-B_{ОК}$ = 00.0011.

Определим знак частного: *SgQ = SgA ^ SgB = 0 ^ 1 = 1* .

Проверим, выполняется ли условие (5.1) и затем определим цифру за цифрой разряды частного:

$A_{ОК}$ = 00.1001
+
$B_{ОК}$ = 11.0011

$S_{ОК}$ = 11.1100 $|A| < |B|$ $q_0 = 1$, так как $SgQ = 1$
+
$-B_{ОК}$ = 00.1100 Восстановление

$A_{ОК}$ = 00.1001
$2A_{ОК}$ = 01.0010
+
$B_{ОК}$ = 11.0011

$A_{1ОК}$ = 00.0110 $q_1 = 0$
$2A_{1ОК}$ = 00.1100
+
$B_{ОК}$ = 11.0011

$A_{2ОК}$ = 11.1111 $q_2 = 0$, так как $A_2 = 0$
$2A_{2ОК}$ = 11.1111
+
$B_{ОК}$ = 11.0011

$A_{3ОК}$ = 11.0011 $q_3 = 1$
+
$-B_{ОК}$ = 00.1100 Восстановление

$A_{3ОК}$ = 11.1111
$2A_{3ОК}$ = 11.1111
+
$B_{ОК}$ = 11.0011

$A_{4ОК}$ = 11.0011 $q_4 = 1$

+

$-B_{ОК}$ = 00.1100 Восстановление

$A_{4\,ОК}$ = 11.1111

Ответ: $Q_{ОК}$ = 11.0011 , Q= -3/4.

Пример 8 - На сумматоре ДК разделить *IA* = -9/64 на *IB* = 12/64 представленные в ДК. Разрядность операндов *n* = 4.

Машинные изображения делимого и делителя: $A_{ДК}$ = 11.0111; $B_{ДК}$ = 00.1100 . Для выполнения операции вычитания потребуется отрицательное значение делителя *B*: $-B_{ДК}$ = 11.0100 .

Определим знак частного: *SgQ = SgA ^ SgB = 0 ^ 1 = 1.*

Проверим, выполняется ли условие (5) и затем определим цифру за цифрой разряды частного:

$A_{ДК}$ = 11.0111

+

$B_{ДК}$ = 00.1100

$S_{ДК}$ = 00.0011 $|A| < |B|$ $q_0 = 1$, так как $SgQ = 1$

+

$-B_{ДК}$ = 11.0100 Восстановление

$A_{ДК}$ = 11.0111

$2A_{ДК}$ = 10.1110

+

$B_{ДК}$ = 00.1100

$A_{1\,ДК}$ = 11.1010 $q_1 = 0$

$2A_{1ДК}$ = 11.0100

+

$B_{ДК}$ = 00.1100

$A_{2\,ДК}$ = 00.0000 $q_2 = 0$, так как $A_2 = 0$

$2A_{2ДК}$ = 00.0000

+

$B_{ДК}$ = 00.1100

$A_{3\,ДК}$ = 00.1100 $q_3 = 1$

+

$-B_{ДК}$ = 11.0100 Восстановление

$A_{3ДК}$ = 00.0000

$2A_{3ДК}$ = 00.0000

+

$B_{ДК}$ = 00.1100

$A_{4\,ДК} = 00.1100$ $\quad q_4 = 1$

\+

$-B_{ДК} = 11.0100$ Восстановление

$A_{4\,ДК} = 00.0000$

Ответ: $Q_{ОК} = 11.0011$, $Q_{ДК} = 11.0100$, Q= -3/4.

2.2.2 Деление чисел без восстановления остатка

Алгоритм выполнения одного $(i+1)$-го цикла деления положительных чисел без восстановления остатка показан на рисунке 3.

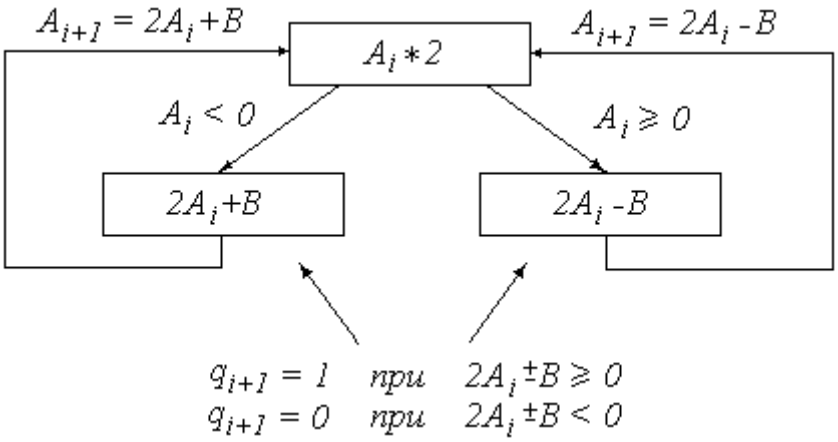


Рисунок 3 - Алгоритм выполнения $(i+1)$-го цикла деления без восстановления остатка

Цикл начинается со сдвига на один разряд влево (удвоения) остатка A_i , полученного во время предыдущего i-го цикла деления. Далее в зависимости от того, каков был знак предыдущего частного остатка A_i, выполняется либо операция вычитания ($2A_i - B$, если $A_i \geq 0$), либо операция сложения ($2A_i + B$, если $A_i < 0$). Результат выполнения операции сложения-вычитания является новым остатком. Очередная цифра частного определяется знаком получившегося частного остатка $A_{i+1}=2A_i \pm B$. Если $A_{i+1} \geq 0$, то в очередной разряд частного записывается 1, а если $A_{i+1} < 0$, то $q_{i+1} = 0$. Таким образом, во время одного цикла выполняется только одна операция сложения-вычитания.

Пример 9 - На сумматоре ДК разделить IA = 9/16 на IB = 12/16, представленные в ПК. Разрядность операндов $n = 4$.

Машинные изображения делимого и делителя, представленные в ПК и ДК будут совпадать $A_{ПК} = A_{ДК} = 0.1001$; $B_{ПК} = B_{ДК} = 0.1100$.

Определим знак частного: $SgQ = SgA \wedge SgB = 0 \wedge 0 = 0$.

Для выполнения операции вычитания потребуется отрицательное значение делителя $-B_{ДК} = 11.0100$.

Проверим, выполняется ли условие (5) и затем определим цифру за цифрой разряды частного:

$A_{ДК} = 00.1001$
+
$-B_{ДК} = 11.0100$

$S_{ДК} = 11.1101$ $\quad |A| < |B|$, $q_0 = 0$ – переполнения нет
$2A_{ДК} = 11.1010$
+
$B_{ДК} = 00.1100$

$A_{1\,ДК} = 00.0110$ $\quad q_1 = 1$
$2A_{1ДК} = 00.1100$
+
$-B_{ДК} = 11.0100$

$A_{2\,ДК} = 00.0000$ $\quad q_2 = 1$, так как $A_2 = 0$
$2A_{2\,ДК} = 00.0000$
+
$-B_{ДК} = 11.0100$

$A_{3\,ДК} = 11.0100$ $\quad q_3 = 0$
$2A_{3ДК} = 10.1000$
+
$B_{ДК} = 00.1100$

$A_{4\,ДК} = 11.0100$ $\quad q_4 = 0$

Частное получается в прямом коде из значений SgQ и q_1 , q_2 , q_3 , q_4 . Ответ: $Q_{ПК} = 0.1100$, Q=3/4.

Если делимое и делитель представлены в ОК или ДК, то деление можно выполнять с учетом их знаков. Правила выполнения алгоритма изменяются следующим образом:

- на каждом шаге сравниваются знаки остатка SgA_i и делителя SgB;
- в зависимости от результата выполняется либо операция вычитания ($2A_i - B$, если знаки совпадают ($SgA \wedge SgB = 0$), либо операция сложения, если знаки противоположные ($SgA \wedge SgB = 1$);
- если $A_i = 0$, то вместо знака остатка используется знак делимого;
- результат выполнения операции сложения-вычитания является новым остатком $A_{i+1}=2A_i \pm B$;
- при $|2A_i| \pm |B| \geq 0$ очередная цифра частного $q_i = \bar{S}gQ$, при $|2A_i| \pm |B| < 0$ $q_i = SgQ$.

Очередную цифру частного можно определить путем сравнения знака получившегося частного остатка A_{i+1} и делимого: если знак A_{i+1} совпадает со

знаком делимого или $A_{i+1}=0$, то очередная цифра частного $q_i = \bar{S}gQ$, иначе - $q_i = SgQ$. Таким образом, условием для $q_i = SgQ$ является

$$(Sg(A_{i+1}) \wedge SgQ \wedge SgB) \,\&\, (2A_{i+1} \neq 0). \qquad (9)$$

Результат деления получается в ОК, если требуется частное, представленное в ДК, то к младшему разряду регистра, содержащего отрицательное частное, нужно прибавить единицу. Результат операции в модифицированном ОК или ДК проще всего формировать путем записи в младший разряд регистра частного значения SgQ с последующим выполнением $n+1$ микроопераций сдвига влево $Q:=L1(Q.q_i)$.

Пример 10 - На сумматоре ДК разделить IA = -9/16 на IB = 12/16, представленные в ДК. Разрядность операндов $n = 4$.

Машинные изображения делимого и делителя, представленные в ДК, $A_{ДК} = 11.0111$; $B_{ДК} = 00.1100$.

Определим знак частного: $SgQ = SgA \wedge SgB = 1 \wedge 0 = 1$.

Для выполнения операции вычитания потребуется отрицательное значение делителя $-B_{ДК} = 11.0100$.

