Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Н. А. Долгий

ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ, СИСТЕМЫ И СЕТИ

Учебно-методическое пособие по выполнению лабораторных работ для студентов направления подготовки 15.03.04 – Автоматизация технологических процессов и производств

Калининград
Издательство ФГБОУ ВО «КГТУ»
2023

УДК 681.5

Рецензент:
кандидат технических наук, доцент, проректор по учебной работе ФГБОУ ВО
«Калининградский государственный технический университет»
В. И. Устич

Долгий, Н. А.
Вычислительные машины, системы и сети : учебно-методическое пособие по выполнению лабораторных работ для студентов направления подготовки 15.03.04 – Автоматизация технологических процессов и производств / Н. А. Долгий. – Калининград : Изд-воФГБОУ ВО «КГТУ», 2023. – 147 с.

В учебно-методическом пособии приводятся рекомендации по выполнению лабораторных работ, связанных с проектированием электронных узлов ЭВМ.
Табл. 32, рис. 81, лит. – 4 наименования

Учебно-методическое пособие рассмотрено и одобрено в качестве локального электронного методического материала кафедрой цифровых систем и автоматики 28 сентября 2022 г., протокол № 2.

Учебно-методическое пособие рекомендовано к использованию в учебном процессе в качестве локального электронного методического материала методической комиссией Института цифровых технологий 17 марта 2023 г., протокол № 2.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов направления подготовки 15.03.04 – Автоматизация технологических процессов и производств, изучающих дисциплину «Вычислительные машины, системы и сети».

Цель лабораторного практикума по дисциплине: получение практических навыков проектирования узлов различной функциональной и структурной организации ЭВМ.

Лабораторный практикум содержит 18 лабораторных работ.

Лабораторные работы проводятся в лабораториях кафедры цифровых систем и автоматики (ЦСА).

В результате выполнения лабораторных работ студенты сформируют навыки проектирования цифровых узлов ЭВМ.

Лабораторная работа № 1
Базовые логические элементы. Реализация простейших логических функций

Цель работы: изучение основных характеристик интегральных микросхем транзисторно-транзисторной логики; обучение навыкам проектирования и синтеза простейших комбинационных схем.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется лабораторный стенд УМ11. Стенд расположен в лаборатории № 261.10.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

Числовые данные и машинные команды представлены в ЭВМ в двоичной системе счисления. Физическими аналогами знаков 0 и 1 двоичного алфавита служат сигналы, способные принимать два хорошо различимых значения, например, напряжения (потенциал) высокого и низкого уровня, отсутствие или наличие электрического импульса в определенный момент времени, противоположные по знаку значения напряженности магнитного поля и т.п.

В цифровых вычислительных устройствах обычно применяют два первых способа физического представления информации, в соответствии с чем, схемы цифровых устройств принято делить на потенциальные, импульсные и импульсно-потенциальные.

Потенциальный сигнал присутствует в определенной точке схемы в течение всего машинного такта и характеризуется величиной верхнего и нижнего уровня напряжения U^1 и U^0.

Наличие фронта и спада у потенциального сигнала связано с процессом перехода от нижнего к верхнему и от верхнего к нижнему уровню. Длительность этих процессов обозначается соответственно t^{01} и t^{10} (рис. 1), причем моменты перехода определяются тактовыми сигналами.

Импульсный сигнал, помимо перечисленных выше параметров, определяется также моментом своего появления и длительностью, не зависящей в общем случае от длительности машинного такта (рис. 1).

Наибольшее распространение в схемотехнике узлов ЭВМ получили импульсные и импульсно-потенциальные системы элементов. Новые разработки показали возможность очередного этапа развития импульсно-

потенциальных систем на базе интегральных разностных преобразователей (РП). Такие РП формируют импульсные сигналы при заранее определенных видах переключений входных булевых переменных. Длительность выходного импульса синхронных РП определяется параметрами их компонентов, обеспечивающих задержку прохождения сигналов. В синхронных РП в качестве элемента задержки используются D-триггеры или регистры сдвига, поэтому длительность выходного импульса определяется периодом тактовых сигналов.

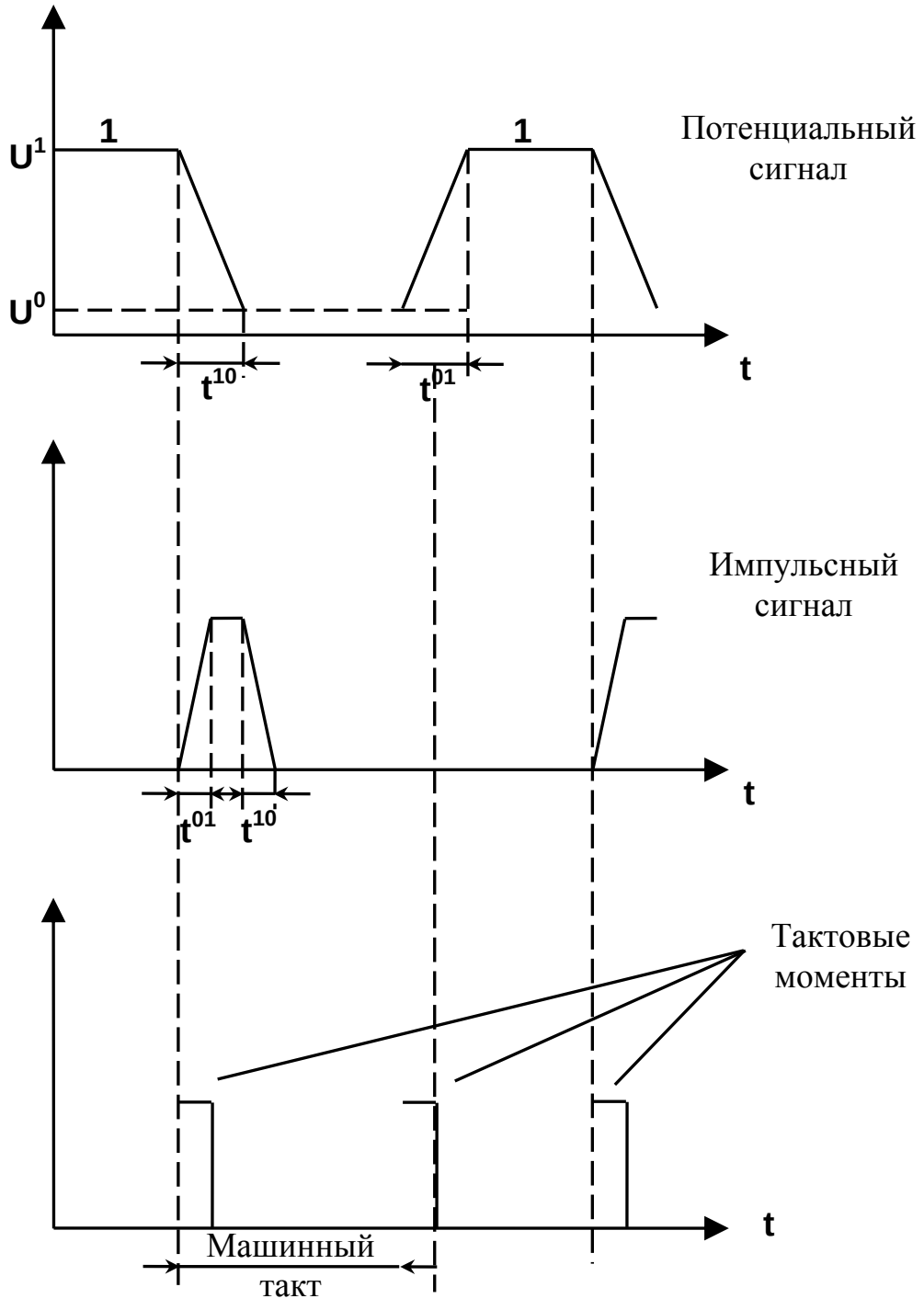


Рис. 1. Потенциальный сигнал верхнего и нижнего уровней напряжения

Системой (серией) логических элементов ЭВМ называется предназначенный для построения цифровых устройств функционально полный набор логических элементов, объединяемый общими электрическими, конструктивными и технологическими параметрами, использующие способ представления информации и одинаковый тип межэлементных связей. Система элементов включает элементы для выполнения логических операций, а также усиления, восстановления и формирования стандартной формы сигналов.

Основными параметрами систем логических элементов узлов ЭВМ (таблица 1) являются:

1. Напряжения питания и логических уровней. Система элементов характеризуется количеством используемых питающих напряжений и их номинальными значениями.

Для логических уровней указываются их полярность и величина. Если не оговорено особо, то логическому нулю соответствует низкий уровень напряжения, а логической единице – высокий (положительная логика).

2. Коэффициент разветвления по выходу (нагрузочная способность К раз) характеризует количество входов аналогичных элементов, которое может быть подключено к выходу логического ключа.

3. Помехоустойчивость. Помехой называется нежелательное электрическое воздействие (например, пульсации напряжения питания, действия паразитных емкостей) на логический элемент, которое может привести к искажению преобразуемых или хранимых данных. Помехоустойчивость есть способность элемента правильно функционировать при наличии помех, она определяется максимально допустимым напряжением помехи, при котором не происходит сбоя в его работе.

4. Быстродействие логических элементов является одним из важнейших параметров и характеризуется средним временем задержки распространения сигнала:

$$t_{зд.р.ср.} = \frac{t^{01}_{зд.р.} + t^{10}_{зд.р.}}{2}$$

Время задержки распространения при включении $t^{01}_{зд.р.}$, и выключении $t^{10}_{зд.р.}$ измеряют как интервал между уровнем 0,5Uвх входного сигнала и уровнем 0,5Uвых выходного сигнала (рис. 2).

5. Потребляемая мощность также является немаловажным фактором, поскольку ее увеличение повышает затраты на создание и эксплуатацию цифровых устройств, а снижение уменьшает быстродействие и помехоустойчивость. В зависимости от типа компонентов наиболее широко распространенные серии элементов характеризуются следующими усредненными параметрами:

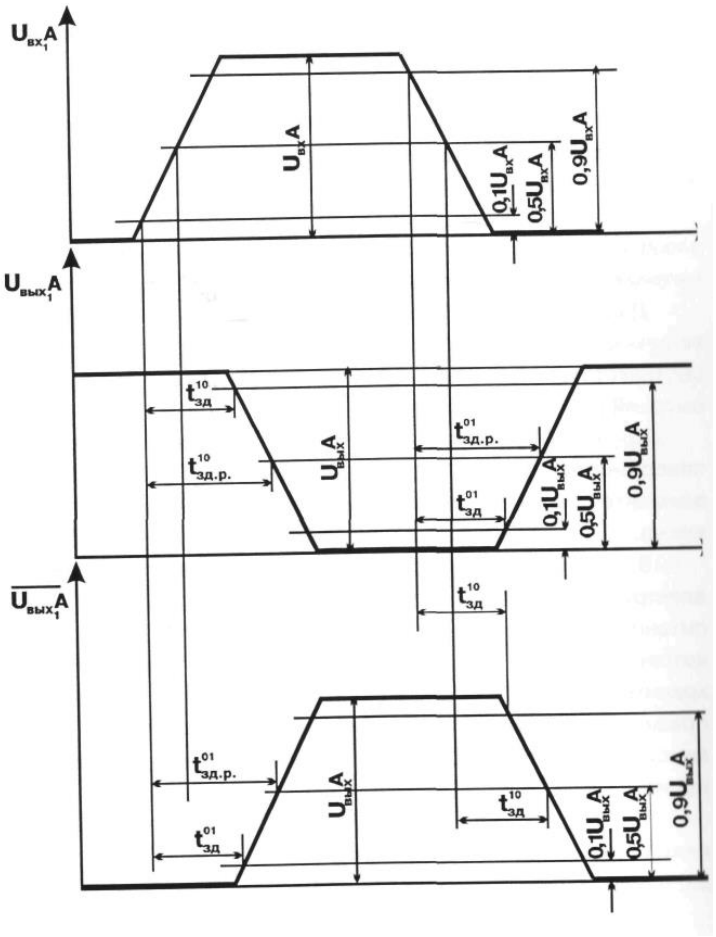


Рис. 2. Время задержки распространения при включении и выключении

Таблица 1

Тип-серия	ТТЛ			ДТЛ	ЭСЛ	кМОП	n-МОП	p-МОП
Характеристика	155	158	131	511	500	176	172	108
1. Напряжение питания	+5	+5	+5	+15	-5,2	+9,0	-27	-27
Напряжение лог. 0	0,4	0,4	0,4	1,5	-1,63	0,3	-2,0	0,85
Напряжение лог. 1	2,4	2,4	2,5	12	-0,98	8,2	7,5	-9,5
2. Коэффициент разветвления	10	10	10	25	-	100	15	10
3. Время задержки распространения, нс	15	15	10	300	2,9	200	600	6000
4. Потребляемая мощность на элемент, мВт	24	4,85	50	187	35	0,001	-	100

Для обеспечения максимального быстродействия и помехоустойчивости неиспользуемые входы должны находиться под постоянным потенциалом. Это позволяет исключить перезарядку емкости разомкнутого эмиттера входного

транзистора относительно выводов схемы, которая увеличивает задержку сигнала. Существует ряд методов создания данного потенциала. Напряжение питания, ограниченное до значения 4,5 В, позволяет подключать неиспользуемые входы непосредственно к источнику питания.

Если напряжение 4,5 В отсутствует, то возможно подключение до 20 неиспользуемых входов через резистор сопротивлением 1кОм к источнику питания +5 В. Неиспользуемые входы можно соединить с используемым, если логическая функция при этом не претерпит изменений и не будет превышена нагрузочная способность при входном напряжении, соответствующем уровню логической единицы.

Рассмотрим возможность применения данных способов подключения неиспользуемых входов для получения инвертора на основе микросхемы К155 ЛА4 (3 элемента «ЗИ-НЕ») (рис. 3–5).

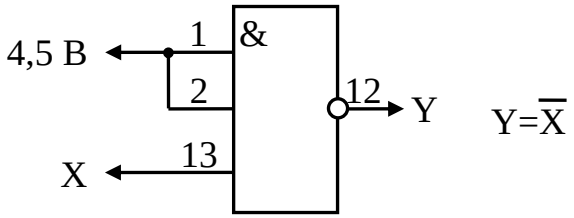


Рис. 3. Подключение неиспользованных входов к дополнительному источнику питания

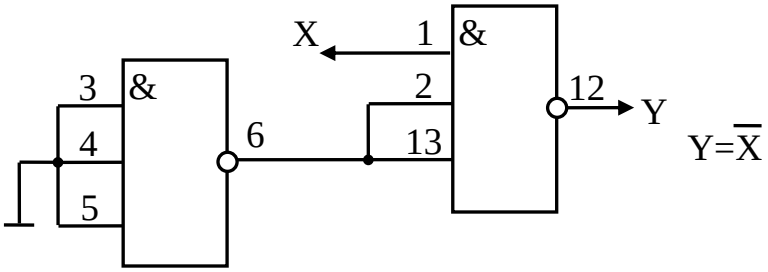


Рис. 4. Подключение неиспользованных входов к выходам неиспользуемой инвертирующей МС

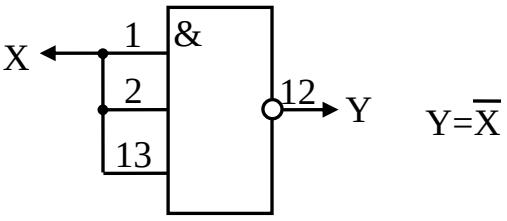


Рис. 5. Подключение неиспользованных входов МС к используемым

Для микросхем серии К155 длительность фронта не должна превышать 150 нс (для ИС 155 ЛА8 длительность фронта и спада критична).

Для обеспечения работоспособности логических микросхем друг от друга при условии сохранения их параметров, оговоренных в ТУ, необходимо выполнять следующие требования:

- выходные напряжения микросхем-генераторов в открытом и закрытом состояниях должны соответствовать входным напряжениям закрытого и открытого состояния микросхемы – нагрузки с учетом величины напряжения помехи;

- значение суммарных токов всех микросхем-нагрузок, подключенных к выходу микросхемы-генератора, не должно превышать значений выходных токов микросхемы-генератора в открытом и закрытом состояниях;

- значение суммарных емкостей входов микросхем-нагрузок, при которых регламентируются временные параметры, не превышает 15 пФ. При необходимости суммарная емкость нагрузки ИС-генератора с учетом емкости монтажа может достигать максимальной емкости нагрузки 200 пФ, но динамические параметры при этом не регламентируются.

Микросхемы 155-й серии, применяемые в данном лабораторном практикуме, могут иметь 14 или 16 выводов. Нумерация выводов показана на рис. 6.

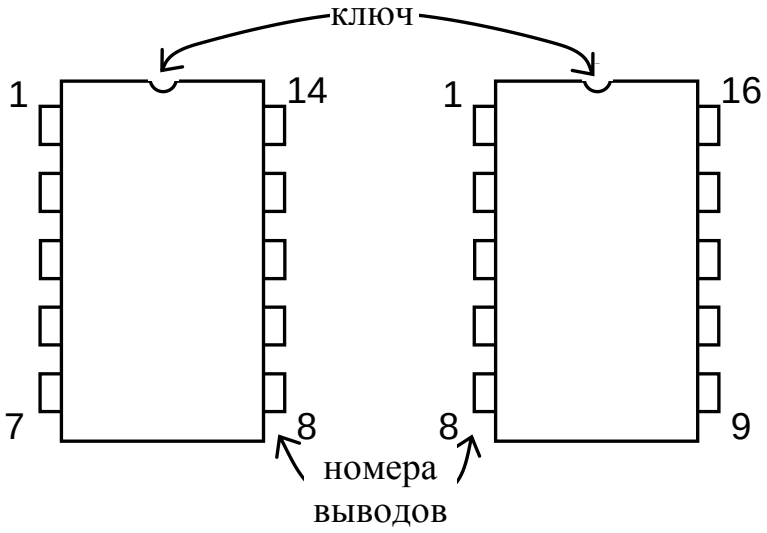


Рис. 6. Нумерация выводов микросхем 155-й серии в корпусе с 14 и 16 выводами

Устройство, преобразующее дискретную информацию, в общем случае имеет n-входов и k-выходов, на которые подаются и с которых снимаются электрические сигналы. Каждый из них представляет собой некоторый символ (букву) входного и выходного алфавита. Устройства, в которых совокупность выходных сигналов (выходное слово Y) в некоторый момент времени t, однозначно определяется входными сигналами (входным словом X), поступившими на входы в тот же момент времени, называются комбинационными схемами. В них результат обработки информации зависит

только от комбинации входных сигналов и вырабатывается сразу при подаче входной информации.

Закон функционирования комбинационной схемы (КС) определен, если задано соответствие между словами ее входного и выходного алфавитов, что может быть осуществлено либо в виде таблицы, либо в аналитической форме, с использованием булевых функций. Таким образом, комбинационная схема, выполняющая соответствующее некоторой булевой функции преобразование информации, является ее техническим аналогом. КС, реализующая элементарную логическую операцию, носит название логического элемента, причем число его входов соответствует числу аргументов воспроизводимой им булевой функции.

В состав элементов ТТЛ 155-й серии входит ряд элементов малой степени интеграции, выполняющих элементарные логические операции. Базовым элементом, т. е. таким, посредством которого могут быть реализованы все остальные логические операции, является элемент И-НЕ. Технология производства позволяет число входов данного элемента варьировать от 2 до 8, при этом, в зависимости от данного числа, наименование элемента будет: 2И-НЕ, 3И-НЕ, 4И-НЕ, 8И-НЕ. Элементы с числом входов 5, 6, 7 – не изготавливаются. Схема базового элемента показана на рис. 7.

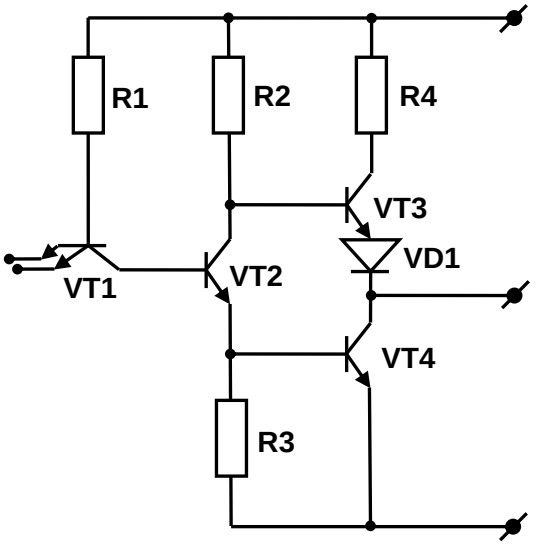


Рис. 7. Базовый элемент «2И-НЕ»

Основной его особенностью является использование многоэмиттерного транзистора, специфичного для интегрального исполнения логических элементов. При подаче на все его эмиттеры, выполняющие роль входов логического элемента, сигнала высокого уровня, ток коллектора смещается в прямом направлении и возникает ток базы транзистора VT_2. За счет падения напряжений на резисторах R_2 и R_3 транзистор VT_3 закрывается, VT_4 – открыт, при этом напряжение на выходе не превышает 0,4 В. При подаче сигнала

низкого уровня хотя бы на один вход - картина противоположная: транзисторы VT_2 и VT_4 закрываются, а VT_3 – открыт. На выходе элемента устанавливается потенциал более 2,4В. Выходной каскад подобного вида иногда называется столбовым. Некоторым недостатком подобного принципа построения выходного каскада является невозможность создания «монтажной функции» путем соединения выходов определенных элементов. Указанное ограничение снимается при использовании микросхем с «открытым коллектором» (рис. 8).

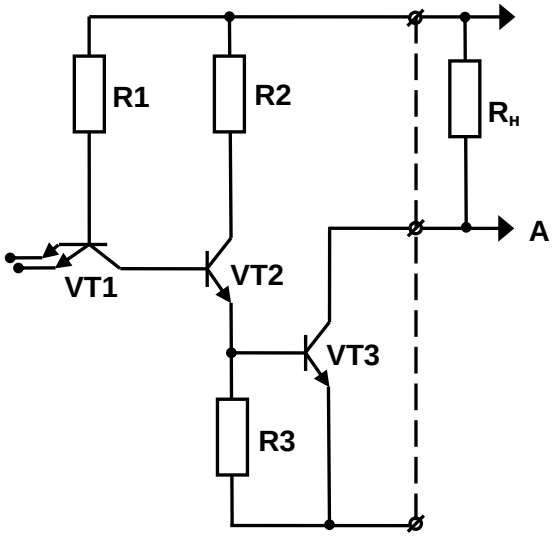


Рис. 8. Элемент «2И-НЕ» с открытым коллекторным выходом

Отсутствие транзистора, сходного по функции с VT_3 согласно рис. 7, компенсируется наличием нагрузочного резистора $R_н$, за счет которого при закрытом транзисторе VT_3 на входах элементов, подключенных к данному, устанавливается единичный потенциал. Подключение к точке А выходов однотипных элементов, транзисторы VT_3 которых закрыты, не изменяет состояние. Если же хотя бы один из указанных транзисторов открыт, в точке А устанавливается низкий потенциал.

Очевидно, что таблица истинности для данного типа микросхем будет включать одну строку, соответствующую случаю, когда на все входы поданы логические единицы, с нулем в графе выхода. В остальных 2^N-1 случаях (где N - число входов) на выходе будет сохраняться высокий уровень.

Порядок выполнения работы

1. Ознакомиться с установкой УМ11. Разобраться с назначением всех гнезд, имеющихся на наборном поле установки.

2. Экспериментальным путем получить таблицы истинности для следующих элементов:

2-И-НЕ,

3-И-НЕ,

4-И-НЕ,

2И-2ИЛИ-НЕ.

В отчете для каждого элемента должно быть приведено его название, математическое представление реализуемой функции, графическое обозначение и таблица истинности.

3. Построить на основе элементов "2-И-НЕ" схему, реализующую функцию элемента "НЕ". Привести 2 различные схемы, их математические модели и графическое изображение.

4. Построить на основе элементов "2-И-НЕ" схему, реализующую функцию элемента "ИЛИ". Привести математическое описание работы схемы и ее графическое изображение.

5. Построить на основе элементов "2-И-НЕ" схему, реализующую функцию элемента "И". Привести математическое описание работы схемы и ее графическое изображение.

6. Построить на основе элементов "2-И-НЕ" схему, реализующую функцию, указанную преподавателем.

7. Определить свой вариант переключательной функции. Для этого необходимо номер варианта перевести в двоичную систему счисления и записать шесть его младших разрядов в виде слова $\alpha 6\ \alpha 5\ \alpha 4\ \alpha 3\ \alpha 2\ \alpha 1$. Определив значение αi, записать их в таблице 2.

Например, если номер варианта 19 (010011), то $\alpha 6=0$, $\alpha 5=1$, $\alpha 4=0$, $\alpha 3=0$, $\alpha 2=1$, $\alpha 1=1$.

Для заданной функции и ее отрицания найти МДНФ. Представить функцию в форме И-НЕ.

Построить указанную схему, учитывая, что на входы могут подаваться с помощью тумблеров прямые и инверсные значения переменных.

Таблица 2

X4	X3	X2	X1	y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	α1
0	0	1	1	1
0	1	0	0	α2
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	α3
1	0	0	1	α4
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	α5
1	1	1	1	α6

Контрольные вопросы

1. Системы логических элементов (серии, комплексы), основные параметры системы логических элементов, значения основных параметров для логических элементов серии 155.

2. Подключение неиспользуемых входов элементов, работоспособность элемента с «висящими в воздухе» выводами.

3. Базовый элемент 155-й серии, реализация элемента «НЕ», «ИЛИ» многовходовых «И» и «ИЛИ», «ИСКЛЮЧАЮЩЕЕ ИЛИ».

4. Комбинационная схема, закон функционирования, способы его задания.

5. Управляемый инвертор.

6. Схемы с «открытым коллектором», «монтажное ИЛИ», возможность соединения логических элементов, минимальные и максимальные величины нагрузочных резисторов для элементов с открытым коллектором.

7. Система обозначения микросхем.

Лабораторная работа № 2
Комбинационные преобразователи логических сигналов. Дешифраторы

Цель работы: изучение принципов построения дешифраторов для управления индикацией; синтез преобразователей кодов.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется лабораторный стенд УМ11. Стенд расположен в лаборатории № 261.10.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

Некоторые функции нескольких переменных встречаются достаточно часто, поэтому они нашли свое отражение в системе логических элементов 155-й серии. К числу подобных элементов можно отнести прежде всего дешифраторы-схемы, способные преобразовать логическую информацию из одного цифрового кода в другой. Двумя наиболее общими типами подобных дешифраторов являются дешифраторы типа «один-из-N» и семисегментные дешифраторы.

Дешифраторы «*один-из-N*». В данном типе дешифраторов в каждый момент времени возможно наличие сигнала лишь на одном из выходов, причем номер выхода определяется двоичной комбинацией на соответствующих входах. Очевидно, что наиболее просто подобный дешифратор синтезирован с помощью схем совпадения, каждая из которых «настроена» на индивидуальную и единственную кодовую комбинацию.

Семисегментные дешифраторы. Более сложными по структуре являются устройства, осуществляющие процесс преобразования кодовых комбинаций с произвольным в общем случае числом нулей и единиц как на входе, так и на выходе. Синтез преобразователей подобного вида представляет весьма сложную задачу и распадается на два этапа:

1. Определение связей входов и выходов схем и генерация структуры системы в заданном классе базовых элементов, которая будет выполнять требуемые функции системы.

2. Решение задачи минимизации, т. е. выбор структуры системы, состоящей из минимального числа базовых элементов.

Минимизация особенно важна, если цифровая система производится серийно, когда желательно обеспечить использование минимального

количества элементов. Минимизация числа элементов уменьшает стоимость и повышает быстродействие и надежность разрабатываемой системы.

Минимизация может оказаться очень трудоемкой процедурой, особенно при проектировании сложной системы, в связи с чем решение задачи синтеза проводится, как правило, на ЭВМ с помощью специального математического обеспечения.

Решение указанной задачи вручную проводится с учетом того, что подобные преобразователи можно представить в виде композиции стандартного дешифратора «один-из-N» (где N соответствует числу входов преобразователя), входы которого в соответствии с определенным законом подключены к выходам дешифратора.

D1 К155ЛА4
D2.D4 К155ЛА3
D3 К155ЛР1

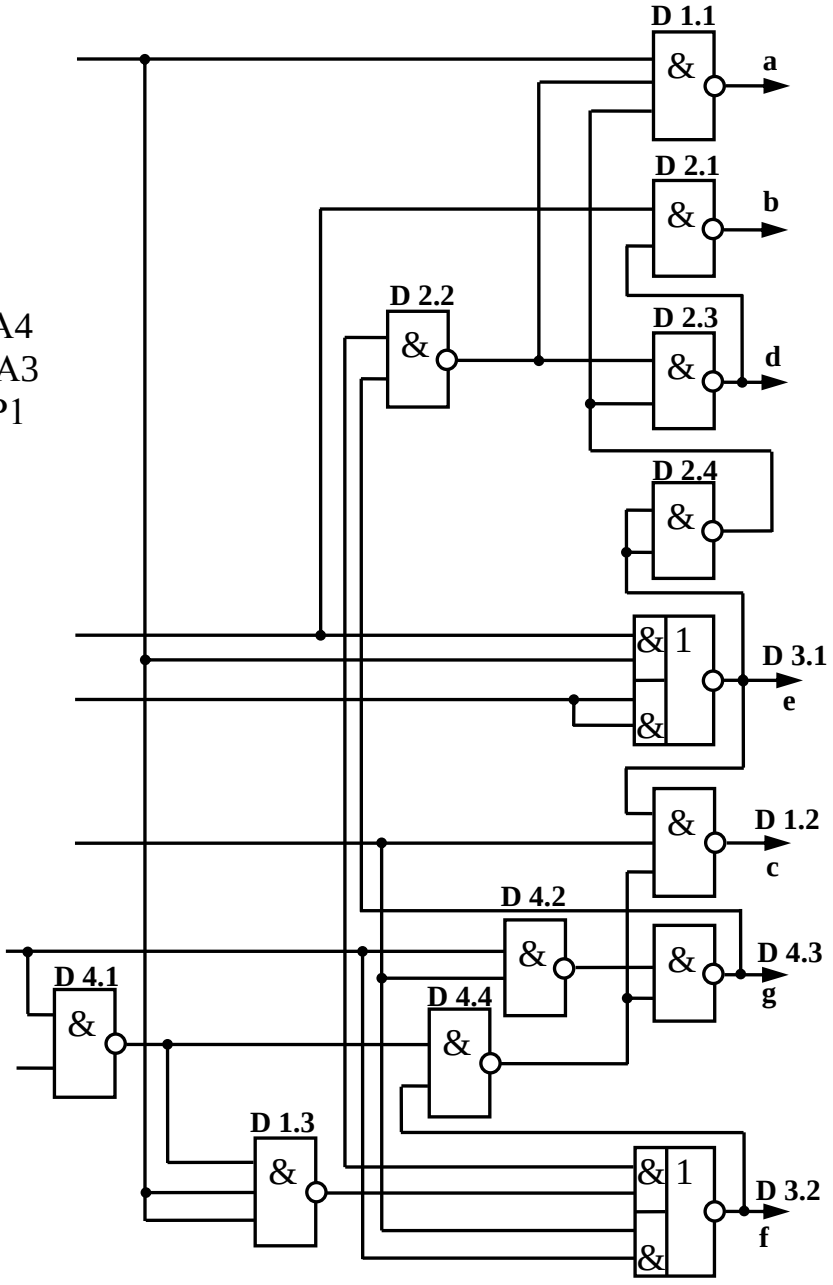


Рис. 9. Семисегментный дешифратор

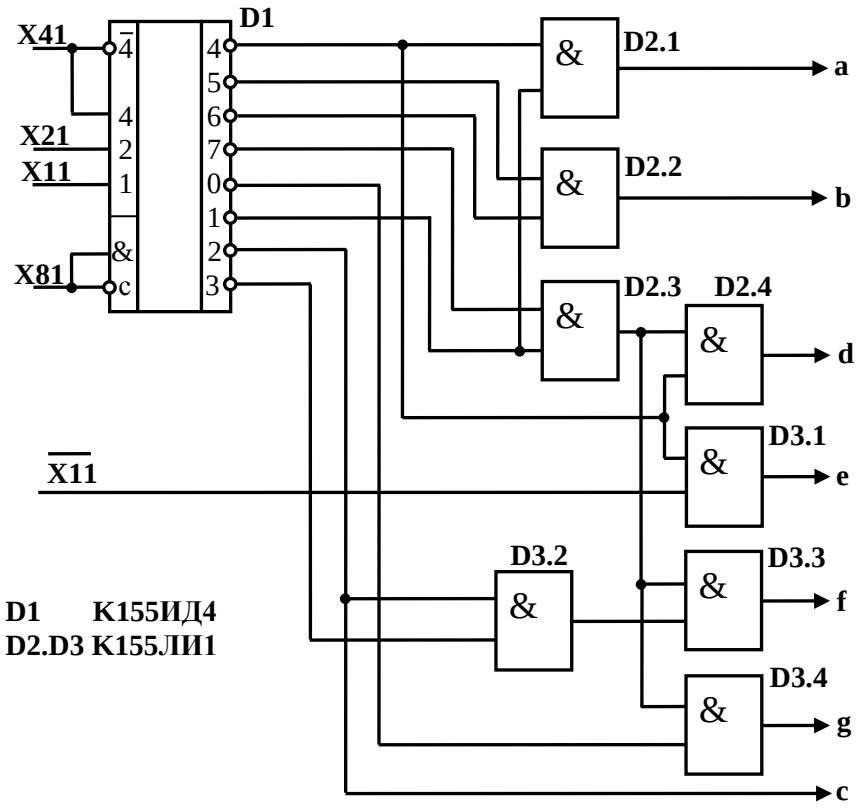


Рис. 10. Семисегментный

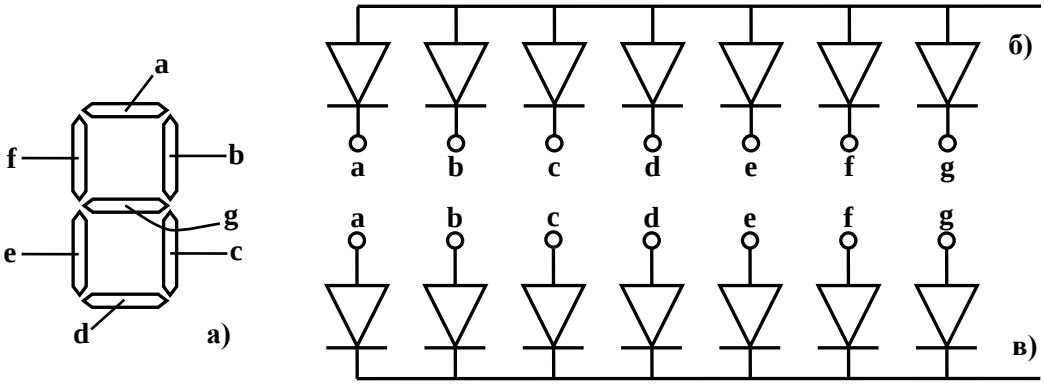


Рис. 11

Примеры преобразователей, выполняющих роль дешифраторов для семисегментных индикаторов, построенных на рассмотренных принципах, приведены на рис. 9–10.

Как показано на рис. 11, каждый из семи сегментов индикатора обозначается буквой латинского алфавита от a до g включительно. Любая цифра от 0 до 9 может быть высвечена в символьной форме на этом индикаторе, для чего необходимо включить соответствующие сегменты. Например, цифра 7 высвечивается, когда горят сегменты а, в, с.

В цифровой измерительной аппаратуре важное место занимают преобразователи кодов. По определению и по сути преобразователем кодов

является любое КЦУ, поскольку, как и КЦУ, преобразователь кодов преобразует n-разрядный входной код в m-разрядный выходной. Частным случаем преобразователей кодов можно рассматривать шифраторы и дешифраторы, выполняющие преобразование десятичного кода в двоичный и обратно (рис. 12, 13, таблицы 3, 4). Однако существует класс устройств, традиционно относящихся по своим функциональным свойствам к категории преобразователей кодов.

Таблица 3

8	4	2	1		7	3	2	1
X_4	X_3	X_2	X_1	Y	X_4	X_3	X_2	X_1
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	0	2	0	0	1	0
0	0	1	1	3	0	1	0	0
0	1	0	0	4	0	1	0	1
0	1	0	1	5	0	1	1	0
0	1	1	0	6	0	1	1	1
0	1	1	1	7	1	0	0	0
1	0	0	0	8	1	0	0	1
1	0	0	1	9	1	0	1	0
1	0	1	0	10	1	1	0	0
1	0	1	1	11	1	1	0	1
1	1	0	0	12	1	1	1	0
1	1	0	1	13	1	1	1	1

Дешифратор Шифратор

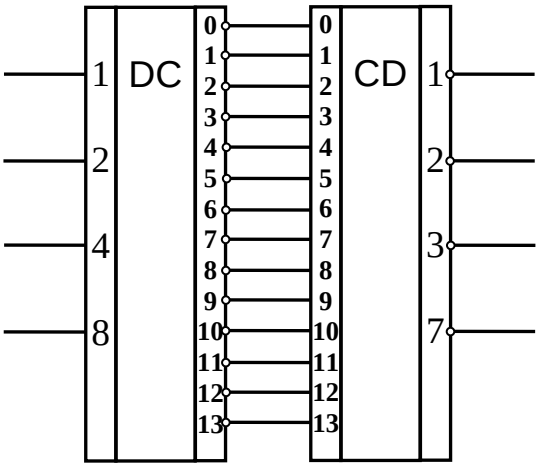


Рис. 12

Таблица 4

8	4	2	1		6	3	2	1
X_4	X_3	X_2	X_1	Y	X_4	X_3	X_2	X_1
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	0	2	0	0	1	0
0	0	1	1	3	0	1	0	0
0	1	0	0	4	0	1	0	1
0	1	0	1	5	0	1	1	0
0	1	1	0	6	1	0	0	0
0	1	1	1	7	1	0	0	1
1	0	0	0	8	1	0	1	0
1	0	0	1	9	1	1	0	0
1	0	1	0	10	1	1	0	1
1	0	1	1	11	1	1	1	0
1	1	0	0	12	1	1	1	1

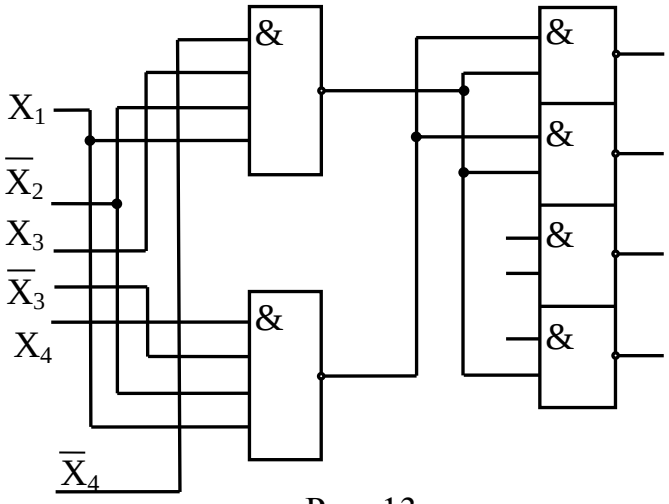


Рис. 13

Задание на лабораторную работу

1. Построить и собрать на стенде схему двоично-десятичного дешифратора.

2. Заполнить перекодирующую таблицу 8421-XXXX, где XXXX-код.

3. Синтезировать преобразователь кодов из элементов, имеющихся на стенде.

4. Заполнить перекодирующую таблицу 8421-сегментный код.

5. Оформить отчет, который должен содержать:

- схему двоично-десятичного шифратора,
- перекодирующую таблицу 8421-XXXX,

- схему преобразователи кодов в соответствии с вариантом, указанным в таблице 5.

Таблица 5

Входной код 8421	Вариант				
	1	2	3	4	5
Выходной код	2421	5221	Джонсона	Грея	8421+3

- рисунок семисегментного индикатора с обозначением сегментов,

- перекодирующую таблицу 8421-сегментный код, взаимное соответствие различных цифр различных кодов указано в таблице 6.

Таблица 6

Десятичная цифра	Коды					
	8421	2421	5221	Джонсона	Грея	8421+3
0	0000	0000	0000	00000	0010	0011
1	0001	0001	0001	00001	0110	0100
2	0010	0010	0010	00011	0111	0101
3	0011	0011	0011	00111	0101	0110
4	0100	0100	0110	01111	0100	0111
5	0101	1011	1000	11111	1100	1000
6	0110	1100	1001	11110	1101	1001
7	0111	1101	1010	11100	1111	1010
8	1000	1110	1011	11000	1110	1011
9	1001	1111	1110	10000	1010	1100

Контрольные вопросы

1. Шифратор: назначение, принципы построения.

2. Дешифратор «один из N»: назначение, принципы построения, работа микросхем К155 ИД1, К155 ИД3, К155 ИД4.

3. Семисегментный код, представление информации на семисегментных индикаторах.

4. Семисегментный дешифратор К155 ИД1, работа схемы.

5. Преобразователи кода на основе запоминающих устройств (ПЗУ).

6. Условные графические обозначения дешифраторов (УГО).

Лабораторная работа № 3
Последовательностные схемы. Триггеры, регистры

Цель работы: изучение принципов построения интегральных триггеров; приобретение навыка синтеза триггеров, регистров и сдвигающих регистров.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется лабораторный стенд УМ11. Стенд расположен в лаборатории № 261.10.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

Характерной особенностью последовательностных (многотактных) структур является зависимость состояния выходов не только от значений входных булевых переменных в данный момент времени, но и от внутренних состояний, определяемых тем, какие условия (последовательности) имели место в предшествующие моменты времени (номер такта).

Последовательностные структуры (логические автоматы с памятью), к числу которых относятся регистры, счетчики и другие системы цифровых устройств, обычно строятся на основе триггерных цепей с тем или иным законом функционирования.

