МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА УиИТСиВТ

Искендерова Э.Г., Тетакаев У.Р.

УЧЕБНО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине

«Теория автоматов»

для студентов направления подготовки бакалавров

09.03.01–Информатика и вычислительная техника

МАХАЧКАЛА–2020

УДК 656.25

Учебно-методические указания к выполнению лабораторных работ по дисциплине «Теория автоматов» для бакалавров направления подготовки 09.03.01-«Информатика и вычислительная техника". Махачкала, ИПЦ ДГТУ, 2020. – 28 с.

Рассматриваются вопросы анализа функциональной структуры и синтеза дискретных устройств с применением программы Multisim 10.1. .

Составители: Искендерова Э.Т., асссистент,
Тетакаев У.Р., к.т.н., ст. преподаватель

Рецензенты: 1. Хазамова М.А., к.т.н., доцент кафедры ТиОЭ ДГТУ;

2. Кобзаренко Д.Н., д.т.н., завлабораторией информационных технологий в энергетике ИПГ ДФИЦ РАН

Рег. № 5026

Печатается по постановлению Ученого совета Дагестанского государственного технического университета от_____2019г.

ЛАБОРАТОРНАЯ РАБОТА №1. СИНТЕЗ УПРАВЛЯЮЩЕГО МИКРОПРОГРАММНОГО АВТОМАТА С ЖЁСТКОЙ ЛОГИКОЙ

1. Цель работы

1. Разработка графа микропрограммы вычислительной процедуры машинной операции для заданной системы микрокоманд.
2. Синтез схемы управляющего автомата для реализации микропрограммы.
3. Моделирование разработанной схемы автомата в программе Multisim 10.1.

2. Структурное представление устройств обработки информации в виде композиции двух автоматов – операционного и управляющего.

При описании работы широкого класса устройств обработки информации широко используется их представление в виде **композиции операционного (ОА) и управляющего автомата (УА).** Та кой подход впервые использовал В.М. Глушков (1962 г).

ОА принимает из внешней среды, хранит и преобразует (в соответствии с кодом заданной операцией (**КОП**)) входные слова из множества **D** и выдаёт во внешнюю среду результат (слова из множества **R**) .

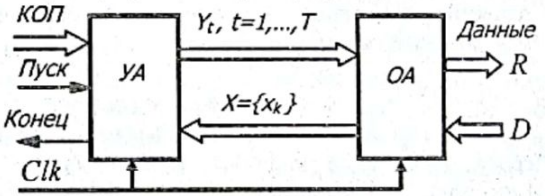


Рис. 1. Декомпозиция устройства обработки информации на операционный и управляющий блоки.

Структурно **ОА** состоит обычно из многоцелевых **управляемых комбинационных схем,** выполняющих какой-либо набор арифметических и логических операторов и **регистровой памяти** для хранения промежуточных результатов в процессе обработки информации. Задачей **УА** является выработка распределённой во времени последовательности управляющих сигналов, под воздействием которых в **ОА** и выполняется некоторая операция.

Каждое элементарное действие по преобразованию или передачи информации выполняемое за один такт автоматного времени назовём **микрооперацией.** Пусть $Y=\{y_1, \ldots, y_N\}$ – множество микроопераций, реализуемых в **ОА.** Они возбуждаются сигналами $y_1, \ldots, y_N$ из **УА** (для обозначения этих сигналов удобно использовать одни и те же символы). Совокупность микроопераций, выполняемых одновременно за один такт автоматного времени, образует **микрокоманду** $Y_t, t=1, \ldots, T.$

В частном случае, микрокоманда (МК) может состоять из одной микрооперации. В общем случае справедливо: $Y_t \subset Y$ т.е. множество Y вк лючает подмножество Y_t.

Для задания порядка следования МК Yt используются специальные переменные, называемые **логическими условиями** x_k из множества $X=\{x_1, \ldots, x_K\}$,

вырабатываемые операционным автоматом **ОА**. Проверка значения логического условия в каждом такте работы **УА** позволяет определить МК, выполняющуюся в следующем такте.

Такая последовательность реализации МК определяется **функциями перехода** – булевыми функциями a_{ij} $(i,j=1,\ldots,T)$, аргументами которых являются логические условия из множества $X=\{x_1,\ldots,x_K\}$. Свяжем с каждой МК Y_i множество функций перехода $(a_{i1},\ldots,a_{iT})$ таких, что если $a_{ii}=1$ после выполнения МК Y_i, то следующей будет выполняться МК Y_i. Множество функций перехода из одной и той же микрокоманды Y_i обладает **свойством ортогональности** (a_{ij} & $a_{ii}=0$ *при* $j\neq t$) **и полноты** $\bigvee_{j=1}^{T} a_{ij} = 1$.

Ортогональность говорит о том, что после данной МК при определённых значениях логических условий может выполниться только одна МК, а полнота – что она выполнится обязательно.

Описание операции в терминах микрокоманд и логических условий будем называть **микропрограммой этой операции,** а сам принцип разбиения выполняемой операции на микрокоманды (микрооперации) – носит название **принципа микропрограммного управления.** В свою очередь УА, осуществляющий управление ОА для реализации микропрограммы операции называется **микропрограммным автоматом.**

3. Граф микропрограммы.

Представление микропрограммы выполнения какой-либо операции в графическом виде получило название **графа микропрограммы (граф МП).** Граф МП представляет собой ориентированный граф, содержащий одну начальную и одну конечную вершины, а также произвольное множество промежуточных вершин, составленных из операторных и условных. Конечная, операторная и условная вершины могут иметь несколько входов. У начальной и операторной вершин - по одному выходу, у условной два, помеченные символами 1 и 0.

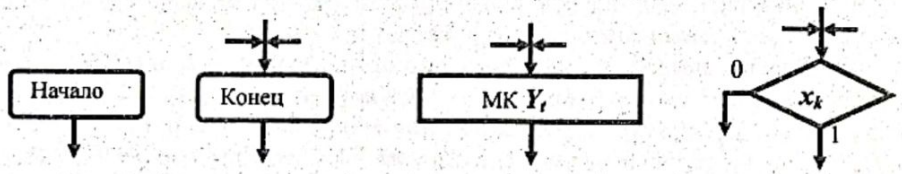


Рис. 2. Виды вершин графа МП: начальная, конечная, операторная и условная.

В каждой операторной вершине записывается оператор или МК Y_t - подмножество множества микроопераций $Y=\{y_1,\ldots,y_N\}$.

Условная вершина соответствует проверяемому логическому условию, в ней помещается один из элементов множества $X=\{x_1,\ldots,x_k\}$. Один из выходов условной вершины может соединяться с её входом, такие вершины будем

называть ждущими или возвратными. При составлении графа МП необходимо руководствоваться следующими правилами:

1) входы и выходы различных вершин соединяются дугами, с указанием направления передачи информации;

2) каждый выход соединяется только с одним входом;

3) для любой вершины графа МП должен быть, по крайней мере, один путь из этой вершины к конечной.

4. Программная модель ОА

Исходными данными для синтеза **УА** являются схема операционного автомата ОА с функциями, определенными в виде списка реализуемых микрокоманд, а также граф микропрограммы операции, заданной для реализации на этом **ОА.** К сожалению, форма проведения практикума по курсу с использованием программы моделирования Multisim 10.1, не даёт возможности воспользоваться каким-либо операционным блоком для выполнения арифметико-логических операций. С другой стороны, использование реальной схемы операционного блока требует наличие определённого интерфейсного блока между **УА** и **ОА** – блока, «расщепляющего» управляющий сигнал микрокоманды Y_t на некоторое семейство микроинструкций.

На рис. 3 представлена одна из возможных декомпозиций УА, включающая три составные части: дешифратор кода вычислительной процедуры, собственно управляющий автомат, генерирующий управляющую последовательность микрокоманд Y_t, и формирователь микроинструкций для управления отдельными узлами операционного устройства (точнее, настройки **ОА** на выполнение заданной микрокоманды Y_t):

$$Y_t \Rightarrow \left(\{b_i\},\{c_j\},\ldots,\{h_p\}\right)_t . \qquad (1)$$

Такой формирователь можно представить в виде памяти, содержание ячейки которой соответствует правой части соотношения (1). Адрес ячейки памяти формируется шифратором, преобразующим унарный код с выхода УА в двоичный. Здесь предполагается, что разрядность унарного кода (т.е. количество входов CD) определяется числом различных микрокоманд Y_t в микропрограмме операции, заданной кодом операции.