Проверим, выполняется ли условие (5) и затем определим цифру за цифрой разряды частного:

$A_{ДК} = 11.0111$
+
$B_{ДК} = 00.1100$

$S_{ДК} = 00.0011$ $\quad |A| < |B|$, $q_0 = 1$ – переполнения нет
$2A_{ДК} = 00.0110$
+
$-B_{ДК} = 11.0100$

$A_{1\,ДК} = 11.1010$ $\quad q_1 = 0$
$2A_{1\,ДК} = 11.0100$
+
$B_{ДК} = 00.1100$

$A_{2\,ДК} = 00.0000$ $\quad q_2 = 0$, так как $A_2=0$
$2A_{2\,ДК} = 00.0000$
+
$B_{ДК} = 00.1100$

$A_{3\,ДК} = 00.1100$ $\quad q_3 = 1$
$2A_{3ДК} = 01.1000$
+
$-B_{ДК} = 11.0100$

$A_{4\,ДК} = 00.1100$ $\quad q_4 = 1$

Частное получается в обратном коде из значений SgQ и q_1 , q_2 , q_3 , q_4 . Ответ: $Q_{ОК}$ = 11.0011, $Q_{ДК}$ = 11.0100, Q = -3/4.

Из приведенных примеров видно, что если процесс деления не останавливается при достижении нулевого остатка, то после выполнения $n+1$ циклов остаток равен $\pm B$, в этом случае он не должен учитываться.

Учитывая, что знак разности $|2A| - |B|$ равен знаку разности $|A| - |B/2|$, рассмотренные алгоритмы деления можно модифицировать, заменив операцию сдвига влево остатка на операцию сдвига вправо делителя. При формировании условий необходимо учитывать, что в процессе деления не будет сохраняться значение делителя.

Подробная информация по методам выполнения арифметических операций приводится в /3, 4/.

2.3 Микропрограммирование операций

Чтобы построить автомат (операционное устройство), выполняющее арифметическую операцию, рассмотренные неформальные описания алгоритмов необходимо представить в виде микропрограммы. Наглядной формой представления микропрограммы является граф-схема. Граф-схема алгоритма (ГСА) микропрограммы строится с использованием вершин четырёх типов и дуг, связывающих вершины.

Начальная вершина отмечает начало алгоритма и имеет единственный выход, из которого исходит дуга к первой выполняемой вершине графа. Операторная вершина определяет действие - микрооператор, совокупность совместимых микроопераций. В операторную вершину может входить любое, не меньшее одной, число дуг, и из её вершины выходит только одна дуга. Условная вершина используется для разветвления вычислительного процесса в одном из двух возможных направлений, в зависимости от значения проверяемого условия. В условную вершину может входить любое число дуг, но выходят всегда две дуги. Конечная вершина отмечает конец микропрограммы. В конечную вершину может входить любое число дуг.

ГСА считается корректной, если выполняются следующие условия:

- ГСА содержит конечное число вершин, каждая из которых принадлежит к перечисленным типам;
- имеет одну начальную и одну конечную вершины;
- выходы и входы вершин соединяются с помощью дуг, направленных от выхода ко входу;
- каждый выход соединён с одним входом;
- из любой вершины существует хотя бы один путь к конечной;
- один из выходов условной вершины может соединяться с её входом, что недопустимо для операторной вершины.

ГСА делятся на содержательные и закодированные. В содержательной ГСА в условных вершинах записываются условия (операции отношения), а в операторных вершинах – наборы совместимых микроопераций.

Наборы совместимых микроопераций образуют микрооператоры. Микропрограммирование сводится к определению такой последовательности микрооператоров, выполнение которых завершается получением требуемого результата вычислений.

Микрооперации называются *функционально совместимыми*, если они присваивают значения разным словам. Условием совместимости *m* микроопераций является совместимость каждой пары микроопераций.

Совместимость микроопераций может быть обусловлена и структурой ОА. Структурная совместимость определяется возможностью ОА параллельно выполнять нескольких микроопераций. В функциональных микропрограммах, описывающих алгоритм выполнения операций безотносительно к структуре ОА, должна учитываться только функциональная совместимость.

Пример содержательной ГСА умножения, начиная с младших разрядов множителя со сдвигом частных сумм в ПК представлен на рисунке 4. Здесь использованы обозначения слов из примера 1. Внутреннее слово *Ct* использовано для обратного счета выполненных циклов.

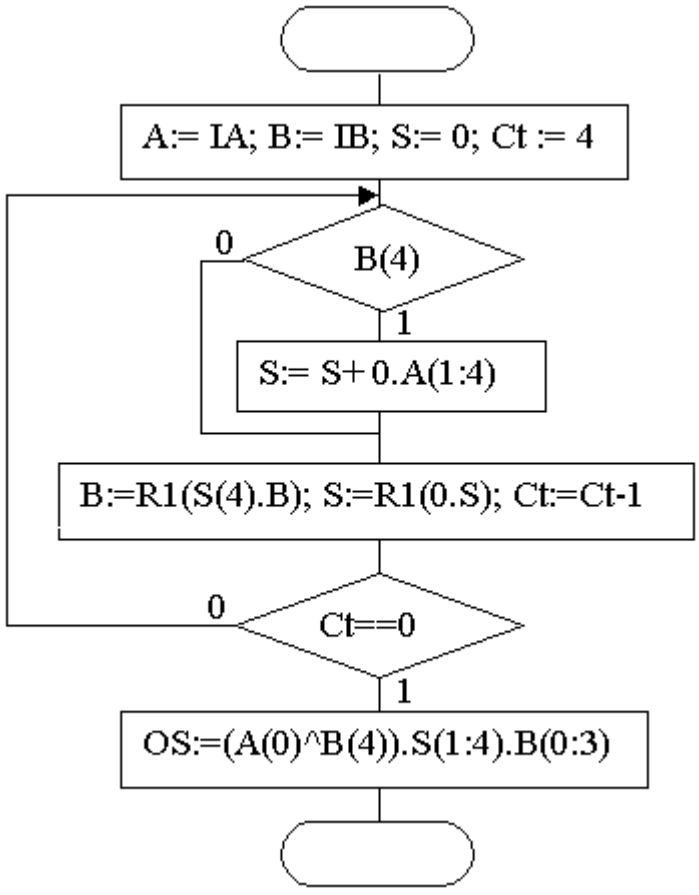


Рисунок 4

От правильности формирования микропрограммы в конечном итоге зависит правильность работы разрабатываемого операционного устройства.

Чтобы исключить возможность неправильной работы микропрограммы из-за ошибок, необходимо выполнить ее тестирование. Полное тестирование обычно требует значительных затрат времени. В курсовом проекте достаточно ограничиться частичным тестированием для нескольких значений операндов, Но эти значения необходимо подобрать таким образом, чтобы можно было проверить работу всех ветвей ГСА микропрограммы. С этой целью могут подойти значения операндов из ранее рассмотренных примеров со всеми возможными комбинациями значений знаковых разрядов. Тестирование микропрограммы можно выполнить с помощью отладчика микропрограмм ТА. В таблице 7 приведены результаты тестирования микропрограммы (рисунок 4).

Таблица 7

№ вар	1	2	3	4
IA	0.0001101	0.0001101	1.0001101	1.0001101
IB	0.0001011	1.0001011	0.0001011	1.0001011
OS	0.00000010001111	1.00000010001111	1.00000010001111	0.00000010001111

Анализ полученных результатов позволяет сделать вывод, что микропрограмма работает правильно для всех четырех вариантов значений операндов.

3 Методические указания по синтезу операционного автомата

Функция ОА определяется совокупностью данных:

- множеством входных слов *{IA, IB},* вводимых в автомат как операнды;
- выходным словом *OS*, представляющим результат операции умножения;
- множеством внутренних слов *{A, B, S, Ct),* используемых для представления информации в процессе выполнения операци;
- множеством микроопераций, реализующих преобразование слов информации *{S := 0, Ct := 4, A := IA, B := IB, B:= R1(S(4).B), S:= R1(0.S), Ct := Ct-1, S := S+0.A(1:4), OS := (A(0)^B(4)).S(1:4).B(0:3)};*
- множеством логических условий *{B(4), Ct==0}*.

Как и всякая задача структурного синтеза, задача построения ОА имеет множество решений. Наиболее просто построить ОА как сеть из подавтоматов (типовых функциональных элементов и узлов), выполняющих подмножество микроопераций алгоритма. Разобьем множество микроопераций на непересекающиеся подмножества микроопераций, изменяющих значение одного и того же слова:

- *{A := IA}* – загрузка данных*;*
- *{B := IB, B:= R1(S(4).B)}* –загрузка данных и сдвиг*;*
- *{S := 0, S:= R1(0.S), S := S+0.A(1:4)}* – сброс, сдвиг и загрузка данных
- *{Ct := 4, Ct := Ct-1}* – загрузка константы и декремент*;*
- *{OS := (A(0)^B(4)).S(1:4).B(0:3)}* – загрузка данных.

Тип необходимого функционального узла определяется составом подмножества микроопераций:

- регистр может выполнять микрооперации загрузки и сброса;
- универсальный сдвиговый регистр может выполнить микрооперации загрузки, сброса и сдвига;
- счетчик может выполнять микрооперации загрузки, сброса и декремента.

Если в правой части микроопераций стоят символы арифметических и логических операций, то для их реализации необходимо использовать соответствующие комбинационные автоматы.

Следует учесть, что число разрядов слов в типовых функциональных узлах фиксировано и обычно принимает значения: 4, 8, 16, 32. Кроме этого, для них принято использовать противоположный порядок нумерации разрядов, например, *D(7)* – старший разряд, а *D(0)* – младший. Это необходимо учитывать при построении ОА путем изменения обозначений разрядов в словах и полях. Для решения данной задачи схема ОА может содержать следующий набор элементов и восьмиразрядных узлов (имена взяты из библиотеки элементов TA.olb):

- регистр dff8 *(A:=IA);*
- универсальный сдвиговый регистр rightsh8 *(B:=IB, B:=R1(S(0).B));*
- сумматор-вычитатель add-sub8 и универсальный сдвиговый регистр rightsh8 *(S := 0, S:= S+0.A(6:0), S:=R1(0.S));*
- вычитающий счетчик с загрузкой count4 *(Ct:=7, Ct:=Ct-1);*
- элемент xor2 и регистр dff16 *(OS:= A(7)^B(0). S(6:0).B(7:1)).*

Регистры и счетчик являются синхронными цифровыми автоматами, их работу можно описать в форме таблиц режимов (таблицы 8 – 10).