Триггер является элементом, который может неограниченно долго находиться в одном из двух устойчивых состояний. Состояние триггера распознается по его выходному сигналу.

Логика работы определяется количеством входов и особенностями схемы. В зависимости от влияния, оказываемого на состояние триггера, его входы имеют следующие обозначения:

S - вход раздельной установки триггера в 1;

R - вход раздельной установки триггера в 0;

T - счетный вход;

D - вход задержки;

J - вход для синхронизируемой установки в 1;

K - вход для синхронизируемой установки в 0;

C - вход синхронизации;

Y - вход разрешения.

Входы и выходы триггеров так же, как и других логических элементов, могут быть прямыми и инверсными, т. е. наличие сигнала определяется высоким или низким уровнем напряжений соответственно.

Часто в структуре триггера присутствуют элементы, реализующие функцию разностного преобразования. В таких случаях можно обеспечить переключение триггера по заранее определенному изменению входных переменных (т. е. по переходу из логического «0» в «1» или наоборот). С наибольшей эффективностью разностные преобразователи используются в цепях синхронизации. Синхровходы такого рода носят название динамических.

Триггер типа D имеет расширенную функциональную схему, представленную на рис. 14.

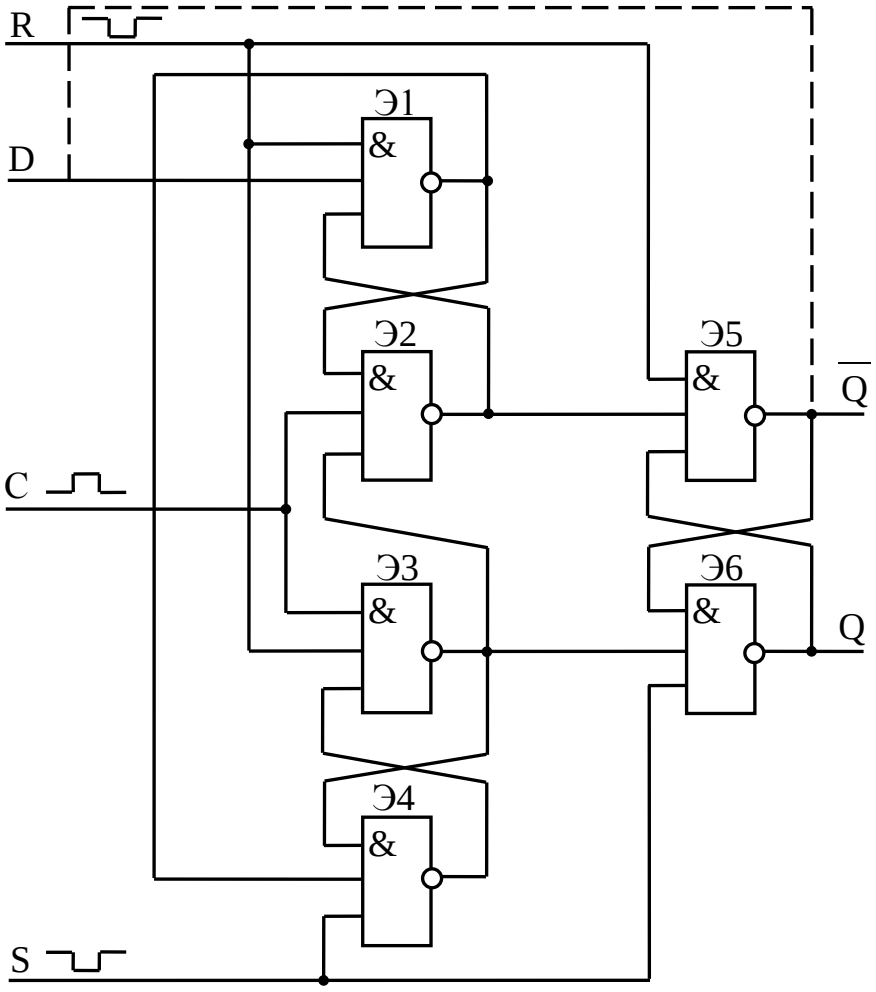


Рис. 14. Функциональная схема D-триггера

Триггер может работать в двух режимах: синхронном, при котором управление производится по входу D, и асинхронном – по R-S-входам.

Таблица переходов триггера в зависимости от сигнала на D-входе и синхроимпульса положительной полярности на входе С представлена в таблице 7.

Таблица 7

C	D	Q^{t+1}
0	0	Q^t
0	1	Q^t
1	0	0
1	1	1

Уровни сигналов для представления 0 и 1 те же, что и для логических элементов «И-НЕ», «И-ИЛИ-НЕ».

Триггер переключается по переднему фронту сигнала синхронизации. Для организации счетного режима необходимо инверсный выход триггера подсоединить к входу D (см. пунктир на рис. 14).

Триггером типа JK называется запоминающий элемент с двумя устойчивыми состояниями и информационными входами J (аналог S) и K (аналог R), которые обеспечивают соответственно раздельную установку состояний «1» и «0». Он функционирует подобно RS-триггеру, однако при совпадении сигналов J = 1 переключается в противоположное состояние, то есть реализует сложение сигналов по модулю два. Триггер типа J-K имеет функциональную схему, представленную на рис. 15. Таблица переходов J-K триггера приведена в таблице 8. J-K триггер по своей структуре является двухступенчатым. По переднему фронту положительного синхроимпульса переключается первая триггерная ступень, построенная на элементах Э3, Э4; по заднему фронту информация с первой ступени передается на вторую, оконечную триггерную ступень Э7, Э8. По асинхронным R-S входам J-K триггер управляется аналогично асинхронным входам триггера D. При каскадировании однотипных триггеров получаются регистры, предназначенные для запоминания слов, а также для выполнения над словами некоторых логических преобразований.

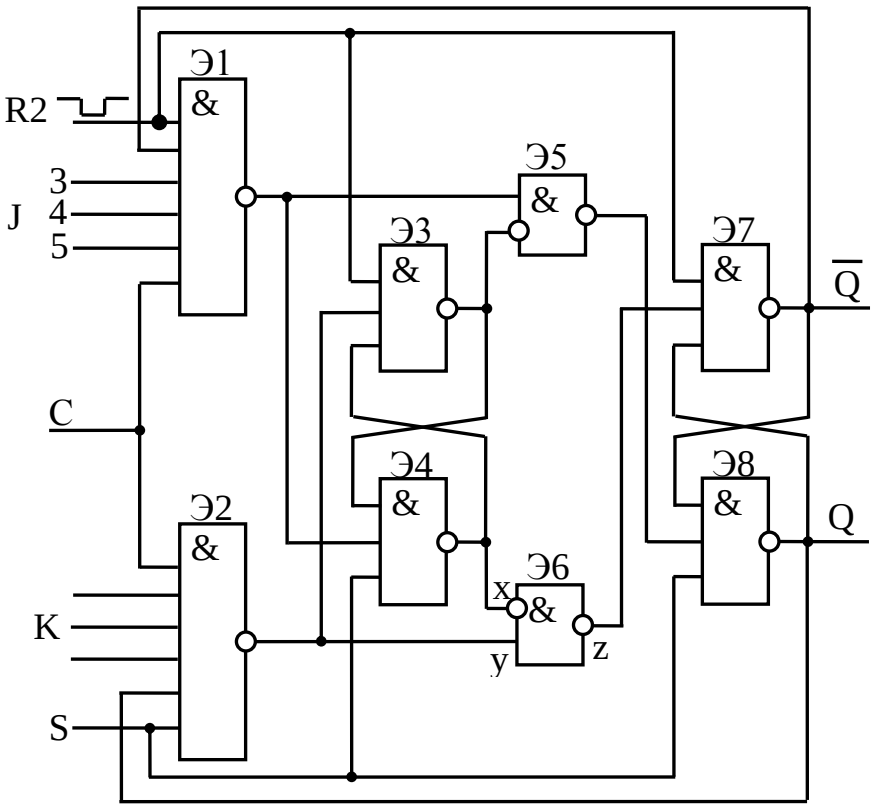


Рис. 15. Функциональная схема J-K-триггера

Таблица 8

C	K	J	Q(t)	Q(t+1)	Пояснение
0	X	X	0	0	Режим хранения информации
0	X	X	1	1	
1	0	0	0	0	Режим хранения информации
1	0	0	1	1	
1	0	1	0	1	Режим установки единицы J=1
1	0	1	1	1	
1	1	0	0	0	Режим записи нуля K=1
1	1	0	1	0	
1	1	1	0	1	K=J=1 счетный режим триггера
1	1	1	1	0	

К числу операций, которые способны выполнять типичные регистры, относятся следующие:

- установка регистра в «0» («сброс»),
- прием слова из другого устройства (регистра, сумматора и т. п.),
- передача слова в другой регистр,

- сдвиг слова вправо или влево на требуемое число разрядов; преобразование, таким образом, параллельного кода слова в последовательный и наоборот.

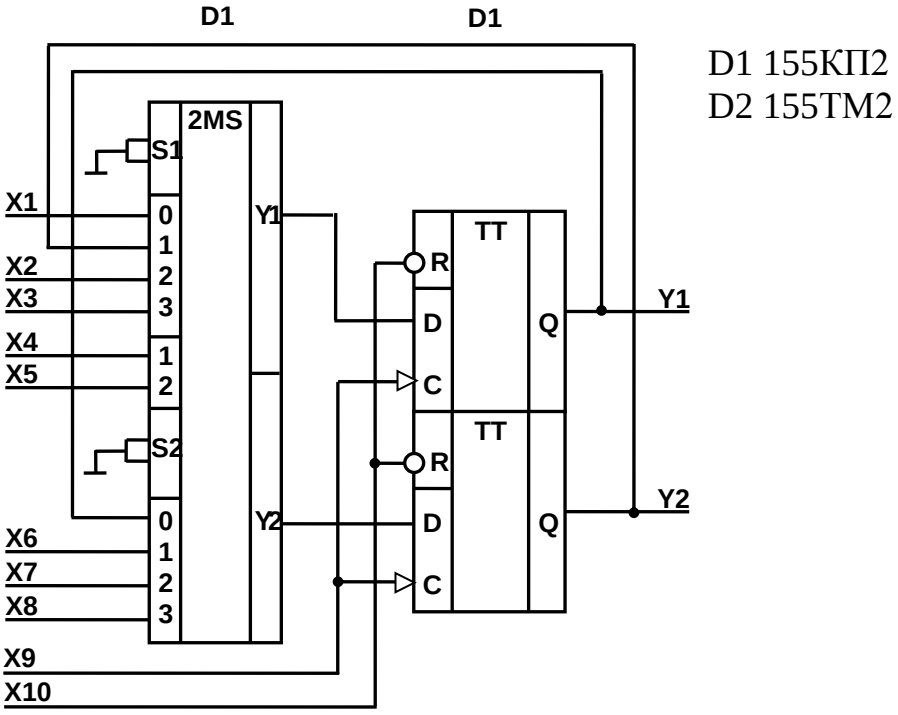


Рис. 16. Многофункциональный регистр

Поскольку один регистр выполняет чаще всего несколько из перечисленных операций, помимо триггеров в его состав входят вспомогательные комбинационные схемы, служащие, как правило, целям коммутации (подключающие, например, информационные входы триггеров к тем или иным источникам сигнала; позволяющие осуществлять управление процессом записи от того или иного синхросигнала).

Пример построения многофункционального регистра, реализующего вышеперечисленные операции, приведен на рис. 16. Синхровходы триггеров, как показано на схеме, являются динамическими.

Операции сдвига выполняются на регистрах сдвига. Регистр сдвига представляет собой схему на триггерах, соединения между которыми, называемые цепями сдвига, обеспечивают передачу двоичной информации от одних триггеров регистра к другим.

Регистр сдвига, в котором для хранения информации используется k триггеров, называется n-разрядным регистром сдвига.

По способу записи информации различают:

1) параллельные регистры, в которых запись числа осуществляется во все разряды одновременно параллельным кодом;

2) последовательные регистры с записью кода числа путем его последовательного сдвига тактирующими сигналами, начиная с младшего или старшего разряда;

3) параллельно-последовательные регистры, которые имеют входы для последовательной и параллельной записи кода.

По количеству входных каналов регистры подразделяются на:

1) парафазные с записью информации по двум каналам: прямому и инверсному;

2) однофазные с записью по одному каналу.

В зависимости от способа тактирования различают регистры однотактного и многотактного действия.

Одной из основных характеристик сдвигающего регистра является быстродействие, которое определяется временем сдвига на один или несколько разрядов одновременно. Время сдвига равно промежутку времени между моментом поступления импульса на шину сдвига и моментом окончания переходного процесса в схеме, вызванного этим импульсом.

Задание на лабораторную работу

1. Изучить и собрать асинхронный RS-триггер на элементах «2-И-НЕ» (рис. 17).

2. Преобразовать собранный триггер в синхронизируемый (рис. 18).

3. Построить двухтактный RS-триггер на элементах «2-И-НЕ» (рис. 19).

4. Ознакомиться с универсальным JK-триггером 155ТВ1 (рис. 20). На его основе получить RS-, D-,T-триггеры (рис. 21–23).

5. Ознакомиться с другими триггерами 155-й серии, в частности ТМ2, ТМ5, ТМ7, ТМ8.

6. Построить схему, реализующую межрегистровую передачу двоичных чисел в единичном коде (рис. 24).

7. Построить схему, реализующую межрегистровую передачу двоичных чисел в парафазном коде (рис. 25).

8. Построить схему сдвигающего регистра.

9. Оформить отчет, который должен содержать схемы всех рассмотренных триггеров, их условные графические обозначения и соответствующие таблицы переходов, а также УГО микросхем 155 ТВ1, ТМ2, ТМ5, ТМ7, ТМ8.

Контрольные вопросы

1. Взаимозаменяемы ли триггеры с динамическими синхровходами и фиксаторы?

2. Каким образом следует включать фиксаторы ТМ7, чтобы они по своим возможностям полностью соответствовали двухступенчатым триггерам ТМ8?

3. Что необходимо предпринять, чтобы триггер, срабатывающий по переднему фронту синхросигнала, стал срабатывать по его спаду?

4. Чем определяется быстродействие триггера?

5. Опишите работу триггеров К155ТВ1, ТМ2, ТМ7, ТМ8.

6. Опишите работу сдвигающего регистра К155 ИР1. Приведите примеры схемных решений.

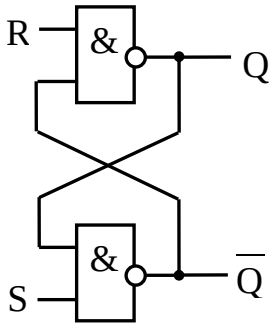


Рис. 17. Асинхронный RS-триггер

R	S	Q^{t+1}	$\overline{Q}^{t+1}$
0	0	-	-
0	1	1	0
1	0	0	1
1	1	Q^t	Q^t

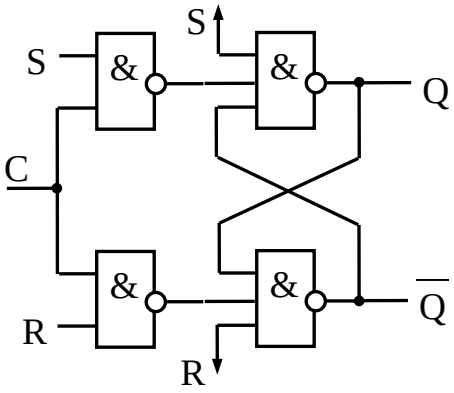


Рис. 18. Синхронный RS-триггер

C	S	R	Q^{t+1}	$\overline{Q}^{t+1}$
1	0	0	Q^t	Q^t
1	0	1	0	1
1	1	0	1	0
~~1~~	~~1~~	~~1~~	~~1~~	~~1~~
0	0	0	Q^t	$\overline{Q}^t$
0	0	1	Q^t	$\overline{Q}^t$
0	1	0	Q^t	$\overline{Q}^t$
0	1	1	Q^t	$\overline{Q}^t$

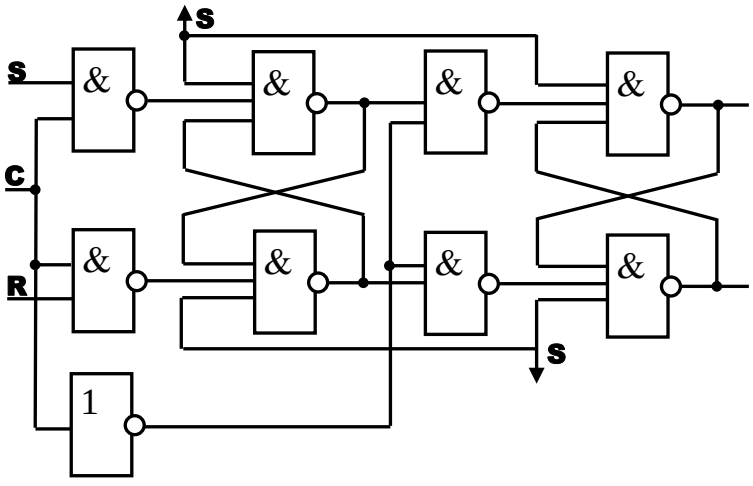


Рис. 19. Двухтактный RS-триггер

S	R	C	Q^{t+1}
0	0	0	Q^t
0	1	0	Q^t
1	0	0	Q^t
1	1	0	Q^t
0	0	1	Q^t
0	1	1	0
1	0	1	1
1	1	1	-

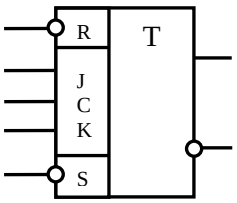


Рис. 20. JK-триггер

C	J	K	Q^{t+1}
0	0	0	Q^t
0	0	1	Q^t
0	1	0	Q^t
0	1	1	Q^t
1	0	0	Q^t
1	0	1	0
1	1	0	1
1	1	1	$\neg Q^t$

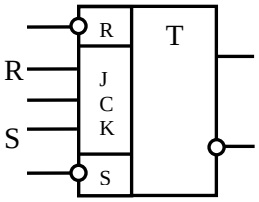


Рис. 21. RS-триггер

C	S	R	Q^{t+1}
0	0	0	Q^t
0	0	1	Q^t
0	1	0	Q^t
0	1	1	Q^t
1	0	0	Q^t
1	0	1	0
1	1	0	1
1	**1**	**1**	**-**

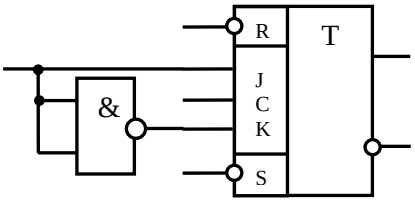


Рис. 22. D-триггер

C	D	Q^{t+1}
0	0	Q^t
0	1	Q^t
1	0	0
1	1	1

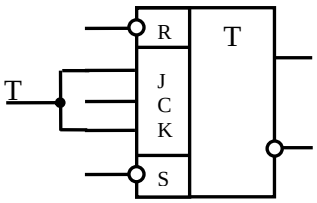


Рис. 23. T-триггер

C	T	Q^{t+1}
1	0	Q^t
1	1	$\neg Q^t$
0	0	Q^t
0	1	Q^t

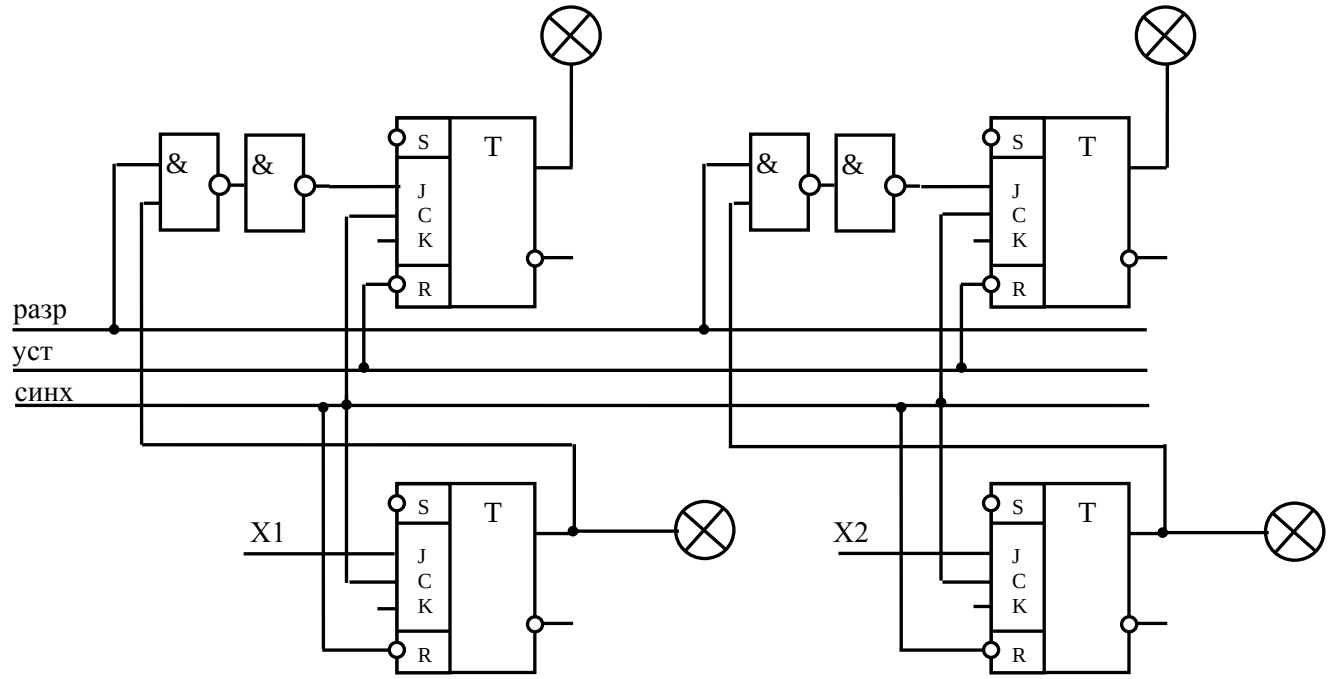


Рис. 24. Межрегистровая передача двоичных чисел в единичном коде

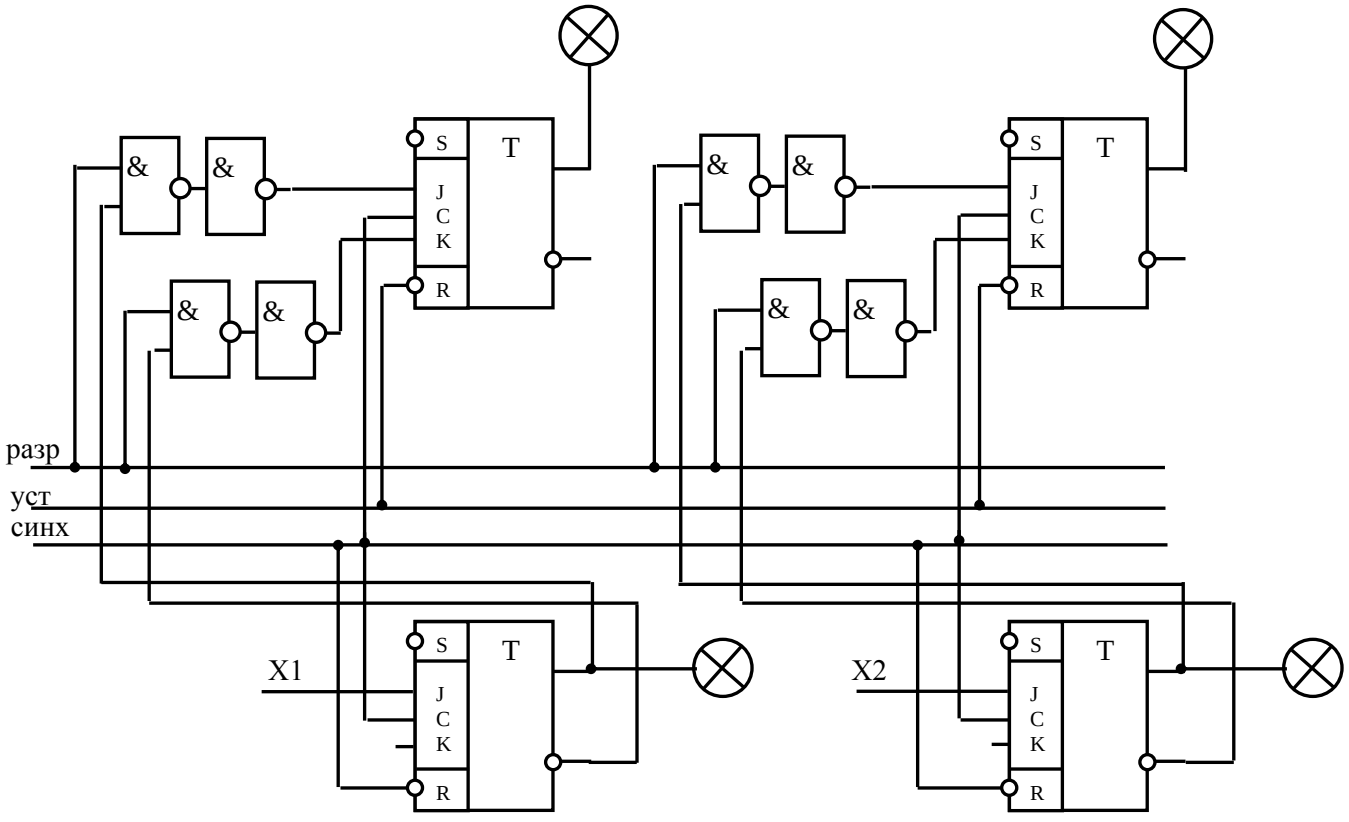


Рис. 25. Межрегистровая передача двоичных чисел в парафазном коде

Лабораторная работа № 4
Элементы операционных устройств. Мультиплексоры. Сумматоры

Цель работы: изучение принципов построения мультиплексоров и сумматоров; синтез блоков операционных устройств.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется лабораторный стенд УМ11. Стенд расположен в лаборатории № 261.10.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Процессоры любых ЭВМ включают в себя операционное и управляющее устройства. Операционное устройство предназначено для обработки операндов и команд, управляющее - для управления вычислительным процессом, состоящим в выборке команд из памяти, дешифрации и преобразовании их в управляющие и синхронизирующие сигналы, координирующие работу всех узлов ЭВМ.

В операционное устройство входят следующие узлы (рис. 26):

- операционные блоки, предназначенные для непосредственного исполнения микроопераций преобразования информации, поступающей на их входы;
- группы регистров, предназначенных для приема и хранения информации, над которой производятся действия в операционных блоках;
- блок формирования состояний, предназначенный для формирования арифметико-логических условий перехода, используемых в микропрограммном устройстве управления для изменения последовательности выборки микрокоманд.

Как видно из рис. 26, важное место в структуре операционного устройства занимают мультиплексоры – узлы, предназначенные для коммутации логических сигналов, поступающих на АЛУ с других блоков.

Распространено использование мультиплексоров для реализации операций арифметического и логического сдвигов.

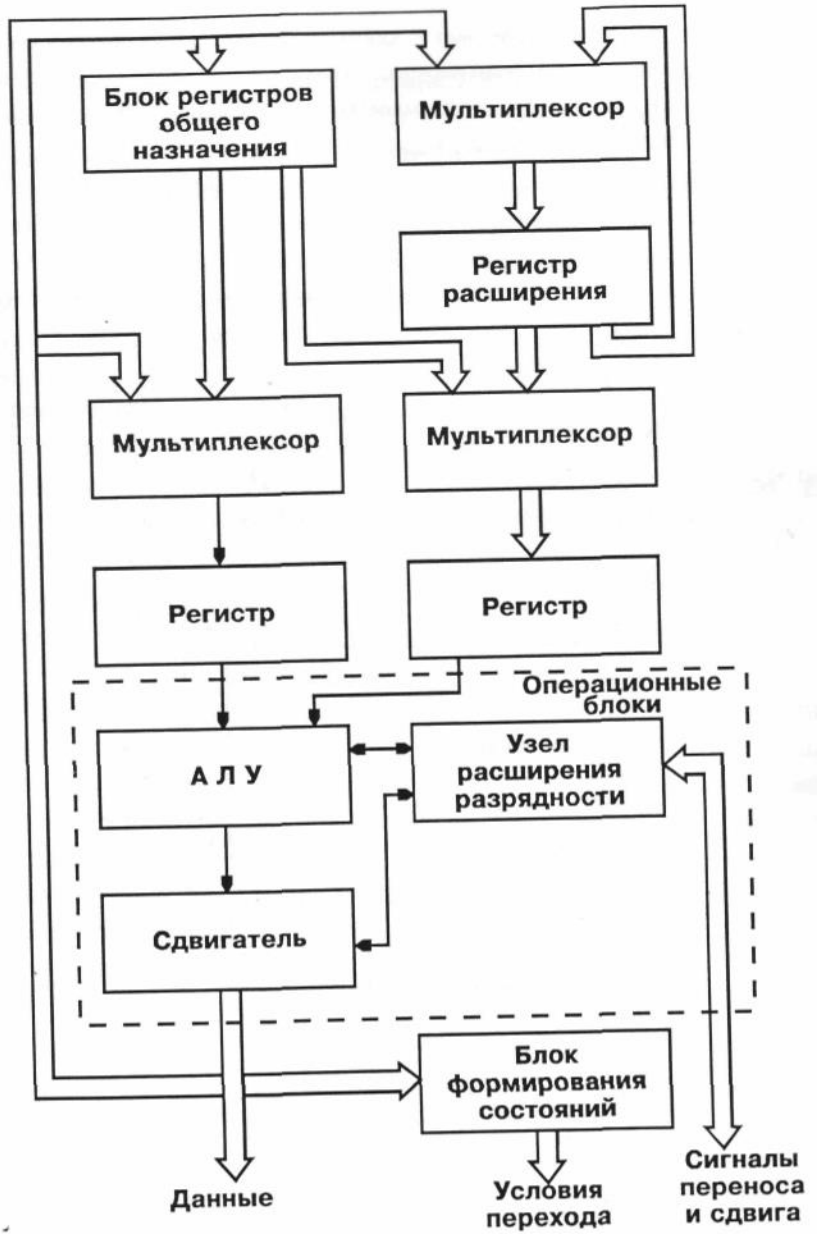


Рис. 26. Структура типового операционного устройства

Мультиплексор – устройство, сигнал на выходе которого определяется сигналом на одном из информационных входов заданным адресным кодом. Мультиплексор представляет собой комбинированное цифровое устройство, обеспечивающее поочередную передачу на один выход нескольких входных сигналов. Он позволяет передавать (коммутировать) сигнал с желаемого входа на выход, в этом случае выбор требующегося входа реализуется определенной комбинацией управляющих сигналов. Число мультиплексных входов принято называть количеством каналов, их может быть от 2 до 16, а число выходов называют разрядами мультиплексора, обычно это 1–4.

Наглядным примером мультиплексора является элемент К155 ЛР3. Очевидно, что адресный код в данном случае должен быть вида «один-из-четырех» (рис. 27).

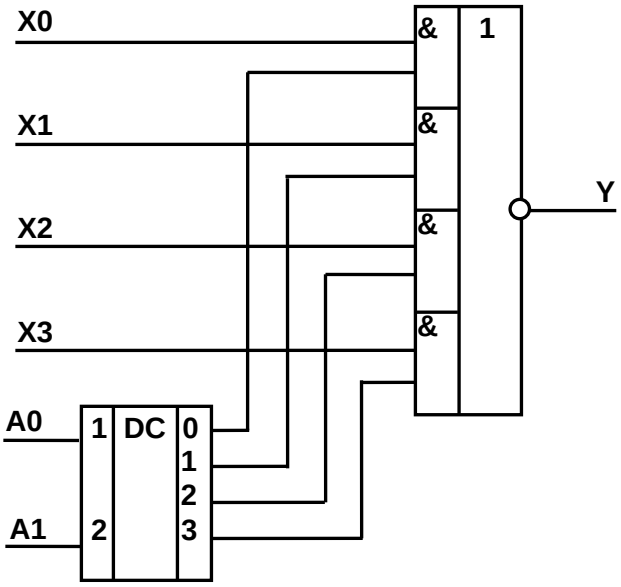


Рис. 27. Элемент К155 ЛР3

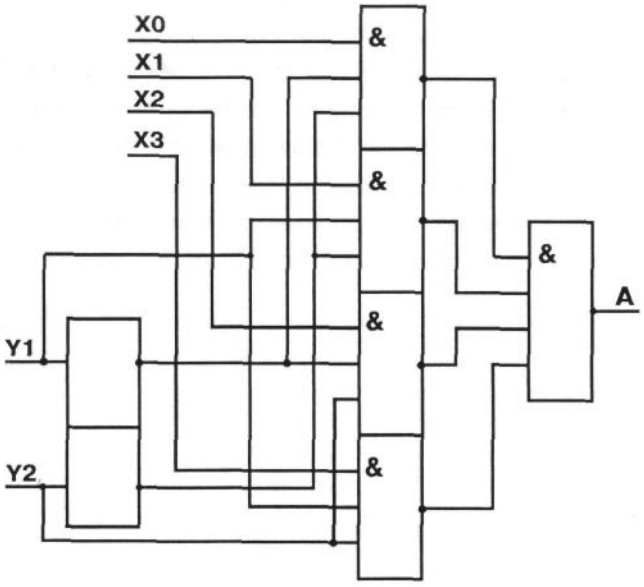


Рис. 28. Схема мультиплексора

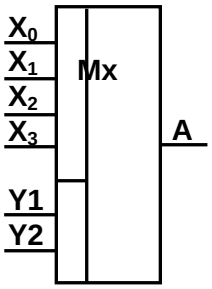


Рис. 29. Мультиплексор с четырьмя информационными входами

Основным узлом АЛУ, входящего в состав операционного устройства, является *сумматор* – логический элемент, реализующий алгебраическое сложение двух чисел.

При сложении двух чисел независимо от системы счисления в каждом разряде производится сложение трех цифр: цифры данного разряда первого слагаемого, цифры данного разряда второго слагаемого и цифры (1 или 0) переноса из соседнего младшего разряда. В результате сложения для каждого разряда получаются цифры суммы для этого разряда и цифры (1 или 0) переноса в следующий старший разряд.

Рассмотрим операцию сложения для одного какого-нибудь разряда складываемых чисел в двоичной системе счисления. В зависимости от значений складываемых цифр и наличия или отсутствия единицы переноса из предыдущего разряда результат сложения будет различным. В таблице 9 приведены восемь возможных вариантов, возникающих при сложении двух двоичных чисел.

Таблица 9

Цифра переноса из предыдущего разряда, p_i	Первое слагаемое, X_i	Второе слагаемое, Y_i	Сумма, S_i	Цифра переноса в старший разряд, $p_i + 1$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

По приведенной таблице можно составить булевы функции для описания одноразрядного сумматора – устройства, вырабатывающего на выходе сигналы суммы и переноса из предыдущего разряда.

Преобразуя выражения для цифры суммы цифры переноса с помощью правил булевой алгебры, можно получать различные соотношения для построения схем полных сумматоров на базе различных элементов.

Поскольку на практике приходится суммировать более чем одноразрядные числа, практический интерес представляют способы построения многоразрядных сумматоров.

Наиболее просто можно осуществить построение параллельных сумматоров, число которых равно числу разрядов слагаемых, путем соединения выхода, на котором фиксируется сигнал переноса данного разряда (рис. 30).

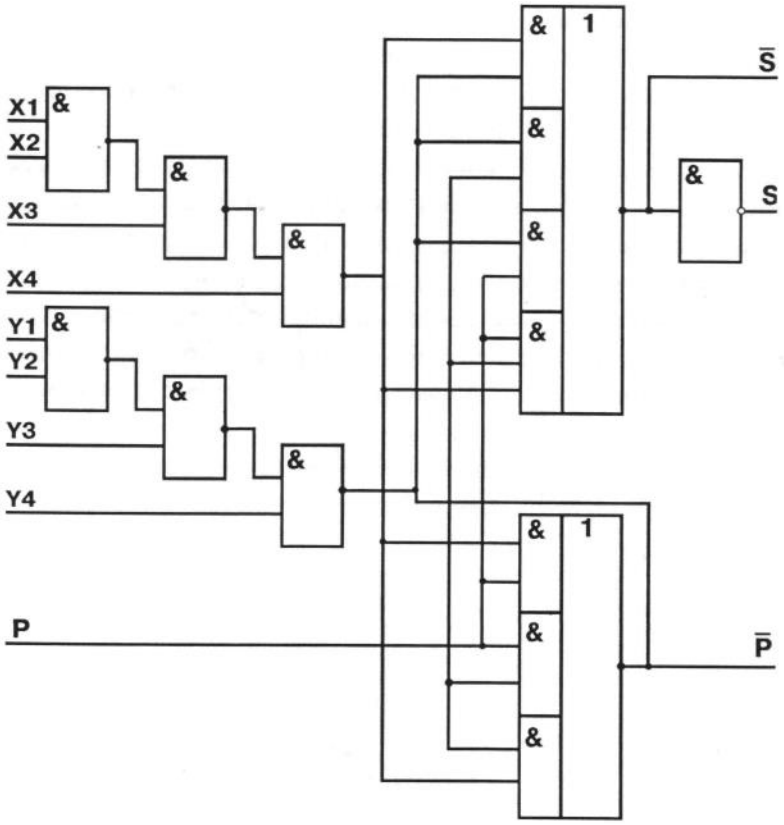


Рис. 30. Одноразрядный сумматор К155ИМ1

В сумматорах этого типа, называемых сумматорами с поразрядным последовательным переносом, перенос распространяется последовательно от разряда к разряду по мере образования цифры суммы в каждом отдельном разряде. При наиболее неблагоприятных условиях для распространения переноса будет иметь место «пробег» единицы переноса через весь сумматор от самого младшего к самому старшему разряду, поэтому время суммирования двух чисел с момента одновременной подачи слагаемых на входы такого сумматора определяется следующей формулой:

$$t_{\Sigma}=t_s+n\cdot t_p\ ,$$

где t_s - время формирования сигнала суммы в одном разряде;

t_p - время задержки сигнала переноса в одном разряде;

n - число разрядов параллельного сумматора.

Из приведенной формулы видно, что повысить быстродействие можно двумя способами: во-первых, уменьшением времени задержки сигнала

переноса в одноразрядном сумматоре; во-вторых, уменьшением влияния числа разрядов на время распространения переноса.

Микросхемы 155ИМ2 и 155ИМ3 представляют собой соответственно двух- и четырехразрядные сумматоры, построенные на указанных принципах. Столь широкое использование последовательного переноса с одной инверсией обусловлено простотой с точки зрения построения цепей распространения переноса.

Второй способ реализуется в сумматорах с параллельным и с групповым переносом, причем сумматоры первого вида, в которых сигналы переноса каждого разряда формируются одновременно с помощью специальной комбинационной схемы, почти не находят применения в чистом виде в многоразрядных сумматорах вследствие значительности затрат на построение данной схемы, а используются для построения сумматоров с групповым переносом.

Сумматор с групповым переносом разбивается на несколько групп примерно равной длины. Сигнал переноса, поступающий на вход младшего разряда группы, при наличии условий распространения переноса во всех разрядах данной группы передается на вход младшего разряда соседней, более старшей группы.

Порядок выполнения работы

1. Собрать мультиплексор с четырьмя информационными входами (рис. 28, 29).
2. Составить булевы функции для описания одноразрядного полного сумматора согласно таблице 9.
3. Собрать 2-х разрядный комбинационный сумматор.
4. Собрать 2-х разрядный сумматор накапливающего типа.

Контрольные вопросы

1. Назначение, принципы построения мультиплексора.
2. Реализация логических функций на основе мультиплексоров.
3. УГО.
4. Работа микросхем 155КР1, КР2, КР5, КРУ.
5. Каскадирование мультиплексоров.
6. Назначение, принципы построения сумматора.
7. Булевы функции, описывающие работу одноразрядного полного сумматора.
8. Полные и неполные сумматоры.
9. Параллельные и последовательные сумматоры.

Лабораторная работа № 5
Счетчики

Цель работы: изучение принципов построения счетчиков; приобретение навыков синтеза цифровых автоматов.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется лабораторный стенд УМ11. Стенд расположен в лаборатории № 261.10.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Счетчиком в цифровой технике называется схема для хранения числа, представляющая собой совокупность триггеров и схем управления ими, позволяющая изменять это число на некоторую константу, а также имеющая, как правило, цепи установки заданной величины, в частности - нуля.

Модулем счетчика (М) называется число разрешенных устойчивых состояний. Сигналы, поступающие на вход счетчика, называются считаемыми; они могут быть, в зависимости от знака, прибавляемыми или вычитаемыми и иметь обозначение +К и -К соответственно. В тех случаях, когда константа, на которую изменяется состояние счетчика, равна единице, указанные входы имеют обозначения +1 и -1.

Установка счетчика в исходное состояние может производиться либо подачей общего для всех триггеров сигнала (как правило, сброс), либо путем установки каждого триггера в отдельности индивидуальным сигналом.

При создании счетчика с модулем М требуется не менее $K=\log_2 M$ триггеров. Следовательно, двоично-десятичный счетчик должен содержать не менее, чем четыре триггера. Схема двоично-десятичного счетчика в коде 8421 (где 8, 4, 2, 1 – веса двоичных разрядов) приведена на рис. 31. В этой системе кодирования каждая десятичная цифра записывается четырьмя двоичными (таблица 10). Работа этого десятичного счетчика поясняется диаграммой (рис. 32).

Данный счетчик является синхронным. Можно представить и асинхронную организацию структуры счетчика, когда на синхронизирующие входы триггеров поступают не считываемые сигналы, а сигналы с входов логических элементов или соседних триггеров. Преимуществом асинхронной организации счетчиков является простота получаемой структуры. Но при

проектировании асинхронных двоично-десятичных счетчиков возникают трудности, обусловленные различием во внутренней организации триггеров. Эти различия появляются только при асинхронной работе. Покажем это на примере асинхронного двоично-десятичного счетчика в коде 8421.