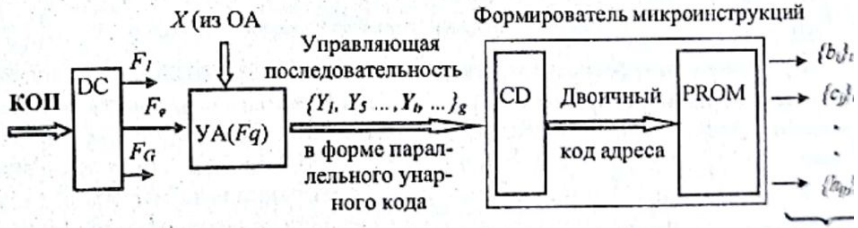


Рис. 3. Структура управляющего автомата

В дальнейшем, для выполнения работы в качестве операционного бло примем ***программную модель***, заданную системой микрокоманд, включающ микрокоманды пересылки, двоичной арифметики, логической обработки, сд

га и вращения (табл.1). На основе данных микрокоманд и будет проводиться разработка графа микропрограммы. Для обозначения каждой микрокоманды введём **мнемокод** ***[mnemonic]***, используя для этого стандартный в практике программирования приём - краткая последовательность букв или символов (от 3 до 5), представляющая собой сокращение соответствующих англоязычных слов или их аббревиатуру, отражающих содержательный смысл микрокоманды. Такой подход с выбором ОА позволяет избавиться от необходимости рассмотрения блока формирования управляющих микроинструкций и сосредоточиться, исключительно, на среднем звене рис. 3

Логические условия $X=\{x_k\}$ $(k=1, ...,K)$ в программной модели принято называть **флагами условий** (табл. 2). Определение флагов условий дано на рис. 4.

Таблица 2

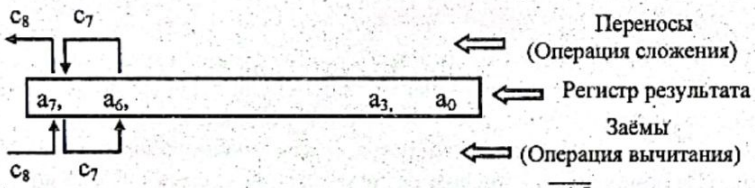
Флаги условий	*CF*	*ZF*	*SF*	*OF*
Логические условия $\{x_k\}$	x_1	x_2	x_3	x_4

Cf – Carry Fl – перенос из старшего разряда регистра результата при сложении или заём для старшего разряда при вычитании.

Of – Overflow – арифметическое переполнение результата при сложении или вычитании чисел в дополнительном коде. При операциях с беззнаковыми числами признаком переполнения является значение флага *Cf* (1 – переполнение, 0 – нет).

Sf – Sign Fl – флаг знака результата.

Zf – Zero Fl – флаг нулевого результата для логических и арифметических операций.



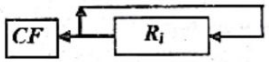
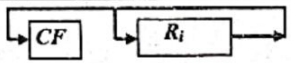
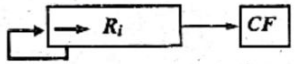
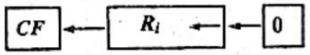
$$Cf = c_8,\ Sf = a_7,\ Of = c_8 \oplus c_7,\ Zf = \overline{\vee_{i=0}^{i=7} a_i},$$

Рис. 4. Определение флагов условий

Набор используемых в ОА микрокоманд МК и анализируемых логических условий должен быть **полным,** т.е. обеспечивать возможность составления микропрограмм для всего множества вычислительных процедур, предназначенных для реализации на данном ОА. Кроме того он должен быть **эффективным,** т.е. обеспечивать минимизацию оного из показателей (время выполнения микропрограммы, аппаратные затраты) при заданных ограничений на значение другого.

Отметим, что поиск набора МК, удовлетворяющих таким требованиям, представляет очень сложную задачу. На практике он подбирается опытным путём. Представленный в табл. 1 набор команд ориентирован на реализацию простых вычислительных процедур, реализуемых в лабораторном практикуме по курсу

Таблица 1

Мнемокод МК	Содержание	Влияние на флаги
IN R_i, *data8*	$(R_i) \leftarrow data8;\ \ i=1,..,8$	Не влияет
MOV R_i, R_j	$(R_i) \leftarrow (R_j);\ \ i,j=1,..,8$	Не влияет
MOV R_i, *const8*	$(R_i) \leftarrow const8;\ \ i=1,...,8$	
ADD R_i, R_j	$(R_i) \leftarrow (R_i) + (R_j);\ \ i,j=1,..,8$	Влияет на все флаги
ADC R_i, R_j	$(R_i) \leftarrow (R_i) + (R_j) + CF;\ \ i,j=1,..,8$	
SUB R_i, R_j	$(R_i) \leftarrow (R_i) - (R_j);\ \ i,j=1,..,8$	
SBB R_i, R_j	$(R_i) \leftarrow (R_i) - (R_j) - CF;\ \ i,j=1,..,8$	
CMP R_i, R_j	$(R_i) - (R_j);\ \ i,j=1,..,8$	
CMP R_i, *const8*	$(R_i) - const8;\ \ i=1,..,8$	
DEC R_i	$(R_i) \leftarrow (R_i) - 1;\ \ i=1,..,8$	Влияет на все флаги, кроме *CF*
INC R_i	$(R_i) \leftarrow (R_i) + 1;\ \ i=1,..., 8$	
AND R_i, R_j	$(R_i) \leftarrow (R_i) \wedge (R_j);\ \ i,j=1,..,8$	Влияет на все флаги, *OF*=*CF*=0
OR R_i, R_j	$(R_i) \leftarrow (R_i) \vee (R_j);\ \ i,j=1,...,8$	
XOR R_i, R_j	$(R_i) \leftarrow (R_i) \oplus (R_j);\ \ i,j=1,..,8$	
NEG R_i	$(R_i) \leftarrow (0 - R_i);\ \ i=1, ..., 8$	Влияет на все флаги
ROL R_i		Только на флаг *CF*
ROR R_i		
SAR R_i		Влияет на все флаги. *OF*=1, если при сдвиге произошло изменение старшего бита
SAL R_i		

Некоторые замечания к использованию предложенного набора МК.

1. Форматы микрокоманд представлены как одноопёрандными, так двухоперандными, использующимися для выполнения бинарных операций. При этом сам операнд представлен содержимым одного из восьми регистров (R_i, .., R_j) **байтового типа.** При выполнении бинарных операций регистр, следующий в записи за мнемокодом МК, является приёмником результата (или как считают в программировании - местом назначения ***(destination)***).

2. Все операнды, используемые в МК, поделим на три группы: **внешние** (они имеют определённые символьные обозначения в вычислительных процедурах), **промежуточные**, получаемые в процессе вычислений в ОА, и некото-

рые **константы**, необходимые для алгоритмической реализации процедуры и вводящиеся непосредственно в качестве операнда **источника** (например, *CMP Ri, const8)*.

Так, например, при реализации процедуры

$$S=\begin{cases}4B-A, \text{если } B<0,\\ -B, \quad \text{если } 0\le B<10\end{cases}$$

операнды (A, B) являются внешними и могут вводиться микрокомандой **IN Ri, data8** в какой-либо регистр ОА:

IN Ri, A *;A→ Ri*

Операнд S, представляющий результат операции, размещается в регистр R1 и остаётся в ОА.

Численная константа «10», может использоваться в микрокомандах, как передачи (*MOV Ri, const8)*, так и сравнения (*CMP Ri, const8)*, например:

CMP *Ri*,10 *;Ri-10, здесь 10* - десятичная запись константы,

CMP *Ri*,00001010 ;00001010b - двоичная запись константы.

5. Пример на построение графа микропрограммы.

Задание. Подсчитать число «0» в байте (операнд А).

Алгоритм процедуры в описательной форме.