Таблица 8 – Режимы работы регистра

Режим работы	*C*	***D***		***Q^+[7. .0]***
Сброс	x	x	1	0
Загрузка	↑	1	0	*D[7..0]*
Хранение	x	0	0	*Q[7..0]*

Таблица 9 – Режимы работы сдвигового регистра

Режим работы	*C*	***D***	***H***		***Q^+[7..0]***	***D OUT>***
Сброс	x	x	x	1	0	0
Загрузка	↑	1	0	0	*D[7..0]*	*D0*
Сдвиг вправо	↑	0	1	0	*DIN>.Q[7..1]*	*Q1*
Хранение	x	0	0	0	*Q[7..0]*	*Q0*

Таблица 10 – Режимы работы счетчика

Режим работы	*C*			***CT^+[7..0]***

		D	***E***	
Загрузка	↑	1	x	*D[7..0]*
Декремент	↑	0	1	*DIN>.Q[7..1]*
Хранение	x	0	0	*CT[7..0]*

Сумматор-вычитатель является комбинационным автоматом, его режим работы определяется значением сигнала *M*:

- *M=1* – сложение;
- *M=0* – вычитание.

Следует помнить, что для сложения (вычитания) чисел в обратных кодах вход переноса *CI* необходимо соединить с выходом переноса *CO*.

Для построения ОА необходимо соединить выходы элементов и узлов, на которых формируются слова или их поля, стоящие в правых частях микроопераций, с входами данных элементов и узлов, выполняющих микрооперацию. Пример выполнения функциональной схемы ОА приведен в приложении Б. В схему дополнительно включены задатчики констант (U1, U2, U4) и буферы (U3, U5) для выполнения конкатенации полей слов. Эти элементы можно назвать «виртуальными» в том смысле, что они могут быть реализованы без аппаратных затрат. С ними функциональная схема более полно поясняет принцип работы ОА. Кроме цепей данных схема содержит цепи синхронизации и управления.

Функциональная микропрограмма (рисунок 4) описывает алгоритм выполнения операции безотносительно к структуре ОА. Структура ОА может вносить ограничения на возможность параллельного выполнения микроопераций. Микрооперации называются *структурно несовместимыми*, если из-за ограничений, порождаемых структурой ОА, они не могут быть выполнены совместно - в одном такте автоматного времени. Структурная несовместимость микроопераций связана с использованием микрооперациями общего оборудования, единственность которого исключает возможность совместного выполнения микроопераций.

В рассматриваемом примере анализ микропрограммы показывает, что для всех микрооператоров, входящие в их состав микрооперации будут выполняться различными элементами и узлами схемы, следовательно, проблем со структурной несовместимостью нет.

Функция ОА не определяет порядок выполнения микрооператоров во времени, он определяется последовательностью микрокоманд, формируемой управляющим автоматом (УА). Необходимо определить коды микрокоманд (комбинации значений управляющих сигналов - микроприказов). Каждой микрооперации можно поставить в соответствие свой микроприказ, но при этом коды микрокоманд будут избыточными. Число разрядов в кодах микрокоманд можно сократить, если для совместимых во всех микрооператорах микроопераций использовать один микроприказ.

Для этого множество микроопераций необходимо разбить на ряд возможно пересекающихся подмножеств таким образом, чтобы элементы каждого подмножества входили в любые микрооператоры только совместно. В данном

случае каждая из микроопераций входит только в один микрооператор, поэтому разбиение совпадает с составом микрооператоров, а код каждой микрокоманды может содержать только один микроприказ. В таблице 11 представлены коды микрокоманд, обеспечивающие выполнение всех микрооператоров.

Таблица 11

Микрооператоры	Микрокоманды
S := 0, Ct := 4, A := IA, B := IB	y1
B:= R1(S(4).B), S:= R1(0.S), Ct := Ct-1	y2
S := S+0.A(1:4)	y3
OS:= A(7)^B(0). S(6:0).B(7:1)	y4

По полученным результатам можно сформировать цепи управления и синхронизации ОА. Для этого необходимо объединить синхровходы всех синхронных функциональных узлов в общую цепь синхронизации *Clk* и подать микроприказы на соответствующие управляющие входы функциональных узлов:

- *y1* на входы *R* DD5 (*S := 0), LD* DD6 (*Ct := 4), LD* DD1 (*A := IA), LD* DD2 (*B := IB);*
- *y2* на входы *SH* DD2 *(B:= R1(S(4).B)), SH* DD5 *(S:= R1(0.S)), CE* DD6 (*Ct := Ct-1);*
- *y3* на вход *LD* DD5 *(S := S+0.A(1:4));*
- *y4* на вход *LD* DD7 *(OS:= A(7)^B(0). S(6:0).B(7:1)).*

В общем случае для формирования осведомительных сигналов необходимо построить комбинационную схему, реализующую функции отношений. В данном примере осведомительные сигналы непосредственно формируются на выходах функциональных узлов:

- *x1* – соответствует логическому условию *B(0);*
- *x2* – соответсвует логическому условию *(Ct = = 0).*

4 Методические указания по абстрактному синтезу управляющего автомата

4.1 Кодированная ГСА микропрограммы

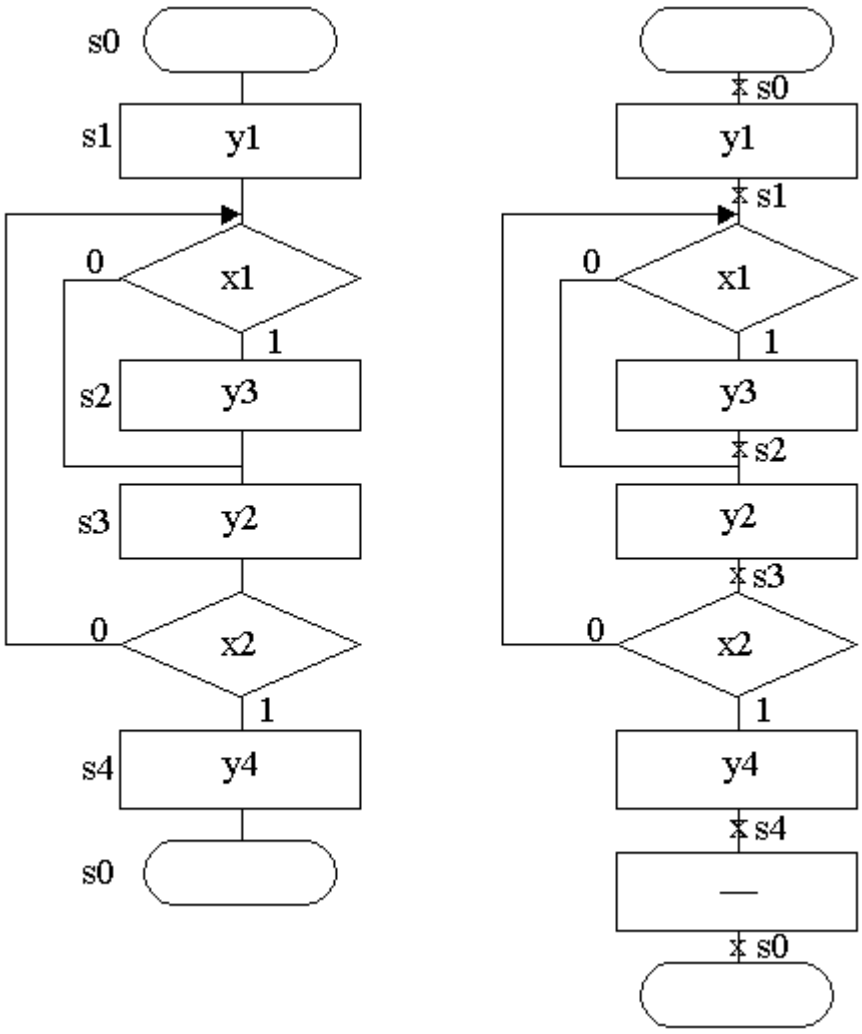
Функция управляющего автомата задаётся кодированной ГСА микропрограммы. Кодированную ГСА получают путём замены в содержательной ГСА микрооператоров (наборов совместимых микроопераций) на коды микрокоманд и логических условий на их идентификаторы.

Заменим наборы микроопераций на коды микрокоманд и логические условия на обозначения осведомительных сигналов. В результате получим кодированную ГСА микропрограммы (рисунок 5, а).

Построение управляющего автомата начинается с отметки внутренних состояний на кодированной ГСА. Отметка состояний должна соответствовать

закону функционирования автомата Мура или Мили, то есть выполняется для них различным образом.

Будем полагать, что автомат начинает работу с состояния s_0, в котором он не вырабатывает никаких выходных сигналов и после выполнения микропрограммы снова оказывается в этом же состоянии. Затем автомат переходит в состояния, предписанные законом функционирования и формирует микрокоманды ***Y***, соответствующие текущим значениям сигналов ***X***. Момент окончания выполнения микропрограммы отмечается возвратом автомата в начальное состояние s_0.



а) б)

Рисунок 5

Поскольку в автомате Мура выходные сигналы связаны только с состоянием автомата, то каждой операторной вершине нужно поставить в соответствие одно из состояний автомата. Следовательно, правило разметки состояний автомата Мура будет выглядеть следующим образом:

- символом s_0 отмечаются начальная и конечная вершины ГСА;
- каждая операторная вершина отмечается единственным символом s_1, s_2, s_3, s_4, s_5, ... ;

- две различные операторные вершины не могут быть отмечены одинаковыми символами.