На рис. 32 приведена схема асинхронного счетчика на синхронных двухступенчатых JK-триггерах. На рис. 33 приведен также асинхронный двоично-десятичный счетчик в коде 8421, но с использованием JK-триггеров с динамическим управлением записью. При сравнении схем (см. рис. 32 и 33) ясно видны отличия в структурах асинхронного счетчика, построенных на основе различных типов JK-триггеров.

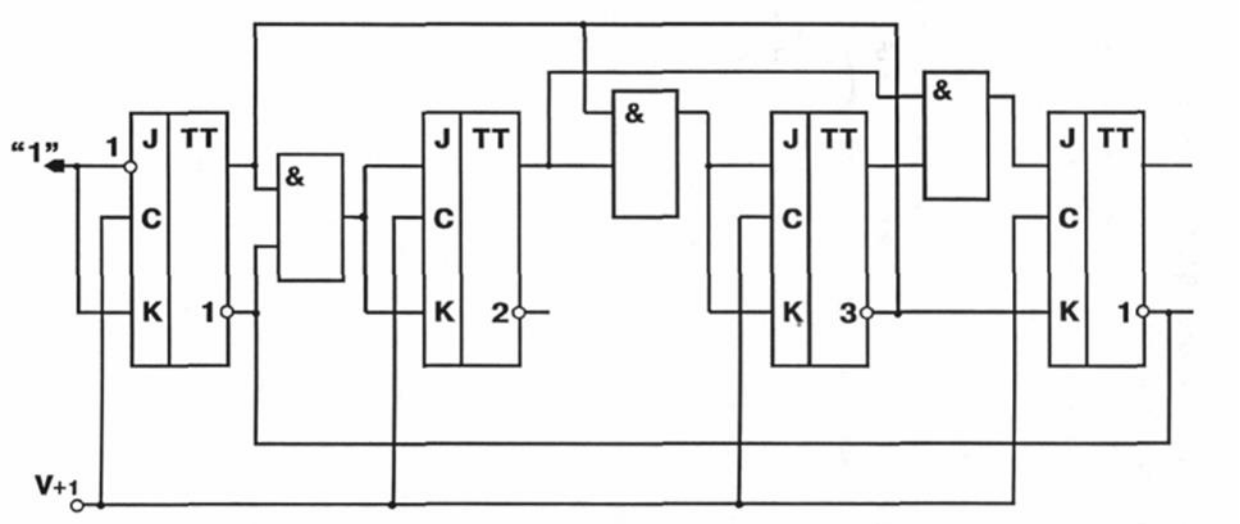


Рис. 31. Синхронный двоично-десятичный счетчик в коде 8421

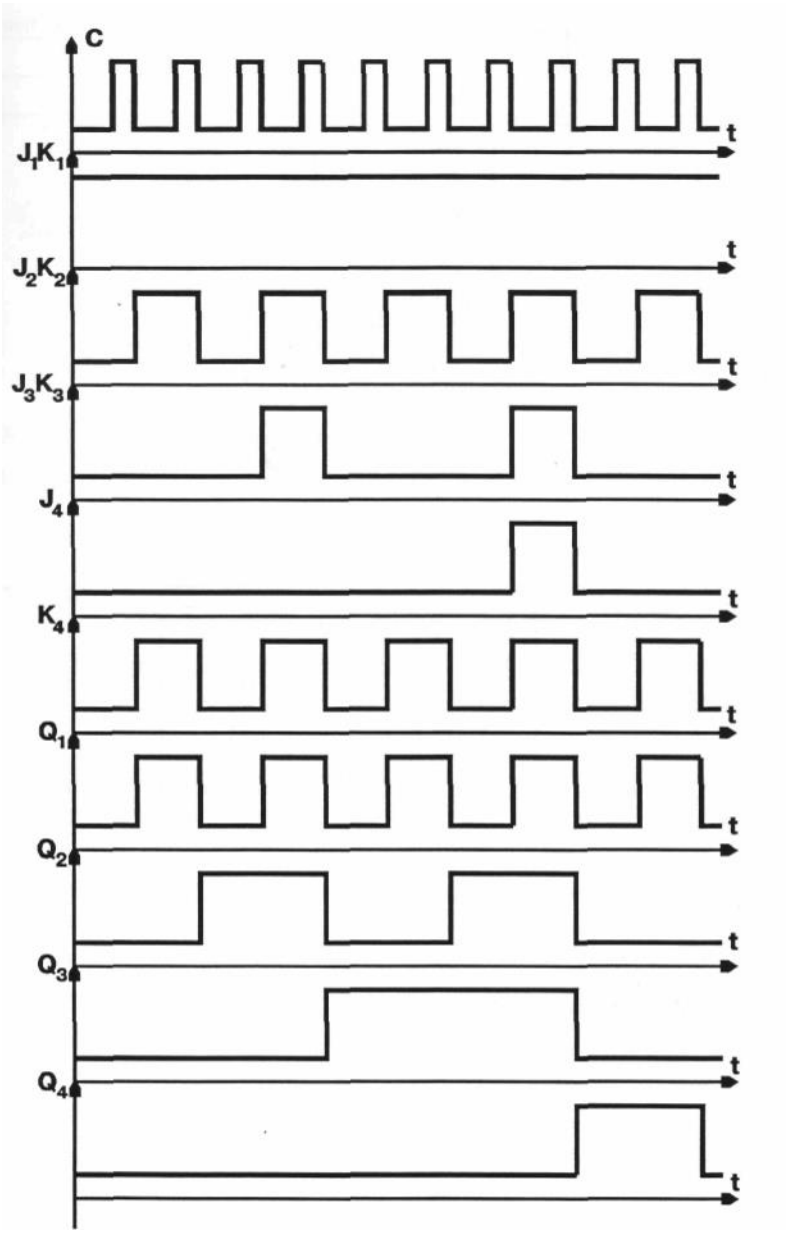


Рис. 32. Временная диаграмма работы счетчика

Таблица 10

Номер п/п	Состояние счетчика	Состояние битов			
		4	3	2	1
1	0	0	0	0	0
2	1	0	0	0	1
3	2	0	0	1	0
4	3	0	0	1	1
5	4	0	1	0	0
6	5	0	1	0	1
7	6	0	1	1	0
8	7	0	1	1	1
9	8	1	0	0	0
10	9	1	0	0	1

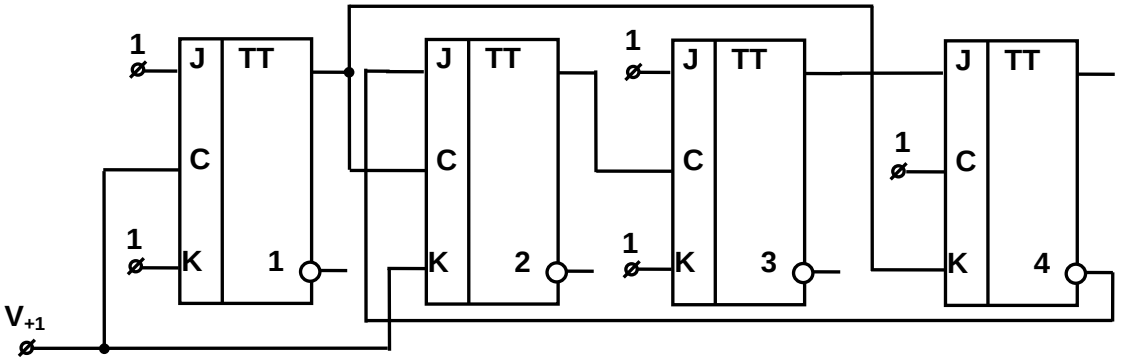


Рис. 32. Асинхронный двоично-десятичный счетчик в коде 8421 на синхронных двухступенчатых JK-триггерах

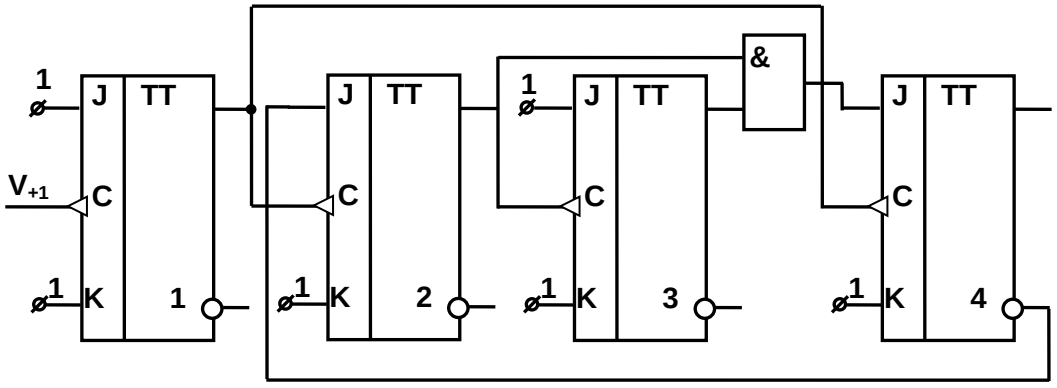


Рис. 33. Асинхронный двоично-десятичный счетчик в коде 8421 на синхронных JK-триггерах с динамическим управлением записью

Структура синхронного счетчика останется неизменной, если двухступенчатые JK-триггеры заменить триггерами с динамическим управлением записью. Если для логического проектирования синхронных счетчиков существуют отработанные методы, то для проектирования

асинхронно работающих счетчиков таких систематизированных методов нет. Все осложняется тем, что различия во внутреннем строении триггеров появляются именно при асинхронной работе. Поэтому проектирующий схемы должен иметь совершенно четкое представление о внутреннем строении используемого им типа триггеров и не ограничиваться таблицей переходов, которая описывает только синхронную работу триггера. Последовательным соединением n счетчиков с модулем M можно получить счетчик с модулем $M_{\Sigma}.=M^n$.

На рис. 34 приведена обобщенная схема логической структуры синхронного счетчика. Из этой схемы можно уяснить принцип работы любого синхронного счетчика. Сигналы с выходов триггеров поступают на входы комбинационной схемы, которая преобразует поступившую информацию. Сигналы с выходов комбинационной схемы подают на входы триггеров. Преобразованная информация не воспринимается триггерами до тех пор, пока на синхронизирующие входы не поступит подсчитываемый сигнал V. Информация, находящаяся на входах каждого триггера, так преобразуется комбинационной схемой, чтобы с приходом очередного подсчитываемого сигнала осуществить требуемый переход счетчика из предыдущего состояния в следующее.

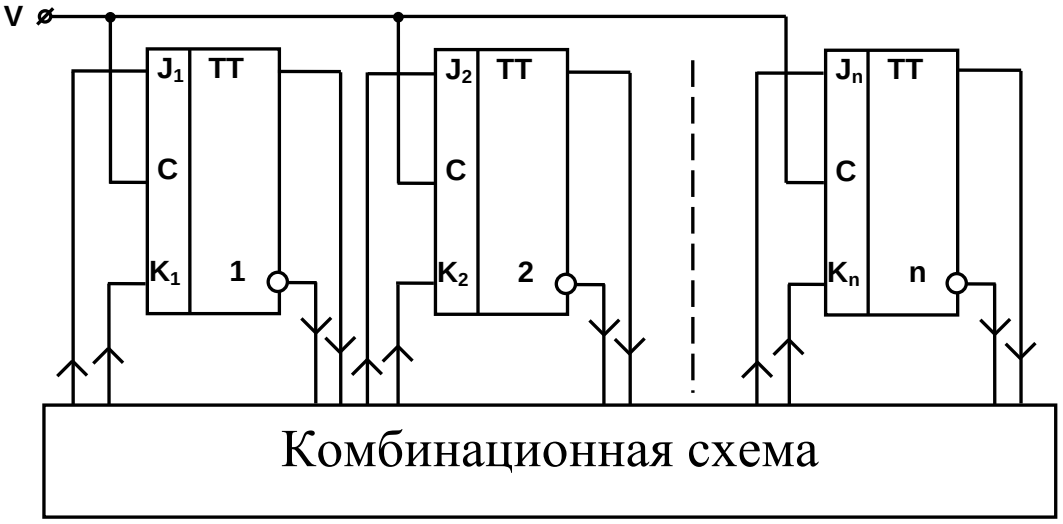


Рис. 34. Обобщенная схема логической структуры синхронного счетчика

Таким образом, функции возбуждения входов триггера можно записать в виде:

$J_{1i}(t)=f_{1i}\,[Q_1(t), Q_2(t), ..., Q_n(t)]$;

$K_{2i}(t)=f_{2i}\,[Q_1(t), Q_2(t), ..., Q_n(t)]$.

Значения всех переменных в этом выражении определены для одного и того же момента времени t. Поэтому функции возбуждения триггеров являются

переключательными функциями, которым соответствуют комбинационные схемы, формирующие входные сигналы для триггеров. Следовательно, если задан тип триггера, то задача логического проектирования схемы счетчика заключается в составлении функции возбуждения каждого триггера и минимизации найденных функций в заданном базисе.

Пример синтеза синхронного счетчика

Спроектировать двухразрядный двоично-десятичный счетчик с системой кодирования 2421 (2, 4, 2, 1 – веса двоичных разрядов) на JK-триггерах. В этой системе кодирования вместо каждой десятичной цифры записываются четыре двоичные.

Для составления функций возбуждения каждого триггера десятичного разряда счетчика на JK-триггерах воспользуемся матрицей переходов JK-триггера (рис. 5.6, 5.7).

Таблица 11. Переходы и функции возбуждения счетчика

Десятичные цифры	Номер набора	Значения прямых выходов триггеров								Время							
		Время t				Время t+1				Функции возбуждения триггеров							
										T4		T3		T2		T1	
		Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	0	0	1	0	a_i	0	a_i	0	a_i	1	a_i
1	1	0	0	0	1	0	0	1	0	0	a_i	0	a_i	1	a_i	a_i	1
2	2	0	0	1	0	0	0	1	1	0	a_i	0	a_i	a_i	0	1	a_i
3	3	0	0	1	1	0	1	0	0	0	a_i	1	a_i	a_i	1	a_i	1
4	4	0	1	0	0	1	0	1	1	1	a_i	a_i	1	1	a_i	1	a_i
5	11	1	0	1	1	1	1	0	0	a_i	0	1	a_i	a_i	1	a_i	1
6	12	1	1	0	0	1	1	0	1	a_i	0	a_i	0	0	a_i	1	a_i
7	13	1	1	0	1	1	1	1	0	a_i	0	a_i	0	1	a_i	a_i	1
8	14	1	1	1	0	1	1	1	1	a_i	0	a_i	0	a_i	0	1	a_i
9	15	1	1	1	1	0	0	0	0	a_i	1	a_i	1	a_i	1	a_i	1

Время		t	t+1
J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Рис. 35. Матрица переходов триггера (составляется по таблице переходов этого триггера)

Q(t)→Q(t+1)	J	K
0→0	0	a_i
0→1	1	a_i
1→0	a_i	1
1→1	a_i	0

Рис. 36. Матрица переходов JK-триггера

Схема одноразрядного десятичного счетчика приведена на рис. 37. При реализации счетчика было учтено, что JK-триггеры имеют три входа J, связанных функцией конъюнкции, и три входа К, также связанных функцией конъюнкции. Это позволяет уменьшить число логических элементов в комбинационной схеме счетчика.

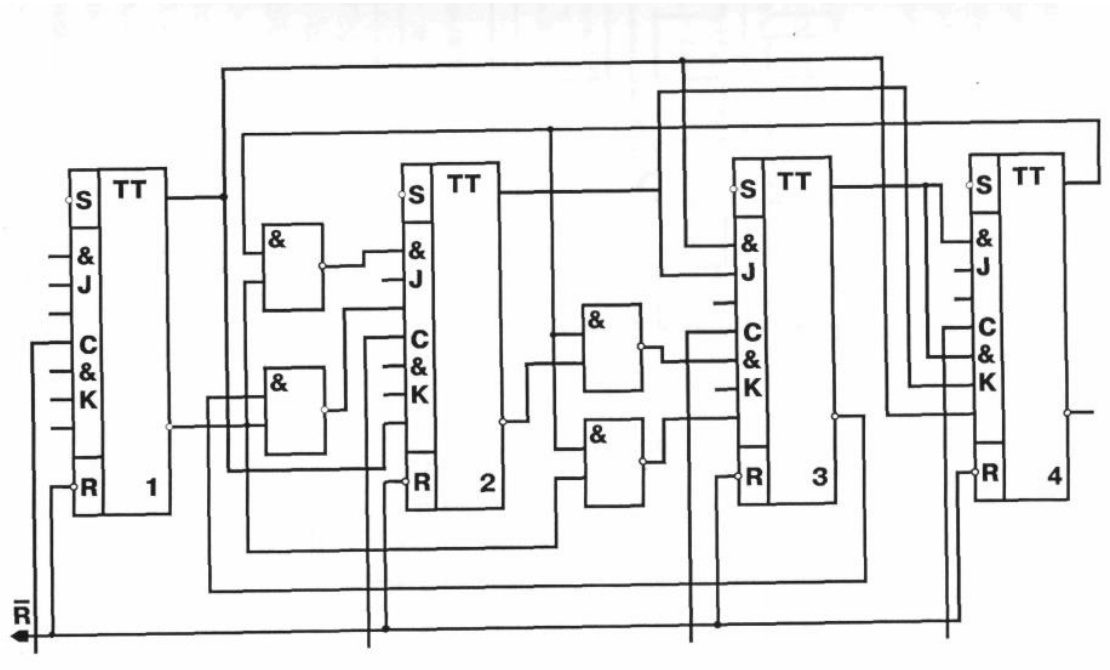


Рис. 37. Двоично-десятичный счетчик в коде 2421

Порядок выполнения работы

1. В соответствии с полученным вариантом и таблицей 12 синтезировать двоично-десятичный счетчик.

Таблица 12

Десятичная цифра	Коды					
	8421	2421	5221	Джонсона	Грея	8421+3
0	0000	0000	0000	00000	0010	0011
1	0001	0001	0001	00001	0110	0100
2	0010	0010	0010	00011	0111	0101
3	0011	0011	0011	00111	0101	0110
4	0100	0100	0110	01111	0100	0111
5	0101	1011	1000	11111	1100	1000
6	0110	1100	1001	11110	1101	1001
7	0111	1101	1010	11100	1111	1010
8	1000	1110	1011	11000	1110	1011
9	1001	1111	1110	10000	1010	1100

2. Оформить отчет, который должен содержать:

- матрицу переходов JK-триггера,
- таблицу переходов и функций возбуждения счетчика,
- эталонную диаграмму Вейча,
- диаграммы Вейча для функций возбуждения JK-триггера,
- минимизированные при помощи диаграмм Вейча функции возбуждения JK-триггеров (все конъюнкции из получившихся выражений необходимо исключить),
- принципиальную схему синтезированного счетчика.

Контрольные вопросы

1. Двоично-десятичный счетчик.
2. Синхронные и асинхронные счетчики.
3. Двухступенчатые триггеры с динамическим управлением записью: различия в работе, возможности при построении синхронных и асинхронных счетчиков.
4. Последовательные соединения счетчиков, модуль полученного каскада.
5. Быстродействие счетчиков.
6. Порядок проектирования синхронного счетчика.
7. Как получена диаграмма Вейча?
8. Сохраняется ли состояние счетчика при отключении питания?

9. Временная диаграмма работы счетчика.

Примеры схемных решений приведены на рис. 38, 39.

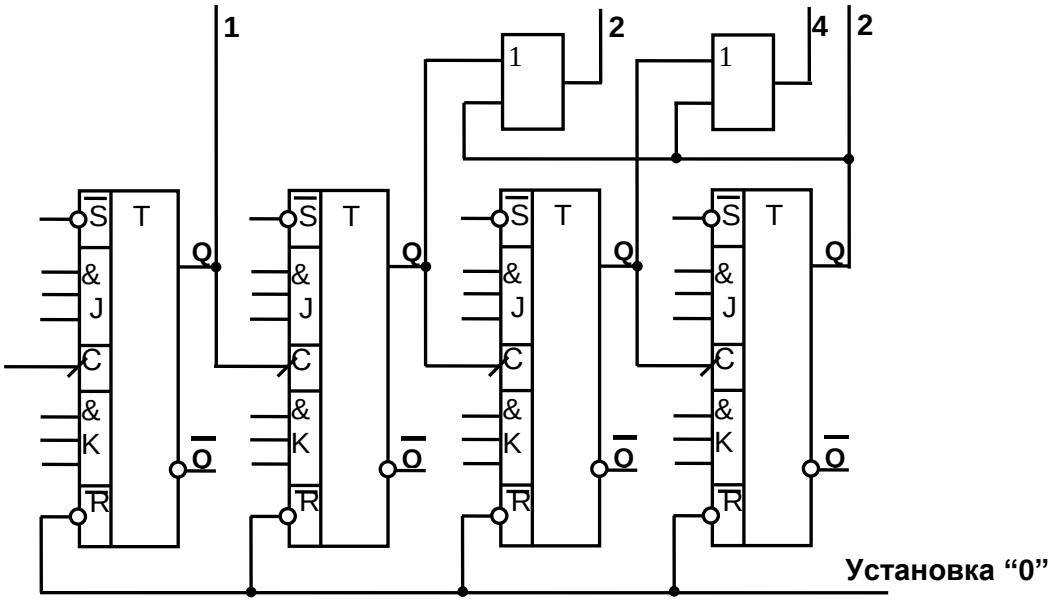


Рис. 38. Четырехразрядный двоично-десятичный счетчик с кодом 2421

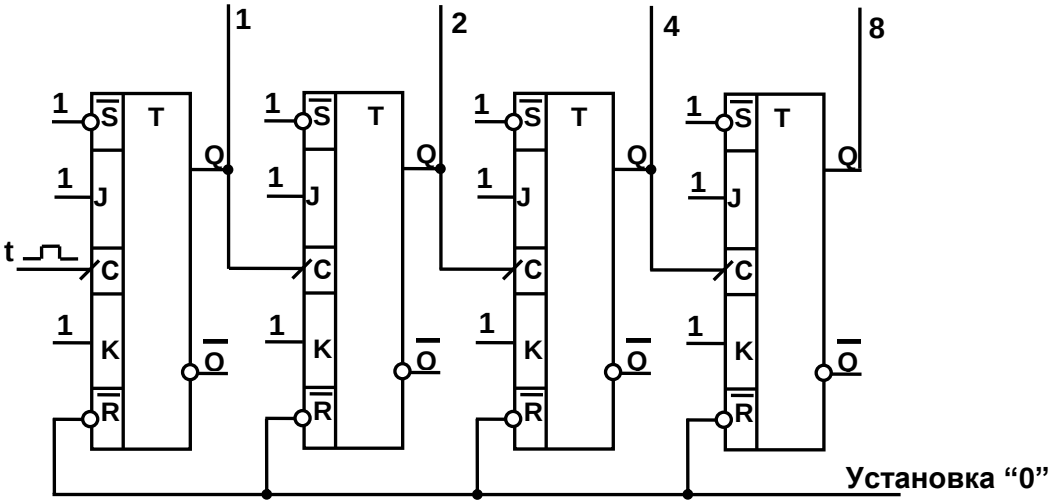


Рис. 39. Четырехразрядный двоичный счетчик

Лабораторная работа № 6
Исследование начальных языков описания цифровых автоматов (Часть 1)

Цель работы: получение навыков описания и исследования цифровых автоматов, заданных с помощью начальных языков.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Язык регулярных выражений алгебры событий

Язык регулярных выражений алгебры событий использует следующие понятия:

1. Входной алфавит: $Z = \{z_1, z_2, \dots z_n\}$.
2. Выходной алфавит: $W = \{w_1, w_2, \dots w_m\}$.
3. Множество событий: $S = \{s_1, s_2, \dots s_k\}$.

Регулярные выражения, описывающие работу автомата, записываются из букв входного алфавита с применением следующих операций:

$Z_i \vee Z_j$– объединение (дизъюнкция),

$Z_i \wedge Z_j$ или $Z_i Z_j$ – конъюнкция, $\{Z_i\}$ – итерация.

Любое регулярное выражение определяет некоторое событие $s_i \in S$. Если это событие наступило, то выдается соответствующая буква $w_{\check{z}} \in S$.

Пример 1

Записать регулярное выражение для автомата, работающего с входным алфавитом $Z = \{z_1, z_2, z_3, z_4, z_5\}$ и имеющего выходной алфавит W={ w1, w2}.

Поведение автомата следующее:

1. Если последовательность входных букв (входное слово) начинается с z_1, а заканчивается цепочкой z_2 z_3, то автомат должен выдать выходную букву w_1 (событие s_1).
2. Если слово начинается с буквы z_2, а заканчивается z_4 или z_5, то автомат должен выдать букву w_2 (событие s_2):

$$s_1/w_1 = z_1\{z_1 \vee z_2 \vee z_3 \vee z_4 \vee z_5\} z_2 z_3;$$

$$s_2/w_2 = z_2\{z_1 \vee z_2 \vee z_3 \vee z_4 \vee z_5\} z_4 \vee z_5.$$

Подобное описание появилось как результат развития теории формальных грамматик и распространения этой теории на автоматы. Оно представляет интерес для теоретических исследований, а при проектировании технических устройств используется редко.

Граф-схемы алгоритмов (ГСА) функционирования автоматов

ГСА – это ориентированный связанный граф, содержащий одну начальную $Y_н$, одну конечную Y_K, множества условных вершин $X=\{x_1 x_2 .. x_n\}$ и операторных вершин $Y=\{y_1 y_2 .. y_m\}$.

ГСА должна удовлетворять следующим условиям:

1. Каждый выход из любой вершины должен быть соединен только с одним входом в какую-либо вершину.
2. Любая вершина лежит хотя бы на одном пути из $Y_н$ в Y_K.

Один из выходов условной вершины может соединяться с ее входом, что недопустимо для операторных вершин.

В операторных вершинах могут помещаться выходные буквы автомата, в условных вершинах – входные буквы (иначе – логические условия). В условных вершинах проверяется наличие на входе автомата соответствующей буквы. В зависимости от того, та ли буква на входе или нет (выходы «1» или «0» условной вершины) будет выполняться соответствующий переход по ГСА. Если при этом встречается операторная вершина, то автомат выдает соответствующую выходную букву.

На рис. 40 приведен пример ГСА. В нашем примере символы x_i и y_i – это просто условное обозначение соответствующих условных и операторных вершин (но не входные и выходные буквы автомата!). Обычно начальную и конечную вершины ГСА обозначают несколько отлично от остальных операторных вершин (смотри рис. 40: вершины $Y_н$ и $Y_к$).

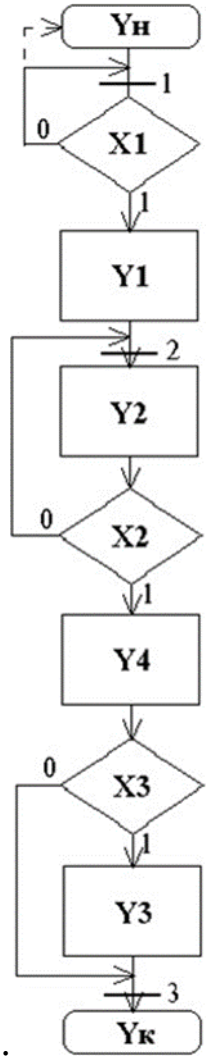


Рис. 40. ГСА функционирования автомата

Логические схемы алгоритмов (ЛСА). Формулы переходов.

ЛСА – это более компактная, но менее наглядная форма представления ГСА. В ЛСА используют:

– символы операторных вершин Y_j;

– символы условных вершин X_j.

– $\uparrow^I$, $\downarrow^I$ – верхние и нижние стрелки. Верхние стрелки $\uparrow^I$ ставятся после символов условных вершин X_j и их надстрочный индекс I указывает номер нижней стрелки $\downarrow^I$, к которой нужно перейти, если $X_j = 0$.

– $W\uparrow^I$- безусловный переход, где i – номера переходов к нижней стрелки $\downarrow^I$.

Для правильной записи ЛСА отметим на ГСА (рис. 1) номерами 1, 2 и 3 вершины, к которым подходят более одной стрелки. Перед символами этих вершин в ЛСА ставятся нижние стрелки с отмеченными на ГСА номерами.

Если возникают затруднения при записи ЛСА, можно вводить

дополнительные нижние и верхние стрелки и использовать безусловный переход $W \uparrow^I$.

ЛСА, соответствующая ГСА на рис. 40, имеет вид:

$$Y_{\text{н}} \downarrow^1 X_1 \uparrow^1 Y_1 \downarrow^2 Y_2 X_2 \uparrow^2 Y_4 X_3 \uparrow^3 Y_3 \downarrow^3 Y_K$$

Формулы переходов описывают все пути между операторными вершинами в ГСА. Они имеют вид:

$$Y_m \to \cup\ b_{ms} Y_S$$

и описывают все возможные переходы из вершины Y_m в вершины Y_S, где $b_{ms} = \cap X_{SJ,}$ и определяется как произведение логических условий, записанных в условных вершинах ГСА, лежащих на пути Y_m в Y_S. Если выход из условной вершины $X_{\text{к}}$ по стрелке «1», то в $\cap$ записывается $X_{\text{к}}$. если «0», то записывается $\neg X_{\text{к}}$.

Формулы переходов, соответствующие ГСА имеют вид:

$$Y_{\text{н}} \to X_1 Y_1 \vee \neg X_1 X_{\text{н}},$$

$$Y_1 \to X_{2,}$$

$$Y_2 \to \neg X_2 Y_2 \vee X_2 Y_4,$$

$$Y_3 \to X_{k,}$$

$$Y_4 \to X_3 Y_3 \vee \neg X_3 Y_k.$$

Матричные схемы алгоритмов (*МСА*)

МСА – это табличное представление формул переходов, где число строк соответствует количеству формул переходов, а число столбцов равно числу операторных вершин, в которые возможны переходы. В таблице 13 приведена МСА, построенная по формулам переходов, соответствующих ГСА на рис. 40.

Таблица 13

Y_m	$Y_{\text{н}}$	Y_1	Y_2	Y_3	Y_4	Y_k
$Y_{\text{н}}$	$\neg X_1$	X_1				
Y_1			1			
Y_2			$\neg X_2$		X_2	
Y_3						1
Y_4				X_3		$\neg X_2$

Преобразование основано на разложении булевых функций переменным X_1.

Пример 2

Приведен фрагмент ГСА (рис. 41, а). Запишем формулу перехода для вершины Y_4:

$$Y_4 \to \cup\ b_{4s}Y_S$$

$$Y_4 \to X_2\neg X_1Y_5 \vee \neg X_2X_1Y_1 \vee \neg X_2X_1Y_1 \vee \neg X_2\neg X_1Y_7,$$

Данному фрагменту ГСА соответствует следующее разложение формулы перехода по переменным X_1 и X_2:

$$Y_4 \to X_2(X_1Y_1 \vee \neg X_1Y_5) \vee \neg X_2(X_1Y_1 \vee \neg X_1Y_7)$$

Разложим исходную формулу по переменной X_1:

$$Y_4 \to X_1(X_2Y_1 \vee \neg X_2Y_1) \vee \neg X_1(X_2Y_5 \vee \neg X_2Y_7) \to X_1(Y_1)\neg X_1(X_2Y_5 \vee \neg X_2Y_7)$$

Этому разложению соответствует фрагмент ГСА, приведенный на рис. 41, б.

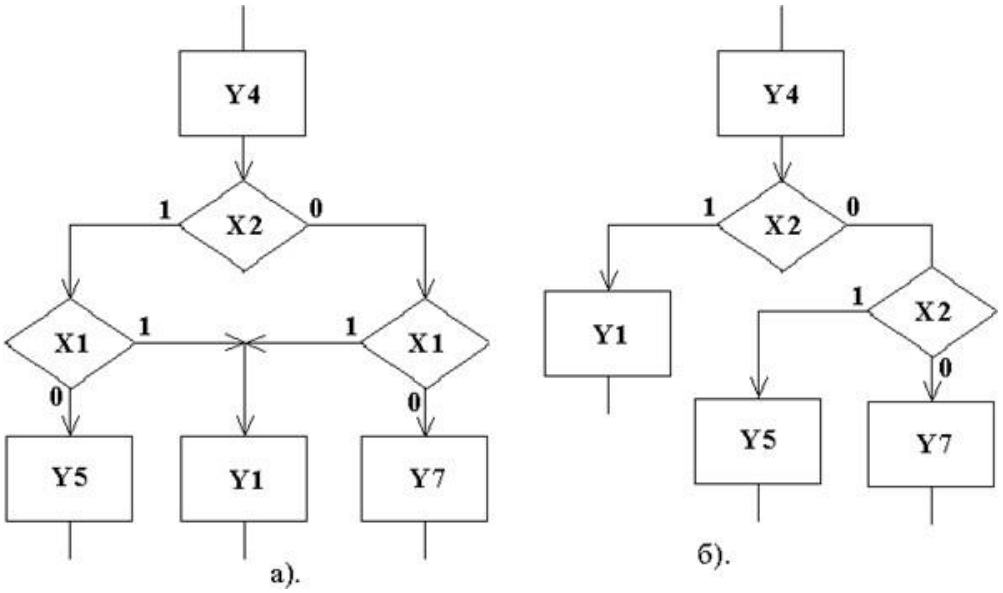


Рис. 41. ГСА для примера 2

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования функционирования автомата и исследовать разработанную модель.

Порядок выполнения работы

1. Разработать схему для ввода описания ГСА (модели ГСА) в среде программирования Java. Для этого необходимо:

1.1.Представить ГСА в виде ЛСА;

1.2.Разработать структуру данных, соответствующую построенной ЛСА и позволяющую хранить и использовать эти данные для моделирования;

1.3.Разработать алгоритм ввода описания ГСА с клавиатуры и из текстового файла;

1.4.Разработать программу ввода модели ГСА в среде программирования Java.

2. Разработать алгоритм моделирования ГСА.

2.1.Предусмотреть 3 режима работы программы моделирования:

– последовательный ввод значений логических условий X и вывод результата каждого шага моделирования;

– ввод значений всех логических условий X и вывод результата моделирования;

– полный перебор всех значений логических условий X и вывод результата моделирования.

3. Разработать программу моделирования ГСА в среде программирования Java.

3.1.Разработать дружественный пользовательский интерфейс.

3.2.Предусмотреть контроль от зацикливания программы при некорректных исходных данных.

4. Исследовать поведение автомата на построенной модели.

4.1.Для всех комбинаций логических условий X проверить правильность переходов по ГСА.

4.2.Исследовать модель ГСА на устойчивость (возможность зацикливания программы моделирования).

4.3.Исследовать модель ГСА на адекватность заданной ГСА.

4.4.Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

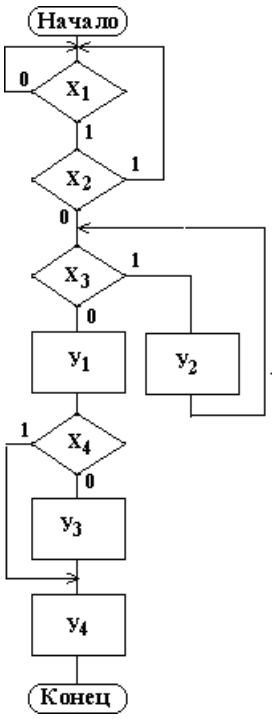
Пример индивидуального задания к лабораторной работе № 6

Задание на разработку программной модели

1. Разработать программу для ввода описания ГСА (модели ГСА) в среде программирования Java (см. рисунок).
2. Разработать алгоритм моделирования ГСА
3. Разработать программу моделирования ГСА в среде программирования Java.

Задание на проведение экспериментов

1. Для всех комбинаций логических условий X проверить правильность переходов по ГСА.
2. Исследовать модель ГСА на устойчивость (возможность зацикливания программы моделирования)
3. Исследовать модель ГСА на адекватность исходной ГСА.
4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата, заданную преподавателем;
- алгоритм моделирования ГСА и структуру данных;
- текст программы моделирования автомата;
- результаты имитационного моделирования автомата.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата?
2. Какие начальные языки являются наиболее наглядными и почему?
3. Какое различие между ГСА и ЛСА?

Лабораторная работа № 7
Исследование начальных языков описания цифровых автоматов (Часть 2)

Цель работы: получение навыков описания и исследования цифровых автоматов, заданных с помощью начальных языков.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

В объединяемых ГСА должны встречаться одинаковые операторные и условные вершины. Вершины, одинаково обозначенные, должны иметь и одинаковый смысл: в одинаково обозначенных вершинах должны быть одинаковые буквы алфавитов W и Z.

Далее рассмотрим переход от граф-схемы автомата к матричной схеме автомата (МСА). Он заключается в следующих действиях:

– для объединяемых ГСА строятся МСА;

– затем строится и заполняется объединенная МСА;

– записываются формулы переходов, и после их разложения по переменным X_{sj} строятся фрагменты объединенной ГСА;

– фрагменты объединенной ГСА объединяются путем совмещения операторных вершин.

При построении объединенной МСА вводятся дополнительные логические условия (вершины), обеспечивающие заданный алгоритм работы автомата.

Например, даны две ГСА: Г1 (рис. 42, а) и Г2 (рис. 42, б). В этих ГСА встречаются одинаковые операторные и условные вершины. Составим для Г1 матрицу МСА1, а для Г2 матрицу МСА2.

Так как объединяются всего две ГСА, то введем одну дополнительную переменную (логическое условие) P. Условие P будет определять, по какому пути будут идти переходы в объединенной ГСА в зависимости от того, какой алгоритм реализуется в настоящий момент.

Пусть для Г1 условие P = 1, а для Г2 – P = 0.

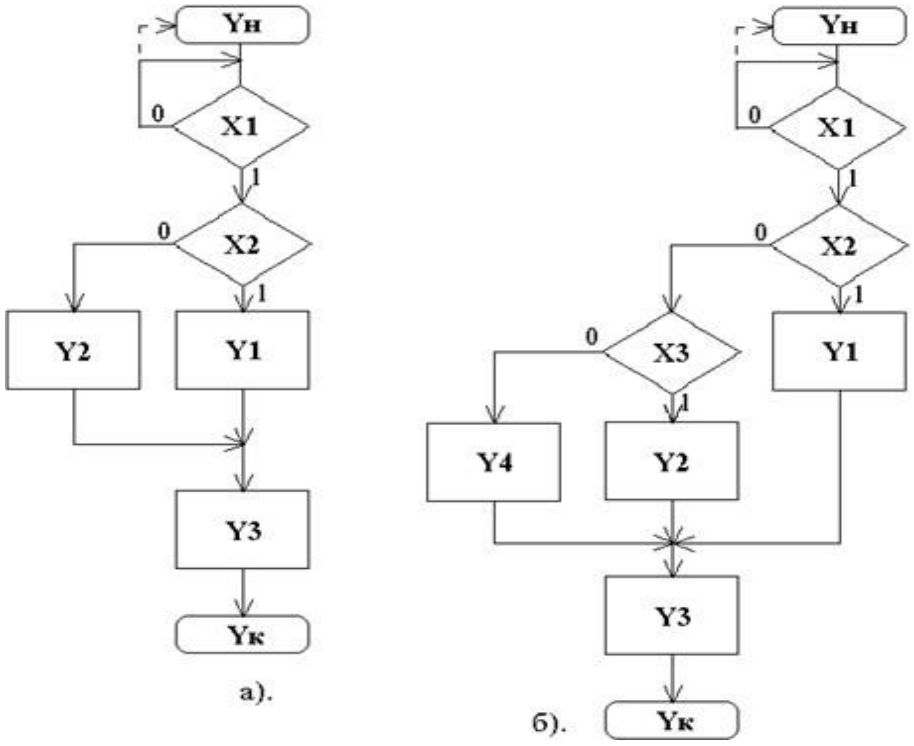


Рис. 43. ГСА Г1 и Г2

В таблице 14 приведена матрица МСА 1, а в таблице 15 – матрица МСА 2.

Таблица 14

$Y_m \backslash Y_s$	$Y_н$	Y_1	Y_2	Y_3	$Y_к$
$Y_н$	$\neg X_1$	X_1X_2	$X_1\neg X_2$	-	-
Y_1				1	
Y_2				1	
Y_3					1

Таблица 15

$Y_m \backslash Y_s$	$Y_н$	Y_1	Y_2	Y_3	Y_4	$Y_к$
$Y_н$	$\neg X_1$	X_1X_2	$X_1\neg X_2X_3$	-	$X_1\neg X_2\neg X_3$	
Y_1				1		
Y_2				1		
Y_3						1
Y_4				1		

Построенная объединенная МСА приведена в таблице 16.