Содержимое регистра R_i, в котором находится искомый байт, подвергается циклическому сдвигу (8 раз) влево МК ***ROL R_1***. Выдвигаемый крайний левый бит фиксируется во флаге ***CF***, нулевое значение которого инициализирует инкремент счётчика нулевых разрядов регистра R_i.

Произведём назначение используемых в МП регистров.

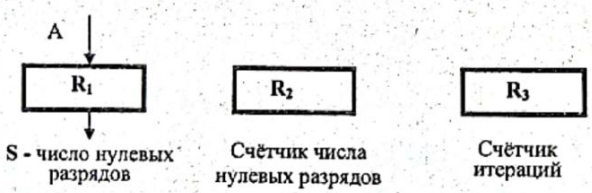


Рис. 5. Назначение используемых регистров.

• Исследуемый байт заносится в регистр R_1, регистр используется для счёта числа нулевых разрядов, а регистр R_3 выполняет функцию счётчика итераций цикла сдвига.

По окончанию процедуры результат (число нулевых разрядов) помещается в регистр R1.

Граф микропрограммы с необходимыми комментариями приведен на рис. 6. Исполнение микропрограммы начинается при выполнении логического условия ПУСК=1, которое реализуется воздействием на УА одноименным командным сигналом.

Имея в своем распоряжении граф микропрограммы можно приступить к синтезу управляющего автомата УА с жесткой логикой, функционирование которого задается моделью Мили или Мура.

6. Синтез управляющего автомата (УА)

Собственно выполнение работы и определяется синтезом управляющего автомата, состоящего из следующих этапов [5]:

Кодированное представление графа микропрограммы или получение граф - схемы алгоритма (ГСА) работы УА.

Разметка ГСА для определения состояний УА, функционирующего в соответствии с моделью автомата Мили (Мура).

Построение графа автомата Мили (Мура).

Выбор типа триггера и кодирование состояний автомата.

Интерпретационный метод синтез УА на основе структурной таблицы

5.1. Составление структурной таблицы автомата (прямой или обратной).

5.2. Запись логических выражений для выходных сигналов управления Yt и сигналов возбуждения триггеров φj.

5.3. Построение структурной схемы автомата.

5.4. Построение функциональной схемы автомата.

Рис. 6. Граф микропрограммы подсчета числа «нулей» в байте.

7. Граф - схема алгоритма (ГСА) работы УА.

Граф - схема алгоритма (ГСА) определяет закон функционирования УА в процессе реализации микропрограммы в операционном устройстве. ГСА (функция УА) представляет собой кодированную форму графа МП и получает-

ся путем замены микрокоманд, указанных в операторных вершинах, управляющими сигналами Yt, а флагов условий в условных вершинах - логическими условиями xk из табл. 2 (сигнал ПУСК также относится к множеству X={xk}). Полученная таким образом, ГСА, представлена на рис. 7, а.

В свою очередь, функция ОА, содержащаяся в графе МП, представляется таблицей, содержащей список микрокоманд и список логических условий (табл. 3).

Таблица 3

Идентификатор МК	МК	Логическое условие	Флаг условия
Y1	$IN\ R_1, A$		
Y2	$MOV\ R_3, 08h$		
Y3	$MOV\ R_2, 00h$	X1	CF
Y4	$ROL\ R_1$	X2	ZF
Y5	$INC\ R_2$		
Y6	$DEC\ R_3$		
Y7	$MOV\ R_1, R_2$		

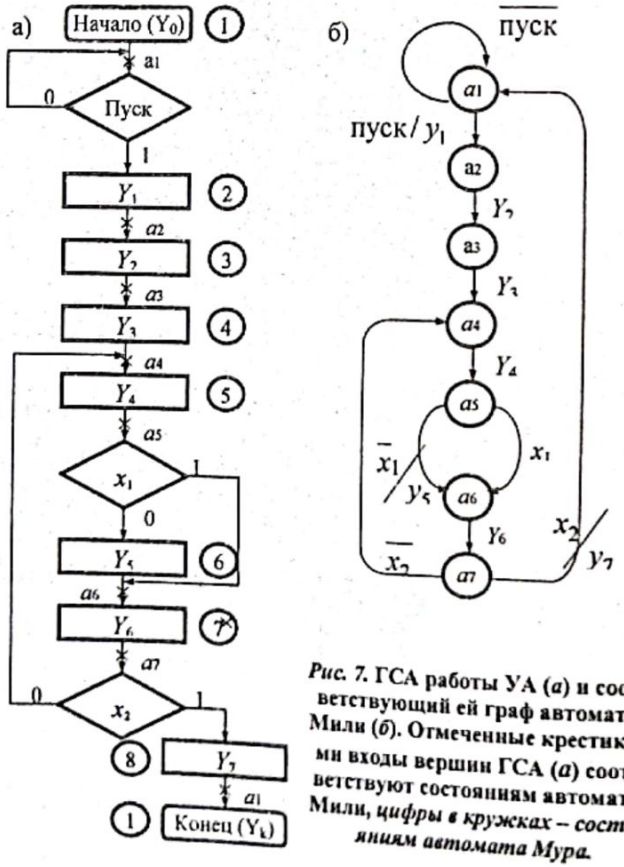


Рис. 7. **ГСА работы УА (*а*) и соответствующий ей граф автомата Мили (*б*). Отмеченные крестиками входы вершин ГСА (*а*) соответствуют состояниям автомата Мили, *цифры в кружках – состояниям автомата Мура.***

8. Матричная схема алгоритма (МСА) работы УА.

Матричная схема алгоритма МСА, соответствующая ГСА, есть прямоугольная матрица, строки которой отмечены символами МК $Y_0, Y_1, \ldots, Y_T$, а столбцы – символами $Y_0, Y_1, \ldots, Y_T, Y_k$. В этой матрице (см. табл. 4) на пересечении строки Y_i и столбца Y_j стоит функция перехода a_{ij} от МК Y_i к Y_j. Требования, которым должны удовлетворять функции переходы, были сформулированы выше в П.2. Повторим их ещё раз:

1. Конъюнкции 2-х любых функций перехода для i-ой строки равны 0 (свойство единственности перехода) $\alpha_{ij} \,\&\, \alpha_{ik} = 0 \ (k \neq j)$, (2)

2. Дизъюнкция всех функций перехода для i-ой строки тождественно равна 1 (свойство обязательности перехода) $\bigvee_{j=1}^{T} \alpha_{ij} = 1$ (3)

Просмотр значений функций (2) и (3) для всех строк МСА подтверждает **корректность** искомого графа ГСА.

Таблица 4.

	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_k
Y_0	$\neg Пуск$	*Пуск*							
Y_1			1						
Y_2				1					
Y_3					1				
Y_4						x_1	$\neg x_1$		
Y_5							1		
Y_6					$\neg x_2$			x_2	
Y_7									1

Матричные схемы алгоритмов используются также при объединении различных ГСА в одну при разработке многопрограммных автоматов.

9. Интерпретация ГСА автоматами Мили и Мура.

9.1. Интерпретация ГСА автоматом Мили.

Как известно, функционирование автомата Мили задается двумя функциями - функцией перехода δ и выхода λ. Применительно к рассмотренной модели автомата запишем:

$$a(t+1) = \delta\,(a(t), X(t)), \qquad Y(t) = \lambda\,(a(t), X(t)), \qquad (4)$$

где t=0, 1, 2, - автоматное время; $a(t) \in A$ состояние автомата из множества $A=\{a_1, a_2, \ldots, a_M\}$ каждое из которых задаётся комбинацией триггеров $Q_1, \ldots, Q_R$;

$X = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_K\}$ - вектор входных сигналов (признаков или условий) $\tilde{x}_k = \{0,1\}$, $k=1,\ldots K$;

$Y=Y_t$ принадлежит множеству управляющих последовательностей для заданной микропрограммы, например, $\{Y_2, Y_3, \ldots\}$.

Необходимый набор состояний автомата определяется путем разметки ГСА по следующим правилам [Л. 1,6]:

– символом a_1 отмечается вход вершины, следующей за начальной, а также вход конечной вершины;

– входы вершин, следующих за операторными, отмечаются символами a_2, a_3, ... a_m, при этом входам различных вершин даются различные символы.