Если для интерпретации закодированной ГСА используется автомат Мили, то разметка граф-схемы производится в следующем порядке:

- символом s_0 отмечается выход начальной и вход конечной вершины;
- символами s_1, s_2, ... отмечаются входы вершин, следующие за операторными вершинами;
- входы двух различных вершин не могут быть отмечены одинаковыми символами;
- входы вершины могут отмечаться только одним символом состояния.

Приведенные правила означают, что если вершина имеет несколько входов, то символом состояния отмечается их подмножество, состоящее из входов, следующих только за начальной или за операторными вершинами.

Если один из входов конечной вершины соединен с выходом операторной вершины, то между ними необходимо ввести пустую операторную вершину, иначе автоматы Мили и Мура, построенные по одной ГСА не будут эквивалентными.

Для рассматриваемого примера на рисунке 5 показаны отметки внутренних состояний для автоматов Мура и Мили. Таким образом, автоматы Мура и Мили будут иметь по пять внутренних состояний.

4.2 Анализ работы операционного автомата

Для проверки правильности работы ОА при выполнении полученной микропрограммы рекомендуется выполнить анализ работы методом имитационного моделирования в среде системы OrCAD Express. Подробную информацию о работе с этой системой можно найти в /5/.

ОА работает в составе операционного устройства. Пример структурной схемы операционного устройства, выполненной в среде редактора схем OrCAD Capture, приведен на рисунке 6. Схема выполнена как иерархическая структура, которая содержит иерархические блоки H1 (ОА) и H2 (УА). Взаимодействие автоматов обеспечивается управляющими сигналами *y1 – y4* и осведомительными сигналами *x1, x2*. Для исключения состязаний в схеме используется двухфазная синхронизация. Синхросигнал *Clk* управляет работой ОА, для синхронизации УА используется инверсный сигнал *Ck*. Сигнал *Res* выполняет инициализацию (системный сброс) ОУ. Структура УА не определена, поэтому его можно представить функциональной VHDL-моделью.

Чтобы ввести схему в среде редактора OrCAD Capture, необходимо его запустить и создать новый проект. Для этого надо выполнить команду **File/New/Project** или щелкнуть на кнопке Create document, расположенной на панели инструментов. Кроме схем проект содержит много другой информации, например описания входных сигналов, модели компонентов, библиотеки символов, кэш проекта (Design cache) и т. п.

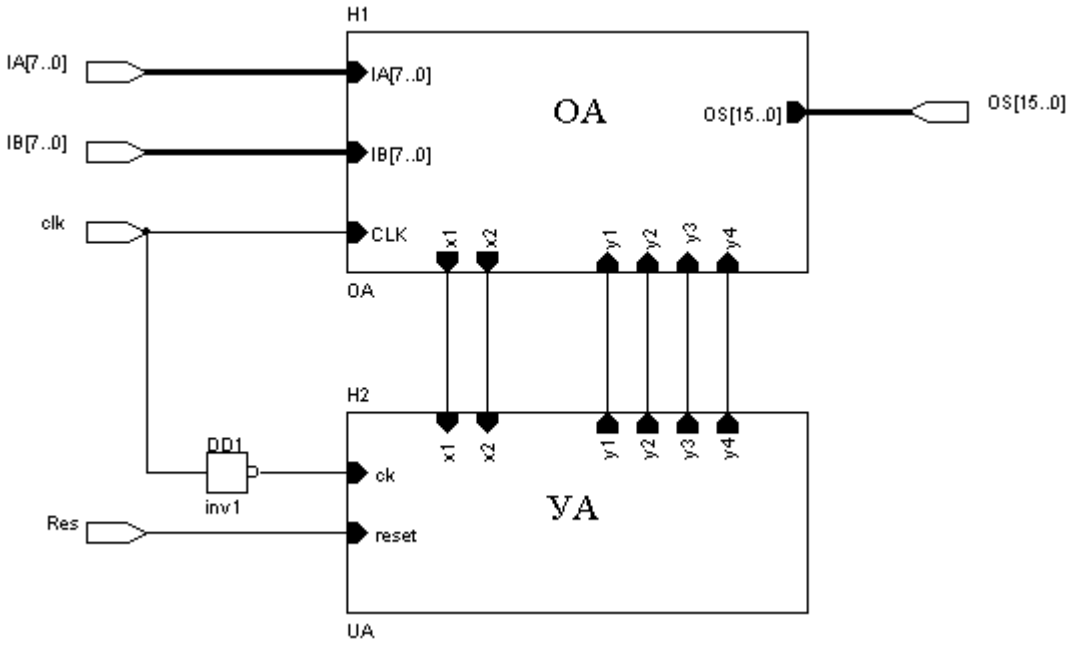


Рисунок 6

На экране появится диалоговая панель New Project, в которой надо задать имя проекта (верхнее поле), например OU, выбрать тип проекта и определить, где он будет располагаться на жестком диске (нижнее поле). Можно указать для проекта несуществующую папку — OrCAD Capture создаст ее автоматически.

В OrCAD определены четыре типа проектов, необходимо выбрать проект типа Programmble Logic Wizard. Этот тип проекта рекомендуется для моделирования схем цифровых устройств на функционально-логическом уровне.

Задав всю необходимую информацию и нажав кнопку OK, вы увидите другую диалоговую панель — Programmble Logic Project Wizard. Панель предлагает выбрать базовое семейство программируемой логики. В данном случае рекомендуется не давать предпочтения ни одному из перечисленных, выбирая Other. После нажатия кнопки «Далее» загружается менеджер проекта с заданным именем (OU). Проект имеет папки Design Resources и Simulations Resources. Чтобы включить в Design Resources схемы, необходимо ввести команду **File/New/Design**. Редактор OrCAD Capture создает ресурс design1.dsn и вложенную в него схему, по умолчанию схема получает имя SCHEMATIC1: PAGE1. Эти имена легко изменить в менеджере проекта, щелкнув правой кнопкой мыши на неугодном имени и указав в открывшемся контекстном меню команду Rename. В данном случае проект должен содержать две схемы:

- верхнего уровня OU: PAGE1 (схема операционного устройства, пример на рисунке 6);

- нижнего уровня OA: PAGE1 (схема OA, пример в приложении Б).

Для добавления в ресурс design1.dsn еще одной схемы необходимо выделить его указателем и выполнить команду **New Schematic**, затем выделить появившуюся папку и выполнить команду **New Page**. Эти команды всплывают при нажатии правой кнопки мыши. Первая из вложенных в ресурс проекта схем

является корневой, то есть более высокого иерархического уровня. При необходимости порядок иерархии можно изменить выпадающей командой **Make Root**.

Для формирования схем потребуется библиотека элементов ta.olb, чтобы подключить ее к проекту необходимо в менеджере проекта найти папку Library, щелкуть на ней правой кнопкой мыши и активизировать выпадающую команду **Add File**. Библиотека ta.olb находится в папке \TA.

После выполнения перечисленных действий в окне OrCAD Capture в левой части будет расположен менеджер проекта, а правую часть занимать окна со схемами. Обратите внимание: содержимое меню команд зависит от того, какое из окон активно.

Рекомендуется сначала сформировать объекты нижнего иерархического уровня, то есть схему ОА и VHDL-модель УА, а только затем формировать схему верхнего уровня. Такой порядок упрощает процедуру установления связей между ними.

Активизируя окно схемы, переведем редактор в режим формирования схемы. При этом появится палитра инструментов Tool Palette, с помощью которой проектируется схема. По умолчанию панель Tool Palette располагается вертикально в правой части экрана и дублирует команды меню Place.

Наиболее часто используемые кнопки расположены в верхней части панели:

- **Select** - для переключения курсора мыши в режим выбора (выделения) объектов схемы (элементов, цепей, имен и т. п.);
- **Place Part** - позволяет размещать элементы;
- **Place wire** - для соединения элементов проводниками (цепями);
- **Place port** - для размещения внешних портов электрической схемы;
- **Place bus** - для размещения линий групповой связи (шин);
- **Place junction** - для электрического соединения пересекающихся проводников (цепей);
- **Place net alias** – для ввода имен проводников (цепей);
- **Place bus entry** - для соединения проводников с «жилами» шины;
- **Place hierarchical block** -для размещения иерархических блоков;
- **Place pin** - для размещения выводов иерархического блока;
- **Place no connect** - для отметки незадействованных выводов элементов.

Формирование схемы рекомендуется выполнять в следующей последовательности:

- разместить элементы и внешние порты (**Place Part** и **Place Port**);
- соединить их выводы между собой (**Place wire**);
- отредактировать имена внешних портов.

Щелкнем на пиктограмме **Place Part** (разместить элемент). Появится диалоговая панель с одноименным названием, на которой видны имена подключенных к проекту библиотек (левое нижнее окно) и список имеющихся в них элементов. Щелкните на любом имени элемента, и его условное графичес-

кое обозначение (УГО) появится в правом нижнем окне. О функции элемента можно судить по его имени или УГО.

Наприме, выделим компонент inv1, выполняющий функцию НЕ. Нажмем кнопку OK и укажем в окне схемы желаемое место. Чтобы зафиксировать положение элемента, щелкнем левой кнопкой мыши. Обратите внимание: рядом с УГО появилось имя DD1 — это позиционное обозначение компонента. Переместим курсор в другое место и опять нажмем левую кнопку. На экране появится еще один такой же элемент с именем DD2. Таким образом, можно размещать сколько угодно копий, пока вы не нажмете клавишу [Esc] или правую кнопку мыши, исполнив затем команду **End Mode** в открывшемся контекстном меню. Есть еще один способ снять активность текущей команды — переместить курсор мыши на пиктограмму **Select** и щелкнуть левой кнопкой.

Чтобы удалить ненужный элемент подведем курсор к нему и выделим его щелчком мыши. Выделенный элемент помечается красным цветом. Нажмем клавишу **Del**, и элемент исчезнет с экрана.