Таблица 16

$Y_m \backslash Y_s$	$Y_н$	Y_1	Y_2	Y_3	Y_4	$Y_к$
$Y_н$	$P\neg X_1$ $\neg P\neg X_1$	PX_1X_2 $\neg P\ X_1X_2$	$PX_1\neg X_2$ $\neg P\ X_1\neg X_2X_3$	-	$\neg PX_1\ \neg X_2\ \neg X_3$	-
Y_1				P $\neg P$		
Y_2				P $\neg P$		
Y_3						P $\neg P$
Y_4				P		

Формулы переходов по объединенной МСА:

$$Y_1 \to (P \vee \neg P) Y_3 \to Y_3,$$

$$Y_2 \to (P \vee \neg P) Y_3 \to Y_3,$$

$$Y_3 \to (P \vee \neg P) Y_k \to Y_k,$$

При этом $Y_4 \to \neg P Y_3 \to Y_3$, так как в строке Y_4 нет P, а значит в Г1 нет вершины Y_4, и условие $\neg P$ из формулы можно исключить.
Далее имеем:

$$Y_н \to (P \vee \neg P)\neg X_1 Y_н \vee (P \vee \neg P) X_1 X_2 Y_1 \vee P X_1 \neg X_2 Y_2 \vee \neg P X_1 \neg X_2 X_3 Y_2 \vee \neg P X_1 \neg X_2$$
$$\neg X_3 Y_4 \to X_1 Y_н \vee X_1 X_2 Y_1 \vee P X_1 \neg X_2 Y_2 \vee \neg P X_1 \neg X_2 X_3 Y_2 \vee \neg P X_1 \neg X_2 \neg X_3 Y_4$$

Выражения для переменных Y_1 – Y_4 – простые, их не надо разлагать. Разложение $Y_н$ удобно начинать с переменной, чаще всего встречающейся в формуле. Это переменная X_1:

$$Y_н \to \neg X_1 (Y_н) \vee X_1 (X_2 Y_1 \vee P \neg X_2 Y_2 \vee \neg P \neg X_2 X_3 Y_2 \vee \neg P \neg X_2 \neg X_3 Y_4)$$

Далее разложим формулу по X_2:

$$Y_н \to \neg X_1 (Y_н) \vee X_1 (X_2 (Y_1) \vee \neg X_2 (P Y_2 \vee \neg P X_3 Y_2 \vee \neg P \neg X_3 Y_4))$$

По полученным объединенным формулам переходов построим фрагменты объединенной ГСА (рис. 44, а):

Объединим фрагменты ГСА, учитывая, что не может быть двух операторных вершин одного вида (рис. 44, б). Вид объединенной ГСА зависит от вида фрагментов (подграфов), которые, в свою очередь, зависят от порядка разложения формул перехода.

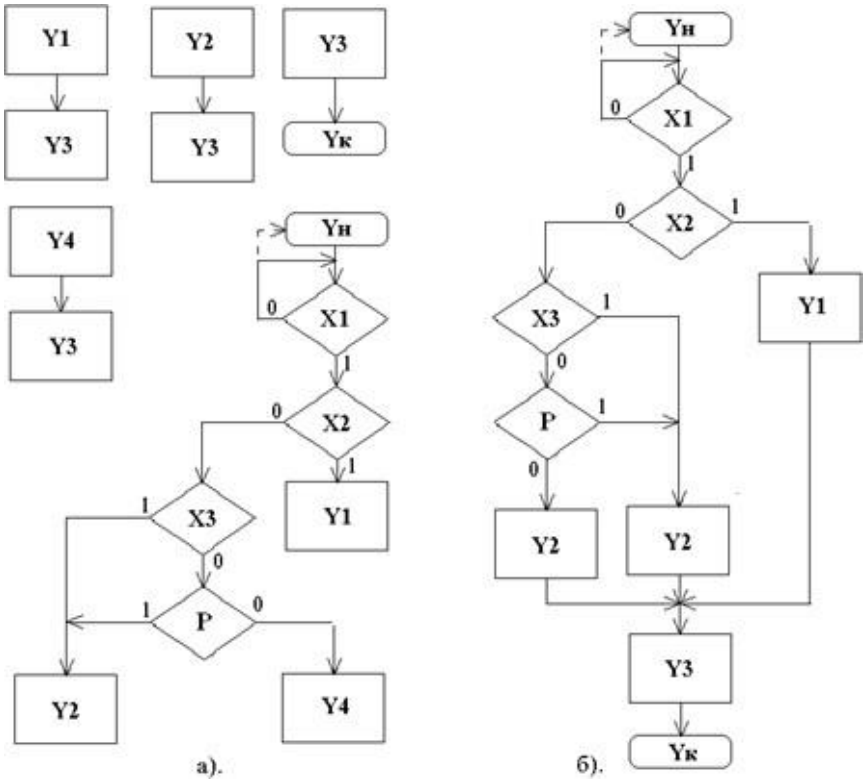


Рис. 44. Фрагменты (а) и объединенная ГСА (б)

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования автомата, функция которого описана матричной схемой алгоритма (МСА) и исследовать разработанную модель.

Порядок выполнения работы

1. Разработать программу для ввода описания ГСА (модели ГСА) в среде программирования Java. Для этого необходимо:

1.1. Представить ГСА в виде ЛСА.

1.2. Разработать структуру данных, соответствующую построенной ЛСА и позволяющую хранить и использовать эти данные для моделирования.

1.3. Разработать алгоритм ввода описания ГСА с клавиатуры и из

текстового файла.

1.4. Разработать алгоритм формирования МСА по описанию автомата в виде ЛСА.

1.5. Разработать программу ввода модели ГСА и формирования МСА в среде программирования Java.

2. Разработать алгоритм моделирования МСА.

2.1. Предусмотреть три режима работы программы моделирования:

– последовательный ввод значений логических условий X и вывод результата каждого шага моделирования;

– ввод значений всех логических условий X и вывод результата моделирования;

– полный перебор всех значений логических условий X и вывод результата моделирования.

3. Разработать программу моделирования МСА в среде программирования Java.

3.1.Разработать дружественный пользовательский интерфейс.

3.2.Предусмотреть контроль от зацикливания программы при некорректных исходных данных.

4. Исследовать поведение автомата на построенной модели.

4.1.Для всех комбинаций логических условий X проверить правильность переходов по МСА.

4.2.Исследовать модель МСА на устойчивость (возможность зацикливания программы моделирования).

4.3.Исследовать МСА на адекватность заданной ГСА.

4.4.Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

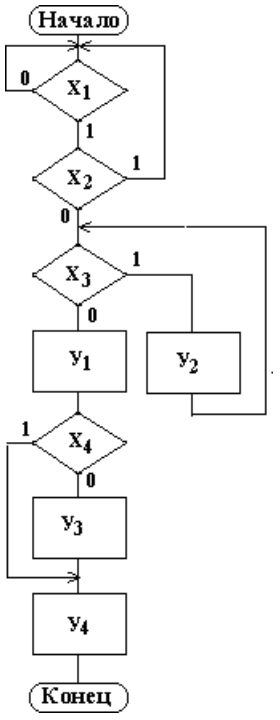
Пример индивидуального задания к лабораторной работе № 7

Задание на разработку программной модели

1. Разработать программу для ввода описания ГСА и формирования МСА в среде программирования Java (см. рисунок).
2. Разработать алгоритм моделирования МСА.
3. Разработать программу моделирования МСА в среде программирования Java.

Задание на проведение экспериментов

1. Для всех комбинаций логических условий X проверить правильность переходов по МСА.
2. Исследовать модель МСА на устойчивость (возможность зацикливания программы моделирования).
3. Исследовать модель МСА на адекватность заданной ГСА.
4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата, заданную преподавателем;
- структуру данных и алгоритм моделирования МСА;
- текст моделирующей программы автомата;
- результаты имитационного моделирования автомата на спроектированной программе.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата?
2. Какие правила необходимо соблюдать при объединении МСА?
3. Как влияет порядок преобразования формул переходов объединенной МСА на вид ГСА?

Лабораторная работа № 8
Исследование автоматных языков описания цифровыхавтоматов

Цель работы: получение навыков описания и исследования цифровых автоматов, заданных с помощью начальных языков.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Среди автоматных языков наиболее распространены *таблицы, графы и матрицы переходов и выходов*.

Рассмотрим примеры таблиц для автоматов Мили и Мура.

Пример автомата S1 Мили:

Таблица переходов: $\delta: A \times Z \to A$;

δ	a_1	a_2	a_3
z_1	a_2	a_3	a_2
z_2	a_3	a_2	a_1

Таблица выходов: $\lambda: A \times Z \to W$;

λ	w_1	w_2	w_3
z_1	w_1	w_3	w_3
z_2	w_2	w_1	w_1

На пересечении столбца a_m и строки z_f ставят:

$a_s = \delta\,(a_m, z_f)$

$w_g = \lambda\,(a_m, z_f)$.

Таблицы переходов и выходов могут быть объединены в единую *совмещенную* таблицу переходов и выходов:

δ / λ	a_1	a_2	a_3
z_1	a_2/w_1	a_3/w_3	a_2/w_3
z_2	a_3/w_2	a_2/w_1	a_1/w_1

Пример автомата S_2 Мура:

Автомат Мура задается одной ***отмеченной таблицей переходов***:

$$\delta: A \times Z \to A;\ \lambda: A \to W$$

	w_1	w_1	w_1	w_1	w_1
	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_5	a_5	a_3	a_3
z_2	a_4	a_2	a_2	a_1	a_1

Для *не полностью определенных автоматов* на месте неопределенных состояний и выходных сигналов ставятся прочерки (не в шапках таблиц, а внутри таблиц).

Граф автомата – это ориентированный граф, вершинам которого соответствуют состояния автомата, а дугам – переходы.

Дуге (a_m, a_s), направленной от вершины a_m к вершине a_s приписывается (в модели Мили) входной сигнал z_f и выходной сигнал $w_g = \lambda (a_m, z_f)$, либо ставится прочерк.

В модели Мура выходной сигнал приписывается вершине, т.е. состоянию $w_g = \lambda (a)$.

Автоматы S_1и S_2 заданные выше в виде таблиц, задаются следующими графами (рис. 45 и 46):

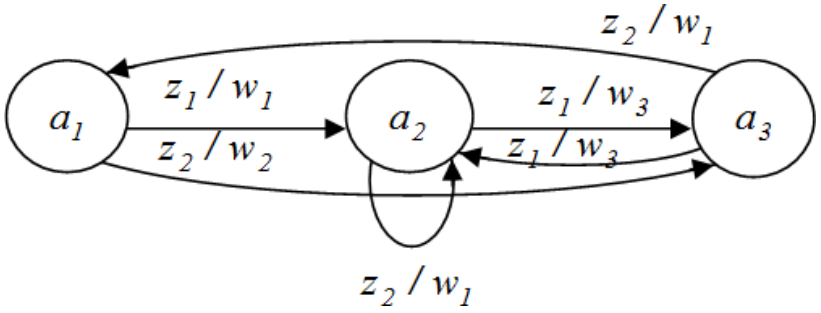


Рис. 45. Автомат S_1

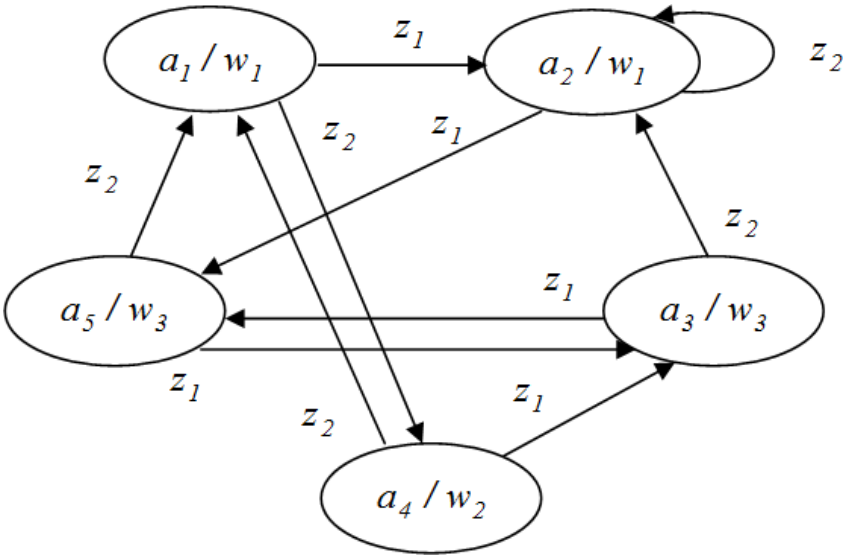


Рис. 46. Автомат S_2

Расширим функции δ и λ, определив их на множестве пар: (состояние, входное слово).

Обозначим $\zeta = z_{i1}z_{i2} \dots z_{ik}$ –входное слово длиной k символов;

e – слово длиной нуль (пустое слово);

E – множество всех конечных входных слов ненулевой длины: $e \notin E$.

Функция заключительного состояния: $\tilde{\delta}(a_m, \zeta)$ на множестве $A \times (E \cup \{e\})$ определяется следующим образом:

1. $\tilde{\delta}(a_m, \text{e}) = a_m$ для всех $a_m \in A$;
2. $\tilde{\delta}(a_m, \zeta) = \tilde{\delta}(a_m, z_{i1} \dots z_{ik})$ равна:

a) $\delta(\delta(\dots \delta(\delta(\delta(a_m, z_{i1}), z_{i2}), z_{i3}), \dots, z_{ik-1}), z_{ik}) = \delta(a_{ik}, z_{ik})$, если $\delta(a_{ij}, z_{ij})$ определена для всех $j = 1,2,\dots, k$; $a_{i1} = a_m$;

b) в противном случае $\tilde{\delta}(a_m, \zeta)$ не определено.

Таким образом, $\tilde{\delta}(a_m, \zeta)$ является заключительным состоянием, в которое переходит автомат из состояния a_m под действием входного слова ζ.

Функция заключительного выхода $\tilde{\lambda}(a_m, \zeta)$.

Автомат Мура:

$$\tilde{\lambda}(a_m, \zeta) = \begin{cases} \lambda\left(\tilde{\delta}(a_m, \zeta)\right), \text{если определена } \tilde{\delta}(a_m, \zeta) \\ \text{не определена, если } \tilde{\delta}(a_m, \zeta) \text{ не определена} \end{cases}$$

Автомат Мили:

$$\tilde{\lambda}(a_m, \zeta) = \begin{cases} \lambda\left(\tilde{\delta}(a_m, \zeta), z_{ik}\right), \zeta = \zeta' z_{ik}\big), \text{если определена } \tilde{\delta}(a_m, \zeta') \\ \text{не определена, если } \tilde{\delta}(a_m, \zeta') \text{ не определена.} \end{cases}$$

Функция заключительного выхода – это выходной сигнал, который появляется в результате действия последнего символа цепочки ζ:

– в модели Мура – это выходной сигнал, отмечающий заключительное состояние;

– в модели Мили – это выходной сигнал, появляющийся на переходе в заключительное состояние.

Пример 3

Для автомата S_1 (Мили) введенного ранее найдем $\tilde{\lambda}\left(a_1, z_1, z_2, z_{1,} z_1\right)$.

Решение: $\tilde{\delta}\left(a_1, z_1, z_2, z_{1,} z_1\right) = \tilde{\delta}\left(\delta(a_1, z_1), z_2, z_{1,} z_1\right) =$

$$= \tilde{\delta}\left(\delta(a_2, z_2), z_{1,} z_1\right) = \tilde{\delta}\left(\delta(a_2, z_1), z_{1,}\right) = \tilde{\delta}\left(a_3, z_{1,}\right) = \delta\left(a_3, z_{1,}\right) = a_2$$

Последний переход равен $\delta\left(a_3, z_{1,}\right)$, поэтому

$$\tilde{\lambda}\left(a_1, z_1, z_2, z_{1,} z_1\right) = \lambda\left(a_3, z_{1,}\right) = w_3$$

Реакция автомата $w(a_m, \zeta)$ в состоянии a_m на входное слово ζ– это цепочка выходных символов (выходное слово) вырабатываемая автоматом.

Для модели Мили:

$$w(a_m, \zeta) = \begin{cases} \lambda\left(a_m, z_{i1,}\right)\lambda\left(\delta\left(a_m, z_{i1,}\right), z_{i2}\right) \dots \lambda\left(\tilde{\delta}(a_m, \zeta'), z_{ik}\right) = w_{i1} w_{i2} \dots w_{ik} \\ \text{не определена, если } \tilde{\delta}(a_m, \zeta') \text{ не определена} \end{cases}$$

Для модели Мура:

$$w(a_m, \zeta) = \begin{cases} \lambda(\delta\left(a_m, z_{i1,}\right) \dots \lambda\left(\delta(\delta\left(a_m, z_{i1,}\right), z_{i2}\right) \dots \lambda\left(\tilde{\delta}(a_m, \zeta')\right) \\ \text{не определена, если } \tilde{\delta}(a_m, \zeta') \text{ не определена} \end{cases}$$

Обратим внимание, что выходной сигнал вырабатываемый автоматом Мура в состоянии a_m не входит в цепочку $w(a_m, z_{i1})$, т. к. он не является следствием действия символа $z_{i1.}$

Под реакцией автомата Мура в состоянии a_m на выходное слово $\zeta = z_{i1} z_{i2} \dots z_{ik}$ понимается выходное слово длины k, но сдвинутое на 1 такт времени по сравнению с выходным словом автомата Мили: $w(a_m, \zeta) = w_{i1} w_{i2} \dots w_{ik+1.}$

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования конечного автомата, функция которого описана таблицами переходов и выходов:

$$\delta = A \times Z \to A, \lambda: A \to W$$

Порядок выполнения работы

1. Разработать программу для ввода описания функций переходов и выходов конечного автомата $\delta = A \times Z \to A, \lambda: A \to W$ в среде программирования Java. Для этого необходимо:

1.1.Разработать структуру данных, соответствующую функциям переходов и выходов конечного автомата и позволяющую хранить и использовать эти функции для моделирования.

1.2.Разработать алгоритм ввода функций переходов и выходов конечного автомата с клавиатуры и из текстового файла.

1.3.Разработать программу ввода функций переходов и выходов конечного автомата в среде программирования Java.

2. Разработать алгоритм моделирования конечного автомата.

2.1. Предусмотреть 2 режима работы программы моделирования:

– последовательный ввод букв из входного алфавита Z и вывод результата каждого шага моделирования;

– ввод слова из букв входного алфавита Z и вывод результата моделирования.

3. Разработать программу моделирования конечного автомата в среде программирования Java.

3.1.Разработать дружественный пользовательский интерфейс.

3.2.Предусмотреть контроль от зацикливания программы при некорректных исходных данных.

4. Исследовать поведение автомата на построенной модели.

4.1.Для различных входных слов проверить правильность работы автомата в соответствии с функциями переходов и выходов.

4.2.Исследовать модель автомата на устойчивость (возможность зацикливания программы моделирования).

4.3.Исследовать модель на адекватность заданному автомату.

4.4.Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 8

Задание на разработку программной модели

1. Разработать программу для ввода описания конечного автомата, приведенного на рисунке, всреде программирования Java. Построить граф переходов.
2. Разработать алгоритм моделирования конечного автомата.
3. Разработать программу моделирования конечного автомата в среде программирования Java.

Задание на проведение экспериментов

1. Для различных входных слов проверить адекватность модели заданному описанию.
2. Исследовать модель на устойчивость (возможность зацикливания программы моделирования).
3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Описание автомата:

$$Z = \{a, b, c\};$$
$$W = \{a, b, c\};$$
$$A = \{d_1, d_2, d_3\};$$
$$a_1 = d_1;$$

δ_1 и λ_1 – функции переходов и выходов автомата, заданные таблицей переходов и выходов:

$$\delta: A \times Z \to A$$
$$\lambda: A \times Z \to W$$

	d_1	d_2	d_3
a	d_2/c	d_1/b	d_2/a
b	d_2/a	d_3/b	d_1/b
c	d_3/a	d_2/a	d_3/c

Подготовка отчета

Отчет должен содержать:

– описание закона функционирования конечного автомата, заданного преподавателем, и граф переходов автомата;

– структуру данных и алгоритм моделирования конечного автомата;

– текст моделирующей программы для конечного автомата;

– результаты имитационного моделирования конечного автомата с помощью разработанной программы.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования конечного автомата?
2. Какие правила необходимо соблюдать при моделировании конечного автомата?
3. Какое различие между автоматами Мили и Мура?

Лабораторная работа № 9
Исследование абстрактного автомата (Часть 1)

Цель работы: получение навыков исследования абстрактных автоматов.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Абстрактный автомат (АА) задается шестеркой (шестикомпонентным вектором):

$$\boldsymbol{S = (A, Z, W, \delta, \lambda, a_1)},$$

где $\boldsymbol{A = \{a_1, a_2, \dots, a_м\}}$ – множество (алфавит) состояний;

$\boldsymbol{Z = \{z_1, z_2, \dots, z_F\}}$ – множество (алфавит) входных сигналов;

$\boldsymbol{W = \{w_1, w_2, \dots, w_G\}}$ – множество (алфавит) выходных сигналов;

$\boldsymbol{\delta = A \times Z \to A}$ – функция переходов автомата.

Функция $\boldsymbol{\delta}$ ставит паре $(\boldsymbol{a_m, z_f})$ в соответствие некоторое состояние $\boldsymbol{a_s}$:

$$\boldsymbol{a_s = \delta(a_m, z_f)}, \quad \boldsymbol{a_s \in A},$$

где λ: $A \times Z \to W$ функция выходов автомата (рис. 47)

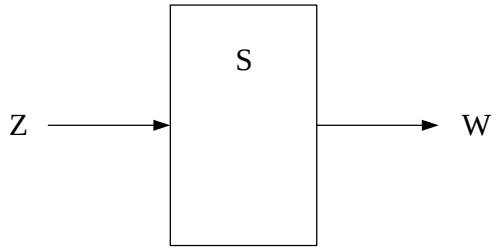


Рис. 47. Функция выходов автомата

Функция $\boldsymbol{\lambda}$ ставит в соответствие паре $(\boldsymbol{a_m, z_f})$ некоторый выходной сигнал $\boldsymbol{w_g = \lambda(a_m, z_f)}$, при этом $\boldsymbol{a_1 \in A}$ – начальное состояние АА.

Автомат называется *конечным*, если конечны множества $\boldsymbol{A, Z}$ и $\boldsymbol{W}$.

Автомат называется *детерминированным*, если для всех пар $(\boldsymbol{a_m, z_f})$ функция $\boldsymbol{\delta(a_m, z_f)}$ имеет единственное значение. На рис. 47 АА имеет один вход и один выход. Автомат работает в дискретном времени, принимающем значения $\boldsymbol{t = 0, 1, 2, 3}$, ... Всегда $\boldsymbol{a(0) = a_1}$.

В момент времени $\boldsymbol{t}$ автомат находится в состоянии $\boldsymbol{a(t)}$. На его вход поступает входной сигнал $\boldsymbol{z(t) \in Z}$. В качестве реакции на входной символ автомат выдает некоторую букву $\boldsymbol{w(t) \in W}$ выходного алфавита $\boldsymbol{w(t) = \lambda(a(t), z(t))}$. Затем в соответствии с функцией перехода $\boldsymbol{\delta}$ переходит в следующее состояние: $\boldsymbol{a(t+1) = \delta(a(t), z(t))}$. На этом заканчивается один цикл работы автомата.

Последовательность символов входного алфавита $\boldsymbol{z(0)z(1)z(2)}$ ... задает *входное слово* (слово входного алфавита).

Последовательность символов выходного алфавита $\boldsymbol{w(0)w(1)w(2)}$ ... задает *выходное слово* (слово выходного алфавита).

Работа АА состоит в преобразовании слов входного алфавита в слова выходного алфавита, при этом его рассматривают его как «черный ящик», не учитывая внутреннюю структуру автомата. При этом основное внимание уделяется поведению автомата относительно внешней среды. В качестве элементов, связывающих автомат с внешней средой, выступают входные и выходные слова.

Введение состояний в описание устройства позволяет запоминать и учитывать предысторию поведения автомата. Это дает возможность строить устройства, которые на одни и те же входные сигналы в разные моменты времени реагируют по-разному.

Поведение комбинационных схем описывается системой булевских функций, в которых аргументам соответствуют только входные сигналы. КС всегда на данный набор входных сигналов вырабатывает определенный набор выходных сигналов, не зависящий от времени. КС можно рассматривать как частный случай автомата S с одним состоянием:

$$\boldsymbol{S = (Z, W, \lambda)}, \qquad \boldsymbol{\lambda := Z \to W},$$

Основными видами соединений двух автоматов являются:

- параллельное;
- последовательное.

Параллельное соединение. На рис. 48 показано параллельное соединение автоматов S_1 и S_2,

где $\left.\begin{matrix} \boldsymbol{S_1} = (\boldsymbol{A_1}, \boldsymbol{Z}, \boldsymbol{W_1}, \boldsymbol{\delta_1}, \boldsymbol{\lambda_1}) \\ \boldsymbol{S_2} = (\boldsymbol{A_2}, \boldsymbol{Z}, \boldsymbol{W_2}, \boldsymbol{\delta_2}, \boldsymbol{\lambda_2}) \end{matrix}\right\}$ Начальное состояние не выделено.

Переменная $\boldsymbol{\varphi}$ – функциональный преобразователь (автомат без памяти): $\boldsymbol{\varphi} = \boldsymbol{W_1} \times \boldsymbol{W_2} \to \boldsymbol{W}$.

Результирующий автомат определяется как $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{\delta}, \boldsymbol{\lambda})$:

1. $\boldsymbol{A} = \boldsymbol{A_1} \times \boldsymbol{A_2}$ – множество всевозможных пар вида:

   $\boldsymbol{a_m} = (\boldsymbol{a_{m1}}, \boldsymbol{a_{m2}}) \big| (\boldsymbol{a_{m1}} \in \boldsymbol{A_1}) \& (\boldsymbol{a_{m2}} \in \boldsymbol{A_2}) \big)$.

2. $\boldsymbol{Z}$ – входной алфавит автоматов S_1 и S_2.
3. $\boldsymbol{W} = \boldsymbol{\varphi}(\boldsymbol{W_1} \times \boldsymbol{W_2})$ – заданное отображение.
4. Функция переходов $\boldsymbol{\delta}$: $\boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{A}$ определяется следующим образом:

   $\boldsymbol{\delta}(\boldsymbol{a_m}, \boldsymbol{z_f}) = \big(\boldsymbol{\delta_1}(\boldsymbol{a_{m1}}, \boldsymbol{z_f}), \boldsymbol{\delta_2}(\boldsymbol{a_{m2}}, \boldsymbol{z_f})\big), \boldsymbol{z_f} \in \boldsymbol{Z}$.

   или $\boldsymbol{\delta}(\boldsymbol{A} \times \boldsymbol{Z}) = \big(\boldsymbol{\delta_1}(\boldsymbol{A_1} \times \boldsymbol{Z}), \boldsymbol{\delta_2}(\boldsymbol{A_2} \times \boldsymbol{Z})\big)$.

5. Функция выходов $\boldsymbol{\lambda}$: $\boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{W}$ определяется следующим образом:

   $$\boldsymbol{\lambda}(\boldsymbol{a_m}, \boldsymbol{z_f}) = \boldsymbol{\varphi}\left(\boldsymbol{\lambda_1}(\boldsymbol{a_{m1}}, \boldsymbol{z_f}), \boldsymbol{\lambda_2}(\boldsymbol{a_{m2}}, \boldsymbol{z_f})\right),$$

   или $\boldsymbol{\lambda}(\boldsymbol{A} \times \boldsymbol{Z}) = \boldsymbol{\varphi}\big(\boldsymbol{\lambda_1}(\boldsymbol{A_1} \times \boldsymbol{Z}), \boldsymbol{\lambda_2}(\boldsymbol{A_2} \times \boldsymbol{Z})\big)$.

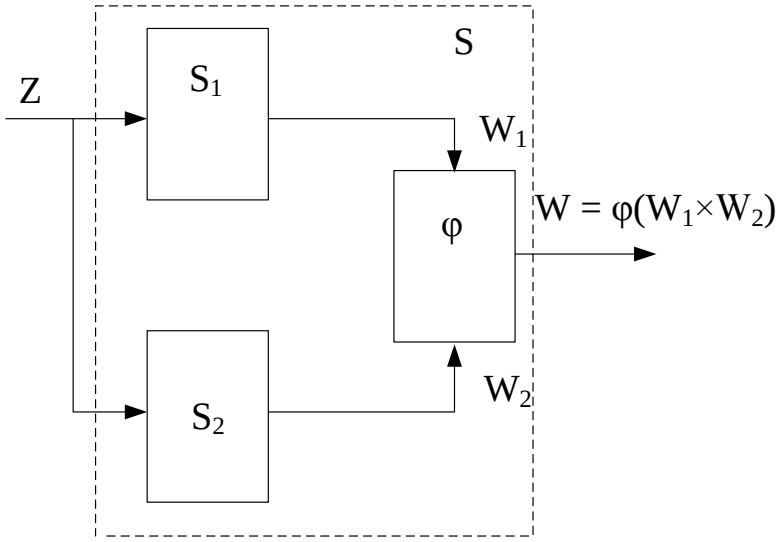


Рис. 48. Параллельное соединение автоматов

Пример 4

Заданы два автомата:

$\boldsymbol{S_1 = (B, Z, U, \delta_1, \lambda_1)}$ и $\boldsymbol{S_2 = (C, Z, V, \delta_2, \lambda_2)}$

$\boldsymbol{\delta_1: (B \times Z) \to B}$

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$
$\boldsymbol{z_1}$	$\boldsymbol{b_1}$	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$
$\boldsymbol{z_2}$	$\boldsymbol{b_3}$	$\boldsymbol{b_3}$	$\boldsymbol{b_2}$

$\boldsymbol{\lambda_1: (B \times Z) \to U}$

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$
$\boldsymbol{z_1}$	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{u_2}$
$\boldsymbol{z_2}$	$\boldsymbol{u_1}$	$\boldsymbol{u_1}$	$\boldsymbol{u_1}$

$\boldsymbol{\delta_2: (C \times Z) \to C}$

	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{c_2}$	$\boldsymbol{c_1}$

$\boldsymbol{\lambda_2: (C \times Z) \to V}$

	$\boldsymbol{c_1}$	$\boldsymbol{c_1}$
$\boldsymbol{z_1}$	$\boldsymbol{v_1}$	$\boldsymbol{v_2}$
$\boldsymbol{z_1}$	$\boldsymbol{v_2}$	$\boldsymbol{v_1}$

$\boldsymbol{\lambda_2: (C \times Z) \to V}$

	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$
$\boldsymbol{v_1}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$
$\boldsymbol{v_2}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$

Построим объединенный автомат $\boldsymbol{S_1 = (A, Z, W, \delta, \lambda)}$, который определен как: $\boldsymbol{\delta: A \times Z \to A}$,

$\boldsymbol{A \times Z = \{(b_1, c_2), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2), a_1, a_2, a_3, a_4, a_5, a_6,\}}$

тогда $\boldsymbol{\delta: A \times Z \to A}$

	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$
$\boldsymbol{z_2}$	$\boldsymbol{b_3c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_2c_1}$

$\boldsymbol{\lambda: A \times Z \to W}$

	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_1c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$
$\boldsymbol{z_2}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$

Пример 5. Построение таблиц:

$$\boldsymbol{\delta(a_3, z_1) = \delta((b_2, c_1), z_1) = \big(\delta_1(b_2, z_1), \delta_2(c_1, z_1)\big) = (b_1, c_1) = a_1}$$

$$\boldsymbol{\lambda(a_3, z_1) = \lambda((b_2, c_1), z_1) = \varphi\big(\lambda_1(b_2, z_1), \lambda_2(c_1, z_1)\big) = \varphi(u_2, v_1) = w_2.}$$

Последовательное соединение

На рис. 49 показано последовательное соединение автоматов S_1 и S_2.

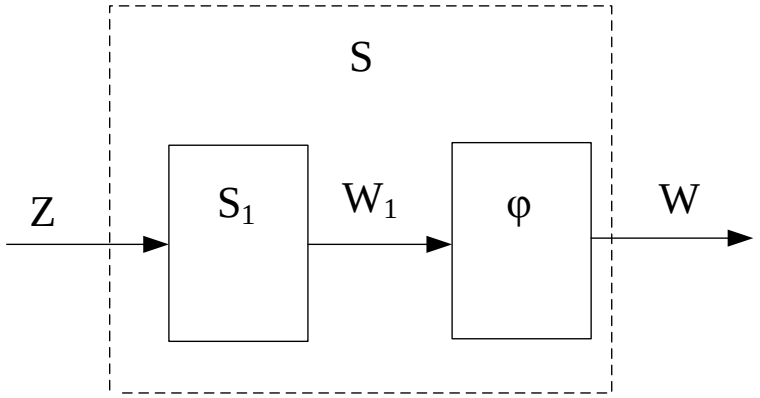


Рис. 49

Автоматы задаются как:

$$\boldsymbol{S_1 = (A_1, Z, W_1, \delta_1, \lambda_1)}$$

$$\boldsymbol{S_2 = (A_2, W_1, W, \delta_2, \lambda_2)}$$

Результирующий автомат:

$\boldsymbol{S = (A, Z, W, \delta, \lambda)}$, при этом:

1. $\boldsymbol{A = A_1 \times A_2}$ или

$$\boldsymbol{A = \{a_m = (a_{m1}, a_{m2}) \,|\, (a_{m1} \in A_1) \& (a_{m2} \in A_2)\}};$$

2. $\boldsymbol{Z = Z}$;
3. $\boldsymbol{W = W}$;
4. Функция $\boldsymbol{\delta}$: $\boldsymbol{A \times Z \to A}$ определяется как

$$\boldsymbol{\delta(a_m, z_f) = \big(\delta_1(a_{m1}, z_f), \delta_2(a_{m2}, \lambda_1(a_{m1}, z_f))\big)} \text{ или}$$

$$\boldsymbol{\delta(A \times Z) = (\delta_1(A_1 \times Z), \delta_2(A_2 \times \lambda_1(A_1 \times Z)))}$$

5. Функция $\boldsymbol{\lambda}$: $\boldsymbol{A \times Z \to W}$ определяется следующим образом:

$$\boldsymbol{\lambda(a_m, z_f) = \lambda_2(a_{m2}, \lambda(a_{m1}, z_f))} \text{ или}$$

$$\boldsymbol{\lambda(A \times Z) = \lambda_2(A_2, \lambda(A_1 \times Z))}.$$

Пример 6

Заданы два автомата:

$\boldsymbol{S_1 = (B, Z, U, \delta_1, \lambda_1)}$ – рассмотрен выше при параллельном соединении,

$\boldsymbol{S_2 = (C, U, W, \delta_2, \lambda_2)}$.

$\boldsymbol{\delta_1: (B \times Z) \to B}$

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$
$\boldsymbol{z_1}$	$\boldsymbol{b_1}$	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$
$\boldsymbol{z_2}$	$\boldsymbol{b_3}$	$\boldsymbol{b_3}$	$\boldsymbol{b_2}$

$\boldsymbol{\lambda_1: (B \times Z) \to U}$

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$
$\boldsymbol{z_1}$	$\boldsymbol{u_1}$	$\boldsymbol{u_2}$	$\boldsymbol{u_2}$
$\boldsymbol{z_2}$	$\boldsymbol{u_1}$	$\boldsymbol{u_1}$	$\boldsymbol{u_1}$

$\boldsymbol{\delta_2: (C \times U) \to C}$

	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$
$\boldsymbol{u_1}$	$\boldsymbol{c_1}$	$\boldsymbol{c_2}$
$\boldsymbol{u_2}$	$\boldsymbol{c_2}$	$\boldsymbol{c_1}$

$\boldsymbol{\lambda_2: (C \times U) \to W}$

	$\boldsymbol{c_1}$	$\boldsymbol{c_1}$
$\boldsymbol{u_1}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$
$\boldsymbol{u_2}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$

Результирующим автоматом будет автомат $\boldsymbol{S = (A, Z, W, \delta, \lambda)}$, который имеет:

1. $\boldsymbol{A = B \times C = \{(b_1, c_2), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_2), (b_3, c_2)\} = \{a_1, a_2, a_3, a_4, a_5, a_6\}}$.
2. $\boldsymbol{Z = \{z_1, z_2\}}$.
3. $\boldsymbol{W = \{w_1, w_2\}}$.
4. $\boldsymbol{\delta: A \times Z \to A}$.

	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_2c_1}$
$\boldsymbol{z_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$

Например,

$\boldsymbol{\delta(a_4, z_2) = (\delta_1(b_2, z_2), \delta_2(c_2, \lambda_1(b_2, z_2))) = (b_3, \delta_2(c_2, u_1)) = (b_3, c_2) = a_6}$.

5. $\boldsymbol{\lambda: A \times Z \to W}$

	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{a_3}$	$\boldsymbol{a_4}$	$\boldsymbol{a_5}$	$\boldsymbol{a_6}$
	$\boldsymbol{b_1c_1}$	$\boldsymbol{b_1c_2}$	$\boldsymbol{b_2c_1}$	$\boldsymbol{b_2c_2}$	$\boldsymbol{b_3c_1}$	$\boldsymbol{b_3c_2}$
$\boldsymbol{z_1}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$
$\boldsymbol{z_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$

Методика выполнения лабораторной работы

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования композиции 2-х конечных абстрактных автоматов, функции которых описаны таблицами переходов и выходов: $\boldsymbol{\delta}: \boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{A}$; $\boldsymbol{\lambda}: \boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{W}$.

Порядок выполнения работы

1. Разработать программу для ввода описания функций переходов и выходов конечного автомата $\boldsymbol{\delta}: \boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{A}$ и $\boldsymbol{\lambda}: \boldsymbol{A} \to \boldsymbol{W}$ в среде программирования Java. Для этого необходимо:

1.1.Разработать структуру данных, соответствующую функциям переходов и выходов конечного абстрактного автомата и позволяющую хранить и использовать эти функции для моделирования.

1.2.Разработать алгоритм ввода функций переходов и выходов конечного автомата с клавиатуры и из текстового файла.

1.3.Разработать программу ввода функций переходов и выходов конечного автомата в среде программирования Java.

2. Разработать алгоритм моделирования одного конечного авто мата и композиции 2-х конечных абстрактных автоматов.

2.1. Предусмотреть 2 режима работы программы моделирования:

– последовательный ввод букв из входного алфавита Z и вывод результата каждого шага моделирования;

– ввод слова из букв входного алфавита Z и вывод результата моделирования.

3. Разработать программу моделирования композиции 2-х конечных абстрактных автоматов в среде программирования Java.

3.1.Разработать пользовательский интерфейс.

3.2.Предусмотреть контроль от зацикливания программы при некорректных исходных данных.

4. Исследовать поведение композиции 2-х конечных абстрактных автоматов на построенной модели.

4.1.Для различных входных слов проверить правильность работы композиции 2-х конечных абстрактных автоматов в соответствии с функциями переходов и выходов.

4.2.Исследовать модель результирующего автомата на устойчивость (возможность зацикливания программы моделирования)

4.3.Исследовать модель результирующего автомата на адекватность заданному описанию.

4.4.Результаты исследований, алгоритмы и тексты программ пред ставить в виде отчета.

Пример индивидуального задания к лабораторной работе № 9

Описание композиции автоматов S_1 и S_2 (рис. 50):

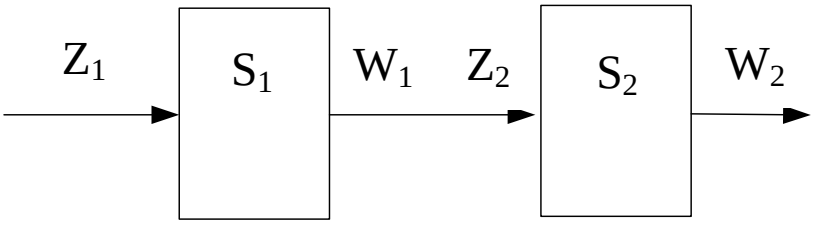


Рис. 50. Структура композиции автоматов S_1 и S_2

$\boldsymbol{S_1}$– конечный абстрактный автомат Мили задан вектором
$\boldsymbol{S_1 = (Z_1, W_1, A_1, \delta_1, \lambda_1, a_1)}$, где:
$\boldsymbol{Z_1 = \{a, b, c\}}$;
$\boldsymbol{W_1 = \{a, b, c\}}$;
$\boldsymbol{A_1 = \{d_1, d_2, d_3\}}$;
$\boldsymbol{a_1 = d_1}$;
$\boldsymbol{\delta_1}$ и $\boldsymbol{\lambda_1}$ – функции переходов и выходов автомата, заданные таблицей переходов и выходов.
$\boldsymbol{\delta_1: A_1 \times Z_1 \to A_1;\ \lambda_1: A_1 \times Z_1 \to W_1}$

	$\boldsymbol{d_1}$	$\boldsymbol{d_2}$	$\boldsymbol{d_3}$
$\boldsymbol{a}$	$\boldsymbol{d_2/c}$	$\boldsymbol{d_1/b}$	$\boldsymbol{d_2/a}$
$\boldsymbol{b}$	$\boldsymbol{d_2/a}$	$\boldsymbol{d_3/b}$	$\boldsymbol{d_1/b}$
$\boldsymbol{c}$	$\boldsymbol{d_3/a}$	$\boldsymbol{d_2/a}$	$\boldsymbol{d_3/c}$

$\boldsymbol{S_2}$ – конечный абстрактный автомат Мили задан вектором
$\boldsymbol{S_2 = (Z_2, W_2, A_2, \delta_2, \lambda_2, a_2)}$, где:
$\boldsymbol{Z_2 = \{a, b, c\}}$;
$\boldsymbol{W_2 = \{a, b, c\}}$;
$\boldsymbol{A_2 = \{h_1, h_2, h_3\}}$;
$\boldsymbol{a_2 = h_2}$;
$\boldsymbol{\delta_2}$ и $\boldsymbol{\lambda_2}$ – функции переходов и выходов автомата, заданные таблицей переходов и выходов.
$\boldsymbol{\delta_2: A_2 \times Z_2 \to A_2;\ \lambda_2: A_2 \times Z_2 \to W_2}$

	$\boldsymbol{h_1}$	$\boldsymbol{h_2}$	$\boldsymbol{h_3}$
$\boldsymbol{a}$	$\boldsymbol{h_2/c}$	$\boldsymbol{h_1/b}$	$\boldsymbol{h_2/a}$
$\boldsymbol{b}$	$\boldsymbol{h_2/a}$	$\boldsymbol{h_3/b}$	$\boldsymbol{h_1/b}$
$\boldsymbol{c}$	$\boldsymbol{h_3/a}$	$\boldsymbol{h_2/a}$	$\boldsymbol{h_3/c}$

Задание на разработку программной модели

1. Разработать программу для ввода описания 2-х конечных абстрактных автоматов, приведенных на рисунке, в среде программирования Java. Построить графы переходов.

2. Разработать программу моделирования композиции 2-х конечных абстрактных автоматов в среде программирования Java.

Задание на проведение экспериментов

1. Для различных входных слов проверить соответствие модели заданному описанию.