Если отметкам a_1, ..., a_m (рис. 7, *а*) поставить в соответствие вершины графа и соединить их дугами, число и направление которых определяется всевозможными переходами между одноименными отметками ГСА, то получим граф автомата Мили (рис. 7, *б*). Каждый переход может включать произвольное число условных вершин, но не более одной операторной. Каждая дуга помечается символом x_k (без инверсии, если путь проходит через выход условной вершины, отмеченный символом "1")[1] и выходным сигналом Y_t, если путь проходит через операторную вершину.

Работа автомата по выполнению микропрограммы является циклической, поэтому рассмотрим его функционирование в течение одного машинного такта, совпадающего с одним тактом синхронизации сигнала ***CLK***. Будем также считать, что временные такты работы УА и ОА совпадают во времени. Последнее условие являются очень существенным и диктуется возможностями моделирования УА в программе Multisim 10.1.

Развертка ГСА в тактах автомата Мили приведена на рис. 8, а временная диаграмма работы – на рис. 9, *а*.

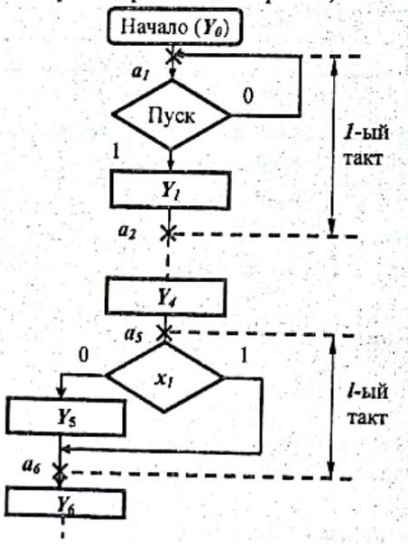


Рис. 8. Развёртка ГСА в тактах автомата Мили

Когда автомат не работает (Пуск=0), он находится в начальном состоянии a_1. При запуске (Пуск=1) автомат сохраняет состояние a_1 в течение всего такта, в начале которого вырабатывается управляющий сигнала Y_1, инициирующий в ОА выполнение соответствующей МК. В конце такта ОА при *Clk=1* фиксирует результаты реализации МК Y_1.

Перейдём к описанию *i*-го такта (см. рис. 8)

В начале *i*-го машинного такта (отрицательный фронт сигнала ***CLK***) автомат переходит в новое состояние a_5 и, руководствуясь значением логического условия x_1 (предыдущий такт, МК Y_4 - ***ROL*** R_1, флаг ***CF***), готовится выполнить действия, соответствующие одному из путей перехода $a_5 \rightarrow a_6$. Пускай

[1] В общем случае дуга помечается конъюнктивным термом перехода $X(a_m a_s)$, если переход содержит несколько условных вершин.

x_1=0. Тогда, спустя некоторое время задержки после перехода автомата в состояние a_5, им будет выработан сигнал Y_5 , под действием которого ОА выполнит МК Y_5 - ***INC*** R_2. В конце *i*-го такта, по завершению МК Y_5 (*Clk=1*) ОА зафиксирует у себя в регистре флагов ***Fl*** логические условия, соответствующие этой команде, которые могут быть использованы УА для интерпретации ГСА в следующем *(i+1)*- такте.

Что является характерным для автомата Мили. Условия $X^{(i-1)}$ выработанные в предыдущем *(i-1)*-м такте **должны быть неизменными в течение всего *i*-го такта**, так как в автомате Мили вид выполняемой микрокоманды зависит не только от состояния автомата (как в автомате Мура), но и от значения признаков $X^{(i-1)}$. Достигается это использованием триггеров с динамическим управлением записью информации для реализации регистра флагов ***Fl*** в ОА.

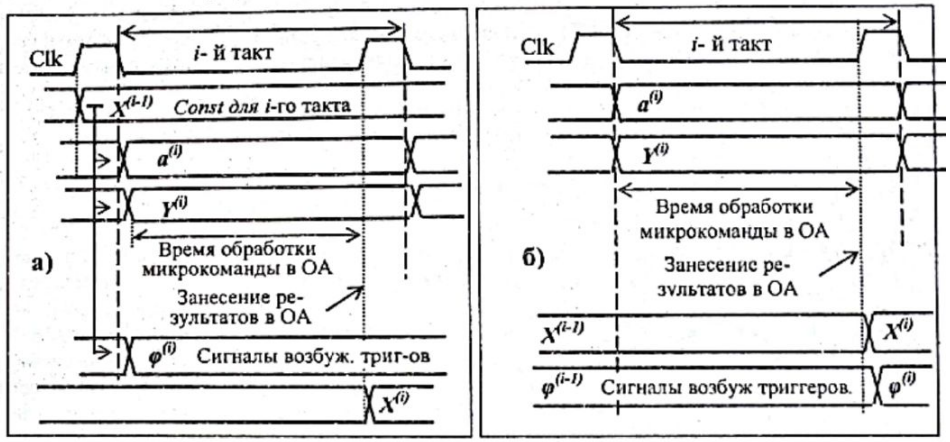


Рис. 9. Временные диаграммы работы ОА под управлением автомата Мили (*а*) и Мура (*б*)

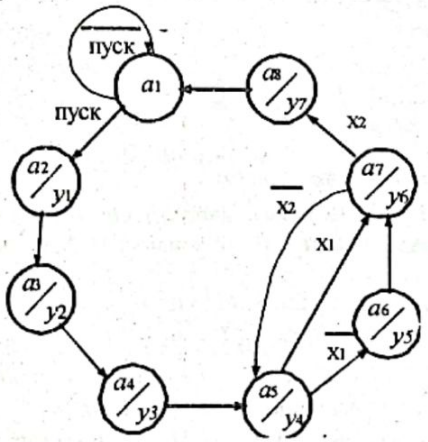


Рис. 10. Граф автомата Мура, для ГСА рис. 7, а

9.2. Интерпретация ГСА автоматом Мура.

ГСА может быть также интерпретирована и автоматом Мура, которому свойственен следующий закон функционирования:

$$a(t+1) = \delta\,(a(t), X(t)),\ Y(t) = \lambda\,(a(t)). \quad (5)$$

Поскольку в автомате Мура выходные сигналы Y_t связаны только с состояниями автомата, то каждой операторной вершине графа ГСА следует поставить в соответствие одно из состояний a_2, a_3, ... Символом a_1 помечаются начальная и конечная вершины. На ГСА рис. 7, *а* символы состояний a_1, ..., a_8 размещены в кружках при соответствующих вершинах.

Граф автомата Мура, интерпретирующий ГСА рис. 7, *а*, представлен на рис. 10. В отличие от графа автомата Мили, в графе автомата Мура выходные сигналы помещаются внутри кружка вместе с состоянием a_j. В общем случае автомат Мура имеет большее число состояний, чем автомат Мили, поэтому его реализация требует больших аппаратных затрат. Однако, взаимодействие управляющего автомата Мура с операционным устройством осуществляется по более простому алгоритму (рис. 11) и отражено на временной диаграмме рис. 9, *б*.

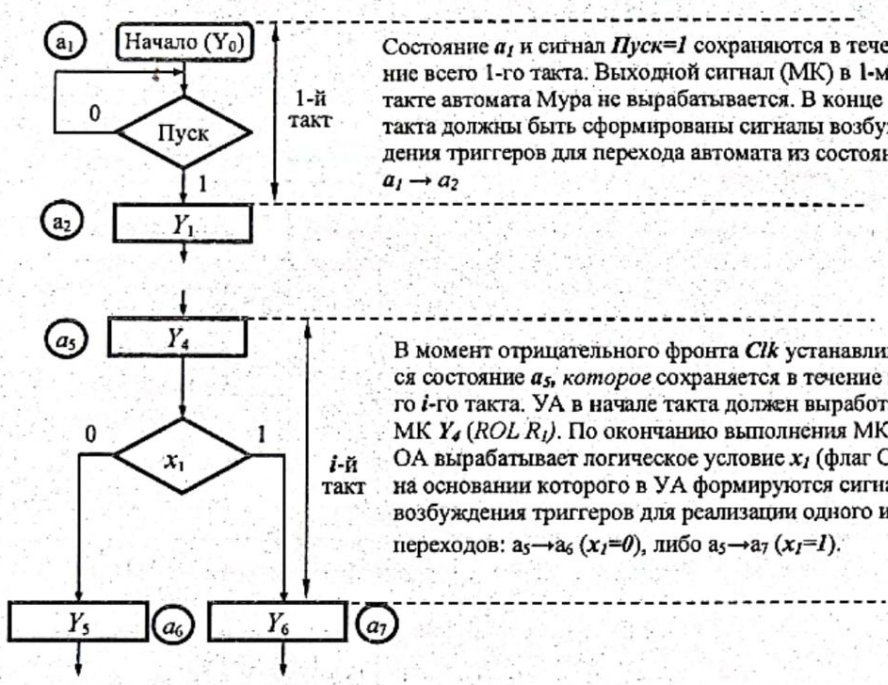


Рис. 11. Развёртка ГСА в тактах автомата Мура

10. Кодирование состояний автомата

Количество триггеров, требуемых для организации памяти управляющего автомата, определяется ближайшим сверху целым от двоичного логарифма числа состояний M, т.е.:

$$R = \lceil \log_2 M \rceil . \qquad (6)$$

Каждому состоянию a_j (т.е. вершине графа автомата) должна соответствовать одна определенная комбинация значений $Q_1, ..., Q_R$. Различают два подхода к кодированию состояний автомата: случайное и экономичное (существует еще понятие противогоночного кодирования, которое актуально для асинхронных автоматов и здесь не рассматривается). Экономичное кодирование, как правило, приводит к уменьшению сложности комбинационной части, которая формирует выходные сигналы управления и сигналы возбуждения элементов памяти.

Существует много способов экономичного кодирования, однако в большинстве случаев они весьма трудоемки в практическом использовании и требуют специальных знаний из области теории дискретной математики. Характерной особенностью этих методов является стремление либо минимизировать число переключений элементов памяти на всех переходах автомата, либо обеспечить условия, при которых каждое изменение состояния автомата вызывалось бы действием как можно меньшего числа сигналов возбуждения элементов памяти. Из простых способов, относящихся к классу экономичного кодирования, отметим три:

1) соседнее кодирование,
2) кодирование с ослабленной зависимостью от входных сигналов,
3) приоритетное кодирование логически смежных состояний.

■ При соседнем кодировании любые два смежных состояния в графе автомата кодируются наборами $Q_1, ..., Q_R$, отличающимися состояниями лишь одного элемента (в теории кодирования такие кодовые комбинации характеризуются минимальным кодовым расстоянием $d = 1$). Обычно для реализации соседнего кодирования граф автомата накладывают на карту Карно соответствующего ранга, как это показано на рис. 12 применительно к графу автомата Мили рис. 7, *б*.

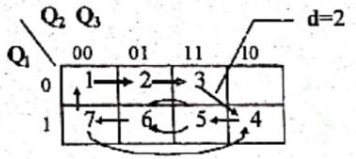


Рис. 12. Применение соседнего кодирования для состояний графа автомата Мили рис.7, *б* (переход «3→4» не удовлетворяет требованиям соседнего кодирования)

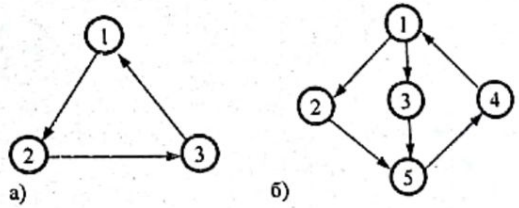


Рис. 13. Графы, не допускающие соседнее кодирование

Соседнее кодирование, требования к которому сформулированы в работе [1], не всегда возможно. На рис. 13 показаны два графа, не допускающие способ соседнего кодирования.

■ Кодирование с ослабленной зависимостью от входных переменных группы $X=\{x_k\}$ предусматривает размещение подмножества состояний автомата $A(a_m)$, образованного всевозможными переходами из состояния a_m, в одном столбце или строке карты Карно.

На рис. 14 показан один из возможных вариантов использования данного способа для автомата Мура (рис. 10). Здесь подмножества $A\{a_3\} \rightarrow (a_6, a_7)$ и

$A\{a_7\} \rightarrow (a_5, a_8)$ расположены в одной строке. Остальные переходы удовлетворяют соседнему кодированию.

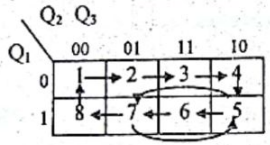


Рис. 14. Кодирование с ослабленной зависимостью от входных переменных состояний **автомата Мура** (рис. 10).

Приоритетное кодирование **логически смежных состояний.**

Правило 1. Два состояния автомата, из которых возможны переходы в одно и то же третье состояние, называются логически смежными (ЛСС-1).

Правило 2. Два состояния, в которые может быть осуществлён переход из одного какого-либо состояния, также называются логически смежными (ЛСС-2).

При экономичном кодировании ЛСС должны кодироваться соседними кодовыми комбинациями (т.е. различаться значением одного триггера). Если при кодировании нельзя удовлетворить всем условиям смежности, то приоритет отдаётся ЛСС-1, определённым по 1-му правилу. В случае, если в группе оказываются не 2 состояния, а три и больше, они по возможности должны находится в соседних клетках карты Карно.

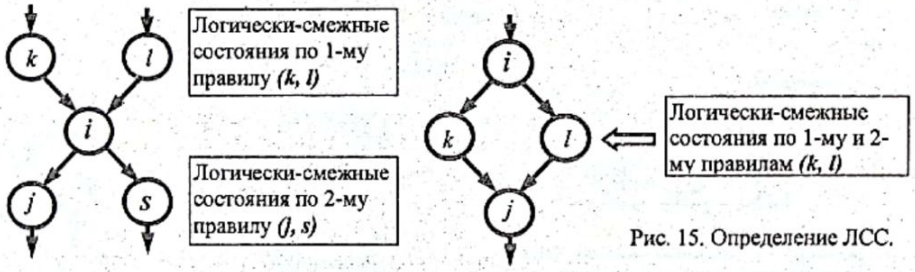


Рис. 15. Определение ЛСС.

Определим ЛСС для исследуемых автоматов.

Для автомата Мили (рис. 7, б): ЛСС-1 {(3,7), (1,7)}; ЛСС-2 {(1,4), (1,2)}.

Для автомата Мура (рис. 10): ЛСС-1 {(1,8), (4,7), (5,6)}; ЛСС-2 {(1,2), (6,7), (5,8)}.

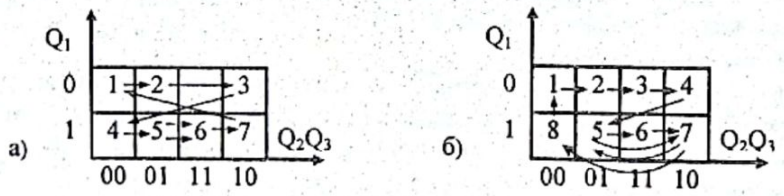


Рис. 16. Возможные варианты приоритетного кодирования логически смежных состояний автомата Мили (а) и Мура (б)

Какой же способ кодирования выбрать для рассмотренных графов автоматов? Если результаты кодирования состояний автомата являются критичными, то *нужно выбрать такой вариант кодирования, который обеспечит наименьшую сложность комбинационной схемы автомата. Это является одним из требований, предъявляемых к проекту.* Эта работа проводится на следующих этапах синтеза управляющего автомата.

11. Интерпретационный метод синтез УА на основе структурной таблицы.

Канонический метод синтеза структурного автомата (Мили или Мура) на основе таблицы истинности для выходных сигналов и сигналов возбуждения триггеров является универсальным методом, позволяющим получить схему автомата с минимальными аппаратными затратами. Однако этот метод становится трудоёмким для реализации ГСА с большим числом операторных вершин, порождающих автоматы с большим числом состояний.

В таких случаях используется **интерпретационный метод синтез УА на основе структурной таблицы.**

Исходной информацией для составления структурной таблицы является граф автомата Мили (или Мура), представленный в стандартной форме, а также результаты кодирования состояний автомата.

Дальнейшие этапы синтеза схемы автомата включают следующие этапы:

1. Выбор типа триггера и составление структурной таблицы (прямой или обратной)
2. Запись логических выражений для выходных сигналов управления Y_i и сигналов возбуждения триггеров φ_j.
3. Составление структурной схемы автомата.
4. Построение функциональной схемы.