Порты схемы размещаются аналогично командой **Place Port**. Входные порты схемы имеют имя PortRight-R, выходные порты PortLeft-L. При выборе порта необходимо задать имя входного (или выходного) сигнала схемы. Эти имена можно отредактировать после размещения портов двойным нажатием кнопки мыши по полю имени порта.

Чтобы получить законченную схему, размещенные элементы необходимо соединить проводниками. Для этого надо щелкнуть на пиктограмме **Place wire** (разместить проводник). Обратите внимание: курсор мыши изменил свою форму, теперь он похож на небольшое перекрестие. Выполним все необходимые соединения, нажимая левую кнопку мыши для обозначения начала и конца каждого проводника.

Чтобы нарисовать сложную цепь, неоднократно меняющую направление, необходимо в точках излома фиксировать уже нарисованную часть проводника щелчком левой кнопки мыши. Чтобы закончить рисование цепи, надо нажать правую кнопку мыши, а затем исполнить команду **End Wire** либо произвести двойной щелчок в точке, где заканчивается проводник. При постоянно нажатой кнопке мыши рисование цепи прекращается, если кнокпу отпустить при достижении вывода элемента. При этом активность команды не снимается. Чтобы закончить процесс рисования проводников надо нажать клавишу [Esc] или щелкнуть на пиктограмме **Select** (или выбрать новую команду).

Линии связи и выводы элементов схемы допускается соединять между собой только встык, без наложения. Визуальный контроль подключения цепи к контакту весьма прост: свободный вывод компонента заканчивается небольшим квадратиком, который исчезает, если произошло соединение. И наоборот, если проводник подключается к другой цепи, то в точке их соприкосновения появляется Junction-соединение (точка).

Модель УА сформируем по кодированной ГСА микропрограммы на языке VHDL. Описание объекта на языке VHDL имеет типовую структуру: в его начале указываются библиотеки стандартных моделей, затем следуют описание

интерфейса объекта и раздел архитектуры, который представлен в поведенческом варианте. Объявим переменные для представления текущего (*current_state*) и следующего (*next_state*) внутреннего состояния. Они могут принимать значения из алфавита {*s0, s1, s2, s3, s4*}, который образует перечисляемый тип *state_type*.

УА является синхронным автоматом, но модель конечного автомата не описывает процесс синхронизации. Поэтому функционирование УА представим в виде двух параллельных процессов:

- управления и синхронизации памяти автомата;
- функционирования конечного автомата.

Для описания первого из этих процессов необходимо задать реакцию памяти автомата на сигналы инициализации *Reset* и синхронизации *Ck*. Особенностью автомата Мили является зависимость значений выходных сигналов не только от текущего состояния, но и от значений входных сигналов. При работе в составе операционного устройства эта особенность может привести сбоям в его работе, так как выходные сигналы УА могут изменяться непосредственно вслед за изменениями осведомительных сигналов, не дожидаясь прихода фронта синхросигнала *Ck.* Чтобы исключить такую возможность, достаточно на выходе автомата Мили включить элементы памяти (регистр), управляемые синхросигналом *Ck.*

Для описания второго из этих процессов необходимо описать функции переходов и выходов автомата с помощью оператора *case*. Здесь модели автоматов Мура и Мили будут отличаться способом задания значений функции выходов. Примеры VHDL – моделей управляющих автоматов Мура и Мили приведены в приложениях В и Г.

В проекте VHDL-модель должна присутствовать в виде текстового файла, рекомендуется использовать встроенный текстовый редактор OrCAD, так как он имеет средства синтаксического контроля. То есть правильность конструкций языка можно проверять при формировании текста.

Формирование нового файла выполняется по команде **File/New/VHDL File**. Готовый файл подключается к ресурсам проекта путем выделения Design Resources и активизации всплывающей команды **Add File** правой кнопкой мыши.

При формировании схемы верхнего уровня необходимо разместить иерархические блоки командой **Place hierarchical block**. Их размещение выполняется почти также как и размещение простых элементов. Отличия состоят в том, что необходимо задать их позиционные обозначения (Reference), типы (Implementation Type) и имена (Implementation Name), а для VHDL модели еще имя и местонахождение файла (Path and filename). В данном случае значения этих параметров могут быть следующими

- H1, Schematic View , OA;
- H2, VHDL, UA, A_moor.vhd.

Если к моменту размещения иерархических блоков в проекте присутствуют указанные ресурсы, то связь с ними устанавливается автоматически, а в УГО блоков автоматически появляются их внешние

контакты, описанные в ресурсе нижнего уровня. Проверить связи между блоками и ресурсами можно командами **View\Ascend Hierarchy**, **View\Descend Hierarchy**.

Далее необходимо подключить к проекту VHDL-библиотеку функциональных моделей элементов ta.vhd. Для этого в менеджере проекта надо выбрать папку In Design, правой кнопкой мыши активизировать выпадающую команду **Add File**. Библиотека ta.vhd находится в папке \TA. Программа предложит выбрать тип файла (Select File Type) – в нашем случае это VHDL SimModel.

Основная подготовка выполнена, и теперь можно запустить программу моделирования. Для этого надо активизировать окно менеджера проекта и выполнить команду **Tools/Simulate**. В качестве ответа на запрос о способе моделирования надо указать верхнюю строчку In design. После этого автоматически запустится программа моделирования OrCAD Simulate. Она сообщит, что проект открыт, в ответ необходимо подтвердить необходимость загрузки схемы.

Активизируя команду **Stimulus/Create Test Bench**, автоматически создается вектор тестов, то есть описания внешних воздействий. В диалоговом окне этой команды выбирается проект в целом и на строке VHDL Output File указывается имя выходного файла. Для подключения файла тестов к проекту указывается опция Add to Project. Созданный таким образом файл тестов необходимо отредактировать в среде текстового редактора OrCAD, чтобы задать значения операндов и описания управляющих сигналов. Значения операндов *IA* и *IB* в процессе моделирования не изменяются, поэтому их значения можно задать в блоке описания внешних сигналов операционного устройства в двоичной или шестнадцатиричной форме. Например, значения *IA= -11* и *IB = 13* можно задать следующим образом

```
signal   os : std_logic_vector(15 downto 0);
signal   ia : std_logic_vector(7 downto 0):=x”8B”;
signal   ib : std_logic_vector(7 downto 0):=x”0D”;
signal   res : std_logic;
signal   clk : std_logic;
```

Описания изменяющихся во времени входных воздействий должны следовать после комментария

```
--Place stimulus and analysis statements here.
 process begin
  -- modify the delay values and clock signal name
  Clk <= '0'; wait for 10 ns;
  Clk <= '1'; wait for 10 ns;
end process;
process begin
  -- modify the delay values and clock signal name
  res <= '1'; wait for 10 ns;
  res <= '0'; wait;
end process;
```

По этим описаниям будут формироваться одиночный импульс *res* длительностью 10 нс и периодический синхросигнал *Clk* (длительность импульса 10 нс, период 20 нс). Для описания стимулов можно использовать образцы, открываемые нажатием правой кнопки мыши (Samples).

Остается определить условия моделирования. Для этого выберем в выпадающем меню Simulate команду **Simulate/Run**, зададим время моделирования, например 280 нс, и нажмем OK. Программа выполнит требуемую работу и сообщит о ее результатах. Через некоторое время они появятся на экране монитора в табличной форме и в виде временных диаграмм. По умолчанию данные выдаются о всех внешних сигналах схемы. Состав и порядок следования сигналов на временных диаграммах можно изменить — эти операции выполняются командой **Trace/Edit Signal Traces**. Для получения наглядного представления о процессе функционирования операционного устройства необходимо вывести временные диаграммы следующих сигналов*: clk, res, B, S, OS, x1, x2, y1, y2, y3, y4, current_state*. Значения сигналов *B, S, OS* удобнее представлять в шестнадцатиричной форме.

Результаты моделирования ОА в составе операционного устройства при умножении чисел *IA* = - 11 = 8Bh *IB* = +13 = 0Dh приведены на рисунке 7. Результат *OS* = 811Eh. Отбрасывая неиспользуемый младший разряд, получим в двоичной и десятичной форме *OS*=1000 0001 0001 111b = - 143, что является правильным значением. Аналогично необходимо выполнить расчеты для других значений операндов из таблицы 7. Если результаты моделирования показывают, что ОУ работает правильно, то схема ОА и функция УА не содержат ошибок, можно приступить к выполнению структурного синтеза УА. В противном случае необходимо найти ошибки и устранить их.