2. Исследовать модель на устойчивость (возможность зацикливания программы моделирования)

3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

- заданное преподавателем описание абстрактного автомата;
- структуру данных и алгоритм моделирования;
- текст моделирующей программы для композиции автоматов;
- результаты исследования поведения объединенного автомата (композиции) на построенной модели.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования абстрактных автоматов?

2. Какие правила необходимо соблюдать при моделировании последовательного и параллельного соединения абстрактных автоматов?

3. Какое различие между абстрактными автоматами Мили и Мура?

Лабораторная работа № 10
Исследование абстрактного автомата (Часть 2)

Цель работы: получение навыков исследования абстрактных автоматов.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Соединение двух автоматов с обратной связью. Структурная схема автоматов, соединенных цепью обратной связи, представлена на рисунке 51.

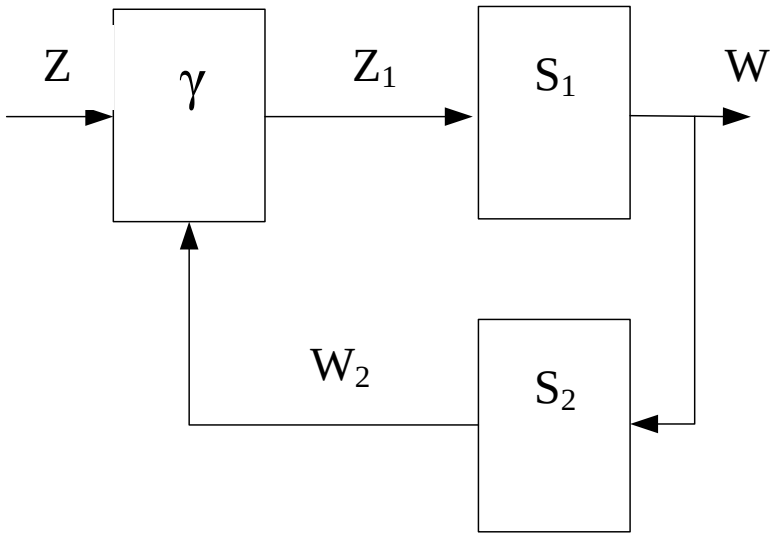


Рис. 51. Структурная схема автоматов, соединенных цепью обратной связи

Для этого автомата имеем:

$\boldsymbol{S_1 = (A_1, Z_1, W, \delta_1, \lambda_1)}$;

$\boldsymbol{S_2 = (A_2, W, W_2, \delta_2, \lambda_2)}$, где $\boldsymbol{\gamma}$ – автомат без памяти – функциональный преобразователь, реализующий отображение $\boldsymbol{\gamma: Z \times W_2 \rightarrow Z_1}$. В данном случае как минимум один из автоматов должен быть автоматом Мура.

Пусть $\boldsymbol{S_2}$ – автомат модели Мура. Тогда результирующий автомат $\boldsymbol{S = (A, Z, W, \delta, \lambda)}$, у которого:

1. $\boldsymbol{A = A_1 \times A_2}$, или
   $\boldsymbol{A = \{a_m = (a_{m1}, a_{m2}) | (a_1 \in A_1) \& (a_2 \in A_2)\}}$
2. $\boldsymbol{Z = Z}$;
3. $\boldsymbol{W = W}$;
4. $\boldsymbol{\delta}$: $\boldsymbol{A \times Z \to A}$ определяется как

$$\boldsymbol{\delta(a_m, z_f) =}$$
$$\boldsymbol{\left(\delta_1(a_{m1}, \gamma\left(z_f, \lambda_2(a_{m2})\right)\right), \delta_2(a_{m2}, \lambda_1(a_{m1}, \gamma\left(z_f, \lambda_2(a_{m2})\right))\right)}$$

или

$$\boldsymbol{\delta(A \times Z) = \left(\delta_1(A_1 \gamma\left(Z, \lambda_2(A_2)\right))\right), \delta_2(A \times \lambda_1(A_1, \gamma(Z, \lambda_2(A_2)))),}$$

5. $\boldsymbol{\lambda}$: $\boldsymbol{A \times Z \to W}$ определяется следующим образом:

$$\boldsymbol{\lambda(a_m, z_f) = \lambda_1(a_{m1}, \gamma(z_f, \lambda_2(a_{m2})))} \text{ или}$$
$$\boldsymbol{\lambda(A \times Z) = \lambda_1(A \times \gamma(Z \times \lambda_2(a_m)))}.$$

Пример 7. На рис. 52 представлен автомат с обратной связью.

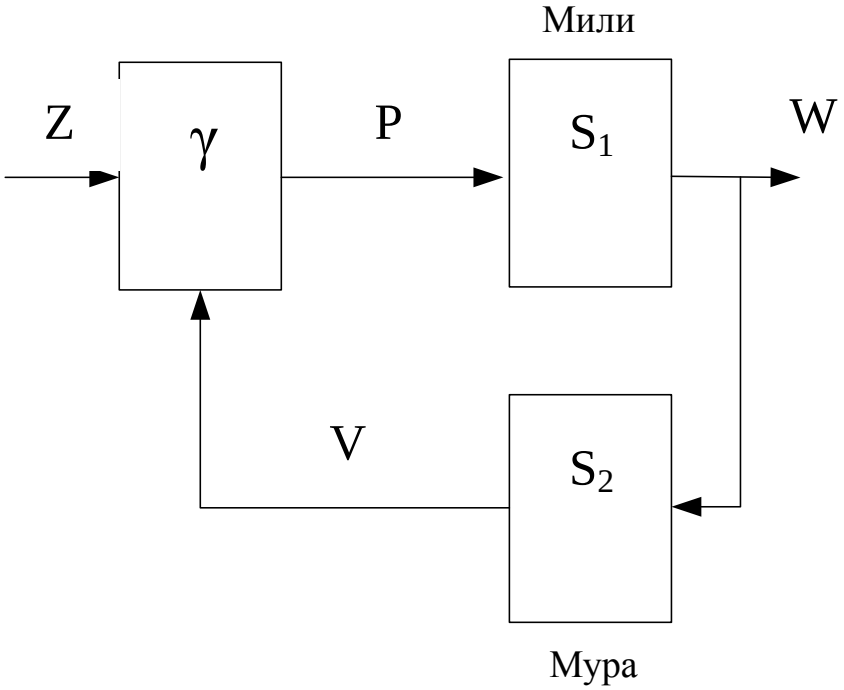


Рис. 52. Автомат с обратной связью

Описание автомата:

$\boldsymbol{\delta_1}: (\boldsymbol{B} \times \boldsymbol{P}) \to \boldsymbol{B}$

	b_1	b_2	b_3
p_1	b_3	b_2	b_3
p_2	b_2	b_1	b_1

$\boldsymbol{\lambda_1}: (\boldsymbol{B} \times \boldsymbol{P}) \to \boldsymbol{W}$

	b_1	b_2	b_3
p_1	w_1	w_2	w_1
p_2	w_3	w_1	w_2

$\boldsymbol{\delta_2}: \boldsymbol{C} \times \boldsymbol{W} \to \boldsymbol{C}$

	v_1	v_2
	c_1	c_2
w_1	c_1	c_2
w_2	c_2	c_2
w_3	c_2	c_1

$\boldsymbol{\lambda_2}: \boldsymbol{C} \to \boldsymbol{V};\ \boldsymbol{\gamma}: (\boldsymbol{Z} \times \boldsymbol{V}) \to \boldsymbol{P}$

	z_1	z_2	z_3
v_1	p_1	p_1	p_1
v_2	p_2	p_2	p_1

Решение:

$\boldsymbol{\delta}: (\boldsymbol{A} \times \boldsymbol{Z}) \to \boldsymbol{A}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_3c_1	b_2c_1	b_2c_2	b_1c_2	b_3c_1	b_1c_2
z_2	b_3c_1	b_2c_1	b_2c_2	b_1c_2	b_3c_1	b_1c_2
z_3	b_3c_1	b_3c_2	b_2c_2	b_2c_2	b_3c_1	b_3c_2

$\boldsymbol{\lambda}: \boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{W}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_1	w_3	w_2	w_1	w_1	w_2
z_2	w_1	w_3	w_2	w_1	w_1	w_2
z_3	w_1	w_2	w_2	w_2	w_1	w_1

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования конечного автомата, функция которого описана таблицами переходов и выходов $\boldsymbol{\delta}: (\boldsymbol{A} \times \boldsymbol{Z}) \to \boldsymbol{A}$, $\boldsymbol{\lambda}: \boldsymbol{A} \times \boldsymbol{Z} \to \boldsymbol{W}$. Разработать программу моделирования композиции 2-х конечных абстрактных автоматов.

Порядок выполнения работы

1. Разработать программу для ввода описания функций переходов и выходов конечного автомата: $(\boldsymbol{A} \times \boldsymbol{Z}) \rightarrow \boldsymbol{A}$, $\boldsymbol{\lambda}$: $\boldsymbol{A} \times \boldsymbol{Z} \rightarrow \boldsymbol{W}$ в среде программирования Java. Для этого необходимо:

1.1. Разработать структуру данных, соответствующую функциям переходов и выходов конечного абстрактного автомата и позволяющую хранить и использовать эти функции для моделирования.

1.2. Разработать алгоритм ввода функций переходов и выходов конечного автомата с клавиатуры и из текстового файла.

1.3. Разработать программу ввода функций переходов и выходов конечного автомата в среде программирования Java.

2. Разработать алгоритм моделирования одного конечного автомата и композиции 2-х конечных абстрактных автоматов.

2.1. Предусмотреть 2 режима работы программы моделирования:

- последовательный ввод букв из входного алфавита Z и вывод результата каждого шага моделирования;

- ввод слова из букв входного алфавита Z и вывод результата моделирования.

3. Разработать программу моделирования композиции 2-х конечных абстрактных автоматов в среде программирования Java.

3.1. Разработать дружественный пользовательский интерфейс.

3.2. Предусмотреть контроль от зацикливания программы при некорректных исходных данных.

4. Исследовать поведение композиции 2-х конечных абстрактных автоматов на построенной модели.

4.1. Для различных входных слов проверить правильность работы композиции 2-х конечных абстрактных автоматов в соответствии с функциями переходов и выходов.

4.2. Исследовать модель результирующего автомата на устойчивость (возможность зацикливания программы моделирования).

4.3. Исследовать модель результирующего автомата на адекватность заданному описанию.

4.4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 10

Описание композиции автоматов S_1 и S_2 (рис. 53):

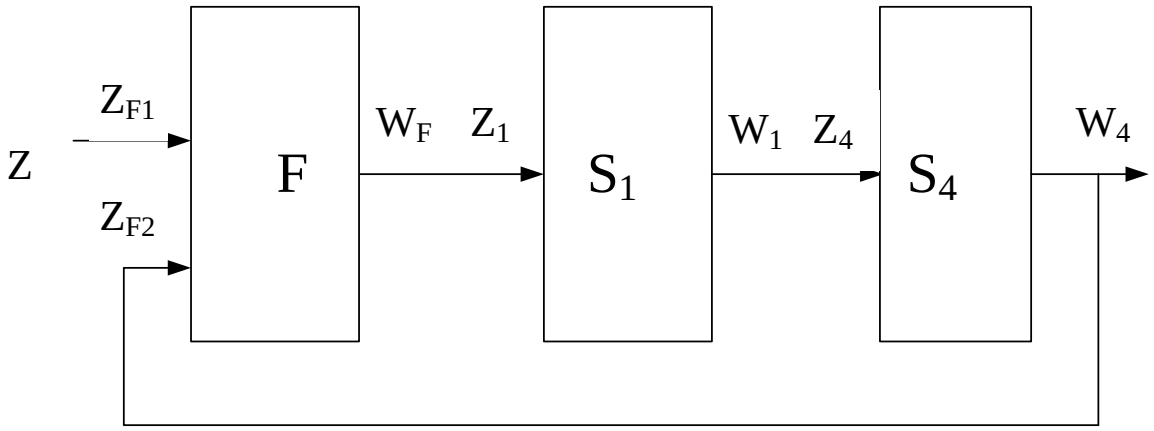


Рис. 53. Схема композиции автоматов

$\boldsymbol{S_1}$ – конечный абстрактный автомат Мили задан вектором
$\boldsymbol{S_1 = (Z_1, W_1, A_1, \delta_1, \lambda_1, a_1)}$, где:
$\boldsymbol{Z_1 = \{a, b, c\}}$;
$\boldsymbol{W_1 = \{a, b, c\}}$;
$\boldsymbol{A_1 = \{d_1, d_2, d_3\}}$;
$\boldsymbol{a_1 = d_1}$;
$\boldsymbol{\delta_1}$ и $\boldsymbol{\lambda_1}$ – функции переходов и выходов автомата, заданные таблицей переходов и выходов.
$\boldsymbol{\delta_1: A_1 \times Z_1 \to A_1}$; $\boldsymbol{\lambda_1: A_1 \times Z_1 \to W_1}$

	$\boldsymbol{d_1}$	$\boldsymbol{d_2}$	$\boldsymbol{d_3}$
$\boldsymbol{a}$	$\boldsymbol{d_2/c}$	$\boldsymbol{d_1/b}$	$\boldsymbol{d_2/a}$
$\boldsymbol{b}$	$\boldsymbol{d_2/a}$	$\boldsymbol{d_3/b}$	$\boldsymbol{d_1/b}$
$\boldsymbol{c}$	$\boldsymbol{d_3/a}$	$\boldsymbol{d_2/a}$	$\boldsymbol{d_3/c}$

$\boldsymbol{S_4}$– конечный абстрактный автомат Мура задан вектором
$\boldsymbol{S_4 = (Z_4, W_4, A_4, \delta_4, \lambda_4, a_4)}$, где:
$\boldsymbol{Z_4 = \{a, b, c\}}$;
$\boldsymbol{W_4 = \{a, b, c\}}$;
$\boldsymbol{A_4 = \{p_1, p_2, p_3\}}$;
$\boldsymbol{a_4 = p_1}$;
$\boldsymbol{\delta_4}$ и $\boldsymbol{\lambda_4}$ – функции переходов и выходов автомата, заданные таблицей переходов и выходов.

$\boldsymbol{\delta_4: A_4 \times Z_4 \to A_4;\ \lambda_4: A_4 \to W_4}$

$W_4 >$	a	b	c
$A_4 >$	p_1	p_2	p_3
a	p_2	p_1	p_2
b	p_2	p_3	p_1
c	p_3	p_2	p_3

Блок F выполняет преобразование:

$$F: Z_{F1} \times Z_{F2} \to W_F$$

Z_{F1} \ Z_{F2}	a	b	c
a	c	b	a
b	a	b	b
c	a	a	c

Задание на разработку программной модели

1. Разработать программу для ввода описания композиции 2-х конечных абстрактных автоматов, приведенных на рисунке, в среде программирования Java. Построить графы переходов.
2. Разработать алгоритм моделирования композиции 2-х конечных абстрактных автоматов (см. рис. 53).
3. Разработать программу моделирования композиции 2-х конечных абстрактных автоматов в среде программирования Java.

Задание на проведение экспериментов

1. Для различных входных слов проверить соответствие модели заданному описанию.
2. Исследовать модель на устойчивость (возможность зацикливания программы моделирования)
3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

- заданное преподавателем описание абстрактного автомата;
- структуру данных и алгоритм моделирования;
- текст моделирующей программы для объединения автоматов;
- результаты исследования поведения объединенного автомата на построенной модели.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования абстрактных автоматов?

2. Какие правила необходимо соблюдать при моделировании абстрактных автоматов?

3. Какое условие необходимо соблюдать при соединении автоматов с обратной связью?

Лабораторная работа № 11
Исследование автомата с магазинной памятью

Цель работы: получение навыков исследования автомата с магазинной памятью (МП).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Автоматы и преобразователи с магазинной памятью играют важную роль при построении автоматно-лингвистических моделей различного назначения, связанных с использованием бесконтекстных (контекстно-свободных) языков [2]. В частности, такие устройства используются в большинстве работающих программ для синтаксического анализа программ, написанных на различных языках программирования, которые во многих случаях можно рассматривать как бесконтекстные. В отличие от конечных автоматов и преобразователей, автоматы с магазинной памятью снабжены дополнительной магазинной памятью.

На рис. 54 представлен такой автомат, в котором конечное управляющее устройство (Конечное УУ) снабжается дополнительной управляющей головкой, всегда указывающей на верхнюю ячейку магазинной памяти.

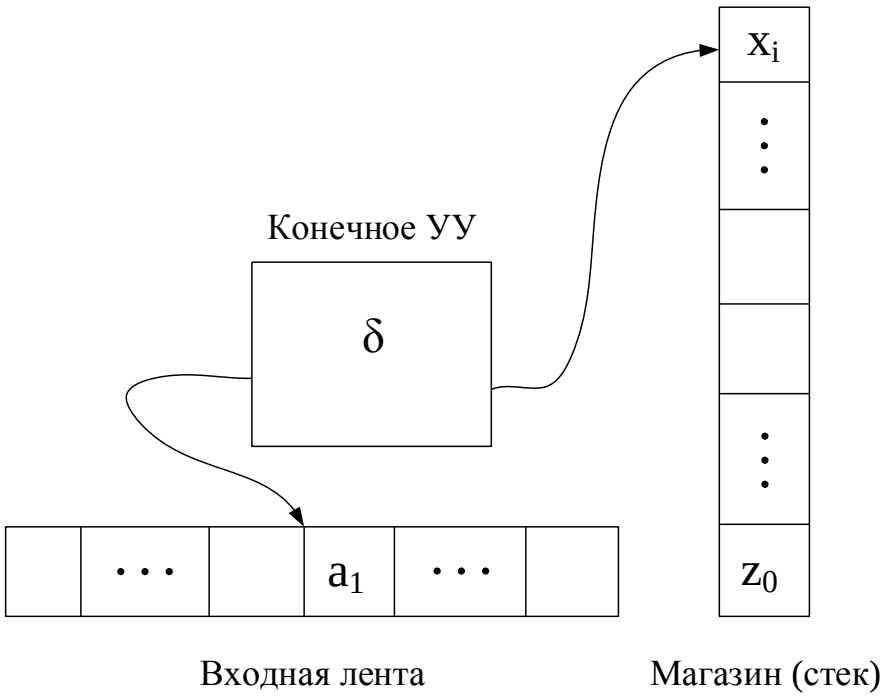


Рис. 54. Автомат с магазинной памятью

За один такт работы автомата (преобразователя) управляющая головка может произвести следующие действия:

1. Считывает символ с ленты и перемещается вправо.
2. Переходит в новое состояние.
3. Заменяет символ на вершине магазина некоторой цепочкой.

Формально детерминированный магазинный автомат определяется как следующая совокупность объектов:

$$\boldsymbol{P} = (\boldsymbol{Q}, \boldsymbol{V}, \boldsymbol{M}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{z_0}, \boldsymbol{F}),$$

где: $\boldsymbol{Q}$ – конечное множество состояний; $\boldsymbol{Q} = \{\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2} \ldots \ \boldsymbol{F}\}$, причем $\boldsymbol{q_0}$ – начальное состояние, а $\boldsymbol{F}$ – множество допускающих (заключительных) состояний;

$\boldsymbol{V}$ – конечной входной алфавит; $\boldsymbol{V} = \{\boldsymbol{a_0}, \boldsymbol{a_1}, \boldsymbol{a_2} \ldots\}$;

$\boldsymbol{M}$ – конечной магазинный алфавит; $\boldsymbol{M} = \{\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2} \ldots \boldsymbol{z_0}\}$, где

$\boldsymbol{z_0}$– маркер дна магазина;

$\boldsymbol{\delta}$ – функция переходов, $\boldsymbol{\delta}(\boldsymbol{q_i}, \boldsymbol{a_i}, \boldsymbol{x_i}) \rightarrow (\boldsymbol{q_j}, \boldsymbol{X_j})$;

где $\boldsymbol{X_j}$ – цепочка магазинных символов, замещающих $\boldsymbol{x_i}$ на вершине магазина. $\boldsymbol{X_j}$ может принимать значения:

$\boldsymbol{X_j} = \boldsymbol{e}$ – верхний магазинный символ снимается;

$\boldsymbol{X_j} = \boldsymbol{x_i}$ – верхний магазинный символ не изменяется;

$\boldsymbol{X_j} = \boldsymbol{YZ}$ – верхний магазинный символ $\boldsymbol{x_i}$ заменяется на $\boldsymbol{Z}$, а на вершину магазина – $\boldsymbol{Y}$.

Функцию δ для недетерминированного магазинного автомата можно

представить, например, совокупностью правил вида:

$$(q, a, x) \rightarrow (q_1, \gamma_1), \dots, (q_m, \gamma_m).$$

Это правило предлагает различные варианты поведения автомата при одинаковых значениях (q, a, x).

Конфигурацией магазинного автомата называется тройка (q, w, γ), где: q – текущее состояние МП;

w – оставшаяся (не прочитанная) часть входной ленты; γ – содержимое магазина.

Существует два способа определения языка, допускаемого магазинным автоматом. Согласно первому способу считается, что входная цепочка w принадлежит языку L1(МП) тогда, когда после просмотра последнего символа, входящего в эту цепочку, в магазине автомата МП будет находиться пустая цепочка ε. Согласно второму способу считается, что входная цепочка принадлежит языку L2 (МП) тогда, когда после просмотра последнего символа, входящего в эту цепочку, автомат М окажется в одном из своих заключительных состояний.

Пример 8. Рассмотрим автомат МП, допускающий язык L, все слова которого являются палиндромами четной длины.

$$P = (Q, V, M, \delta, q_0, z_0, F) = (\{q_0, q_1, q_2\}), \{0, 1\}, \{0, 1, z_0\}, \{q_2\}).$$

Функция δ определяется следующими правилами:

1. $\delta(q_0, 0, z_0) \rightarrow (q_0, 0, z_0)$;
2. $\delta(q_0, 1, z_0) \rightarrow (q_0, 1, z_0)$;
3. $\delta(q_0, 0, 0) \rightarrow (q_0, 0, 0)$;
4. $\delta(q_0, 0, 1) \rightarrow (q_0, 0, 1)$;
5. $\delta(q_0, 1, 0) \rightarrow (q_0, 1, 0)$;
6. $\delta(q_0, 1, 1) \rightarrow (q_0, 1, 1)$;
7. $\delta(q_0, \varepsilon, z_0) \rightarrow (q_1, z_0)$;
8. $\delta(q_0, \varepsilon, 0) \rightarrow (q_1, 0)$;
9. $\delta(q_0, \varepsilon, 1) \rightarrow (q_1, 1)$.

Правила 7, 8 и 9 позволяют МП спонтанно, не считывая символ с входной ленты, переходить в состояние q1, не изменяя состояния магазина.

10. $\delta(q_1, 0, 0) \rightarrow (q_1, \varepsilon)$;
11. $\delta(q_1, 1, 1) \rightarrow (q_1, \varepsilon)$.

Правила 10 и 11: МП проверяет на совпадение входного символа с символом на вершине магазина. При совпадении – верхний символ магазина снимается.

12. $\delta(q_1, \varepsilon, z_0) \rightarrow (q_2, z_0)$.

Если прочитана вся строка (ε) и обнаружен маркер магазина (z_0), то строка является палиндромом четной длины.

Примеры палиндромов четной длины: 00, 1111, 100001, 11000011 и т.п.

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования автомата с магазинной памятью (МП) в среде программирования Java.

Порядок выполнения работы

1. Разработать программу для ввода описания МП в среде программирования Java. Для этого необходимо:

1.1. Разработать структуру данных, содержащих полное описание МП и позволяющую хранить и использовать эти данные для моделирования генератора слов.

1.2. Разработать алгоритм ввода полного описания МП с клавиатуры и из текстового файла.

1.3. Разработать программу ввода описания МП в среде программирования Java.

2. Разработать алгоритм моделирования МП.

3. Разработать программу моделирования МП в среде программирования Java.

3.1. Разработать дружественный пользовательский интерфейс.

3.2. Предусмотреть контроль от зацикливания программы и контроль некорректных исходных данных.

4. Исследовать поведение МП на построенной модели.

4.1. Исследовать модель МП на устойчивость (возможность зацикливания программы моделирования).

4.2. Исследовать модель МП на адекватность заданному описанию.

4.3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 11

Задание на разработку программной модели

1. Разработать программу ввода описания МП в среде программирования Java.

Исходные данные:

Автомат с магазинной памятью (МП):

$$\boldsymbol{MP} = (\boldsymbol{Q}, \boldsymbol{V}, \boldsymbol{M}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{z_0}, \boldsymbol{F}) = (\{\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}\}), \{\boldsymbol{0}, \boldsymbol{1}\}, \{\boldsymbol{0}, \boldsymbol{1}, \boldsymbol{z_0}\}, \{\boldsymbol{q_2}\}).$$

должен допускать язык L, все слова которого имеют вид:

$1^k\ 0^k$ – префикс содержит k– единиц, суффикс содержит k – нулей.

2. Разработать правила работы автомата (систему команд δ) и алгоритм заданного МП

3. Разработать программу функционирования заданного МП в среде

программирования Java.

Задание на проведение экспериментов

1. Исследовать модель данного МП на адекватность заданному описанию.

2. Исследовать модель на устойчивость (возможность зацикливания программы моделирования).

3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

– заданную преподавателем операцию, выполняемую автоматом с магазинной памятью;

– структура данных МП;

– программу функционирования автомата с магазинной памятью;

– результаты исследования поведения автомата.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата с магазинной памятью?

2. Какие правила необходимо соблюдать при моделировании автомата с магазинной памятью?

3. Какое различие между конечными абстрактными автоматами и автомата с магазинной памятью?

Лабораторная работа № 12

Исследование языков описания микропрограммных цифровых автоматов

Цель работы: получение навыков описания, преобразования и исследования цифровых автоматов, описанных с помощью микропрограмм (МП).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и язык программирования Java. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Микропрограммный управляющий автомат – это конечный автомат, обеспечивающий выполнение микропрограмм операционным устройством.

Микропрограмма описывает алгоритм выполнения какой-либо сложной операции (арифметической, логической и т.п.) в терминах микроопераций и логических условий.

К микрооперациям относятся простейшие действия над словами информации, выполняемые элементами операционного автомата (ОА): загрузка и хранение слова, инверсия слова, сложения двух слов, инкремент, декремент, сдвиг на 1 разряд влево или вправо и т.п.

Микропрограмма часто записывается в виде ГСА (подобно ГСА в программировании). Идентификаторами в микрооперациях являются обозначения операционных элементов ОА, например, RA – регистр А, СТ – счетчик и т.д., и идентификаторы переменных. Микрооперации записываются в виде операторов присваивания с использованием идентификаторов и символов микроопераций.

Пример 9. Рассмотрим ряд микроопераций:

RC: = 0 – в регистр RC записать (загрузить) число 0.

RA: = A – в регистр RA записать (загрузить) число A.

RA: = $\overline{\mathbf{RA}}$ – инвертировать содержимое регистра RA.

RC: = RA + RB – в регистр RC поместить результат сложения чисел, хранящихся в регистрах RA и RB.

RA: = L1 (RA). 0 – сдвинуть влево на 1 разряд содержимое регистра RA, а в освободившийся крайний правый разряд записать 0.

CT: = CT + 1 – изменить состояние счетчика CT на +1.

Все операционные элементы характеризуются разрядностью. Разряды нумеруются по направлению справа налево начиная с нуля. Например, запись CT[3..0] означает, что счетчик CT – четырехразрядный. Его разряды (начиная со старших) обозначаются так: CT[3], CT[2], CT[1], CT[0].

Микрооперации, допускающие совместное выполнение, называются совместимыми. Они могут записываться в одной и той же операторной вершине ГСА.

В микропрограмме кроме микроопераций (операторные вершины ГСА) используются логические условия (условные вершины ГСА). Логические условия, как правило, вырабатываются операционным автоматом и могут принимать значения «0» (ложно) или «1» (истинно).

Пример 10. Рассмотрим некоторые логические условия в микрооперациях:

CT = 0 – содержимое счетчика CT равно нулю.

RA(0) – содержимое младшего разряда регистра RA: если оно равно нулю – переход по выходу «0» условной вершины ГСА, если равно «1» – переход по выходу «1» условной вершины ГСА.

Микропрограмма, записанная в терминах микроопераций и логических условий, называется содержательной или функциональной.

Пример 11. Рассмотрим выполнение операции умножения двух целых положительных 8-разрядных двоичных чисел: C = A * B.

Алгоритм выполнения операции умножения двух целых 8-разрядных положительных двоичных чисел можно описать следующим образом.

Имеются переменные A и B – операнды операции C = A * B – и переменная C, в которую будем записывать промежуточные суммы и в которой будет, в итоге, сформированы результат и счетчик циклов CT.

Алгоритм выполнения операции:

1. C = 0, CT = 0.
2. Если B[0] = 1, то C = C + A.
3. A = L1(A). 0 – сдвиг числа A влево на 1 разряд (умножение на 2).
4. B = R1(B) – сдвиг числа B вправо на 1 разряд.
5. CT = CT + 1.
6. Если CT = 7 – перейти к п. 8.

7. Перейти к п. 2.

8. Операция выполнена.

Совершенно очевидно, что для выполнения этой операции необходим ряд операционных элементов (элементов ОА) соответствующей разрядности, выполняющих простейшие функции: хранения, суммирования, счета циклов сложения и т.п. Это следующие элементы:

1. RB[7..0] – 8-разрядный регистр множителя – числа B. Функции регистра:

– загрузка числа: RB := B;

– сдвиг числа вправо на один разряд: RB := R1(B).

2. RA[15..0] – 16-разрядный регистр множимого – числа A. Разрядность регистра RA в два раза больше разрядности числа A, так как в процессе выполнения операции число A должно сдвигаться влево n-1 раз.

Функции регистра:

– загрузка числа: RA := A;

– сдвиг числа влево на один разряд: RA := L1(RA).

3. RC[15..0] – 16-разрядный регистр промежуточных сумм и результа-та. Разрядность регистра RC в два раза больше разрядности чисел A и B, так как в процессе выполнения операции число C может иметь разрядность в два раза большую, чем у чисел A и B.

Функции регистра:

– загрузка числа: RC := C;

– обнуление регистра: RC := 0.

4. CT[2..0] – 3-разрядный двоичный счетчик для подсчета числа циклов (до семи).

Функции счетчика:

– обнуление счетчика: CT := 0;

– счет: CT := CT + 1.

5. CM[15..0] – 16-разрядный сумматор – комбинационная схема для выполнения микрооперации: RC := RC + RA.

Содержательная микропрограмма выполнения операции умножения двух целых 8-разрядных двоичных чисел A и B и структурная схема операционного автомата приведены на рис. 55.

Каждая микрооперация в ОА инициируется микрокомандой. Микрокоманды вырабатываются управляющим автоматом (УА) в

зависимости от того, какая микрооперация должна выполняться в данный момент.

По микропрограмме, записанной в форме ГСА, строится ГСА управляющего аппарата. Для этого вместо микроопераций в операторных вершинах записываются микрокоманды (*Yi*), а вместо логических условий – их коды – *Xi*.

Кроме того, в ГСА УА часто предусматривают две дополнительные вершины – одну операторную (перед вершиной «конец») и одну условную (после вершины «начало»). В первой записывается микрокоманда завершения выполнения операции. Это микрокоманда не для операционного автомата, а для информирования о том, что операция выполнена и ОУ готово к выполнению следующей операции (в нашем примере – $У_7$). Вторая соответствует внешней команде

«Пуск» (в нашем примере – X_3), по которой ОУ начинает выполнение операции.

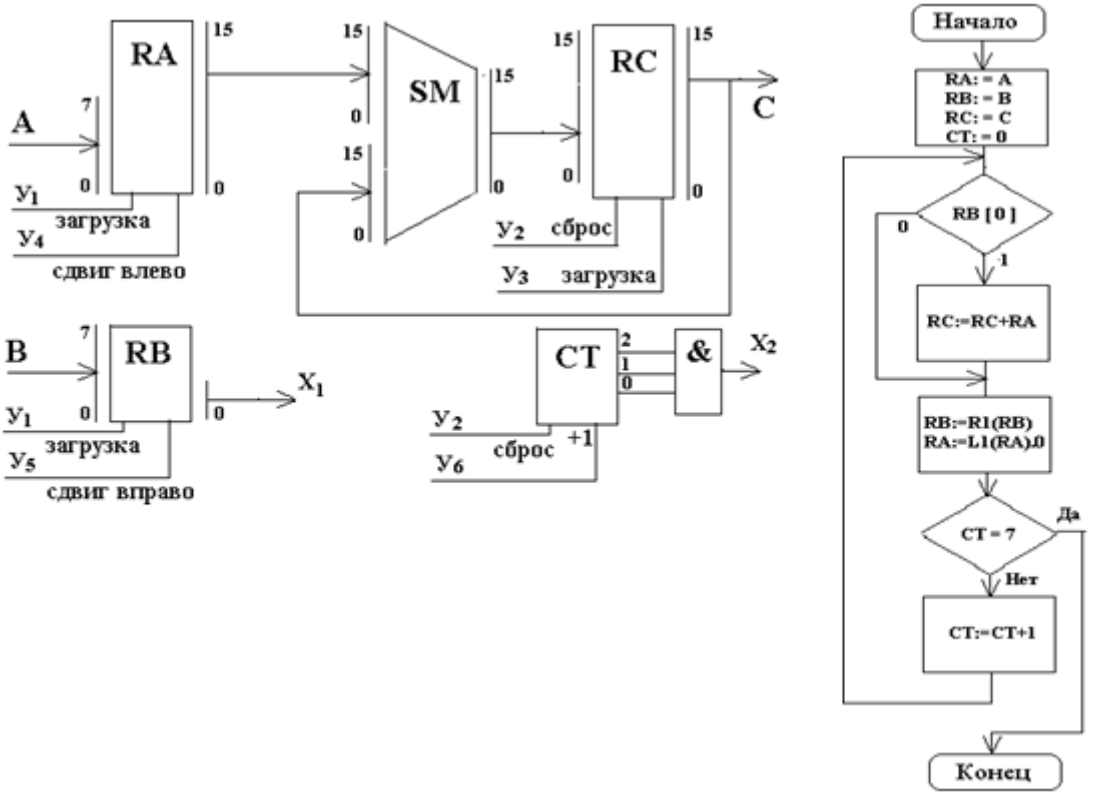


Рис. 55

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования микропрограммных цифровых автоматов в среде программирования Java.

Порядок выполнения работы

1. Разработать программу для ввода описания микропрограмм (МП) в среде программирования Java. Для этого необходимо:

1.1. Разработать структуру данных, содержащих полное описание микропрограммы и позволяющую хранить и использовать эти данные для моделирования генератора слов.

1.2. Разработать алгоритм ввода полного описания микропрограмм с клавиатуры и из текстового файла.

1.3. Разработать программу ввода описания МП в среде программирования Java.

2. Разработать алгоритм моделирования МП.

3. Разработать программу моделирования МП в среде программирования Java.

3.1. Разработать дружественный пользовательский интерфейс.

3.2. Предусмотреть контроль от зацикливания программы и контроль некорректных исходных данных.

4. Исследовать поведение МП на построенной модели.

4.1. Исследовать модель МП на устойчивость (возможность зацикливания программы моделирования).

4.3. Исследовать модель МП на адекватность заданному описанию.

4.4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 12

Задание на разработку программной модели

1. Разработать программу ввода описания МП в среде программирования Java.

Исходные данные:

Микропрограммный автомат (МПА) должен выполнять операцию сравнения двух целых положительных чисел A и B. Если A > B, то результат сравнения C = A – B; Если A < B, то результат C = B – A.

2. Разработать структурную схему и содержательную микропрограмму (МП) заданного МПА.

3. Разработать программу функционирования заданного МПА в среде программирования на Java.

Задание на проведение экспериментов

1. Исследовать модель данного МПА на адекватность заданному описанию.

2. Исследовать модель на устойчивость (возможность зацикливания программы моделирования).

3. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

- словесное описание функционирования микропрограммного автомата, заданное преподавателем;
- структурную схему автомата;
- микропрограмму, реализуемую заданным автоматом;
- текст моделирующей программы на языке Java;
- результаты моделирования заданного микропрограммного автомата.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования микропрограммного автомата?

2. Какие правила необходимо соблюдать при описании закона функционирования автомата с помощью микропрограмм?

3. Какие микрооперации автомат может выполнять одновременно?

4. Почему выполнение одного и того же алгоритма может занимать различное число тактов автомата?

Лабораторная работа № 13
Исследование автоматов Мура на жесткой логике

Цель работы: получение навыков структурного синтеза, исследования и моделирования цифровых управляющих автоматов (модель Мура).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час

Теоретические сведения

Рассмотрим порядок синтеза цифрового управляющего автомата на примере автомата, заданного ГСА на рис. 56.

Каждой операторной вершине ГСА автомата Мура соответствует определенное состояние – $\boldsymbol{a_i}$. Начальное состояние автомата соответствует началу ГСА, а точнее – входу в первую вершину ГСА. Первая вершина ГСА обычно соответствует логическому условию «Пуск», поэтому начальное состояние можно отметить на входе этой вершины. Поскольку этому состоянию не соответствует никакая операторная вершина, то и автомат в начальном состоянии не выдает никаких микрокоманд. При такой отметке начального состояния конечное состояние автомата соответствует завершению ГСА (перед вершиной «Конец»).

Для корректной работы автомата необходимо, чтобы после завершения выполнения алгоритма автомат вернулся в начальное состояние. Таким образом можно совместить начальное и конечное состояния автомата и обозначить их одинаково $\boldsymbol{a_0}$.

Часто дополнительно введенную операторную вершину, соответствующую микрокоманде «Операция выполнена», отмечают как конечное состояние автомата ($\boldsymbol{a_0}$), а так как конечное состояние автомата должно совпадать с начальным, в ГСА вводится еще одна дополнительная операторная вершина, которая отмечается состоянием $\boldsymbol{a_0}$, так же, как конечная.

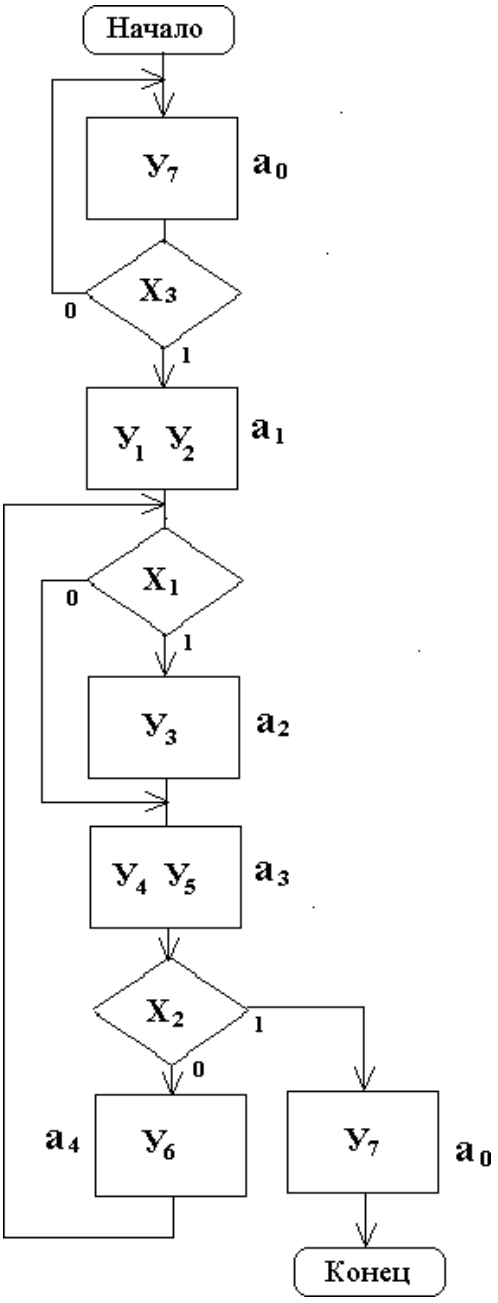


Рис. 56. ГСА автомата

Такое преобразование ГСА имеет определенные преимущества: если автомат «свободен», он не «молчит», а выдает микрокоманду «Операция выполнена», что говорит о его готовности к работе. Остальным операторным вершинам соответствуют состояния, которые можно пронумеровать так: a_1, a_2, a_3 и т.д. (см. рис. 56). Таким образом, ГСА автомата Мура отличается наличием еще одной дополнительной операторной вершиной с состоянием a_0 и дополнительной условной (x_3).

Построение графа переходов автомата Мура (по ГСА на рис. 56)

Вершины графа соответствуют состояниям автомата, дуги – переходам из состояния a_m в состояние a_s. У вершин графа записываются микрокоманды, соответствующие состояниям, в начале дуги – логические условия, определяющие переход из состояния a_m в состояние a_s (рис. 57).

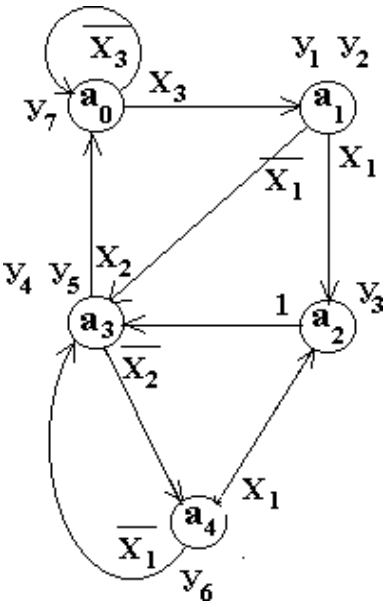


Рис. 57. Граф переходов автомата Мура

Построение прямой таблицы переходов автомата Мура. Прямая таблица переходов (таблица 17) строится по графу, представленному на рис. 57. Количество строк в таблице равно количеству переходов в графе. В столбце a_m записываются состояния, из которых начинается переход, в столбце a_s – состояния, в которые перешел автомат из a_m. В столбце Ya_s записываются Y_i – микрокоманды, вырабатываемые автоматом в состоянии a_s. В столбце $X\ a_m\ a_s$ записываются логические условия (их конъюнкция), обеспечивающие переход из состояния a_m в состояние a_s. Прямая таблица позволяет проверить полноту переходов, показанных на графе переходов: дизъюнкция всех $X\ a_m\ a_s$ из состояния a_m должна быть равна «1». В нашем примере дизъюнкция всех $X\ a_0\ a_s = \neg\, x_3 \vee\ x_3 = 1$.