Вопрос выбора типа триггера в искомой постановке синтеза УА является риторическим, так как мы не можем привести какие-либо обоснованные требования к типу логического функционирования триггера. Однако используемые триггеры должны быть **синхронного типа с динамическим управлением записью информации,** так как только этот тип триггера отвечает требованиям построения синхронных автоматов. Кроме того, принятый нами способ синхронизации устройства обработки информации (рис. 9), предполагает смену состояний УА по отрицательному фронту (срезу) сигнала синхронизации. Следовательно, выбранный тип триггера должен соответствовать этому требованию.

11.1. Автомат Мили.

■ **Выбор типа триггера и составление структурной таблицы автомата Мили (прямой или обратной).**

В качестве примера, при составлении структурной таблицы автомата Мили выберем синхронный JK-триггер, тактируемый срезом синхросигнала CLK (допустим SN7476). Примем так же вариант соседнего кодирования состояний автомата, изображённого на рис. 12 (правда, как это видно из рисунка, принцип соседнего кодирования в полной мере для данного графа автомата реализовать не удалось).

$Q^t \to Q^{t+1}$		J^t	K^t
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

Триггерный словарь JK-триггера

В прямой структурной таблице (табл. 5), в графе «Исходные состояния» перечисляются все состояния автомата, начиная с первого (в обратной таблице указанная последовательность перечислений состояний автомата производится в графе «Состояния переходов»).

Переход автомата из состояния a_m в a_S контролируется частной функцией перехода $F_i(a_m, a_S) = a_m X(a_m, a_S)$, которая и определяет значения выходных сигналов Y_t и функций возбуждения φ_j для каждого перехода. Здесь $X(a_m, a_S)$- конъюнкция логических условий, определяющая путь перехода $a_m \to a_s$ в графе автомата (рис 7,б). Если переход $a_m \to a_s$ является безусловным, то $X(a_m, a_S)=1$.

В колонке «Сигналы возбуждения $\varphi_i (a_m, a_s)$» выписываются значения J_i и K_i, принимающие **единичные значения** в триггерном словаре.

- **Запись логических выражений для выходных сигналов управления Y_t и сигналов возбуждения триггеров φ_j.**

Аналитические выражения для определения функций Y_t и сигналов возбуждения триггеров φ_j записываются на основе объединения по ИЛИ соответствующих частных функций переходов (в данной таблице отсутствуют одинаковые выходные сигналы для разных функций перехода). Данные выражения представлены в табл. 6.

Таблица 5. Прямая структурная таблица автомата Мили

№ перехода	Исходные состояния $K(a_m)=Q_1 Q_2 Q_3$				Состояния переходов $K(a_s)=Q_1 Q_2 Q_3$				Фуикции переходов	Выходные сигналы	Сигналы возбуждения
	a_m	Q_1	Q_2	Q_3	a_S	Q_1	Q_2	Q_3	$F_i(a_m, a_S)$	$Y_i(a_m, a_S)$	$\varphi_i(a_m, a_S)$
1	a_1	0	0	0	a_1	0	0	0	$a_1 \overline{пуск}$	------	------
2		0	0	0	a_2	0	0	1	a_1 пуск	Y_1	J_3
3	a_2	0	0	1	a_3	0	1	1	a_2	Y_2	J_2
4	a_3	0	1	1	a_4	1	1	0	a_3	Y_3	J_1, K_3
5	a_4	1	1	0	a_5	1	1	1	a_4	Y_4	J_3
6	a_5	1	1	1	a_6	1	0	1	$a_5 \overline{x_1}$	Y_5	K_2
7		1	1	1	a_6	1	0	1	$a_5 x_1$	------	K_2
8	a_6	1	0	1	a_7	1	0	0	a_6	Y_6	K_3
9	a_7	1	0	0	a_1	0	0	0	$a_7 x_2$	Y_7	K_1
10		1	0	0	a_4	1	1	0	$a_7 \overline{x_2}$	------	J_2

Таблица 6

- **Структурная схема**

Структурная схема управляющего автомата Мили включает три составные части (рис. 17): регистр состояний (состоит из выбранного типа триггеров), дешифратор состояний и комбинационную часть, предназначенную для реализа-

ции выражений для выходных сигналов управления Y_t и сигналов возбуждения памяти φ_j.

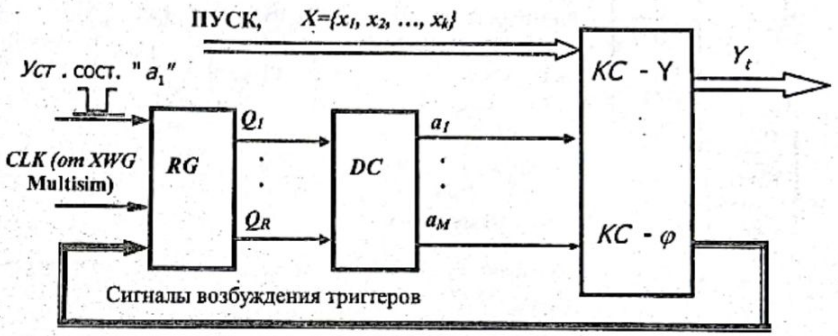


Рис. 17. Структурная схема автомата Мили

Построение функциональной схемы.

Здесь ограничимся лишь детализацией двух фрагментов структурной схемы (рис. 18), связанными с регистром состояний автомата и схемой задания входных управляющих сигналов для отладки и проверки работоспособности автомата в программе Multisim 10.1 (рис. 19).

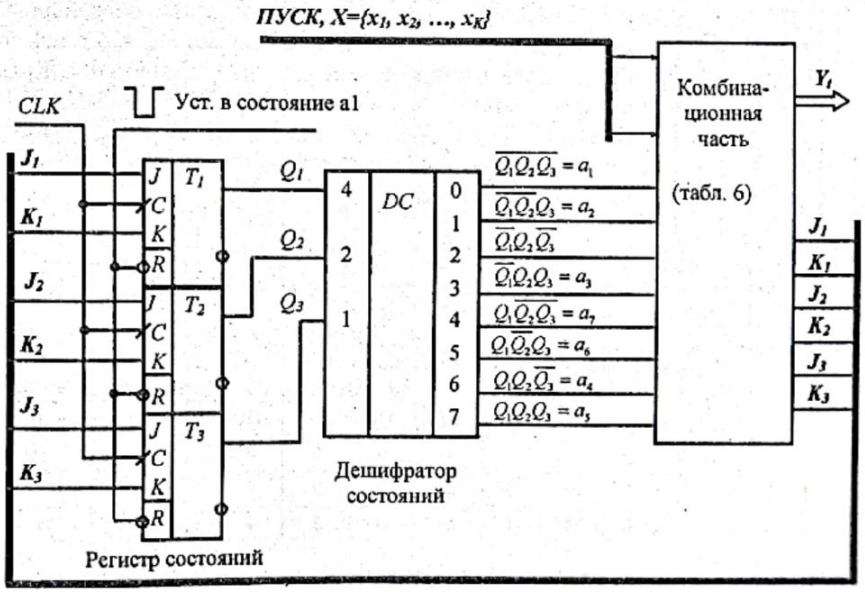


Рис. 18. Структурно-функциональная схема автомата Мили

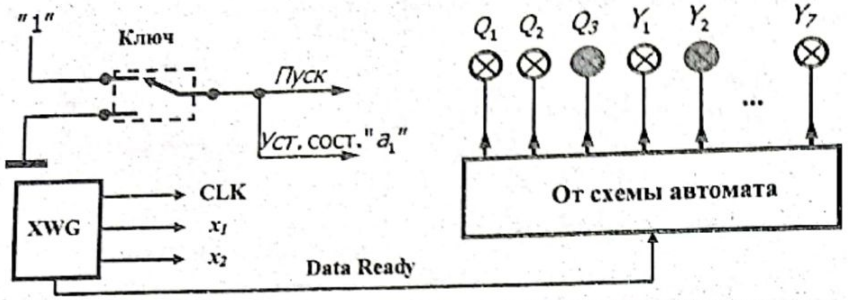


Рис. 19. Схема задания входных управляющих сигналов для отладки и проверки работоспособности автомата

11.2. Автомат Мура.