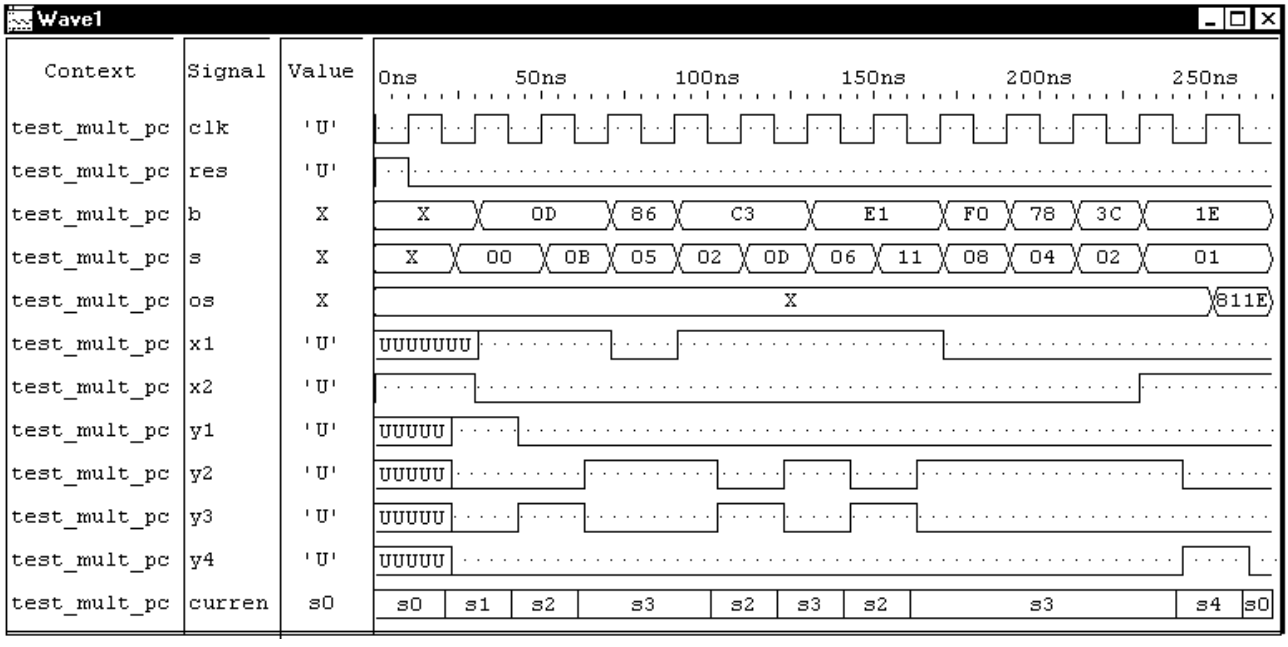


Рисунок 7

4.3 Таблица переходов-выходов

Для построения УА удобно представить его характеристические функции в табличной форме. Для автомата Мура в ячейках таблицы переходов-выходов для каждой пары значений аргументов ***(X(t), s(t))*** проставляются будущие внутренние состояния *s(t+1)*. Значения выходных сигналов ***Y(t)*** представляются в отдельном столбце. Для автомата Мили в ячейках таблицы переходов-выходов для каждой пары значений аргументов ***(X(t), s(t))*** проставляются будущие внутренние состояния *s(t+1)* и текущие значения выходных сигналов ***Y(t)***. Для рассматриваемого примера характеристические функции автомата Мура приведены в таблице 12.

Таблица 12

s	x_1x_2				***Y***
	00	01	11	10	
s_0	s_1	s_1	s_1	s_1	-
s_1	s_3	s_3	s_2	s_2	*y1*
s_2	s_3	s_3	s_3	s_3	*y3*
s_3	s_3	s_4	s_4	s_2	*y2*
s_4	s_0	s_0	s_0	s_0	*y4*

5 Методические указания по структурному синтезу управляющего автомата

При заданных типах элементов памяти структурный синтез управляющего автомата сводится к выполнению следующих проектных операций:

- кодирование внутренних состояний;
- формированиение функций внешнего перехода;
- формирование и минимизация функций возбуждения элементов памяти и функций выходов;
- построение комбинационной схемы автомата в выбранном базисе логических элементов и функциональной схемы автомата.

Структурный синтез начинается с двоичного кодирования внутренних состояний автомата - установления взаимно однозначного соответствия между состояниями автомата и комбинациями состояний элементов памяти. Информация о внутреннем состоянии представляется многомерным двоичным сигналом $\boldsymbol{Q} = <q_{l-1}, \ldots, q_1, q_0>$, $l \geq \lceil \log_2 L \rceil$, L – число внутренних состояний.

Функции внешнего перехода определяют изменение состояний каждого из элементов памяти в зависимости от изменения состояния всех элементов памяти и приходящих на автомат входных сигналов.

Чтобы каждый элемент памяти работал в соответствии со своей функцией внешних переходов, необходимо, чтобы на его входы приходили строго определенные управляющие сигналы. Эти сигналы формируются по логическим выражениям, которые называют функциями возбуждения элементов памяти. Функции возбуждения зависят не только от функции внешних переходов элемента памяти, но и от его собственного, внутреннего функционирования.

Аргументами функций выходов автомата Мура являются только элементы вектора ***Q,*** для автомата Мили в их число также входят входные (осведомительные) сигналы.

Для рассматриваемого в качестве примера варианта задания рассмотрим выполнение структурного синтеза автомата Мура, заданного таблицей 12 на синхронных D-триггерах.

5.1 Кодирование состояний

Для синхронных автоматов обычно выполняют экономичное кодирование состояний, которое обеспечивает наиболее простую реализацию комбинационной схемы (КС) автомата. Используем метод соседнего кодирования, основанный на поиске соседних состояний и назначении им соседних кодов.

Практическое определение соседей первого и второго рода удобно выполнять по инверсной таблице переходов. Строки такой таблицы, как обычно, отмечаются состояниями автомата, а столбцы - входными сигналами. В каждую клетку таблицы, соответствующую состоянию s_i и входному сигналу x, заносят-

ся все состояния, из которых автомат выполняет переход в s_i под действием x. Из способа построения инверсной таблицы переходов следует, что состояния, расположенные в одной клетке такой таблицы, являются соседями первого рода. Определение соседей второго рода выполняется по инверсной таблице переходов следующим образом. Если состояния s' и s'' являются соседями первого рода и если эти два состояния встречаются в таблице еще раз в разных строках и в одном и том же столбце, то состояния s_i и s_j, отмечающие строки, в которых расположены s' и s'', являются соседями второго рода.

Из анализа обратной таблицы переходов (таблицы 13) для рассматриваемого примера следует, что соседями первого рода являются состояния s_1, s_2, s_3, а соседеми второго рода s_3, s_4. Определим число разрядов кода $l \geq \lceil \log_2 5 \rceil = 3$. Результат кодирования состояний, полученный по правилам соседнего кодирования приведен в таблице 14.

Таблица 13

s	x_1x_2			
	00	01	11	10
s_0	s_4	s_4	s_4	s_4
s_1	s_0	s_0	s_0	s_0
s_2	s_3	-	s_1	s_1 s_3
s_3	s_1 s_2 s_3	s_1 s_2	s_2	s_2
s_4	-	s_3	s_3	-

Таблица 14

s	q_2	q_1	q_0
s_0	0	0	0
s_1	0	0	1
s_2	0	1	1
s_3	1	0	1
s_4	1	0	0

5.2 Формирование функций возбуждения и выходов

Табличные формы представления функций внешнего перехода, функций возбуждения и функций выходов можно получить непосредственно из таблицы переходов-выходов 12 и таблицы кодов состояний 14. Для этого символы состояний необходимо заменить соответсующими кодами и установить порядок следования строк и столбцов в соответствии с циклическим кодом Грея. Это позволит упростить формирование и минимизацию логических функций.

Для наглядности выполняемых преобразований построим структурную таблицу автомата Мура. Заполняется структурная таблица с учетом функционирования заданного элемента памяти (таблица 9), в данном случае - D-триг-гера.

Таблица 9

$Q(t)\to Q(t+1)$	D	T	S	R	J	K
00	0	0	0	x	0	x
01	1	1	1	0	1	x
11	1	0	x	0	x	0
10	0	1	1	0	x	1

В структурной таблице автомата (таблица 10) отображаются значения функций возбуждения и выходов для всех рабочих наборов. С целью упрощения их аналитического представления отобразим их на картах Карно и выполним минимизацию (рисунок 8).

Таблица 10

Пе-ре-ход	Исх. сост	Код исходн. состояния			След. сост	Код след. состояния			Вх. сигн.	Вых. сигн.	Функции возбуждения		
		q2	*q1*	*q0*		*q2*	*q1*	*q0*			*D2*	*D1*	*D0*
1	s_0	0	0	0	s_1	0	0	1	-	-	0	0	1
2	s_1	0	0	1	s_3	1	0	1	$\overline{x1}$	*y1*	1	0	1
3	s_1	0	0	1	s_2	0	1	1	*x1*	*y1*	0	1	1
4	s_2	0	1	1	s_3	1	0	1	-	*y3*	1	0	1
5	s_3	1	0	1	s_3	1	0	1	$\overline{x1}\,\overline{x2}$	*y2*	1	0	1
6	s_3	1	0	1	s_2	0	1	1	$x1\,\overline{x2}$	*y2*	0	1	1
7	s_3	1	0	1	s_4	1	0	0	*x2*	*y2*	1	0	0
8	s_4	1	0	0	s_0	0	0	0	-	*y4*	0	0	0

Минимизацию функций целесообразно выполнять по критериями оптимальности совместной минимизации: минимум числа различных термов (конъюнкций или дизъюнкций), используемых для покрытия всех функций системы и минимум рангов этих термов. При этом один и тот же терм может входить в покрытие нескольких функций.

D2:

Q	x_1x_2 00	01	11	10
000	0	0	0	0
001	1	1	0	0
011	1	1	1	1
010	-	-	-	-
110	-	-	-	-
111	-	-	-	-
101	1	1	1	0
100	0	0	0	0

D1:

Q	x_1x_2 00	01	11	10
000	0	0	0	0
001	0	0	1	1
011	0	0	0	0
010	-	-	-	-
110	-	-	-	-
111	-	-	-	-
101	0	0	0	1
100	0	0	0	0

D0:

Q	x_1x_2 00	01	11	10
000	1	1	1	1
001	1	1	1	1
011	1	1	1	1
010	-	-	-	-
110	-	-	-	-
111	-	-	-	-
101	1	0	0	1
100	0	0	0	0

Рисунок 8

В результате минимизации получим ДНФ функций возбуждения

$$D2 = q_1 + q_0 \cdot \bar{x}_1 + q_2 \cdot q_0 \cdot x_2,$$
$$D1 = q_2 \cdot q_0 \cdot x_1 \cdot x_2 + q_2 \cdot q_1 \cdot q_0 \cdot x_1,$$
$$D0 = q_2 + x_2 \cdot q_0.$$

Очевидно, что минимизация функций выходов невозможна, так как каждая из них представлена только одним минтермом

$$y1 = \bar{q}_2 \cdot \bar{q}_1 \cdot q_0,$$
$$y2 = q_2 \cdot q_1 \cdot q_0,$$
$$y3 = q_2 \cdot q_1 \cdot q_0,$$
$$y4 = q_2 \cdot q_1 \cdot q_0.$$

5.3 Формирование функциональной схемы управляющего автомата

На основе полученных функций возбуждения и функций выходов можно построить функциональную схему микропрограммного автомата Мура. На практике чаще всего используют базисы Буля (элементы И, ИЛИ, НЕ), Шеффера (элементы И-НЕ) и Пирса (элементы ИЛИ-НЕ). Качество решения задачи синтеза КС оценивают по затратам оборудования и быстродействию.