Кодирование состояний автомата. Выбор элементов памяти. Так как поведение автомата всегда зависит от его текущего состояния a_m, необходимо хранить код состояния a_m в памяти состояний автомата. Объем памяти зависит от способа кодирования состояний. При минимальном кодировании каждому состоянию соответствует число в двоичном представлении, причем количество разрядов этого числа n определяется выражением $n =]\ log_2 \mid A \mid [$. Другой крайний случай – унитарное кодирование, при котором $n = \mid A \mid$. От выбранного способа кодирования и самого кодирования состояний может зависеть сложность схемы автомата.

Таблица 17

№ п/п	a_m	a_s	Ya_s	$X\ a_m\ a_s$
1	a_0	a_0	y_7	$\neg\ x_3$
2		a_1	$y_7,\ y_2$	x_3
3	a_1	a_2	y_3	x_1
4		a_3	$y_4,\ y_5$	$\neg x_1$
5	a_2	a_3	$y_4,\ y_5$	1
6	a_3	a_4	y_6	$\neg\ x_2$
7		a_0	y_7	x_2
8	a_4	a_2	y_3	x_1
9		a_3	$y_4,\ y_5$	$\neg x_1$

Используем для нашего примера минимальное кодирование состояний. Так как автомат имеет пять состояний, то минимальное количество элементов памяти $n =]\ log_2 \mid A \mid [=]\ log_2\ 5\ [= 3,$

где] ... [– обозначение «большее целое».

Выберем в качестве элементов памяти D-триггеры (описание D-триггера смотри далее). Для нашего примера их количество равно трем. Обозначим их как T_2 , T_1 , T_0 , причем T_2 соответствует старшему разряду кода состояний. Выходы триггеров обозначаются соответственно $Q_2Q_1Q_0$. Значение числа $Q_2Q_1Q_0$ на этих выходах есть код состояния автомата.

Закодируем состояния автомата произвольно $K\ a_i = \ Q_2Q_1Q_0$:

$K\ a_0$ = 100. $K\ a_1$ = 001. $K\ a_2$ = 010.

$K\ a_3$ = 000. $K\ a_4$ = 011.

Обратная структурная таблица автомата Мура. Обратная структурная таблица автомата Мура (таблица 18) строится на основе прямой таблицы переходов путем упорядочивания строк по столбцу a_s и добавления столбцов:

$K\ a_m$ – код состояния a_m; $K\ a_s$ – код состояния a_s.

$F\ a_m\ a_s$– функции управления элементами памяти при переходе из состояния a_m в состояние a_s. Поскольку в качестве элементов памяти используем D-триггеры, в этом столбце записываем только D_i, соответствующие триггерам, которые необходимо установить в состояние «1», чтобы обеспечить переход в состояние с кодом $K\ a_s$.

Таблица 18

№ п/п	a_m	Ka_m	a_s	Ka_s	$X\ a_m\ a_s$	Ya_s	$F\ a_m\ a_s$
1	a_0	100	a_0	100	$\neg\, x_3$	y_7	D_2
2	a_3	000			x_2		D_2
3	a_0	100	a_1	001	x_3	$y_1,\ y_2$	D_0
4	a_1	001	a_2	010	x_1	y_3	D_1
5	a_4	011			x_1		D_1
6	a_1	001	a_3	000	$\neg\, x_1$	$y_4,\ y_5$	-
7	a_2	010			1		-
8	a_4	011			$\neg\, x_1$		-
9	a_3	000	a_4	011	$\neg\, x_2$	y_6	D_1, D_0

Функции управления элементами памяти и функции выходов автомата

Функции управления элементами памяти записываются по обратной структурной таблице автомата:

$$D_i = F(\ a_m, X\ a_m\ a_s)$$

Смысл этого выражения следующий (например, для D_2): значение функции D_2 должно быть равно 1 (см. обратную структурную таблицу 2) в двух случаях (1-я и 2-я строки таблицы): если автомат находился в состоянии a_0, а значение $x_3 = 0$; если автомат находился в состоянии a_3, а значение $x_2 = 1$. Таким образом, функция D_2 имеет вид:

$$D_2 = a_0 \neg x_3 \vee a_3\ x_2$$

Остальные функции D_1 и D_0 записываются аналогично:

$$D_1 = a_1\ x_1 \vee a_4\ x_1 \vee a_3\ \neg x_2 = x_1(\ a_1 \vee a_4) \vee a_3\ \neg x_2.$$

$$D_0 = a_0\ x_3 \vee a_3\ \neg x_2.$$

Функции выходов также записываются по обратной структурной таблице автомата: $y_i = F(\ a_s)$.

Так как y_i в автомате Мура зависят только от текущего состояния автомата, то для нашего примера они имеют вид:

$$y_1 = y_2 = a_1;\ y_3 = a_2;\ y_4 = y_5 = a_3;\ y_6 = a_4;\ y_7 = a_0.$$

Это означает следующее: в состоянии a_1 автомат вырабатывает микрокоманды y_1 и y_2 , в состоянии a_2 – микрокоманду y_3 и т.д.

Структурная схема автомата Мура на жесткой логике.

Структурная схема автомата Мура состоит из цифровых узлов, показанных на рис. 58.

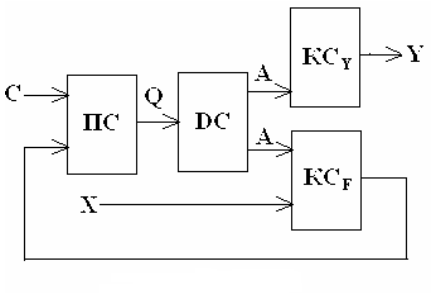


Рис. 58

На рис. 58 используются следующие обозначения: память состояний (ПС), дешифратор состояний (DC), комбинационная схема формирования сигналов управления элементами памяти состояний (КСF), комбинационная схема формирования выходных сигналов автомата (КСУ). Взаимодействие узлов автомата следующее. Автомат находится в некотором состоянии a_m, код которого Ka_m в виде значений Q на выходе триггеров памяти состояний (ПС) подается на вход дешифратора состояний (DC), на выходе которого собственно и формируются значения переменных a_m. На выходах комбинационной схемы $КС_У$ формируются микрокоманды $У$, а на выходах схемы $КС_F$ формируются значения функций управления элементами памяти, которые обеспечивают переход автомата в новое состояние a_s при поступлении импульса синхронизации C на вход синхронизации ПС.

Функциональная схема автомата Мура на жесткой логике

Функциональная схема автомата Мура на рис. 59 состоит из следующих цифровых узлов:

- память состояний. В нашем примере – это триггеры T_2, T_1, T_0.
- дешифратор состояний DC. Дешифратор необходим для преобразования двоичного кода состояний автомата Ka_m в унитарный код, соответствующий переменным a_i , используемым в записанных выше функциях. В нашем примере дешифратор DC имеет 3 входа и 8 выходов. На вход DC подается Ka_m – код состояния a_m, а на выходах DC формируется унитарный код состояния автомата a_m: единица на i-ом выходе дешифратора DC формируется при $Ka_m = i$.
- комбинационная схема формирования сигналов управления элементами памяти состояний автомата реализует функции $D_i = F(\ a_m, Xa_m\ a_s)$.
- комбинационная схема формирования выходных сигналов автомата реализует функции $y_i = F(\ a_s)$.

В функциональной схеме (рис. 59) использованы «шины». Шины представляют собой множество соединений схемы, изображенных в виде одной утолщенной линии. Вход в шину и выход из нее конкретного соединения обозначаются либо одним и тем же числом, либо содержательным обозначением сигнала, передаваемого поэтому соединению. Применение шин в схемах позволяет избежать большого числа пересечений на схеме и делает ее более простой для чтения.

С целью упрощения схемы в ней не показаны элементы, обеспечивающие установку автомата в начальное состояние a_0 с кодом $Ka_0 = 100$. Этот вопрос будет рассмотрен ниже.

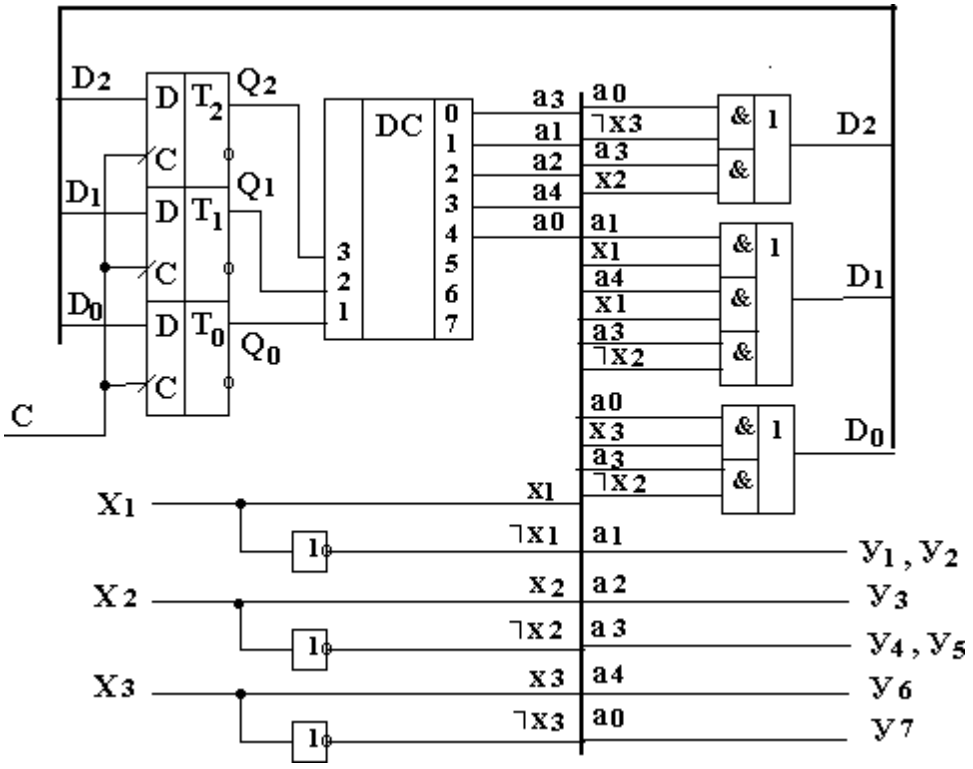


Рис. 59. Функциональная схема автомата Мура

На рис. 60 приведены временные диаграммы, поясняющие работу автомата Мура. Находясь в некотором состоянии a_i, автомат вырабатывает выходной сигнал (микрокоманду) y_j, соответствующий этому состоянию. В это же время формируются сигналы управления элементами памяти D_i, которые определяют следующее состояние автомата в зависимости от текущего состояния и значений логических условий x_i. При поступлении на вход синхронизации автомата положительного фронта импульса C автомат переходит в новое состояние, определяемое значениями D_i на входах триггеров T_2 T_1 T_0.

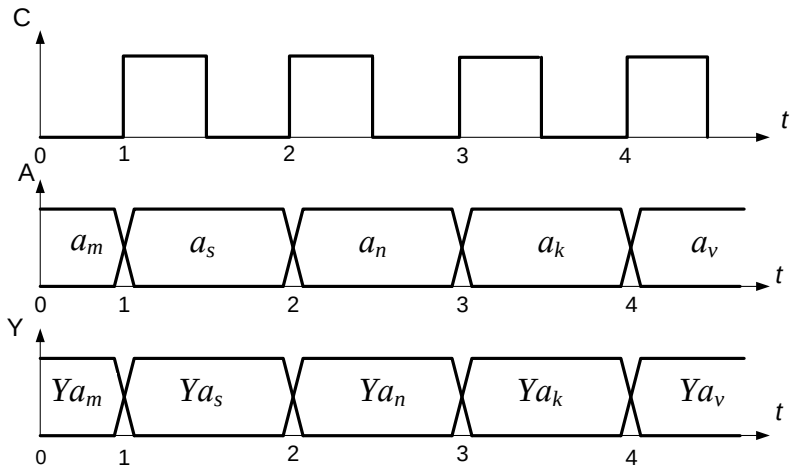


Рис. 60. Временная диаграмма

Элементная база автоматов на жесткой логике состоит из простейших логических элементов: конъюнкторов (элементы «И»), дизъюнкторов (элементы «ИЛИ»), инверторов (элементы «НЕ»), их комбинаций (элементы «И-НЕ», «ИЛИ-НЕ», «И-ИЛИ-НЕ» и т.п.). Кроме того, она также включает в себя и более сложные элементы – дешифраторы (DC) и элементы памяти (триггера).

Простейшие логические элементы реализуют вычисление простейших булевых функций. Примеры некоторых элементов, их обозначения и реализуемые ими функции приведены на рис. 61. Символ « ¬ » обозначает инверсию, «&» – конъюнкцию, « ∨ » – дизъюнкцию. Часто вместо символа «&» используют «*», а в случае однобуквенных переменных вообще его опускают, например, функции, приведенные на рис. 61, можно записать так: $y = a\ b;\ \ y = \neg(a\ b \vee c\ d \vee e)$.

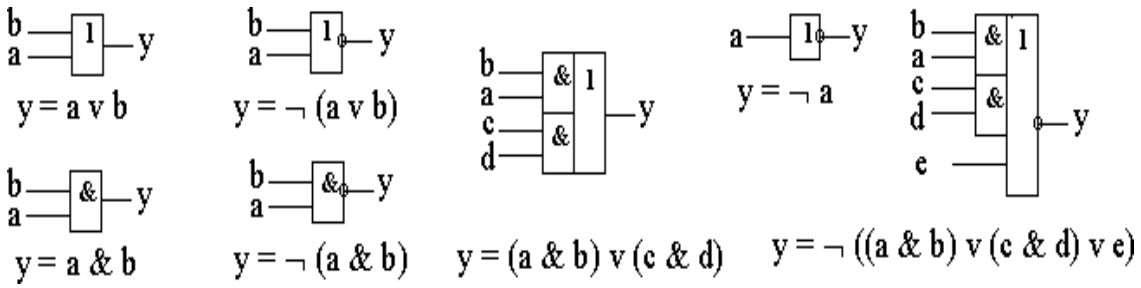


Рис. 61. Функции логических элементов

Используя простейшие логические элементы можно строить функциональные схемы, реализующие сложные булевы функции, например, построить функциональную схему, реализующую вычисление следующей системы булевых функций:

$$y_1 = x_2 \neg x_1 \, x_0 \vee \neg x_2 \, x_1 \, x_0 \vee \neg x_2 \neg x_1 \neg x_0;$$
$$y_2 = x_2 \neg x_1 \, x_0 \vee \neg x_2 \, x_0 \vee \neg x_2 \neg x_0;$$
$$y_3 = x_2 \, \neg x_0 \vee \neg x_2 \, x_0 \vee \neg x_1.$$

В приведенном примере в функциях y_1 , y_2 , y_3 встречаются одинаковые конъюнкции, которые обозначим через вспомогательные переменные Z_i и выразим через x_i:

$$Z_1 = x_2 \neg x_1 \, x_0; \; Z_2 = \neg x_2 x_1 \, x_0; \; Z_3 = \neg x_2 \neg x_1 \, \neg x_0; \; Z_4 = \neg x_2 \, x_0;$$
$$Z_5 = x_2 \, \neg x_0.$$

Функциональные схемы строятся по следующим правилам.

1. Входы схемы (переменные x_i) – с левой стороны схемы, выходы (переменные y_i) – с правой стороны.

2. Входы схемы (переменные x_i), значения которых используются в функциях с инверсиями, соединяются с входами инверторов, на выходах которых формируются значения $\neg x_i$.

3. Схемы, вычисляющие значения Z_i , строятся на конъюнкторах, на входы которых подаются соответствующие переменные x_i или $\neg x_i$.

4. Схемы, вычисляющие значения y_1 , строятся на дизъюнкторах, на входы которых подаются соответствующие переменные x_i , $\neg x_i$ или Z_i .

5. Чтобы схема была легко читаема, при ее изображении используют «шины». Шина изображается более толстой линией на схеме, в которую входят и из которой выходят тонкие линии, отображающие связи между элементами. Для идентификации этих связей непосредственно у шины ставятся либо обозначения соответствующих переменных, либо номера этих связей (первое предпочтительней).

На рис. 62 приведена функциональная схема, реализующая функции y_1, y_2 , y_3.

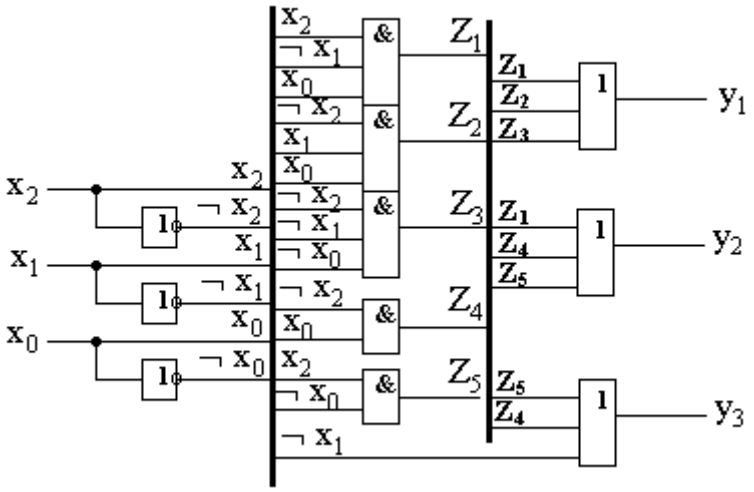


Рис. 62. Функциональная схема, реализующая функции $y_1,\ y_2\ ,\ y_3$

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования цифрового управляющего автомата Мура (ЦУА) и исследовать разработанную модель.

Порядок выполнения работы

1. Спроектировать автомат Мура на жесткой логике по заданной ГСА:

1.1. Разметить состояния автомата по ГСА.

1.2. Построить граф переходов и прямую таблицу переходов.

1.3. Построить обратную структурную таблицу автомата.

1.4. Сформировать функции выходов и управления элементами памяти

1.5. Построить функциональную схему автомата.

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели.

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель ЦУА на устойчивость (возможность зацикливания программы моделирования).

2.4. Исследовать модель ЦУА на адекватность заданной ГСА.

2.5. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 13

Задание на разработку модели ЦУА

1. Разметить состояния автомата Мура по ГСА.

2. Построить граф переходов и прямую таблицу переходов ЦУА.

3. Построить обратную структурную таблицу автомата ЦУА.

4. Сформировать функции выходов и управления элементами памяти ЦУА.

5. Построить функциональную схему ЦУА и реализовать ее, используя программу моделирования электронных схем Multisim.

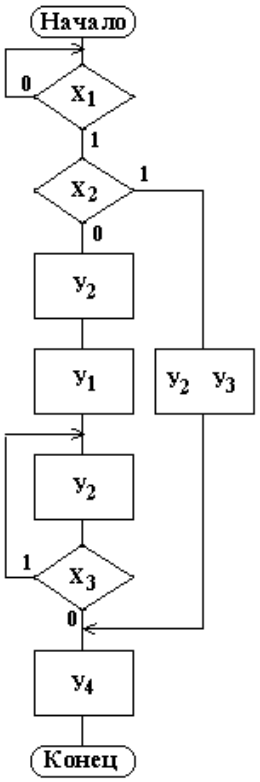
Задание на проведение экспериментов

1. Для всех комбинаций логических условий X проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата Мура на устойчивость (возможность зацикливания программы моделирования).

3. Исследовать модель автомата Мура на адекватность заданной ГСА.

4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата, заданного преподавателем;
- разметку состояний автомата;
- граф переходов и прямую таблицу переходов;
- обратную структурную таблицу автомата;
- функции выходов и управления элементов памяти;

- функциональную схему автомата;

- результаты исследования автомата Мура с жесткой логикой с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования структурного автомата Мура?
2. Какие правила необходимо соблюдать при разметке состояний автомата Мура?
3. Какое различие между структурными автоматами Мили и Мура?
4. Какие элементы памяти могут использоваться в автоматах Мура, законы их функционирования?

Лабораторная работа № 14
Исследование автоматов Мили на жесткой логике

Цель работы: получение навыков структурного синтеза, исследования и моделирования цифровых управляющих автоматов (модель Мили).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

Рассмотрим порядок синтеза автомата на примере автомата, заданного с помощью ГСА, изображенной на рис. 63.

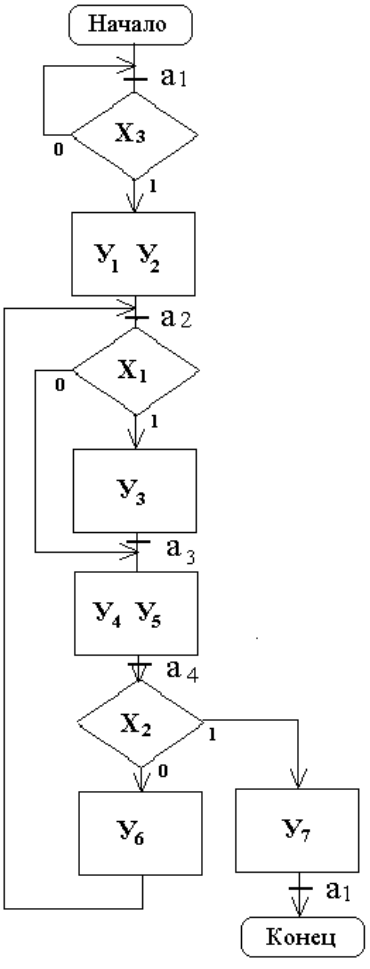


Рис. 63. ГСА для синтеза автомата Мили

Разметка состояний автомата по ГСА

В отличие от автомата Мура состояния автомата Мили не соответствуют операторным вершинам ГСА, а отмечаются на дугах ГСА перед вершинами, следующими за операторными. Исключение составляет начальное (оно же конечное) состояние автомата. Его удобно обозначать символом a_0 или a_1. Символом a_0 или a_1 отмечают вход вершины, следующей за начальной, и вход конечной вершины ГСА. Входы всех остальных вершин, следующих за операторными, также отмечаются символами: $a_1, a_2, a_3 \ldots$.

Используем для примера ГСА УА (рис. 63) для синтеза автомата Мили. Обозначим начальное состояние как a_1, а остальные – a_2, a_3, a_4.

В случае, когда в вершину, следующую за операторной вершиной, входит более чем одна дуга, состояние необходимо отметить на дуге так, чтобы для всех входящих дуг соблюдалось правило разметки состояний. На ГСА на рис. 63 это состояния a_2 и a_3.

Состояние a_2 необходимо отметить ниже входящей слева стрелки, а состояние a_3 – выше входящей справа стрелки. В первом случае в a_2 сошлись пути из двух операторных вершин, а во втором – путь из a_2 не приводит в состояние a_3 (этот переход был бы «пустым», без прохода через операторную вершину), а приводит в состояние a_4 (после операторной вершины).

Построение графа переходов автомата Мили по ГСА

Вершины графа соответствуют состояниям автомата, дуги – переходам из состояния a_m в состояние a_s. У выхода дуги из вершины графа a_m записываются логические условия, определяющие переход из состояния a_m в состояние a_s, а у входа дуги в состояние a_s – микрокоманды, вырабатываемые автоматом при переходе из состояния a_m в состояние a_s (рис. 64).

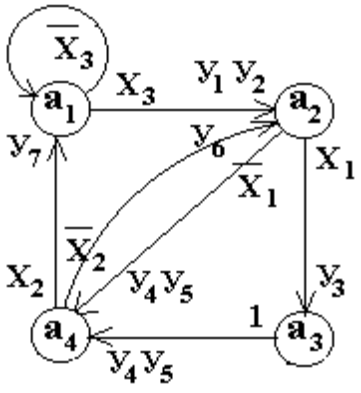


Рис. 64. Граф переходов

Построение прямой таблицы переходов автомата Мили

Прямая таблица 19 переходов строится по графу переходов на рис. 64.

Таблица 19

№ п/п	a_m	a_s	$Xa_m a_s$	$Ya_m a_s$
1	a_1	a_1	$\neg x_3$	-
2		a_2	x_3	$y_1\ y_2$
3	a_2	a_3	x_1	y_3
4		a_4	$\neg x_1$	$y_4\ y_5$
5	a_3	a_4	1	$y_4\ y_5$
6	a_4	a_1	x_2	y_7
7		a_2	$\neg x_2$	y_6

Количество строк в таблице равно количеству переходов в графе переходов. В столбце a_m записываются состояния, из которых начинается переход, в столбце a_s – состояния, в которые перешел автомат из состояния a_m.

В столбце $Ya_m a_s$записываются Y_i – микрокоманды, вырабатываемые автоматом при переходе из состояния a_m в состояние a_s. В столбце $Xa_m\, a_s$ записываются логические условия (их конъюнкция), обеспечивающие переход из состояния a_m в состояние a_s.

Прямая таблица позволяет проверить полноту переходов, показанных на графе переходов: дизъюнкция всех $Xa_m a_s$ из состояния a_m должна быть равна «1» ($Xa_m a_s = 1$). В нашем примере дизъюнкция всех $Xa_1 a_s$ равна

$$Xa_1 a_s = \cup\, Xa_1 a_s = Xa_1 a_1 \vee Xa_1 a_2 = \neg\, x_3 \vee x_3 = 1;$$

Аналогично $\cup\, Xa_2 a_s = Xa_2 a_3 \vee Xa_2 a_4 = x_1 \vee \neg x_1 = 1;$

$$\cup\, Xa_3 a_s = Xa_3 a_4 = 1;$$

$$\cup\, Xa_4 a_s = Xa_4 a_1 \vee Xa_4 a_2 = x_2 \vee \neg x_2 = 1.$$

Кодирование состояний автомата. Выбор элементов памяти

Кодирование состояний автомата Мили производится так же, как и автомата Мура.

Используем для нашего примера минимальное кодирование состояний. Так как автомат имеет четыре состояния, то минимальное количество элементов памяти:

n =] *log*2 | *A* | [=] *log*2 4 [= 2.

Выберем в качестве элементов памяти синхронные RS-триггеры. Для нашего примера их количество равно двум. Обозначим их как T_1, T_0 , причем T_1 соответствует старшему разряду кода состояний. Выходы триггеров обозначаются соответственно Q_1, Q_0. Значение числа Q_1Q_0 на этих выходах есть код состояния автомата. Закодируем состояния автомата произвольно $Ka_i = Q_1Q_2$: $Ka_1 = 00;\ Ka_2 = 01; Ka_3 = 10; Ka_4 = 11.$

Обратная структурная таблица автомата Мили

Обратная структурная таблица 20 автомата Мили строится так же, как и для автомата Мура.

Таблица 20

№ п/п	a_m	Ka_m	a_s	Ka_s	$X\ a_m\ a_s$	$Y\ a_m a_s$	$F\ a_m\ a_s$
1	a_1	00	a_1	00	$\neg\, x_3$	-	-
2	a_4	11			x_2	y_7	R_1R_0
3	a_1	00	a_2	01	x_3	$y_1,\ y_2$	S_0
4	a_4	11			$\neg\, x_2$	y_6	R_1
5	a_2	01	a_3	10	x_1	y_3	S_1R_0
6	a_2	01	a_4	11	$\neg\, x_1$	$y_4,\ y_5$	S_1
7	a_3	10			1	$y_4,\ y_5$	S_0

При заполнении столбца $F\ a_m\ a_s$ следует обратить внимание на то, что управление RS-триггерами отличается от управления D-триггерами. Если состояние некоторых разрядов RS-триггеров памяти автомата не изменяется при переходе из a_m в a_s, то нет необходимости вырабатывать соответствующие сигналы управления S=1 или R=1, так как комбинация S=0 и R=0 соответствует режиму хранения в RS-триггерах. Например, в третьей строке структурной таблицы описан переход из состояния a_1 с кодом $Ka_m = 00(Q_1 = 0; Q_2 = 0)$ в состояние a_2 с кодом $Ka_s = 01(Q_1 = 0; Q_2 = 1)$. Чтобы обеспечить переход из a_1 в a_2, нужно сохранить значение $Q_1 = 0$, а в младший разряд памяти состояний Q_0 – установить «1», поэтому в столбце $Fa_m\ a_s$ третьей строки записано «S_0», что означает $S_0 = 1$. При поступлении на вход синхронизации памяти состояний синхроимпульса С триггер T_1 не изменит своего состояния (так как R_1 =0 и S_1 =0), а триггер T_0 перейдет из состояния «0» в состояние «1» (так как R_0=0, а S_0=1). В столбце $Fa_m\ a_s$ не будем записывать $R_i = 0$ или $S_i = 0$, а будем записывать только те R_i и S_i, значения которых должны быть равны «1».

Функции управления элементами памяти и функции выходов автомата

Функции управления элементами памяти записываются по обратной структурной таблице автомата:

$$R_i = F(a_m, X\ a_m\ a_s);$$
$$S_i = F(a_m, X\ a_m\ a_s).$$

Смысл этих выражений следующий (например, для R_1). Значение функции R_1 должно быть равно «1» в двух случаях (2-я и 4-я строки таблицы 20): если автомат находился в состоянии a_4, а значение $x_2 = 1$, или, если

автомат находился в состоянии a_4, а значение $\neg x_2 = 1$. Таким образом, функция R_1 имеет вид:

$$R_1 = a_4 x_2 \vee a_4 \neg x_2 = a_4.$$

Остальные функции R_i и S_i записываются аналогично:

$$S_1 = a_2 x_1 \vee a_2 \neg x_1 = a_2;$$
$$R_0 = a_4 x_2 \vee a_2 x_1;$$
$$S_0 = a_1 x_3 \vee a_3.$$

Функции выходов автомата $Y a_m a_s$ также записываются по обратной структурной таблице автомата:

$$y_i = F(a_m, X a_m a_s).$$

Смысл этого выражения следующий (например, для y_4). Значение функции y_4 должно быть равно «1» при переходе автомата из состояний a_2 или a_3 в состояние a_4 (6-я и 7-я строки таблицы 20). Иначе: если автомат находился в состоянии a_2, а значение $\neg x_2 = 1$, или, если автомат находился в состоянии a_3, то при переходе автомата из состояний a_2 или a_3 в состояние a_4 значение y_4 должно быть равно «1». Таким образом, функция y_4 имеет вид:

$$y_4 = y_5 = a_2 \neg x_1 \vee a_3.$$

Остальные функции выходов имеют вид:

$$y_1 = y_2 = a_1 x_3;\ y_3 = a_2 x_1;\ y_6 = a_4 \neg x_2;\ y_7 = a_4 x_2.$$

Структурная схема автомата Мили на жесткой логике

Структурная схема автомата Мили состоит из следующих цифровых узлов (рис. 65):

– память состояний (ПС),

– дешифратор состояний (DC),

– комбинационная схема формирования сигналов управления элементами памяти состояний ($КС_F$),

– комбинационная схема формирования выходных сигналов автомата ($КС_У$).

Взаимодействие узлов автомата следующее. Автомат находится в некотором состоянии a_m, код которого Ka_m в виде значений Q на выходе триггеров памяти состояний (ПС) подается на вход дешифратора состояний (DC), на выходе которого собственно и формируются значения переменных a_m. На выходах комбинационной схемы $КС_F$ формируются значения функций управления элементами памяти $Fa_m a_s$, которые обеспечивают переход автомата в новое состояние a_s при поступлении импульса синхронизации C на вход синхронизации ПС, а на выходах комбинационной схемы $КС_У$ при этом формируются значения функций выходов автомата $Ya_m a_s$.

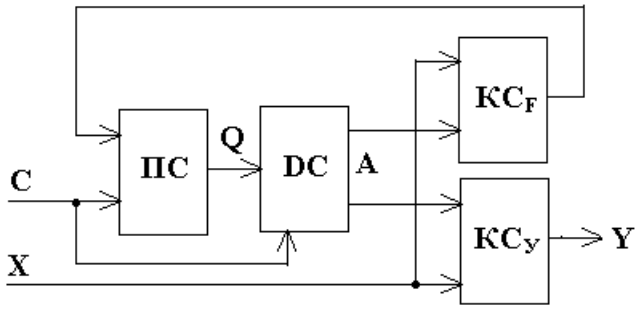


Рис. 65. Структурная схема автомата Мили

Функциональная схема автомата Мили на жесткой логике

Функциональная схема автомата Мили состоит из следующих цифровых узлов (рис. 66):

– Память состояний. В нашем примере – это два триггера T_1 и T_0.

– Дешифратор состояний DC. В нашем примере дешифратор DC имеет 2 входа и 4 выхода. На вход DC подается Ka_m – код состояния a_m, а на выходах DC формируется унитарный код состояния автомата a_m: единица на i-ом выходе дешифратора DC формируется при $Ka_m = i$.

– Комбинационная схема формирования сигналов управления элементами памяти состояний автомата. В нашем примере реализует функции $R_i = F(a_m, X\ a_m\ a_s)$; $S_i = F(a_m, X\ a_m\ a_s)$.

– Комбинационная схема формирования выходных сигналов автомата. В нашем примере реализует функции $y_i = F(a_m, X\ a_m\ a_s)$.

– Память логических условий. В нашем примере это три D-триггера Tx_1 Tx_2 Tx_3.

Значения логических условий на входе автомата Мили могут измениться во время формирования микрокоманды y_i, что может привести к формированию «ложных» (лишних) микрокоманд. Поэтому необходимо зафиксировать значения x_i, поступившие на входы автомата к моменту прихода импульса синхронизации, на время формирования микрокоманд y_i. Таким образом, по положительному фронту импульса синхронизации С значения x_i, запоминаются на триггерах Txi, при С = 1 формируются микрокоманды y_i и функции управления элементами памяти R_i и S_i , а по отрицательному фронту импульса С автомат переходит в следующее состояние, определяемое значениями R_i и S_i на входах памяти состояний автомата.

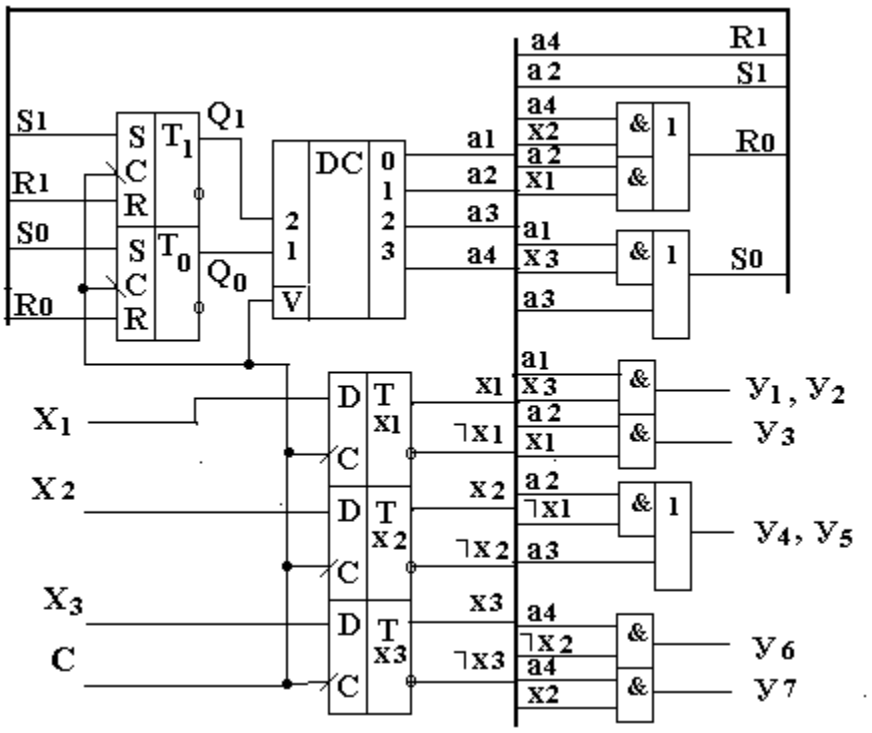


Рис. 66. Функциональная схема автомата Мили

Временная диаграмма, поясняющая работу автомата Мили, приведена на рис. 67.

Из временной диаграммы видно, что по положительному фронту импульса синхронизации С значения логических условий X на входе автомата запоминаются на триггерах T*xi*. Значения логических условий с выходов этих триггеров и текущее состояние автомата a_m используются для вычисления *Yms* – микрокоманд, вырабатываемых автоматом на переходе из состояния a_m в состояние a_s . Дешифратор состояний DC (см. рис. 66) имеет вход разрешения V: при V=1 дешифратор выдает на одном из своих выходов значение «1», при V= 0 – на всех выходах DC логический «0». Это означает, что при C=0 (а значит и V=0) все выходные сигналы автомата *Yms* равны нулю. Автомат вырабатывает микрокоманды только при C=1.

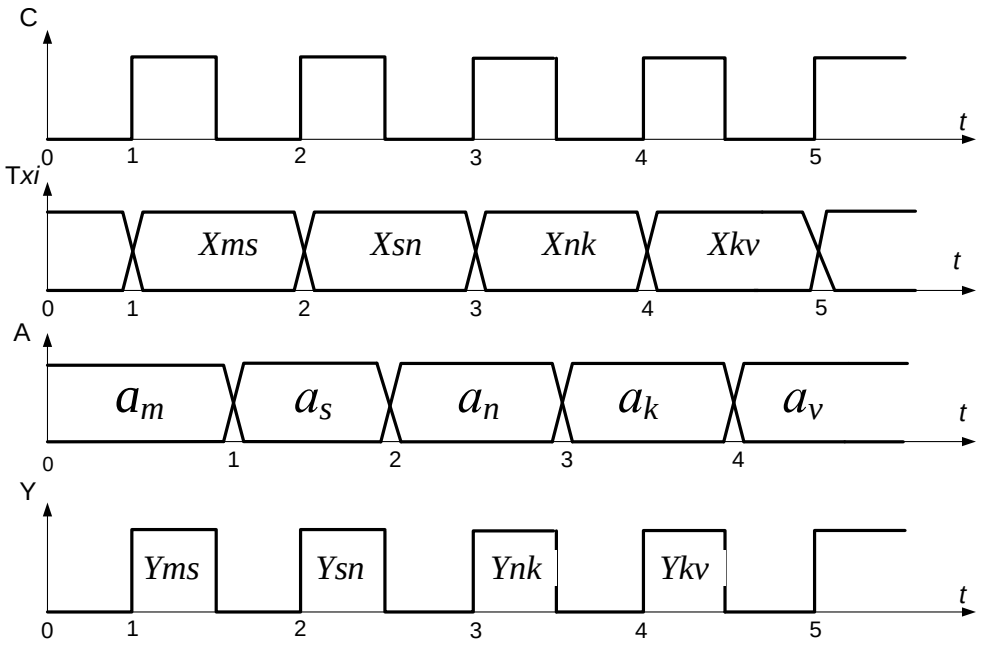


Рис. 67. Временная диаграмма

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования цифрового управляющего автомата Мили (ЦУА) на жесткой логике и исследовать разработанную модель.

Порядок выполнения работы

1. Спроектировать автомат Мили на жесткой логике по заданной ГСА:

1.1. Разметить состояния автомата по ГСА.

1.2. Построить граф переходов и прямую таблицу переходов.

1.3. Построить обратную структурную таблицу автомата.

1.4. Сформировать функции выходов и управления элементами памяти.

1.5. Построить функциональную схему автомата.

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели.

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель автомата Мили на устойчивость (возможность зацикливания программы моделирования).

2.4. Исследовать модель автомата Мили на адекватность заданной ГСА.

2.5. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 14

Задание на разработку модели ЦУА

1. Разметить состояния автомата по ГСА.

2. Построить граф переходов и прямую таблицу переходов ЦУА.

3. Построить обратную структурную таблицу автомата ЦУА.

4. Сформировать функции выходов и управления элементами памяти ЦУА.

5. Построить функциональную схему ЦУА и реализовать ее, используя программу моделирования электронных схем Multisim.

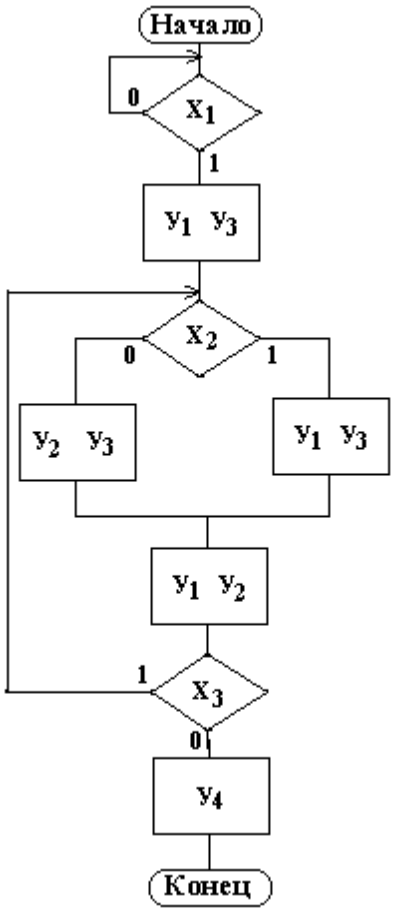
Задание на проведение экспериментов

1. Для всех комбинаций логических условий X проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата Мили на устойчивость (возможность зацикливания программы моделирования).

3. Исследовать модель автомата Мили на адекватность заданной ГСА.

4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата, заданного преподавателем;
- разметку состояний автомата;
- граф переходов и прямую таблицу переходов;
- обратную структурную таблицу автомата;
- функции выходов и управления элементов памяти;
- функциональную схему автомата;
- результаты исследования автомата Мили с жесткой логикой с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования структурного автомата Мили?
2. Какие правила необходимо соблюдать при разметке состояний автомата Мили?
3. Какое различие между структурными автоматами МИЛИ и МУРА?
4. Какие элементы памяти могут использоваться в автоматах? Законы их функционирования.