■ **Выбор типа триггера и составление структурной таблицы автомата Мура (прямой или обратной).**

В качестве примера, при составлении структурной таблицы автомата Мура выберем синхронный D- триггер (**допустим SN7474, дополненный инвертором на входе синхронизации**), и способ приоритетного кодирования логически-смежных состояний автомата. Результат кодирования состояний автомата этим способом показан на рис. 16,б

Прямая структурная таблица автомата Мура (табл. 7) подобна табл. 5 для автомата Мили за тем лишь исключением, что в ней будет отсутствовать столбец ВЫХОДНЫЕ СИГНАЛЫ $Y_i(a_m a_S)$, которые войдут в расширенный столбец ИСХОДНЫЕ СОСТОЯНИЯ с обозначением a_m/Y_t, так как для автомата Мура выходной сигнал есть функция только состояния (см. выражение (5).

$Q^t \to Q^{t+1}$		D^t
0	0	0
0	1	1
1	0	0
1	1	1

Триггерный словарь D-триггера

Таблица 7. Прямая структурная таблица автомата Мура

№ перехода	Исходные состояния $K(a_m)=Q_1\,Q_2\,Q_3$				Состояния переходов $K(a_s)=Q_1\,Q_2\,Q_3$				Функции переходов	Сигналы возбуждения
	a_m/Y_t	Q_1	Q_2	Q_3	a_S	Q_1	Q_2	Q_3	$F_i(a_m, a_S)$	$\varphi_i(a_m, a_S)$
1	a_1	0	0	0	a_1	0	0	0	$a_1\ \overline{пуск}$	------
2		0	0	0	a_2	0	0	1	a_1 пуск	D_3
3	a_2/Y_1	0	0	1	a_3	0	1	1	a_2	D_2, D_3
4	a_3/Y_2	0	1	1	a_4	0	1	0	a_3	D_2
5	a_4/Y_3	0	1	0	a_5	1	0	1	a_4	D_1, D_3
6	a_5/Y_4	1	0	1	a_6	1	1	1	$a_5\ \overline{x_1}$	D_1, D_2, D_3
7		1	0	1	a_7	1	1	0	$a_5\ x_1$	D_1, D_2

№ перехода	Исходные состояния $K(a_m)=Q_1\,Q_2\,Q_3$				Состояния переходов $K(a_s)=Q_1\,Q_2\,Q_3$				Функции переходов	Сигналы возбуждения
	a_m/Y_t	Q_1	Q_2	Q_3	a_S	Q_1	Q_2	Q_3	$F_i(a_m, a_S)$	$\varphi_i(a_m, a_S)$
8	a_6/Y_5	1	1	1	a_7	1	1	0	a_6	D_1, D_2
9	a_7/Y_6	1	1	0	a_5	1	0	1	$a_7\,\overline{x_1}$	D_1, D_3
10		1	1	0	a_8	1	0	0	$a_7\,x_2$	D_1
11	a_8/Y_8	1	0	0	a_1	0	0	0	a_8	------

■ **Запись логических выражений для выходных сигналов управления Y_t и сигналов возбуждения триггеров φ_j.**

Выходные сигналы в автомате Мура отождествляются с инициализацией соответствующих состояний автомата, а сигналы возбуждения триггеров φ_j записываются на основе объединения по ИЛИ соответствующих функций переходов, так как это было сделано для автомата Мили (см. табл. 6)

■ **Структурная схема**

Структурная схема управляющего автомата Мура также включает три составные части (рис. 20): регистр состояний (состоит из выбранного типа триггеров), дешифратор состояний и комбинационную часть, предназначенную для реализации выражений для сигналов возбуждения φ_j триггеров.

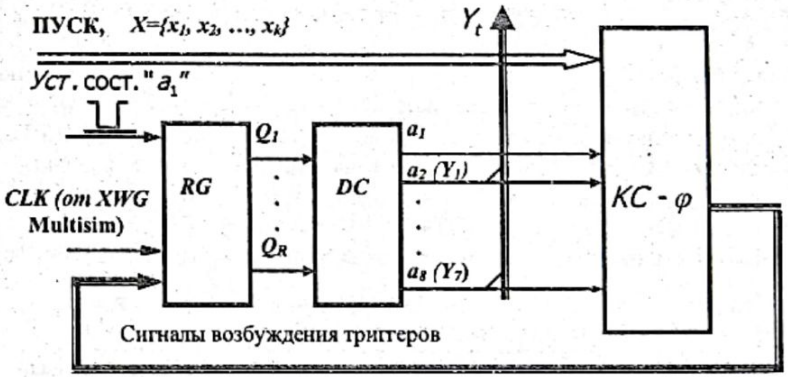


Рис. 20. Структурная схема автомата Мура

■ **Построение функциональной схемы.**

Здесь ограничимся рассмотрением лишь интерфейса регистра состояний дешифратора состояний (рис. 21).

■ Преобразовать логические выражения для комбинационной части автомата к виду, соответствующему выбранному типу логических элементов.

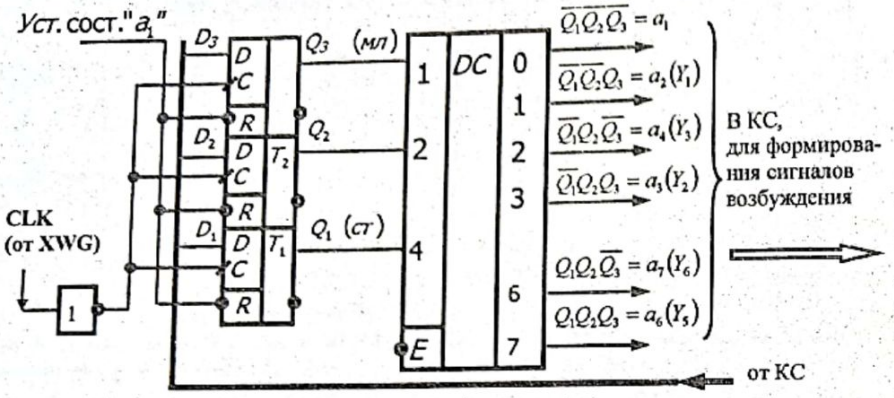


Рис. 21. Схема интерфейса регистра состояний с дешифратором состояний

- Разработать функциональную схему управляющего автомата Мили (Мура).

1. При построении функциональной схемы УА в рабочем поле программы *Multisim* **рекомендуется использовать следующую элементную базу**:

- в качестве регистра состояний – 4 -х разрядный регистр памяти на динамических D триггерах ИС ***SN74xx175*** (если исходное состояние K(a_1) закодирована *0*-ми состояниями триггеров) или два корпуса ИС ***SN74xx74*** ();
- в качестве дешифратора состояний – ИС ***SN74xx138 (3->8)*** или ИС ***SN74xx154*** (***4->16)*** (в этом случае не забудьте включить инверторы на выходы ИС, т.к. обе схемы имеют инверсные выходы);
- комбинационную часть автомата выполнить на ИС с логикой TTL.
- сигналы синхронизации **CLK** и значения признаков x_i задавать генератором ***XWG***, а сигналы «**ПУСК**» и «**Уст. в сост. *a1***» с помощью ключей (см. рис. 19).

2. Выполнить набор схемы УА в рабочем поле программы Multisim 10.1.

Таблица 8. Таблица моделирования комплекса «УА Мили ↔ ОА» по тактам синхросигнала.