Для примера используем базис логических элементов Буля. Схему управляющего автомата получаем путем соединения входов и выходов КС с выходами и входами D-триггеров. Построенная схема приведена в приложении Д.

6 Методические указания по анализу работы операционного устройства

Для проверки правильности работы операционного устройства при выполнении микропрограммы рекомендуется выполнить анализ его работы методом имитационного моделирования в среде программы OrCAD Simulate. Процесс моделирование выполняется аналогично моделированию ОА. Отличия определяются только готовностью схемы УА, которой он и должен быть представляется в модели. Поэтому в сформированный ранее проект необходимо внести следующие изменения:

- удалить из проекта VHDL - модель УА;
- добавить в проект функциональную схему УП;
- удалить из схемы операционного устройства (рисунок 6) иерархические блоки H2 (УА) и заново его создать, но указать тип Schematic View.

Результаты моделирования рассмотренного операционного устройства при умножении чисел *IA* = - 11 = 8Bh и *IB* = +13 = 0Dh приведены на рисунке 9. Результат *OS* = 811Eh, что является правильным значением. На рисунке 9 приведены диаграммы управляющих *(y1-y4),* осведомительных *(x1-x2)* сигналов, а также сигналов, кодирующих внутренние состояния УА (*4-6*).

Заключение о правильности выполненного проекта выполняют по результатам моделирования для нескольких значений операндов. При получении неправильных результатов, ошибки следует искать в схеме УА, так как схема ОА уже была протестирована ранее.

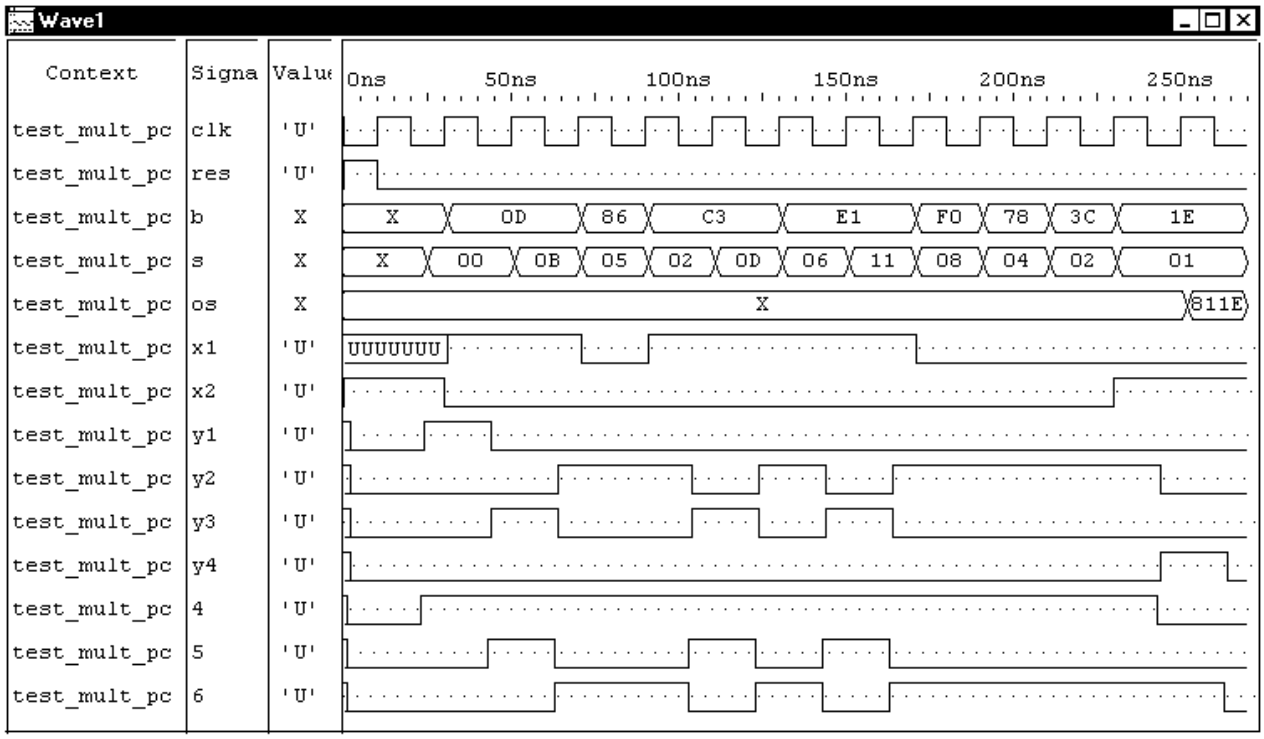


Рисунок 9

7 Вопросы к защите курсового проекта по теории автоматов

7.1 В чем состоит проектирование автомата?

7.2 Каким образом связаны понятия «алгоритм» и «автомат»?

7.3 Как можно представить модель операционного устройства В.М. Глушкова?

7.4 В чем состоит принцип микропрограммного управления?

7.5 Какой автомат называется микропрограммным?

7.6 Что такое микропрограмма?

7.7 В какой форме представляются микропрограммы?

7.8 Как задается функция операционного автомата?

7.9 Что понимают под совместимостью микроопераций?

7.10 Чем определяется функциональная и структурная совместимость?

7.11 Как формируется структура операционного автомата?

7.12 Как задается функция управляющего автомата?

7.13 Какие формы задания конечного автомата пригодны для выполнения его синтеза?

7.14 Как можно получить эти формы для автоматов Мили и Мура?

7.15 Чем отличаются автоматы Мили и Мура?

7.16 Каким образом обеспечевается совместная устойчивая работа операционного и управляющего автоматов в операционном устройстве?

7.17 Что понимают под внутренним состоянием конечного автомата?

7.18 Как выполняется кодирование внутренних состояний конечного автомата?

7.19 Чем отличается структурный синтез автоматов Мили и Мура.

7.20 Каким образом можно проверить правильность формирования микропрограммы и операционного устройства?

Список использованных источников

1 СТП 101-00. Общие требования и правила оформления выпускных квалификационных работ, курсовых проектов (работ), отчетов по РГР, по УИРС, по производственной практике и рефератов. – Оренбург: ОГУ, 2000. – 62 с.

2 Инженерная и компьютерная графика: Учеб. для вузов/ Под ред. Э.Т. Романычевой. – М.: Высш. шк., 1996. – 376 с.

3 Савельев А.Я. Прикладная теория цифровых автоматов. - М.: Высшая школа,1987. - 272 с.

4 Постников А.И. Основы теории цифровых автоматов: Учеб. пособие для вузов по спец. Вычислительные машины, комплексы, системы и сети. - Красноярск, 2000. - 296 с.

5 Разевиг В.Д.Система проектирования цифровых устройств OrCAD.– М.: «Солон-Р», 2000.

Приложение А
(обязательное)
Пример оформления бланка задания на курсовой проект

Министерство образования Российской Федерации
Государственное образовательное учреждение высшего профессионального образования «Оренбургский государственный университет»

Факультет информационных технологий

Кафедра вычислительной техники

Задание на курсовой проект

Микропрограммный автомат

Исходные данные: Операция - умножение чисел в формате с ФТ;
Код операндов и результата – прямой;
Метод - обработки разрядов множителя, начиная с младших при неподвижном множимом (со сдвигом частных сумм);
Тип автомата: автомат Мура;
Тип элемента памяти: D-триггер.

Разработать: 1) Граф-схему алгоритма микропрограммы;
2) Функциональную схему операционного автомата;
3) Функциональную схему управляющего автомата.

Дата выдачи задания "____"_____200__г.

Руководитель Петров А.Б.