Лабораторная работа № 15
Исследование автоматов Мили на ПЛМ

Цель работы: получение навыков структурного синтеза, моделирования и исследования цифровых управляющих автоматов Мили на программируемых логических матрицах (ПЛМ).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 2 академических часа.

Время самостоятельной подготовки: 1 академический час.

Теоретические сведения

На этапе логического проектирования сложных цифровых устройств управления большие трудности возникают из-за их нерегулярности и малой повторяемости отдельных узлов. В предыдущих лабораторных работах были рассмотрены управляющие автоматы с жесткой логикой. «Жесткость» заключается в том, что любое изменение в алгоритме работы автомата приводит к изменению в комбинационных схемах, реализующих функции переходов и выходов автомата.

Существуют регулярные, настраиваемые пользователем структуры, называемые программируемыми логическими матрицами (ПЛМ). ПЛМ содержат в себе две матрицы – матрицу «И» ($М_{\&}$) и матрицу «ИЛИ» ($М_1$), соединенные последовательно. Матрица «И» вычисляет конъюнкцию логических переменных, а матрица «ИЛИ» – дизъюнкцию полученных термов. Таким образом, пару матриц «И» и «ИЛИ» удобно использовать для вычисления булевых функций, заданных в виде ДНФ (дизъюнктивной нормальной формы).

В простейшем случае ПЛМ представляет матрицу – сеть горизонтальных и вертикальных шин. В узлах матрицы могут быть (а могут и не быть) полупроводниковые диоды.

Если в узле есть диод, то горизонтальная шина через него связана с вертикальной, если диода нет – то не связана. Каждая вертикальная шина такой матрицы – это простейший диодный элемент «И» или «ИЛИ» (в зависимости от направления включения диода и значения напряжения на резисторах матрицы).

На рис. 68 приведен пример двух матриц – «И» и «ИЛИ». Каждая вертикальная линия в этих матрицах – это один диодный логический элемент «И» или «ИЛИ».

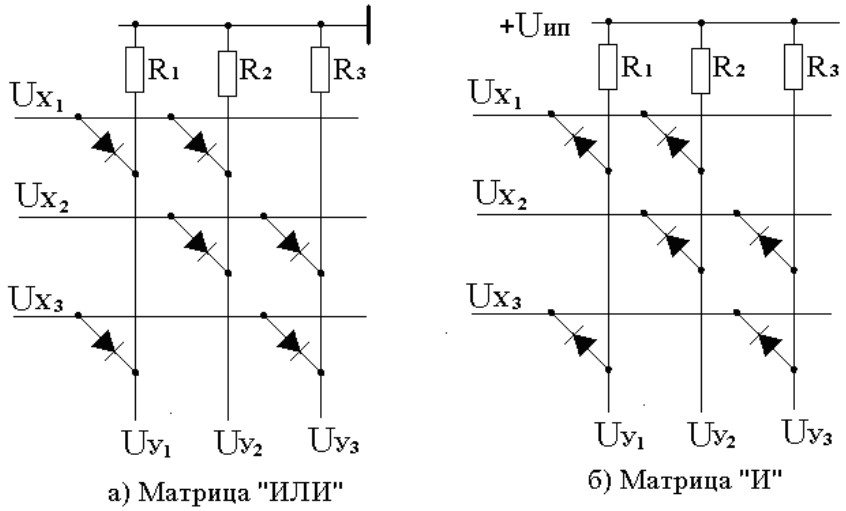


Рис. 68. Матрицы «И» и «ИЛИ»

На входы матриц подаются напряжения Ux_i, соответствующие логическим переменным x_i. Значению $x_i = 0$ в матрице соответствует $Ux_i = 0$ [В], значению $x_i = 1$ соответствует $Ux_i = U_{\text{ИП}}$ [В], где $U_{\text{ИП}}$ – напряжение источника питания в вольтах. Выходные напряжения матриц Uy_i соответствуют значениям функций y_i, реализуемых данными матрицами. Очевидно, что в матрице «ИЛИ» $Uy_i = U_{\text{ИП}}$ ($y_i = 1$), если в узле матрицы на пресечении линий Uy_i и Ux_j есть диод и $Ux_j = U_{\text{ИП}}$ ($x_j = 1$). В матрице «И» $Uy_i = U_{\text{ИП}}$ ($y_i = 1$), если на всех входах, соединенных диодами с шиной Uy_i, значения $Ux_j = U_{\text{ИП}}$ ($x_j = 1$) и $Uy_i = 0$В ($y_i = 0$), если хотя бы на одном из входов $Ux_j = 0$В ($x_j = 0$).

Таким образом, в примере (рис. 68) матрица «ИЛИ» реализует три функции Y_1, Y_2, Y_3 от трех переменных x_1, x_2, x_3:

$Y_1 = x_1 \vee x_3$; $Y_2 = x_1 \vee x_2$; $Y_3 = x_2 \vee x_3$;

а матрица «И» реализует функции

$Y_1 = x_1 \wedge x_3$; $Y_2 = x_1 \wedge x_2$; $Y_3 = x_2 \wedge x_3$.

На рис. 68 матрицы изображены фактически как принципиальные схемы, в которых логические переменные x_i представлены напряжениями Ux_i, а функции y_i – напряжениями Uy_i . В функциональных схемах будем использовать более простое изображение матриц: в узлах вместо диодов будем ставить точки, а входные и выходные сигналы будем обозначать как x_j и y_i (рис. 69).

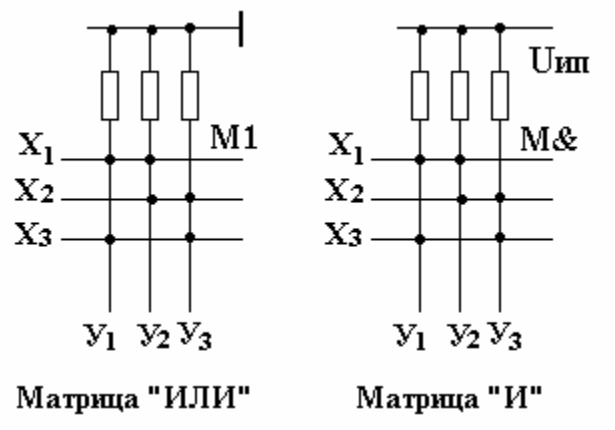


Рис. 69

Обычно матрицы используют тогда, когда необходимо реализовать семейство булевых функций от одних и тех же аргументов. Рассмотрим пример такой реализации.

Пусть дана система из трех булевых функций $y_1,\ y_2,\ y_3$ от четырех переменных $x_1,\ x_2,\ x_3,\ x_4$:

$$y_1 = \neg x_1\, x_2 \vee x_1 \neg x_2 \neg x_4 \vee x_1\, x_3\, x_4;$$
$$y_2 = \neg x_1 x_2 \vee \neg x_1 \neg x_3\, x_4 \vee \neg x_1 \neg x_2 \neg x_3 \vee \neg x_2 \neg x_4;$$
$$y_3 = x_1 \neg x_2 \neg x_4 \vee x_1 \neg x_3 \vee \neg x_1 \neg x_2 \neg x_3;$$

Введем вспомогательные переменные T_i, через которые выразим все дизъюнктивные термы, встречающиеся в функциях $y_1,\ y_2,\ y_3$:

$T_1 = \neg x_1 x_2;\ T_2 = x_1 \neg x_2 \neg x_4;\ T_3 = x_1 x_3 x_4;\ T_4 = \neg x_1 \neg x_3 x_4;$
$T_5 = \neg x_1 \neg x_2 \neg x_3;\ T_6 = \neg x_2 \neg x_4;\ T_7 = x_1 \neg x_3;$

Выразим функции $y_1,\ y_2,\ y_3$ через переменные T_i:

$$y_1 = T_1 \vee T_2 \vee T_3;$$
$$y_2 = T_1 \vee T_4 \vee T_5 \vee T_6;$$
$$y_3 = T_2 \vee T_7 \vee T_5.$$

В этих функциях через переменные T_i обозначены термы (логические произведения), которые вычисляются матрицей «И». С выходов матрицы «И» ($M_{\&}$), сигналы, соответствующие термам T_i, подаются на входы матрицы «ИЛИ» (M_1), где вычисляются значения функций $y_1,\ y_2$ и y_3. Схема приведена на рис. 70.

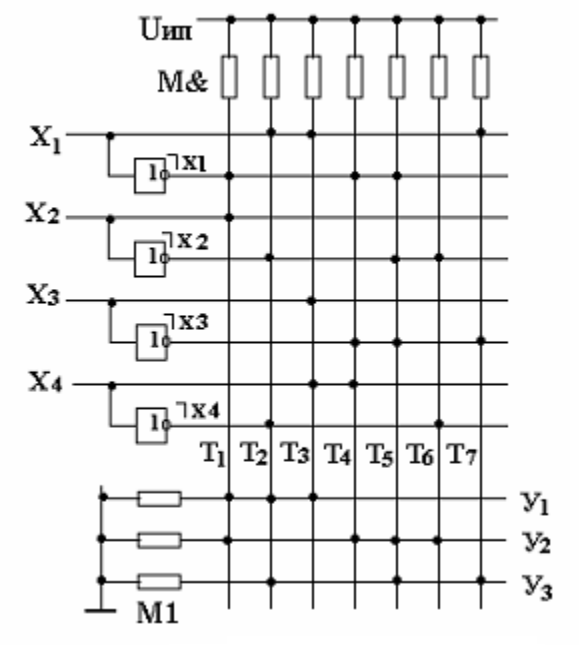


Рис. 70

С помощью матриц $M_{\&}$ («И») и M_1 («ИЛИ») можно реализовывать не только дизъюнктивные нормальные форм, но и скобочные выражения, которые часто являются более компактными.

Например, даны две функции:

$y_1 = (x_1x_2 \vee \neg x_1\neg x_2)x_3 \vee \neg(x_1x_2 \vee \neg x_1\neg x_2)\neg x_3;$

$y_2 = x_1x_2 \vee \neg(x_1x_2 \vee \neg x_1\neg x_2)x_3.$

Выражение в скобках $(x_1x_2 \vee \neg x_1\neg x_2)$ три раза встречается в функциях y_1 и y_2.

На матрицах $M_{\&}$ («И») и M_1 («ИЛИ») сформируем функцию

$f = (x_1x_2 \vee \neg x_1\neg x_2)$, и используем ее в матрице $M_{\&}$ («И») для формирования термов, а затем и значений функций y_1 и y_2. Введем переменные T_i:

$T_1 = x_1x_2;\ T_2 = \neg x_1\neg x_2.$

Через T_1 и T_1 выразим функцию f:

$$f = T_1 \vee T_2.$$

Введем еще три переменные T_i , в которых кроме $x_1,\ x_2,\ x_3$ используем функцию f:

$T_3 = f \wedge x_3;\ T_4 = \neg f \wedge \neg x_3;\ T_5 = \neg f \wedge x_3.$

Преобразуем функции y_1 и y_2 и запишем их в следующем виде:

$y_1 = (T_1 \vee T_2)\ x_3 \vee \neg(T_1 \vee T_2)\ \neg x_3 = f \wedge x_3 \vee \neg f \wedge \neg x_3 = T_3 \vee T_4$

$$y_2 = T_1 \vee \neg(T_1 \vee T_2)\, x_3 = T_1 \vee \neg f \wedge x_3 = T_1 \vee T_5$$

Таким образом, порядок вычислений функций y_1 и y_2 следующий. Матрица $M_{\&}$ вычисляет $T_1 = x_1 x_2$ и $T_2 = \neg x_1 \neg x_2$, затем матрица M_1 вычисляет $f = T_1 \vee T_2$. Значение f в прямом и инверсном виде подается на вход матрицы $M_{\&}$, где вычисляются T_3, T_4 и T_5 , а затем матрица M_1 вычисляет $y_1 = T_3 \vee T_4$ и $y_2 = T_1 \vee T_5$. На рис. 71 приведена схема, реализующая данный способ вычисления скобочных булевых функций.

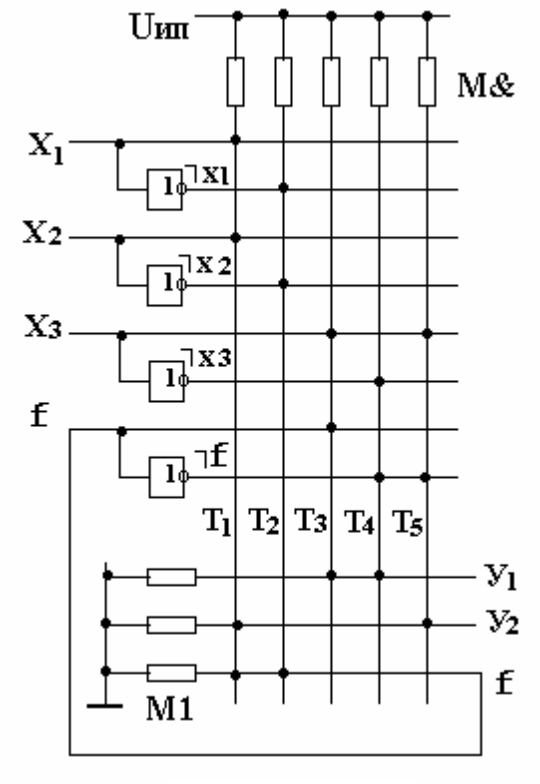


Рис. 71. Матрица для вычисления скобочных функций

Рассмотрим реализацию автомата Мили на ПЛМ. Отличие в синтезе автомата Мили на ПЛМ состоит только на этапах построения обратной структурной таблицы и функциональной схемы автомата.

Возьмем за основу структурную таблицу автомата Мили на жесткой логике. Будем использовать в качестве элементов памяти состояний D-триггеры. Добавим в таблицу еще один столбец $Ta_m a_s$, в который будем записывать функции переходов: $Ta_m a_s = a_m \wedge Xa_m a_s$, где $Xa_m a_s$ – конъюнкция логических условий, обеспечивающих данный переход из a_m в a_s; a_m – состояние автомата, из которого начинается переход, представленное кодом состояния Ka_m, заданного значениями Q_1Q_0 на выходах элементов памяти T_1 T_0. Если, например, код состояния a_m имеет значение $Ka_m = 10$, то значения на выходах элементов памяти состояний Q_1Q_0 также равны 10. Так как здесь Q_1=1, то Q_1 войдет в функцию перехода без инверсии, а Q_0 = 0 поэтому

войдет в функцию с инверсией. Например, для пятой строки структурной таблицы автомата функция перехода из a_2 в a_3 имеет вид $T_2 = a_2 x_1$, так как $Ka_2 = 01$, то $T_1 = \neg Q_1 Q_0 x_1$.

Построим обратную структурную таблицу автомата Мили с памятью состояний на D-триггерах (для матричной реализации), приведенную в таблице 21.

Таблица 21

№ п/п	a_m	Ka_m	a_s	Ka_s	$Xa_m a_s$	$Ya_m a_s$	$Fa_m a_s$	$Ta_m a_s$
1	a_1	00	a_1	00	$\neg x_3$	-	-	$T_1 = \neg Q_1 \neg Q_0 \neg x_3$
2	a_4	11			x_2	y_7	-	$T_2 = Q_1 Q_0 x_2$
3	a_1	00	a_2	01	x_3	y_1, y_2	D_0	$T_3 = \neg Q_1 \neg Q_0 x_3$
4	a_4	11			$\neg x_2$	y_6	D_0	$T_4 = Q_1 Q_0 \neg x_2$
5	a_2	01	a_3	10	x_1	y_3	D_1	$T_5 = \neg Q_1 Q_0 x_1$
6	a_2	01	a_4	11	$\neg x_1$	y_4, y_5	D_1, D_0	$T_6 = \neg Q_1 Q_0 \neg x_1$
7	a_3	10			1	y_4, y_5	D_1, D_0	$T_7 = Q_1 \neg Q_0$

Структурная схема автомата Мили на матрицах (простейшая реализация) имеет вид, показанный на рис. 72. В схеме автомата используются две матрицы – $M_{\&}$ («И»), которая используется для вычисления функций переходов $Ta_m a_s$, и M_1 («ИЛИ»), выполняющая две функции – вычисление функций ($Fa_m a_s$) управления элементами памяти состояний ПС) D_1 и D_0 и вычисление функций выходов ($Ya_m a_s$) автомата y_1.

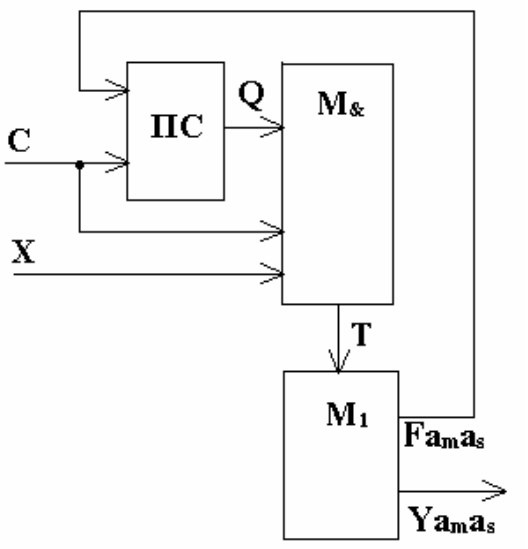


Рис. 72. Структурная схема автомата Мили на ПЛМ

Функции D_1 , D_0 и y_i для нашего примера находятся как дизъюнкции соответствующих функций переходов $Ta_m a_s$:

$D_1 = T_5 \vee T_6 \vee T_7$; $D_0 = T_3 \vee T_4 \vee T_6 \vee T_7$;

$y_1 = y_2 = T_3$; $y_3 = T_5$; $y_4 = y_5 = T_6 \vee T_7$; $y_6 = T_4$; $y_7 = T_2$.

Как видно из этих выражений, функция переходов T_1 не используется ни в одной из функций Di и y_i, поэтому ее можно не вычислять. Таким образом, функциональная схема автомата Мили на ПЛМ (для нашего примера) будет состоять из следующих элементов:

– память состояний. В нашем примере – два D-триггера: T_1 T_0.

– инверторы для формирования инверсных значений логических условий x_i.

– матрица $M_{\&}$, имеющая шесть вертикальных шин (по количеству формируемых функций переходов $Ta_m a_s$) и одиннадцать горизонтальных (удвоенное количество элементов памяти и логических условий плюс один – шина для сигнала синхронизации С).

– матрица M_1, имеющая шесть вертикальных шин (по количеству используемых функций переходов $Ta_m a_s$) и семь горизонтальных (количество функций управления элементами памяти и функций переходов).

Функциональная схема автомата приведена на рис. 73.

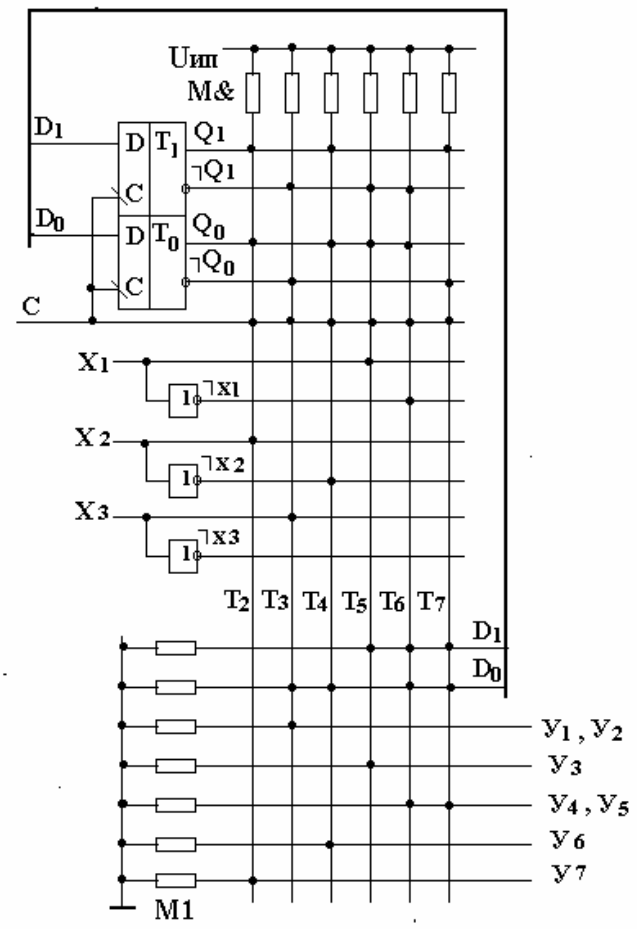


Рис. 73. Функциональная схема автомата Мили на ПЛМ

Для упрощения схемы в ней не показаны элементы, используемые для установки автомата в начальное состояние и фиксации значений логических условий при поступлении импульса синхронизации С (смотри автомат Мили на жесткой логике).

Матрицы $M_{\&}$ и M_1 характеризуются площадями $S_{M\&}$ и S_{M1}, вычислить которые для каждого конкретного примера можно по структурной таблице автомата (или по функциональной схеме). Например, площадь $S_{M\&}$ = 66: 11 горизонтальных шин и 6 вертикальных. Как видно из простого примера на рис. 36, площади матриц $M_{\&}$ и M_1 используются неэффективно по двум причинам:

1) при общем количестве логических условий $|X| = 3$ в каждом конкретном переходе из a_m в a_s участвует всего одно;

2) при общем количестве $|Y| = 7$, на каждом переходе автомат вырабатывает всего одну микрокоманду.

Матрицы $M_{\&}$ и M_1 сильно разрежены: в большом количестве узлов нет диодов. Вопросы более рационального использования площади матриц $M_{\&}$ и M_1 решаются специальными методами оптимизации автоматов на ПЛМ.

Методика выполнения

Используя материал из п. 1, по индивидуальному заданию, выданному преподавателем, разработать программу моделирования цифрового управляющего автомата Мили (ЦУА) на ПЛМ и исследовать разработанную модель.

Порядок выполнения работы

1. Спроектировать автомат Мили на ПЛМ по заданной ГСА:

1.1. Разметить состояния автомата по ГСА.

1.2. Построить граф переходов и прямую таблицу переходов

1.3. Построить обратную структурную таблицу автомата

1.4. Сформировать функции выходов и управления элементами памяти

1.5. Построить функциональную схему автомата

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель ЦУА на устойчивость (возможность зацикливания программы моделирования)

2.4. Исследовать модель ЦУА на адекватность заданной ГСА.

2.5. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 15

Задание на разработку модели ЦУА

1. Разметить состояния автомата по ГСА.

2. Построить граф переходов и прямую таблицу переходов ЦУА.

3. Построить обратную структурную таблицу автомата ЦУА.

4. Сформировать функции выходов и управления элементами памяти ЦУА.

5. Построить функциональную схему ЦУА и реализовать ее, используя программу моделирования электронных схем Multisim.

Задание на проведение экспериментов

1. Для всех комбинаций логических условий X
проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата на устойчивость (возможность зацикливания программы моделирования)

3. Исследовать модель автомата на адекватность заданной ГСА.

4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

- ГСА автомата, заданного преподавателем;
- разметку состояний автомата;
- граф переходов и прямую таблицу переходов;
- обратную структурную таблицу автомата;
- функции выходов и управления элементов памяти;
- функциональную схему автомата;
- результаты исследования автомата Мили на ПЛМ с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата Мили, реализованного на ПЛМ?

2. Какие правила необходимо соблюдать при разметке состояний автомата Мили?

3. Какое различие между структурными схемами автоматов МИЛИ и МУРА на ПЛМ?

4. Какие элементы памяти могут использоваться в автоматах Мили, законы их функционирования?

Лабораторная работа № 16
Исследование автоматов Мура на ПЛМ

Цель работы: получение навыков структурного синтеза, моделирования и исследования цифровых управляющих автоматов Мура на программируемых логических матрицах (ПЛМ).

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

За основу рассмотрим структурную таблицу автомата Мура на жесткой логике. Будем использовать в качестве элементов в памяти состояний D-триггеры. Добавим в таблицу еще один столбец $Ta_m\, a_s$, в который будем записывать функции переходов: $Ta_m\, a_s = \ a_m \wedge\ Xa_m\, a_s$. (таблица 22).

В схеме автомата Мура используются следующие матрицы:

$M_{\&(1)}$ («$И_1$») – для вычисления функций переходов $Ta_m\, a_s$;

$M_{1(1)}$ («$ИЛИ_{(1)}$») – для вычисления функций управления элементами памяти D_2 , D_1 и D_0;

$M_{\&(2)}$ («$И_2$») – дешифратор состояний; так как в автомате Мура y_i зависит только от состояния a_s, функции переходов $Ta_m\, a_s$ нельзя использовать для вычисления y_i;

$M_{1(2)}$ («$ИЛИ_{(2)}$») – для вычисления функций y_i как дизъюнкция выходных сигналов (переменных a_s) с матрицы $M_{\&2}$.

Функции D_2, D_1, D_0 для нашего примера находятся как дизъюнкции соответствующих функций переходов $Ta_m\, a_s$, а y_i зависят только от состояний автомата a_s:

$D_2 = T_1 \vee T_2;\ D_1 = T_4 \vee T_5 \vee T_9;\ D_0 = T_3 \vee T_9;$

$y_1 = y_2 = a_1;\ y_3 = a_2;\ y_4 = y_5 = a_3;\ y_6 = a_4;\ y_7 = a_0.$

Структурная схема автомата Мура на матрицах (простейшая реализация) имеет вид, показанный на рис. 74.

Таблица 22

№ п/п	a_m	Ka_m	a_s	Ka_s	$X\ a_m\ a_s$	Ya_s	$F\ a_m\ a_s$	$T\ a_m\ a_s$
1	a_0	100	a_0	100	$\neg\, x_3$	y_7	D_2	$T_1 = Q_2 \neg\, Q_1 \neg\, Q_0 \neg\, x_3$
2	a_3	000			x_2		D_2	$T_2 = \neg Q_2 \neg\, Q_1 \neg\, Q_0\ x_2$
3	a_0	100	a_1	001	x_3	$y_1,\ y_2$	D_0	$T_3 = Q_2 \neg\, Q_1 \neg\, Q_0\ x_3$
4	a_1	001	a_2	010	x_1	y_3	D_1	$T_4 = \neg Q_2 \neg\, Q_1\ Q_0\ x_1$
5	a_4	011			x_1		D_1	$T_5 = \neg Q_2\ Q_1\ Q_0\ x_1$
6	a_1	001	a_3	000	$\neg\, x_1$	$y_4,\ y_5$	-	$T_6 = \neg Q_2 \neg\, Q_1\ Q_0 \neg x_1$
7	a_2	010			1		-	$T_7 = \neg Q_2\ Q_1 \neg Q_0$
8	a_4	011			$\neg\, x_1$		-	$T_8 = \neg Q_2\ Q_1\ Q_0 \neg x_1$
9	a_3	000	a_4	011	$\neg\, x_2$	y_6	$D_1,\ D_0$	$T_9 = \neg Q_2 \neg Q_1 \neg Q_0 \neg x_2$

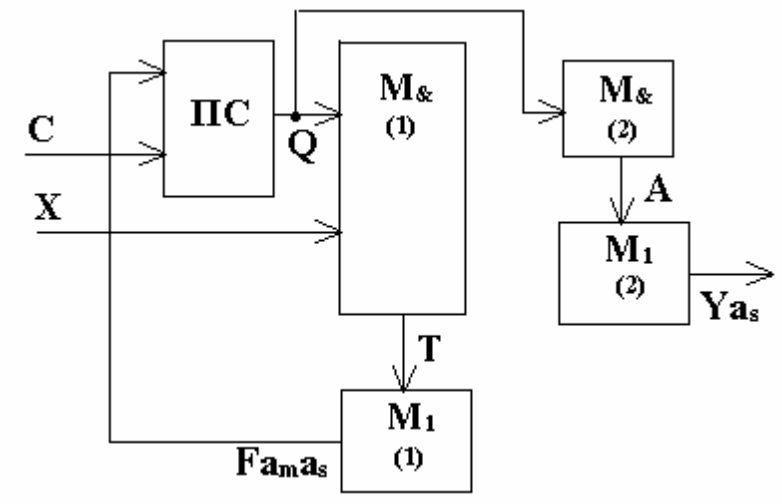


Рис. 74. Структурная схема автомата Мура на ПЛМ

Матрицы $M_{\&(1)}$ и $M_{1(1)}$ выполняют только функцию вычисления $F\ a_m\ a_s$. Матрица $M_{\&(2)}$ на входе имеет коды состояний автомата $Q_2Q_1Q_0$, а на выходе формирует значения переменных a_0, a_1 и т.д., соответствующих состояниям автомата. Матрица $M_{1(2)}$ используется для реализации функций выходов y_i в случае, если они выражены через дизъюнкции a_i.

Функциональная схема автомата приведена на рис. 75. Для упрощения схемы в ней не показаны элементы, используемые для установки автомата в начальное состояние.

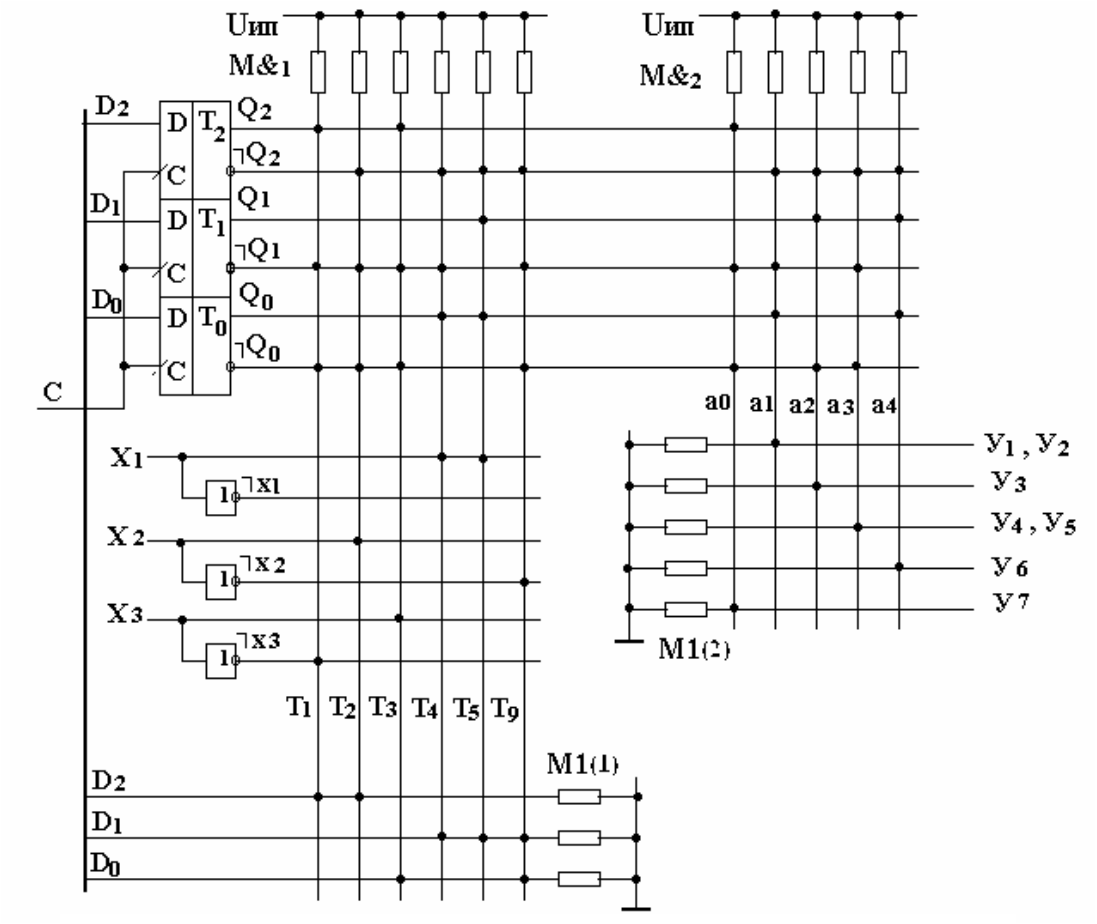


Рис. 75. Функциональная схема автомата Мура на ПЛМ

Как видно из схемы на рис. 75, матрицу М $_{1(2)}$ можно было не использовать, так как в нашем примере каждая функция y_i зависит только от одной переменной a_i. Следует заметить, что использование оптимального кодирования состояний позволяет выразить функции y_i как конъюнкции Q_2 Q_1 Q_0, - кодов состояний автоматов.

Методика выполнения

1. Спроектировать автомат Мура на ПЛМ по заданной ГСА:

1.1. Разметить состояния автомата по ГСА.

1.2. Построить граф переходов и прямую таблицу переходов.

1.3. Построить обратную структурную таблицу автомата.

1.4. Сформировать функции выходов и управления элементами памяти.

1.5. Построить функциональную схему автомата.

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели.

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель ЦУА на устойчивость (возможность зацикливания программы моделирования).

2.4. Исследовать модель ЦУА на адекватность заданной ГСА.

2.5. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 16

Задание на разработку модели ЦУА

1. Разметить состояния автомата по ГСА.

2. Построить граф переходов и прямую таблицу переходов ЦУА.

3. Построить обратную структурную таблицу автомата ЦУА.

4. Сформировать функции выходов и управления элементами памяти ЦУА

5. Построить функциональную схему ЦУА и реализовать ее, используя программу моделирования электронных схем Multisim.

Задание на проведение экспериментов

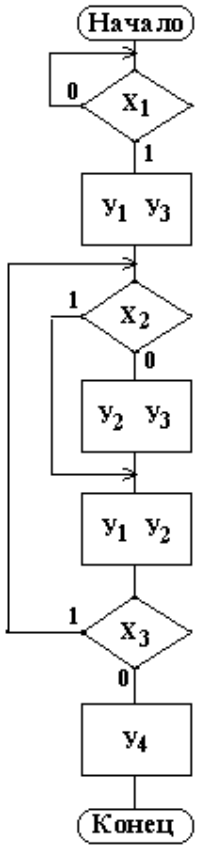
1. Для всех комбинаций логических условий X.

проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата на устойчивость (возможность зацикливания программы моделирования).

3. Исследовать модель автомата на адекватность заданной ГСА.

4. Результаты исследований, алгоритмы и тексты программ представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата Мура, заданную преподавателем;
- разметку состояний автомата;
- граф переходов и прямую таблицу переходов;
- обратную структурную таблицу автомата;
- функции выходов и управления элементов памяти;
- функциональную схему автомата;
- результаты исследования автомата Мура на ПЛМ с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата Мура, реализованного на ПЛМ?
2. Какие правила необходимо соблюдать при разметке состояний автомата Мура?
3. Какое различие между структурными схемами автоматов Мили и Мура на ПЛМ?
4. Какие элементы памяти могут использоваться в автоматах Мура на ПЛМ, законы их функционирования?

Лабораторная работа № 17
Исследование автоматов с программируемой логикой и принудительной адресацией

Цель работы: получение навыков синтеза, микропрограммирования, исследования и моделирования цифровых управляющих автоматов с программируемой логикой и принудительной адресацией.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

Закон функционирования УА с программируемой логикой задается микропрограммой (МП), состоящей из последовательности микрокоманд (МК). МП хранится в ЗУ (запоминающем устройстве любого типа).

Принцип управления по хранимой программе. Напомним, что функция микропрограммного управляющего автомата определяется следующим образом:

- множеством входных сигналов (логических условий):

$X = \{x_1, x_2, \dots x_n\}$;

- множеством выходных сигналов (микрокоманд):

$Y = \{y_1, y_2, \dots y_k\}$;

- микропрограммой (ГСА), задающей порядок следования выходных сигналов У в зависимости от значений входных сигналов Х.

Если автомат Мура описан прямой таблицей переходов, то для каждого состояния a_m известно:

- множество выходных сигналов (микрокоманд) $Y(a_m)$, вырабатываемых автоматом в состоянии a_m;

- множество входных сигналов (логических условий) $X(a_m)$, определяющих все возможные переходы автомата из состояния a_m в состояние a_s;

- множество состояний a_s, в которые может перейти автомат из состояния a_m (обозначим $A(a_m)$).

Для каждого состояния автомата Мура (а значит для каждой операторной вершины ГСА) эту информацию, необходимую и достаточную для описания

закона функционирования автомата, можно закодировать и представить двоичным словом (микрокомандой – не путать с выходным сигналом автомата y_i, который часто также называют микрокомандой!), которое имеет следующую структуру:

№	y_{m1}	y_{m2}	…	y_{mk}	x_{m1}	x_{m2}	…	x_{mn}	a_{m1}	a_{m2}	…	a_{ms}

где:

№ – номер двоичного слова, соответствующего состоянию автомата a_m (или его адрес ЗУ, где будет храниться микропрограмма);

y_{mi} – выходной сигнал (микрокоманда), который должен вырабатывать автомат в данном состоянии a_m; в общем случае таких сигналов *k*; эта часть двоичного слова называется *операционной*;

x_{mi} – входной сигнал (логическое условие), участвующее хотя бы в одном переходе из a_m в a_s; в общем случае таких сигналов *n*; эта часть двоичного слова называется управляющей;

a_{mi} – одно из состояний a_s , в которое может перейти автомат из состояния a_m при определенных значениях логических условий x_{mi}. Поскольку описание одного состояния – это одно двоичное слово, имеющее свой адрес в ЗУ, a_{mi} можно считать адресом состояния ami в ЗУ; в общем случае таких состояний *s*. Если количество логических условий в слове равно *n*, то количество a_{mi} равно 2^n. Эта часть двоичного слова называется адресной.

Воспользуемся прямой таблицей переходов автомата Мура:

Таблица 23

№ п/п	a_m	a_s	Ya_s	Xa_ma_s
1	a_0	a_0	y_7	$\neg x_3$
2		a_1	y_1, y_2	x_3
3	a_1	a_2	y_3	x_1
4		a_3	y_4, y_5	$\neg x_1$
5	a_2	a_3	y_4, y_5	1
6	a_3	a_4	y_6	$\neg x_2$
7		a_0	y_7	x_2
8	a_4	a_2	y_3	x_1
9		a_3	y_4, y_5	$\neg x_1$

Если использовать приведенную выше структуру двоичного слова, то для описания данного автомата достаточно пяти двоичных слов (по числу состояний автомата), имеющих следующий формат:

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$

При этом номер (№) в слово помещать не нужно, так как это адрес слова в ЗУ. В слове под y_{mi} отведено два поля, так как максимальное количество y_{mi} в состоянии a_m равно двум; под x_{mi} – достаточно отвести одно поле, так как в любом переходе из a_m в a_{mi} (или a_s) участвует только одно логическое условие. Под a_{mi} достаточно отвести 2 поля: для адреса перехода при $x_m = 0$ ($a_{x=0}$) и при $x_m = 1$ ($a_{x=1}$). Используя выбранный формат двоичного слова можно составить в виде таблицы 24 содержательную микропрограмму, описывающую работу автомата.

Таблица 24

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$	a_m
0	y_7	-	x_3	0	1	a_0
1	y_1	y_2	x_1	3	2	a_1
2	y_3	-	x_3	3	3	a_2
3	y_4	y_5	x_2	4	0	a_3
4	y_6	-	x_1	3	2	a_4

В этой таблице каждая строка соответствует одному из состояний автомата (столбец a_m); № – соответствует адресу слова, описывающего это состояние в ЗУ. В колонках y_{m1} и y_{m2}записаны микрокоманды, вырабатываемые автоматом в состоянии a_m, в колонке x_m записаны логические условия, определяющие переходы из состояния a_m в состояния, адреса которых в ЗУ записаны в колонках $a_{x=0}$ и $a_{x=1}$, в колонке a_m записаны пояснения: какому состоянию автомата соответствует данная строка.

Каждая из колонок этой таблицы (кроме a_m) должна быть представлена двоичными кодами. Количество двоичных разрядов для каждой колонки можно выбрать таким:

– y_{m1} – 3 разряда: здесь будем записывать номер y_{m1} (от 1 до 7);

– y_{m2} – 3 разряда: здесь будем записывать номер y_{m2} (от 1 до 7);

– x_m– 2 разряда: здесь будем записывать номер x_m (от 1 до 3);

– $a_{x=0}$– 3 разряда: здесь будем записывать адрес $a_{x=0}$, по которому в автомате должен произойти переход к следующему слову при $x_m = 0$ (от 0 до 4);

– $a_{x=1}$– 3 разряда: здесь будем записывать адрес $a_{x=1}$, по которому в автомате должен произойти переход к следующему слову при $x_m = 1$ (от 0 до 4).

Таким образом, разрядность двоичного слова в нашем примере равна 14. Закодированная микропрограмма имеет вид, приведенный в таблице 25.

Таблица 25

№	y_{m1}	y_{m2}	x_m	$a_{x=0}$	$a_{x=1}$
0	111	000	11	000	001
1	001	010	01	011	010
2	011	000	11	011	011
3	100	101	10	100	000
4	110	000	01	011	010

В каждой колонке записаны двоичные числа в соответствии с принятыми выше соглашениями. Требуемый объем памяти – 70 бит. Рассмотрим подробнее 2-ю строку в таблицах 15 и 16, в которых в строке № 2 описывается состояние автомата a_2. Из этого состояния – безусловный переход автомата в состояние a_3. Поэтому в столбце x_mможет быть записан номер любой переменной x_m, а в обоих столбцах $a_{x=0}$ и $a_{x=1}$одинаковый адрес перехода. Так как записанный в столбце x_m обязательно имеет одно из значений: 0 или 1, то переход будет безусловным по адресу $a_{x=0} = a_{x=1}$.