№ такта	Пуск (ключ)	Входные сигналы УА Мили (от XWG Multisim)			Состояние УА Мили и вырабатываемые им МК		ОА (Выполнение МК Y_i при CLK=0 и уст. флагов при CLK=1)
		x_1	x_2	*CLK*	a_m, $K(a_m)=Q_1\,Q_2\,Q_3$	Y_i	
1	1	x	x	0	a_1 0 0 0	Y_1	
		x	x	1			
2	0 (x)	x	x	0	a_2 0 0 1	Y_2	
		x	x	1			
3	0 (x)	x	x	0	a_3 0 1 1	Y_3	
		x	x	1			
4	0 (x)	x	x	0	a_4 1 0 0	Y_4	
		0	x	1			Уст. CF (x_1) =0
5	0 (x)	0	x	0	a_5 1 1 1	Y_5	
		0 (x)	x	1			Изменяет все флаги, кроме CF (x_1)
6	0 (x)	0 (x)	x	0	a_6 1 0 1	Y_6	
		0 (x)	0	1			Уст. ZF (x_2)=0
7	0 (x)	0 (x)	0	0	a_7 1 0 0	—	
		0 (x)	0 (x)	1			
8	0 (x)	0 (x)	0 (x)	0	a_4 1 1 0	Y_4	
		1	0 (x)	1			Уст. CF (x_1) =1
9	0 (x)	1	0 (x)	0	a_5 1 1 1	—	
		1 (x)	0 (x)	1			
10	0 (x)	1 (x)	0 (x)	0	a_6 1 0 1	Y_6	
		1 (x)	1	1			Уст. ZF (x_2) =1
11	0 (x)	1 (x)	1	0	a_7 1 0 0	Y_7	
		1 (x)	1 (x)	1			
12	0	1 (x)	1 (x)	0	a_1	—	

3. Составить таблицу работы комплекса **«УА↔ОА»**, отражающей особенности интерпретации работы автомата Мили (Мура) по тактам синхросигнала **Clk** в процессе реализации вычислительной процедуры. Данная таблица (табл. 8 для автомата Мили (рис. 7,б и 8) и табл. 9 автомата Мура (рис. 10 соответственно)) фактически определяет набор двоичных комбинаций генератора слов **XWG**, использующимся для формирования входных сигналов при моделировании схемы автомата. **Имеющиеся комбинации входных сигналов x_i в таблице моделирования, должны соответствовать всевозможным переходам (дугам) графа автомата.**

4. В соответствии с разработанной таблицей моделирования произвести функциональное моделирование работы УА в программе Multisim 10.1 в пошаговом режиме «Step». ***Именно в этом формате будет приниматься ваша работа преподавателем.***

Лабораторная работав должна отражать содержание этапов синтеза УА, его схему, таблицу моделирования, отображение рабочего окна с УА в Multisim 10.1, а также выводы по результатам моделирования.

Таблица 9. Таблица моделирования комплекса «УА Мура ↔ ОА» по тактам синхросигнала.

№ такта	Пуск (ключ)	Входные сигналы УА Мура (от XWW Multi-sim)			Состояние УА Мура и вырабатываемые им МК		ОА (Выполнение МК Y_t при CLK=0 и уст. флагов при CLK=1)
		x_1	x_2	CLK	a_m, $K(a_m)=Q_1\,Q_2\,Q_3$	Y_t	
1	1	x	x	0	a_1 0 0 0	—	
		x	x	1			
2	0 (x)	x	x	0	a_2 0 0 1	Y_1	
		x	x	1			
3	0 (x)	x	x	0	a_3 0 1 1	Y_2	
		x	x	1			
4	0 (x)	x	x	0	a_4 0 1 0	Y_3	
		x	x	1			
5	0 (x)	x	x	0	a_5 1 0 1	Y_4	
		0	x	1			Уст. CF (x_1)=0
6	0 (x)	0	x	0	a_6 1 1 1	Y_5	
		0 (x)	x	1			Изменяет все флаги, кроме CF (x_1)
7	0 (x)	0 (x)	x	0	a_7 1 1 0	Y_6	
		0 (x)	0	1			Уст. ZF (x_2)=0
8	0 (x)	0 (x)	0	0	a_5 1 0 1	Y_4	
		1	0 (x)	1			Уст. CF (x_1)=1
9	0 (x)	1	0 (x)	0	a_7 1 1 0	Y_6	
		1 (x)	1	1			Уст. ZF (x_2)=1
10	0 (x)	1 (x)	1	0	a_8 1 0 0	Y_7	
		1 (x)	1 (x)	1			
11	0	1 (x)	1 (x)	0	a_1	—	

13. Индивидуальные задания к работе

Индивидуальное задания к работе (табл. 10) определяется выражением, микропрограмму выполнения которого надо реализовать с помощью программной модели операционном устройстве (табл. 1 и 2) под управлением спроектированного микропрограммного автомата.

***) ПОЯСНЕНИЯ к заданиям 4 – 5.**

■ В задании 4 необходимо вычислить значение функции, определённое конъюнктивным термом. Логические переменные x_i задаются соответствующими разрядами «байтового» регистра. Для вычисления предполагается использовать маски D и T, выбранные следующим образом:

В регистрах РОН находятся маски D и T, выбранные следующим образом:
– маска D предназначена для селективного сброса битов в тех разрядах регистра РОН с вектором входных переменных $X=\{x_7, x_6, \ldots, x_0\}$, номера которых соответствуют отсутствующим переменным в исходном терме;
– маска T включает «1» в позициях, которые соответствуют переменным без инверсии в исходном терме и «0» в остальных позициях.

Процедура вычисления заканчивается проверкой соотношения:

$$(D \wedge X) \oplus T = \begin{cases} 0, & f = 1, \\ \neq 0, & f = 0. \end{cases}$$

Маски D и T предварительно занести в соответствующие регистры.

■ В задании 5 маска D выбирается так же, а маска T включает «1» в позициях, соответствующих инверсным переменным, а в остальных - нули. Процедура заканчивается проверкой соотношения:

$$(D \wedge X) \oplus T = \begin{cases} 0, & f = 0, \\ \neq 0, & f = 1. \end{cases}$$

■ В ряде заданий (например, 1 и др.) используются множители, значения которых отличны от значений 2^n. В этом случае необходимый результат умножения можно получить путём применения комбинаций из операций арифметического сдвига влево и сложения.

Таблица 10

№ варианта задания	Функция или процедура	Тип автомата
1	$S = \begin{cases} 6A + B, \text{если } B \geq 0 \\ A - B, \text{если } B < 0 \end{cases}$	Мили
2	$S = \begin{cases} 4A - B, \text{если } B \geq 0 \\ A + B, \text{если } B < 0 \end{cases}$	Мура
3	$S = \begin{cases} -B, & \text{если } B > 0 \\ 5B + A, & \text{если } B \leq 0 \end{cases}$	Мили
4	$f = x_7 \overline{x_6} x_3 x_1$)*	Мура

5	$f = x_6 \vee x_4 \vee \overline{x_3} \vee x_0$)*	Мили
6	Подсчитать число «1» в байте (операнд А)	Мура
7	Подсчитать число «0» в байте (операнд А)	Мили
8	$S = \begin{cases} \textit{Переставить} \text{ тетрады, если } A \geq 0 \\ A = 0, \quad \text{если } A < 0 \end{cases}$	Мура
9	$S = \begin{cases} 2B - A, & \textit{если } B < 0 \\ -B, \textit{если } & 0 \leq B < 10 \end{cases}$	Мили
10	$S = \begin{cases} A + B/4, \textit{если } A > 0 \\ 2A + B/2, \textit{если } A \leq 0 \end{cases}$	Мура
11	$S = \begin{cases} 8B + A, \textit{если } B \geq A \\ -B, \textit{если } \quad B < A \end{cases}$	Мили
12	$S = \begin{cases} 5B, \quad \textit{если } B \geq A \\ A - B, \textit{если } B < A \end{cases}$	Мура

ЛИТЕРАТУРА

1. *Скляров В. А.* Синтез автоматов на матричных БИС / Под. Ред. С. И. Баранова. – Минск: 2010. - 272 с.
2. *Бойко В. И. и др.* Схемотехника электронных систем. Цифровые устройства. - СПб.:БХВ - Петербург, 2017. - 512 с.(33 экз)
3. *Угрюмов Е. П.* Проектирование элементов и узлов ЭВМ: Учеб. пособие для вузов. – М.: Высшая Школа, 2017. - 318 с.: ил.(16 экз)
4. *Угрюмов Е. П.* Цифровая схемотехника. - СПб.: БХВ – Санкт-Петербург, 2000.(2005 г. - второе издание, расширенное и дополненное) – 528 с.: ил.
5. *Пухальский Г.И.*, Новосельцева Т. Я. Проектирование дискретных устройств на интегральных микросхемах: Справочник. -М.: Радио и связь, 1990. - 304 с.: ил.(3 экз)
6. *Уэйкерли Дж.* Проектирование цифровых устройств. В 2-х т. Пер. с англ. - М.: Постмаркет, 2002.