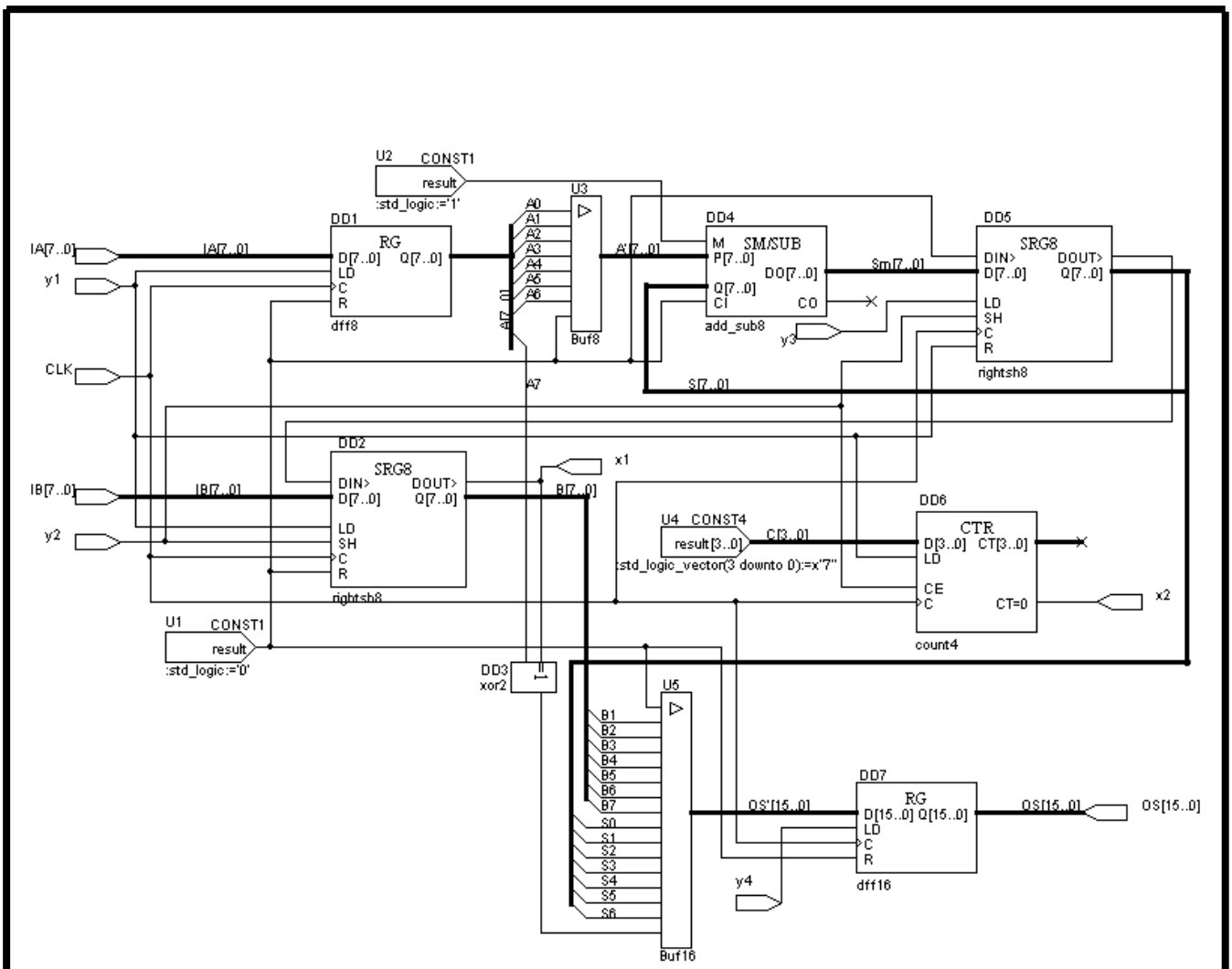
Исполнитель

студент группы 01ВМК1 Иванов Д.И.

Срок защиты проекта "____ "_____200__г.

Приложение Б
(обязательное)

Пример функциональной схемы операционного автомата



					ГОУ ОГУ 2201.4102.01 Э2	
Изм	Лист	№ докум.	Подпись	Дата	Операционный автомат Схема электрическая функциональная	Литера / Масса / Масшт.
Разраб.						
Провер.						Лист 1 / Листов 1
Т. Контр.						
Н. Контр.						ФИТ 01ВМК1
Утв.						

Приложение В
(справочное)

VHDL-модель управляющего автомата Мура

```
-- Moore State machines
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity UA is
  port(
      x1, x2  : in std_logic;
          ck, reset : in std_logic;
      y1, y2, y3, y4 : out std_logic );
end UA;

architecture behavior of UA is
  type state_type is (s0, s1, s2, s3, s4);
  signal current_state, next_state : state_type;
begin
  --  register block
  process (ck, reset)
  begin
      if reset = '1' then
        current_state <= s0;
      elsif ck = '1' and ck'event then
        current_state <= next_state;
      end if;
  end process;

  -- state machine process
  process (current_state, x1, x2)
  begin
    -- default outputs
    y1 <= '0';
    y2 <= '0';
    y3 <= '0';
    y4 <= '0';

case current_state is

      when s0 =>
          next_state <= s1;

      when s1 =>
          y1 <= '1';
```

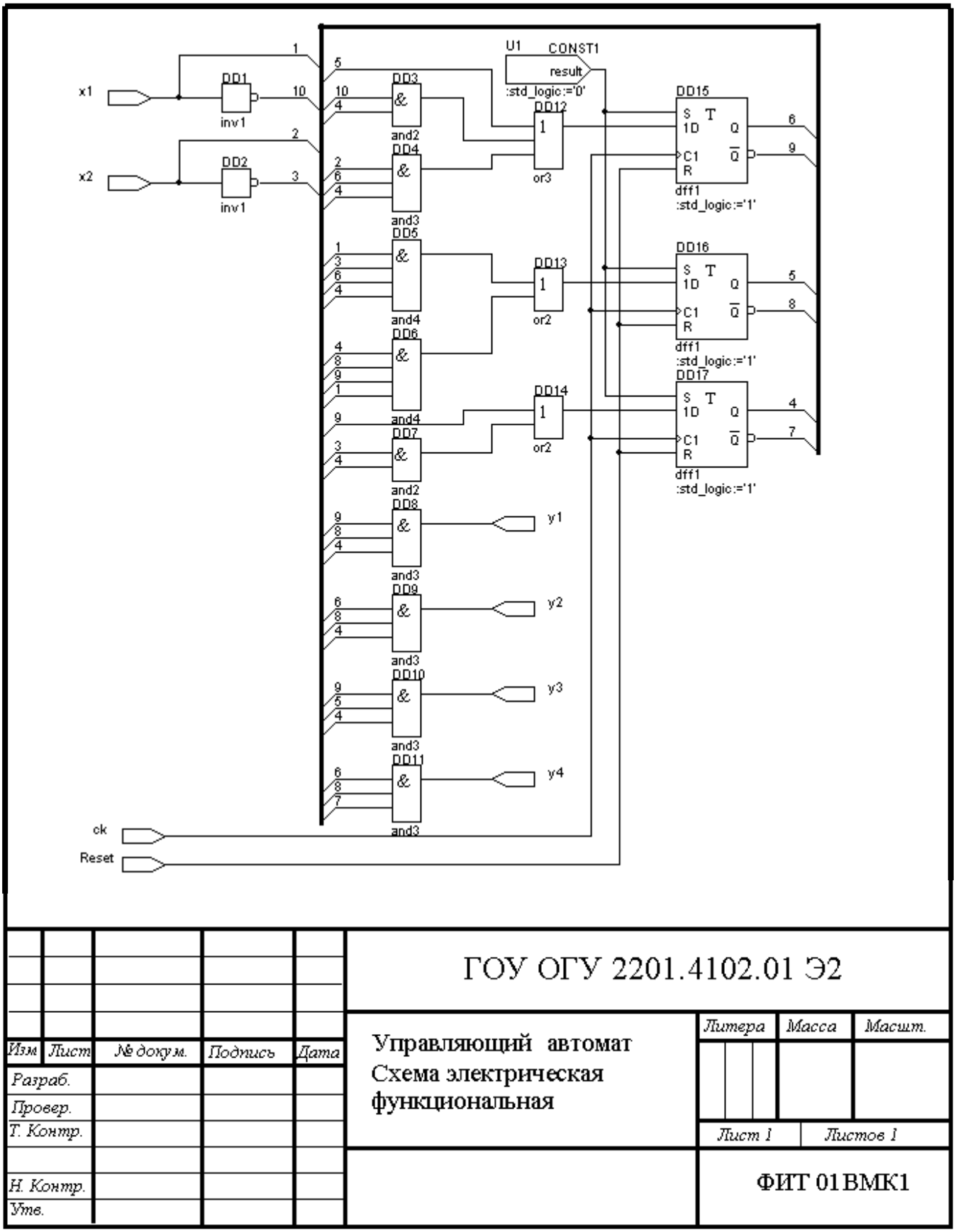
```
        if      x1 = '1' then next_state <= s2;
        else                  next_state <= s3;
        end if;

      when s2 =>
        y3 <= '1';
        next_state <= s3;

      when s3 =>
        y2 <= '1';
         if      x2 = '1' then next_state <= s4;
        elsif   x1 = '1' then next_state <= s2;
        else                  next_state <= s3;
        end if;

       when s4 =>
        y4 <= '1';
        next_state <= s0;

       end case;
  end process;

end;
```

Приложение Г
(справочное)
VHDL-модель управляющего автомата Мили

```
-- Mealy State Machines

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity UA is
  port(
     x1, x2  : in std_logic;
        ck, reset : in std_logic;
     y1, y2, y3, y4 : out std_logic );
end UA;

architecture behavior of UA is
  type state_type is (s0, s1, s2, s3, s4);
  signal current_state, next_state : state_type;
  signal iy1, iy2, iy3, iy4 : std_logic;

begin
  --  register block
  process (ck, reset)
  begin
     if reset = '1' then
       current_state <= s0;
     elsif ck = '1' and ck'event then
       current_state <= next_state;
       y1 <= iy1 after 2 ns;
       y2 <= iy2 after 2 ns;
       y3 <= iy3 after 2 ns;
       y4 <= iy4 after 2 ns;
     end if;
  end process;

  -- state machine process
  process (current_state, x1, x2)
  begin
    -- default outputs
    iy1 <= '0';
    iy2 <= '0';
    iy3 <= '0';
```

```
    iy4 <= '0';
    case current_state is

      when s0 =>
          iy1 <= '1';
          next_state <= s1;

      when s1 =>
         if x1 = '1' then iy3 <= '1';
                          next_state <= s2;
           else           iy2 <= '1';
                          next_state <= s3;
         end if;

      when s2 =>
          iy2 <= '1';
          next_state <= s3;

      when s3 =>
         if    x2 = '1' then iy4 <= '1';
                             next_state <= s4;
         elsif x1 = '1' then iy3 <= '1';
                             next_state <= s2;
         else                iy2 <= '1';
                             next_state <= s3;
         end if;

      when s4 => next_state <= s0;
    end case;
  end process;

end;
```

Приложение Д
(справочное)
Функциональная схема управляющего автомата Мура



					ГОУ ОГУ 2201.4102.01 Э2	
Изм	Лист	№ докум.	Подпись	Дата	Управляющий автомат Схема электрическая функциональная	Литера / Масса / Масшт.
Разраб.						
Провер.						Лист 1 / Листов 1
Т. Контр.						
Н. Контр.						ФИТ 01ВМК1
Утв.						

Лицензия N ЛР020716 от 02.02.93
Подписано в печать .....
Формат ..... Бумага .....
Усл. печ. л. 2,5 Тираж .... Заказ .....

Отпечатано в Оренбургском государственном университете
460352, г. Оренбург, ГСП, пр. Победы, 13,
Издательство ОГУ