Схема, обеспечивающая работу с закодированной микропрограммой, должна содержать следующие функциональные узлы:

– память для хранения микропрограммы (ЗУ) объемом не менее пяти слов; чтение слова из памяти осуществляется по адресу, подаваемому на адресный вход ЗУ; разрядность слов – 14 бит;

– схему, преобразующую двоичный код y_{m1}и y_{m2}в унитарный код, соответствующий закодированному y_i; эта схема – дешифратор (DCy);

– мультиплексор (MSx), выбирающий из множества входных сигналов X всего один, номер которого записан в столбце x_m;

– мультиплексор (MSa), выбирающий из слова адрес $a_{x=0}$ (если выбранное мультиплексором MSx значение $x_m = 0$) или адрес $a_{x=1}$ (если выбранное мультиплексором MSx значение $x_m = 1$);

– регистр адреса (RG), в который необходимо по импульсу синхронизации записать выбранный мультиплексором MSa адрес и подать его на адресный вход ЗУ, из которого будет считано следующее слово (иначе – переход в следующее состояние).

Пронумеруем разряды двоичного слова с 1 по 14 (как показано в таблице); номера разрядов будут соответствовать номерам выходов ЗУ.

y_{m1}			y_{m2}			x_m		$a_{x=0}$			$a_{x=1}$		
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Функциональная схема автомата приведена на рис. 76.

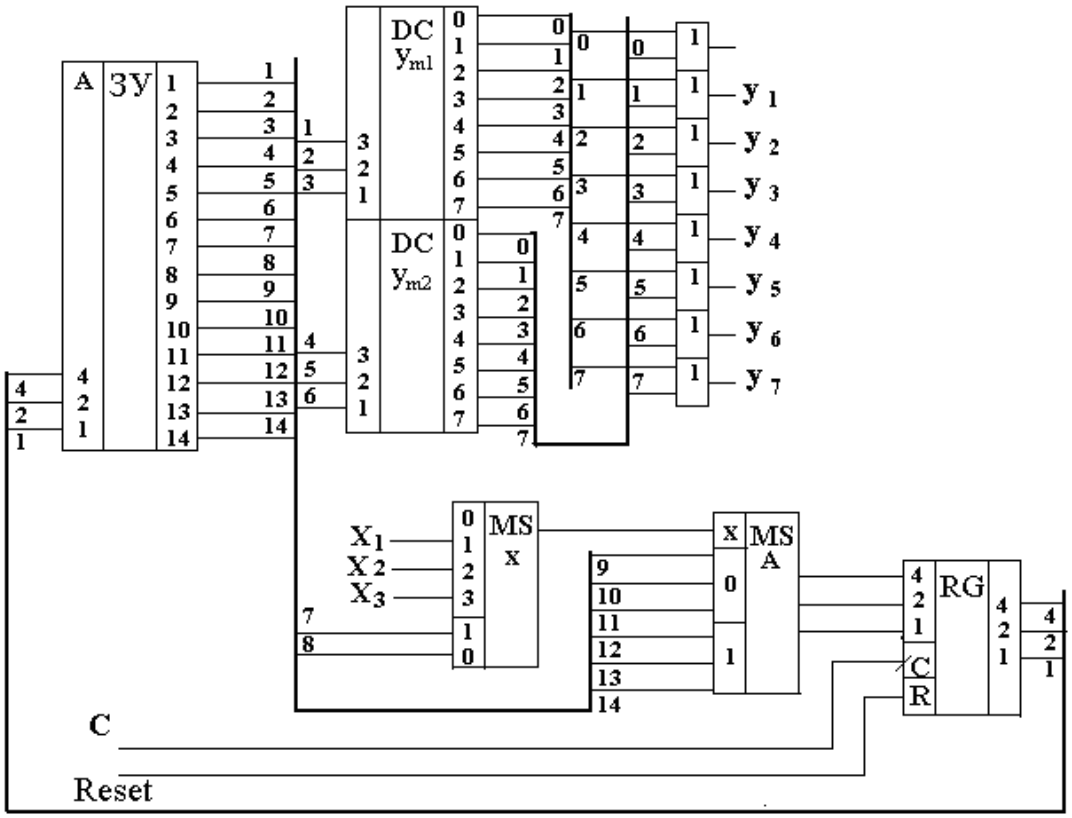


Рис. 76. Функциональная схема автомата

На этой схеме кроме описанных выше элементов имеется 8 дизъюнкторов, объединяющих одноименные выходы дешифраторов DCy_{m1} и DCy_{m2}. Эти элементы необходимы в случае, если одна и та же микрокоманда y_i будет записана в разных строках микропрограммы в разных столбцах y_{m1} и y_{m2}.

Автомат работает следующим образом. Сигнал Reset=1 устанавливает значение «0» во всех разрядах регистра адреса RG. Код (или двоичное число) A=000 с выхода RG подается на адресный вход ЗУ. На выходах ЗУ (разряды 1...14) «появляется» слово, записанное по адресу 0. В этом слове разряды имеют следующий смысл:

– разряды 1, 2, 3 – подаются на вход DCy_{m1} и, поскольку в этих разрядах записано число 111 (7), на выходе «7» DCy_{m1} формируется 1, а значит и выходной сигнал $y_7 = 1$;

– разряды 4, 5, 6 – подаются на вход DCy_{m2} и, поскольку в этих разрядах записано число 000 (0), (которое мы не использовали для кодирования y_i) на выходе «0» DCy_{m1} формируется «1», но ни на каких либо других выходах y_i эта «1» не появится;

– разряды 7, 8 – подаются на управляющие входы мультиплексора MSx; в этих разрядах записано число 11 (3). Значение логического условия x_3

появляется на выход MSx;

– мультиплексор MSa, в зависимости от значения x_3 («0» или «1») подает на входы регистра адреса RG либо адрес перехода, записанный в разрядах 9,10,11 (если $x_3 = 0$), либо адрес перехода, записанный в разрядах 12,13,14 (если $x_3 = 1$);

– при поступлении на вход C импульса синхронизации, этот адрес запишется в регистр RG и с его выходов поступит на адресный вход ЗУ, на выходе которого появится слово, записанное по этому адресу.

Таким образом за один цикл (время между синхроимпульсами C) автомат считает из ЗУ некоторое двоичное слово, выработает микрокоманды y_m, проанализирует входные сигналы x_m, и в зависимости от их значения выберет адрес следующего двоичного слова в ЗУ.

Спроектированный нами интуитивно автомат содержит много избыточности и был приведен в качестве примера, как естественное воплощение наших рассуждений. Недостатками такого подхода являются, прежде всего, сложность схемы (при достаточно простых функциях автомата) и не оптимальная длина микрокоманды (двоичного слова), что приводит к неэффективному использованию памяти ЗУ.

Рассмотрим некоторые приемы, позволяющие сделать автомат с программируемой логикой оптимальным.

Оптимизация автоматов с программируемой логикой. Выходные сигналы y_i (микрокоманды), вырабатываемые управляющим автоматом, сгруппированы в операторных вершинах ГСА в виде наборов, инициирующих выполнение операционным автоматом (ОА) совместимых микроопераций. Совместимые микрооперации – это такие, которые могут быть выполнены ОА одновременно.

Пусть автомат должен генерировать n различных наборов микрокоманд из множества $Y = \{ y_1, y_2 \dots y_k\}$ обозначим *i*-ый набор как:

$$Y_i = \{ y_{i1}, y_{i2} \dots y_{ik}\}$$

Составим матрицу M_{N*K}, где N – количество различных наборов микрокоманд, а К – количество различных микрокоманд y_i, вырабатываемых автоматом. Элемент матрицы $m_{ij} = 1$, если микрокоманда y_j входит в набор Y_i, а если не входит, то $m_{ij} = 0$. Если строки матрицы М (а значит и наборы Y_i) не пересекаются, то в операционной части двоичного слова можно обойтись одним полем, в которое записывается номер набора Y_i. Этот номер будет подаваться на один дешифратор DCy, на выходе которого будут сформированы микрокоманды y_i.

В нашем примере все микрокоманды y_i можно представить в виде следующих наборов:

$$Y_0 = \{ y_7\};\ \ Y_1 = \{ y_1, y_2\};\ Y_2 = \{ y_3\};\ \ Y_3 = \{ y_4, y_5\};\ Y_4 = \{ y_6\};$$

Матрицу М представим в виде таблицы:

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
Y_0							1
Y_1	1	1					
Y_2			1				
Y_3				1	1		
Y_4						1	

Из матрицы видно, что пересечение строк для всех пар Y_i и Y_j (при $i \neq j$) пусто. Закодируем наборы Y_i следующим произвольным образом:

К(У₀) = 000; *К(У₁)* = 001; *К(У₂)* = 010; *К(У₃)* = 011; *К(У₄)* = 100, и изменим формат микрокоманды:

№	$K(Y_i)$	x_m	$a_{x=0}$	$a_{x=1}$

Количество разрядов двоичного слова при этом уменьшится до 11, а значит и уменьшится емкость необходимой памяти ЗУ до 55 бит. Пронумеруем разряды двоичного слова с 1 по 11 (как показано в таблице); номера разрядов будут соответствовать номерам выходов ЗУ на функциональной схеме автомата (рис. 40).

$K(Y_i)$			x_m		$a_{x=0}$			$a_{x=1}$		
1	2	3	4	5	6	7	8	9	10	11

Схема на рис. 77 незначительно отличается от схемы на рис. 76 по своему функционированию. Отличие – один дешифратор микрокоманд Y_i, назначение которого описано выше.

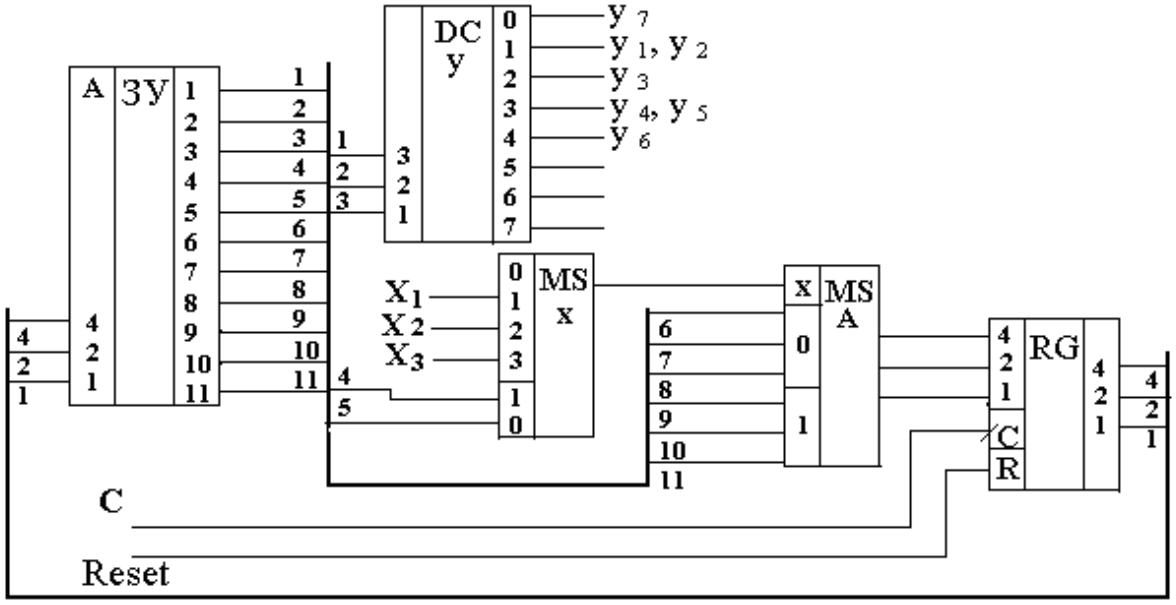


Рис. 77. Модифицированный автомат

Если наборы микрокоманд в матрице М пересекаются, то описанный в последнем примере подход к их формированию невозможен. Поэтому можно поступить следующим образом.

1. Если общее количество микрокоманд Y_i невелико, то в операционной части двоичного слова можно выделить по одному разряду на каждую микрокоманду. В этом случае отпадает необходимость в дешифраторе DCy, так как в соответствующие разряды операционной части двоичного слова записываются значения (0 или 1) для всех Y_i. Таким образом сигналы Y_i берутся непосредственно с соответствующих выходов ЗУ. Если использовать такой подход для нашего примера, то разрядность двоичного слова увеличится до 15 бит, емкость ЗУ – до 75 бит, но из схемы автомата исчезнет дешифратор DCy. Формат двоичного слова приведен ниже.

y_1	y_2	y_3	y_4	y_5	y_6	y_7	x_m		$a_{x=0}$			$a_{x=1}$		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2. Если общее количество микрокоманд Y_i достаточно велико и использование приема, описанного в п.1, приведет к значительному увеличению длины операционной части двоичного слова, то можно поступить следующим образом: закодировать пересекающиеся наборы микрокоманд минимальным образом (например, как в предыдущих примерах), в операционной части двоичного слова записать коды этих наборов, а в схеме автомата вместо DCy использовать дополнительное ЗУу. Разрядность адреса этого ЗУу равна разрядности кодов наборов микрокоманд, разрядность слов, записанных в ЗУу, равна количеству Y_i в автомате. Каждый разряд слова в этом ЗУу соответствует определенному Y_i и в него записывается «0» или «1» в зависимости от значения соответствующего Y_i в данном наборе микрокоманд. Например, если использовать данный прием для предыдущего примера, то дополнительное ЗУу должно иметь разрядность адреса равное 3, разрядность слов – 5. Вид такого ЗУ приведен на рис. 78.

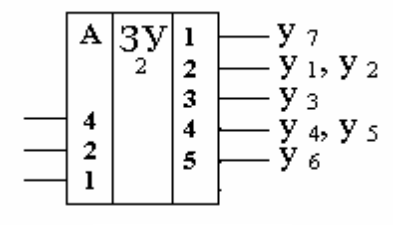


Рис. 78. Микросхема ЗУу

Если использовать кодирование наборов микрокоманд такое же, как в предыдущих примерах, то таблица программирования этого ЗУ будет иметь вид:

	y_7	y_1, y_2	y_3	y_4, y_5	y_6
Адр\бит	1	2	3	4	5
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1

Есть различные способы уменьшения длины адресной части в двоичном слове. Один из них заключается в следующем. В адресной части слова записывают адрес перехода при $X_i = 0$ (Адр $x_i = 0$), а если $X_i = 1$, то адрес перехода Адр вычисляется так:

Адр = Адр $_{xi=0}$ + 1

Если использовать логическое значение переменной X_i как арифметическое (0 или 1), то адрес перехода можно вычислить так:

Адр = Адр $_{xi=0}$ + X_i.

Таким образом, сократив вдвое длину адресной части двоичного слова, мы не потеряли возможность осуществлять условный переход в автомате с естественной адресацией. Однако при этом получили некоторое неудобство: двоичные слова, к которым происходит переход, всегда находятся в ЗУ в соседних ячейках, так как адреса у них различаются всегда на 1. Поэтому в микропрограмме автомата с описанным способом адресации может появиться избыточность в виде дублирования некоторых слов и безусловных переходов.

Воспользуемся прямой таблицей переходов автомата Мура из предыдущего примера (таблица 26) и форматом микрокоманды из первого примера текущего раздела, в котором в адресной части оставим только одно поле $a_{x=0}$.

Таблица 26

№ п/п	a_m	a_s	Ya_s	$Xa_m a_s$
1	a_0	a_0	y_7	$\neg x_3$
2		a_1	y_1, y_2	x_3
3	a_1	a_2	y_3	x_1
4		a_3	y_4, y_5	$\neg x_1$
5	a_2	a_3	y_4, y_5	1
6	a_3	a_4	y_6	$\neg x_2$
7		a_0	y_7	x_2
8	a_4	a_2	y_3	x_1
9		a_3	y_4, y_5	$\neg x_1$

Формат микрокоманды:

$K(Y_i)$			x_m		$a_{x=0}$		
1	2	3	4	5	6	7	8

Используем кодирование наборов микрокоманд и дополнительное ЗУ. Микропрограмма с комментариями приведена ниже в таблице 27:

$Y_0 = \{y_7\}$; $Y_1 = \{y_1, y_2\}$; $Y_2 = \{y_3\}$; $Y_3 = \{y_4, y_5\}$; $Y_4 = \{y_6\}$;

К(У₀) = 000; *К(У₁)* = 001; *К(У₂)* = 010; *К(У₃)* = 011; *К(У₄)* = 100.

Таблица 27

Адр	Комментарий / разряды*	1	2	3	4	5	6	7	8
0	У₀; IF *X₃* = 0 THEN GOTO 0 / 1;	0	0	0	1	1	0	0	0
1	У₁; IF ¬*X₁* = 0 THEN GOTO 2 / 3;	0	0	1	0	1	0	1	0
2	У₂; IF *X₀* = 0 THEN GOTO 3 / 4;	0	1	0	0	0	0	1	1
3	У₃; IF *X₂* = 0 THEN GOTO 4 / 5;	0	1	1	1	0	1	0	0
4	У₄; IF ¬*X₁* = 0 THEN GOTO 2 / 3;	1	0	0	0	1	0	1	0
5	У₀; IF *X₃* = 0 THEN GOTO 0 / 1;	0	0	0	1	1	0	0	0

*В таблице через символ «/» записан альтернативный адрес перехода.

Для организации безусловного перехода в этом примере введено дополнительное логическое условие X_0. В схеме автомата на вход X_0 подадим значение «0», а значит переход в этом случае будет всегда по адресу ах=0, записанному в микрокоманде.

Из таблицы 27 видно, что на состояние a_0 пришлось использовать два слова в ЗУ, тем не менее общий объем ЗУ в нашем примере 48 бит (меньше, чем у двухадресной микрокоманды).

Функциональная схема автомата приведена на рис. 79.

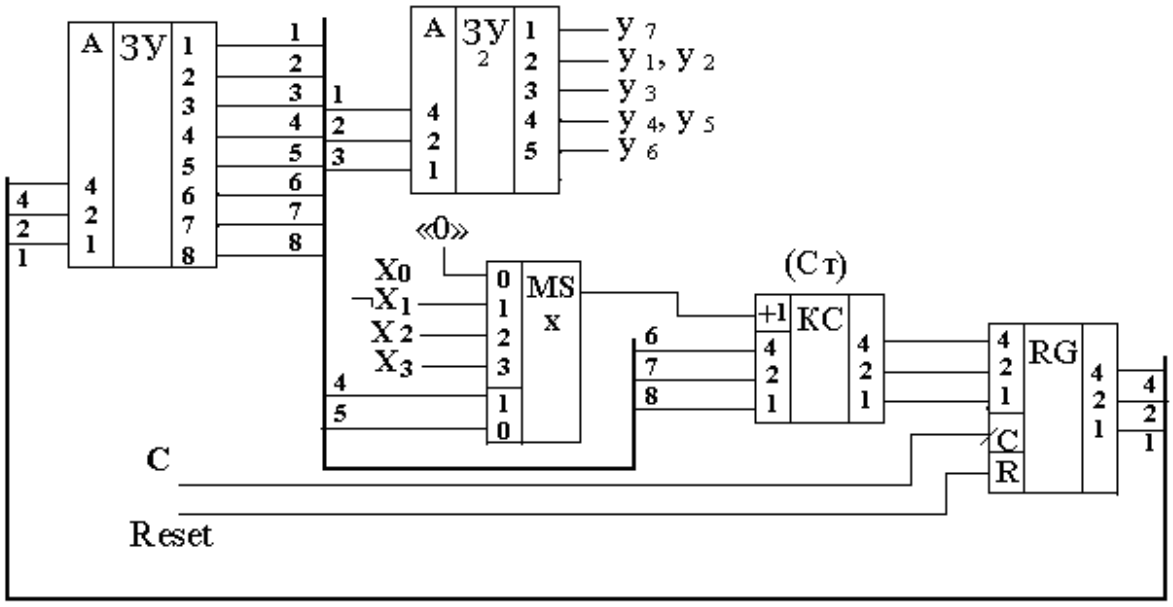


Рис. 79. Функциональная схема автомата

Отличие этой схемы от рис. 77 состоит в использовании вместо мультиплексора адреса MSa комбинационного счетчика (*Cm*). Комбинационная схема (КС), обозначенная как *Cm*, выполняет функцию: *Cm:=A + Xi*, где:

Cm – двоичное число на выходе КС;

A – адрес перехода при *Xi* = 0 (из микрокоманды);

Xi – значение логического условия *Xi*, выбранного мультиплексором MSx в соответствии с кодом, записанным в разрядах 4 и 5 микрокоманды и интерпретируемым как число 0 или 1.

Функционирование схемы на рис. 79 практически не отличается от описанной выше схемы на рис. 77.

Используя описанный выше прием формирования адресов переходов, можно еще на один разряд сократить адресную часть микрокоманды. Так как адреса переходов всегда отличаются на 1, то в качестве младшего разряда адреса очередной микрокоманды можно использовать значение логического условия *Xi*. Формат микрокоманды в этом случае имеет вид, приведенный в таблице. Старшие разряды адресной части (a_C) записаны в микрокоманде, а младший разряд формируется из значения логического условия *Xi*:

$K(Y_i)$			x_m		a_C		*Xi*
1	2	3	4	5	6	7	8

В таблице 28 приведена микропрограмма (на основе предыдущего примера), в которой функцию младшего разряда адреса перехода выполняет соответствующее логическое условие *Xi*, а на рис. 80 – функциональная схема автомата.

Таблица 28

Адр	Комментарий / разряды	1	2	3	4	5	6	7	8
0	a_0; IF X_3 = 0 THEN GOTO 0/1;	0	0	0	1	1	0	0	X_3
1	a_1; IF $\neg X_1$ = 0 THEN GOTO 2/3;	0	0	1	0	1	0	1	$\neg X_1$
2	a_2; IF X_0 =0 THEN GOTO 6/7;	0	1	0	0	0	1	1	X_0
3	a_3; IF X_2 = 0 THEN GOTO 4/5;	0	1	1	1	0	1	0	X_2
4	a_4; IF $\neg X_1$ = 0 THEN GOTO 2/3;	1	0	0	0	1	0	1	$\neg X_1$
5	a_0; IF X_3 = 0 THEN GOTO 0/1;	0	0	0	1	1	0	0	X_3
6	a_3; IF X_2 = 0 THEN GOTO 4/5;	0	0	0	1	1	0	0	X_2

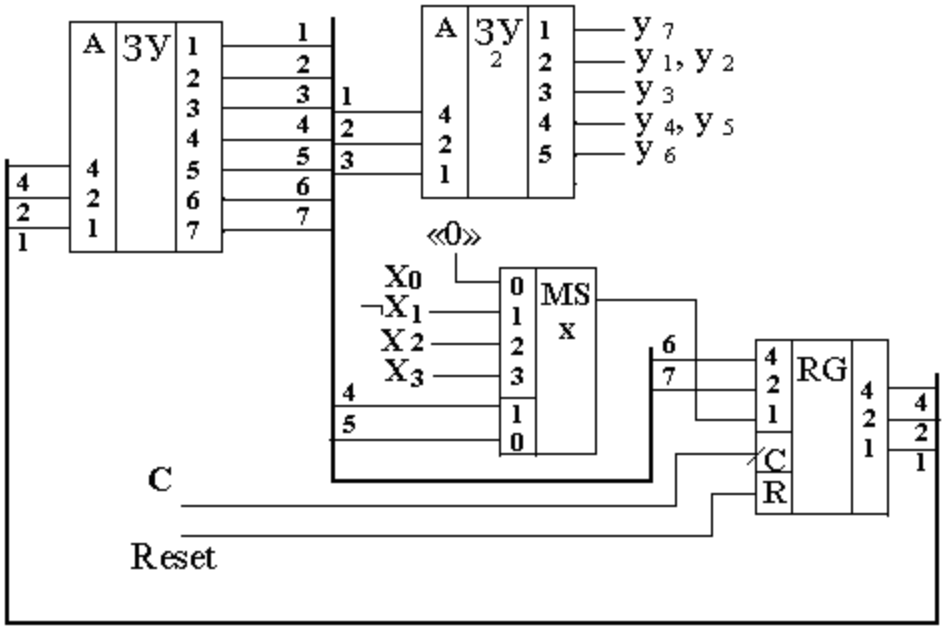


Рис. 80. Оптимизированный автомат

Схема не содержит комбинационного счетчика КС, однако такой формат микрокоманды накладывает на микропрограмму ограничение: при $X_i = 0$ – адрес перехода всегда четный, а при $X_i = 1$ – всегда нечетный.

Методика выполнения

1. Спроектировать автомат программируемой логикой и принудительной адресацией по заданной ГСА:

1.1. Разработать содержательную микропрограмму работы автомата по ГСА.

1.2. Разработать формат микрокоманды.

1.3. Разработать закодированную микропрограмму работы автомата по ГСА.

1.4. Разработать функциональную схему автомата с программируемой логикой и принудительной адресацией.

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели.

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель ЦУА на адекватность заданной ГСА.

2.4. Результаты исследований представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 17

Задание на разработку модели ЦУА

1. Разработать содержательную микропрограмму работы автомата по ГСА.

2. Разработать формат микрокоманды.

3. Разработать закодированную микропрограмму работы автомата по ГСА.

4. Разработать функциональную схему автомата с программируемой логикой и принудительной адресацией

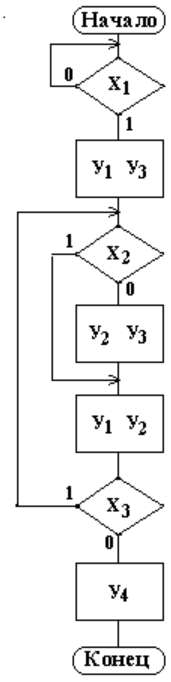
5. Используя программу моделирования электронных схем Multisim, собрать схему разработанного автомата.

Задание на проведение экспериментов

1. Для всех комбинаций логических условий X.
проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата на адекватность заданной ГСА.

3. Результаты исследований представить в виде отчета.



Подготовка отчета

Отчет должен содержать:

- ГСА автомата с программируемой логикой, заданную преподавателем;
- форматы микрокоманд;
- микропрограмму функционирования автомата;
- функциональную схему автомата;
- результаты исследования автомата с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона функционирования автомата с программируемой логикой и принудительной адресацией?
2. Какие правила необходимо соблюдать при описании закона функционирования автомата с программируемой логикой и принудительной адресацией?
3. Какое различие между структурными автоматами и автоматами с программируемой логикой?
4. Какие элементы используются в автоматах с программируемой логикой и принудительной адресацией?

Лабораторная работа № 18
Исследование автоматов с программируемой логикой и естественной адресацией

Цель работы: получение навыков синтеза, микропрограммирования, исследования и моделирования цифровых управляющих автоматов с программируемой логикой и естественной адресацией.

Материалы, оборудование, программное обеспечение:

Для выполнения работы требуется IBM PC-совместимый компьютер и программа схемотехнического моделирования Multisim. Оборудование расположено в лаборатории кафедры ауд. 143а.

Критерии положительной оценки:

Для успешной сдачи лабораторной работы должны быть выполнены все задания и продемонстрированы полученные навыки.

Планируемое время выполнения: 4 академических часа.

Время самостоятельной подготовки: 2 академических часа.

Теоретические сведения

Формат микрокоманд (двоичных слов) при естественной адресации может быть следующим. Операционная микрокоманда (слово) содержит два поля: поле признака вида микрокоманды P (операционная или управляющая; пусть для операционной микрокоманды P = 0) и поле, в котором любым из возможных способов записаны значения y_i , которые должен вырабатывать автомат.

Управляющая микрокоманда (слово) содержит три поля: поле признака P (пусть для управляющей микрокоманды P =1), поле логического условия (может содержать № логического условия X_i) и поле адреса перехода при $X_i = 1$.

Формат операционного слова:

0	Y_i

Формат управляющего слова:

0	X_i	Адр $a_{xi=1}$

Поскольку для хранения и операционных и управляющих слов используется одно и то же ЗУ, разрядность слов должна быть одинаковой.

Возьмем, для примера, ГСА управляющего автомата, приведенную на рис. 56, и запишем алгоритм работы автомата, используя следующие конструкции:

Оператор присваивания вида: y_i:= 1, для описания операторных вершин ГСА;

Оператор условного перехода вида: IF *Xi* = 1 THEN GOTO *Axi*=1, для

описания условных вершин ГСА.

Алгоритм, представленный в виде таблицы, напоминает программу на языке BASIC. Для более эффективного представления, в строках с адресами 1 и 3 перед логическими условиями X_3 и X_1 поставлен знак инверсии (¬), что позволяет сократить количество строк в записи алгоритма. В строке 8 использована конструкция: GOTO 3 – безусловный переход к строке 3. Для реализации безусловного перехода введем дополнительное логическое условие, например, X_0, а в схеме автомата предусмотрим, чтобы всегда $X_0 = 1$. В этом случае, конструкцию «GOTO 3» можно заменить на конструкцию: IF $X0$ = 1 THEN GOTO 3.

В таблице 29 приведен пример описания алгоритма работы автомата.

Таблица 29

Адрес	Оператор
0	$У_7$:= 1;
1	IF X_3 = 1 THEN GOTO 0;
2	$У_1$:= 1; $У_2$:= 1;
3	IF X_1 = 1 THEN GOTO 5;
4	$У_3$:= 1;
5	$У_4$:= 1; $У_5$:= 1;
6	IF X_2 = 1 THEN GOTO 0;
7	$У_6$:= 1;
8	GOTO 3; (IF X_0 = 1 THEN GOTO 3;)

Из таблицы 29 видно, что закодированная микропрограмма должна содержать 9 строк с адресами в ЗУ от 0 до 8. Поэтому адресное поле управляющей микрокоманды (в которое будем записывать адрес перехода – А) должно содержать не менее четырех разрядов (возьмем ровно 4), а поле логического условия (в которое будем записывать номер логического условия – Нх) должно содержать не менее двух разрядов (так как в нашем примере – 4 различных логических условия). В таблице 30 показано соответствие разрядов управляющей микрокоманды и информации, которую она содержит.

Таблица 30

Разряды слова	1	2	3	4	5	6	7
Значения	1	$Нх_1$	$Нх_0$	$А_3$	$А_2$	$А_1$	$А_0$

В первом разряде всегда записывается 1 – признак управляющего слова. В разрядах 2 и 3 записывается номер логического условия Нх в виде двоичного числа: $Нх_1$ $Нх_0$, где индексы 0 и 1 – указывают младший и старший разряды

числа Нх. В разрядах 4, 5, 6 и 7 записывается адрес перехода А в виде двоичного числа A_3 A_2 A_1 A_0, где индексы А также указывают старшинство разрядов в числе А.

В таблице 31 показан формат операционного слова (микрокоманды) для нашего примера.

Таблица 31

Разряды слова	1	2	3	4	5	6	7
Значения	0	y_7	y_1, y_2	y_3	y_4, y_5	y_6	-

В первом разряде всегда записывается 0 – признак операционного слова. В остальных разрядах будем записывать 1 или 0, в зависимости от того, какое значение имеют микрокоманды y_i в данном слове. В следующей таблице 2, в колонках, обозначенных номерами разрядов двоичного слова от 1 до 7, приведена закодированная микропрограмма управляющего автомата с естественной адресацией. Микропрограмма работы автомата представлена в таблице 32. Объем памяти ЗУ – 63 бита.

Таблица 32

Адрес	Оператор	Разряды слова						
		1	2	3	4	5	6	7
0	Y_7 := 1;	0	1	0	0	0	0	0
1	IF ¬ X_3 = 1 THEN GOTO 0;	1	1	1	0	0	0	0
2	Y_1 := 1; Y_2 := 1;	0	0	1	0	0	0	0
3	IF ¬ X_1 = 1 THEN GOTO 5;	1	0	1	0	1	0	1
4	Y_3 := 1;	0	0	0	1	0	0	0
5	Y_4 := 1; Y_5 := 1;	0	0	0	0	1	0	0
6	IF X_2 = 1 THEN GOTO 0;	1	1	0	0	0	0	0
7	Y_6 := 1;	0	0	0	0	0	1	0
8	IF X_0 = 1 THEN GOTO 3;	1	0	0	0	0	1	1

Схема, обеспечивающая работу с закодированной микропрограммой автомата с естественной адресацией, должна содержать следующие функциональные узлы:

– память для хранения микропрограммы (ЗУ) объемом не менее девяти слов; чтение слова из памяти осуществляется по адресу (четырехразрядному), подаваемому на адресный вход ЗУ; разрядность слов – 7 бит;

– схему, обрабатывающую операционную микрокоманду – преобразующую двоичный код в разрядах слова с 2-го по 6-ой в сигналы y_i на

выходе автомата. Эта схема может быть реализована пятью элементами «И»;

– мультиплексор (MSx), выбирающий из множества входных сигналов X всего один, номер которого записан в разрядах 2 и 3 управляющего слова; мультиплексор должен иметь дополнительный вход разрешения V; при V=1 – режим нормальной работы MSx; при V=0 – режим запрета: на выходе MSx значение «0» независимо от сигналов на его входах;

– четырехразрядный двоичный счетчик (СТ), имеющий два режима работы: параллельная загрузка адреса перехода (из разрядов 4…7 управляющего слова) в счетчик (когда выполняется условие *Xi* = 1) и режим счета: СТ := СТ+1 – во всех остальных случаях. Для этого счетчик СТ имеет вход управления V; при V = 0 – режим счета, при V = 1 – режим параллельной загрузки. У счетчика СТ должны быть еще один вход R (Reset, для установки СТ в начальное состояние СТ := 0) и вход синхронизации С.

Функциональная схема автомата с программируемой логикой и естественной адресацией приведена на рис. 81.

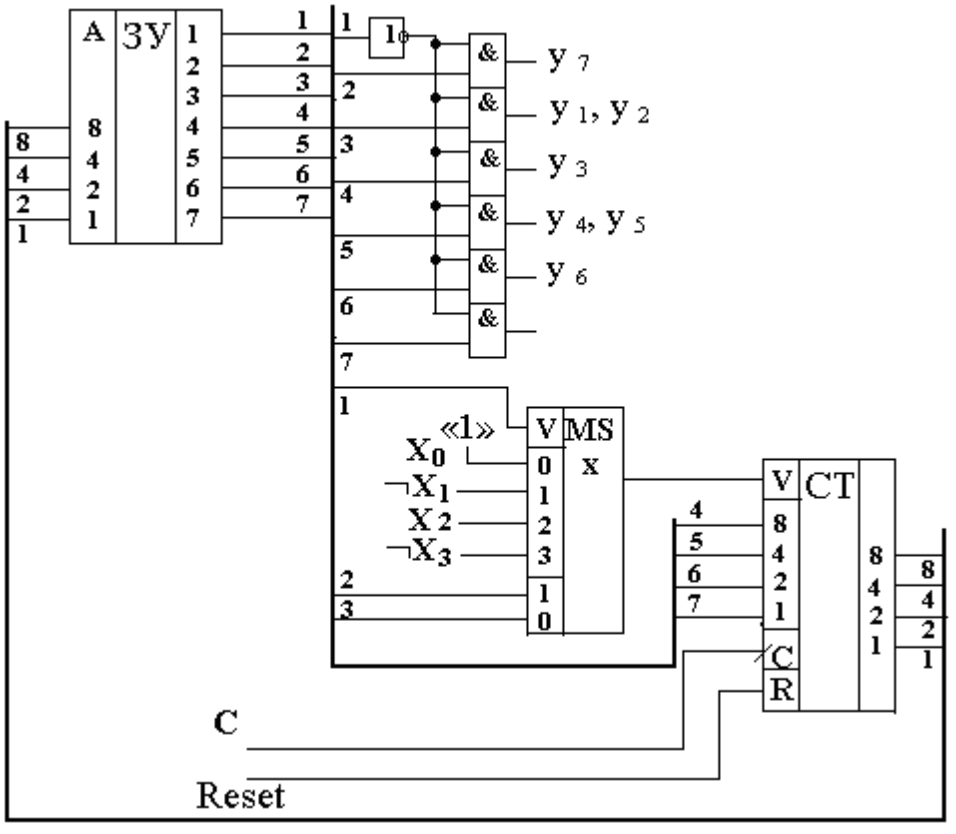


Рис. 81. Автомат с программируемой логикой и естественной адресацией

Автомат работает следующим образом. Сигнал Reset = 1 устанавливает значение «0» во всех разрядах двоичного счетчика СТ. Код (или двоичное число) А=0000 с выхода СТ подается на адресный вход ЗУ. На выходах ЗУ (разряды 1…7) «появляется» слово, записанное по адресу А=0 (см. таблицу 32).

В нашем примере по адресу A=0 записана операционная микрокоманда, в которой:

– первый разряд имеет значение 0 (признак операционной микрокоманды), поэтому на выходе мультиплексора MSx так же значение 0, а счетчик СТ находится в режиме счета; инвертированное значение первого разряда (1) подается на входы элементов «И», выполняющих функцию электронных ключей, и на выходах этих элементов – значения 2…7 разрядов микрокоманды, т. е. – значения y_i ; в нашем примере – $У_7$;

– так как счетчик в режиме счета (V=0), то при поступлении на вход С счетчика СТ импульса синхронизации, его состояние изменится: СТ: = СТ+1, а значит изменится на +1 адрес: A=0001; этот адрес поступит на адресный вход ЗУ, на выходе которого появится слово, записанное по данному адресу;

– в нашем примере слово по адресу A=0001 – управляющая микрокоманда; первый разряд в этом слове имеет значение 1 (признак управляющей микрокоманды); инвертированное значение первого разряда (0) подается на входы элементов «И», и на выходах всех этих элементов – значения «0» (все $y_i = 0$); на разрешающий вход мультиплексора MSx подается значение «1»: MSx находится в режиме нормальной работы;

– значения разрядов 2 и 3 – подаются на управляющие входы мультиплексора MSx; в нашем примере в этих разрядах записано двоичное число 11 (3); значение логического условия X_3 появляется на выходе MSx;

– если $\neg X_3 = 1$ ($X_3 = 0$), то на вход V счетчика СТ подается «1» (режим параллельной загрузки счетчика) и, при подаче на его вход С импульса синхронизации, в СТ запишутся значения разрядов 4..7 микрокоманды (в нашем примере – это адрес 0000); из ЗУ будет считано слово по этому адресу;

если $\neg X_3 = 0$ ($X_3 = 1$), то на вход V счетчика СТ подается «0» (режим счета) и, при подаче на его вход С импульса синхронизации, состояние СТ изменится: СТ := СТ+1; в нашем примере СТ:= 0001+1=0010; запишутся значения разрядов 4..7 микрокоманды (в нашем примере – это адрес 0000); из ЗУ будет считано слово по адресу 0010 (2); и т.д.

Методика выполнения

1. Спроектировать автомат программируемой логикой и естественной адресацией по заданной ГСА:

1.1. Разработать содержательную микропрограмму работы автомата по ГСА.

1.2. Разработать формат микрокоманды.

1.3. Разработать закодированную микропрограмму работы автомата по ГСА.

1.4. Разработать функциональную схему автомата с программируемой логикой и естественной адресацией.

2. Используя программу моделирования электронных схем Multisim, собрать схему автомата и исследовать ее работу в соответствии с заданной преподавателем ГСА.

2.1. Исследовать поведение автомата на построенной модели

2.2. Для всех комбинаций логических условий X проверить правильность переходов по ГСА и формирования автоматом микрокоманд У.

2.3. Исследовать модель ЦУА на адекватность заданной ГСА.

2.4. Результаты исследований представить в виде отчета.

Пример индивидуального задания к лабораторной работе № 16

Задание на разработку модели ЦУА

1. Разработать содержательную микропрограмму работы автомата по ГСА.

2. Разработать формат микрокоманды

3. Разработать закодированную микропрограмму работы автомата по ГСА.

4. Разработать функциональную схему автомата с программируемой логикой и естественной адресацией

5. Используя программу моделирования электронных схем Multisim, собрать схему разработанного автомата.

Задание на проведение экспериментов

1. Для всех комбинаций логических условий X
проверить правильность переходов ЦУА по ГСА.

2. Исследовать модель автомата на адекватность заданной ГСА.

3. Результаты исследований представить в виде отчета.

Подготовка отчета

Отчет должен содержать:

- ГСА автомата с программируемой логикой и принудительной адресацией, заданную преподавателем;
- форматы микрокоманд;
- микропрограмму функционирования автомата;
- функциональную схему автомата;
- результаты исследования автомата с помощью программы моделирования электронных схем Multisim.

Контрольные вопросы

1. Какие объекты и функции используются для описания закона

функционирования автомата с программируемой логикой и естественной адресацией?

2. Какие правила необходимо соблюдать при описании закона функционирования автомата с программируемой логикой и естественной адресацией?

3. Какое различие между структурными автоматами и автоматами с программируемой логикой?

4. Какие элементы используются в автоматах с программируемой логикой и естественной адресацией?

ЛИТЕРАТУРА

1. Парфенкин, А. И. Схемотехника : учеб. пособие / А. И. Парфенкин, О. А. Белов. – Москва : МОРКНИГА, 2017. – 366 с.

2. Пуховский, В. Н. Схемотехника высокопроизводительных вычислительных систем : учебное пособие: [16+] / В. Н. Пуховский, А. О. Пьявченко, С. А. Черный; Южный федеральный университет. – Ростов-на-Дону; Таганрог : Южный федеральный университет, 2019. – 231 с.: ил., табл., схем. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=598636 (дата обращения: 29.03.2023). – Библиогр.: с. 189–191. – ISBN 978-5-9275-3432-6. – Текст: электронный.

3. Хоровиц, П. Искусство схемотехники / П. Хоровиц, У. Хилл; перевод с англ. Б. Н. Бронина [и др.]. – Изд. 6-е. – Москва : Мир, 2003. – 704 с.

4. Новиков, Ю. В. Введение в цифровую схемотехнику: учебное пособие: [16+] / Ю. В. Новиков. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2007. – 344 с: табл., схем. – (Основы информационных технологий). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=233202 (дата обращения: 29.03.2023). – ISBN 5-9556-0082-5. – Текст: электронный.

Локальный электронный методический материал

Николай Алексеевич Долгий

Вычислительные машины, системы и сети

Редактор М. А. Дмитриева

Уч.-изд. л. 6,7. Печ. л. 9,2.

Издательство федерального государственного бюджетного образовательного
учреждения высшего образования
«Калининградский государственный технический университет».
236022, Калининград, Советский проспект, 1.