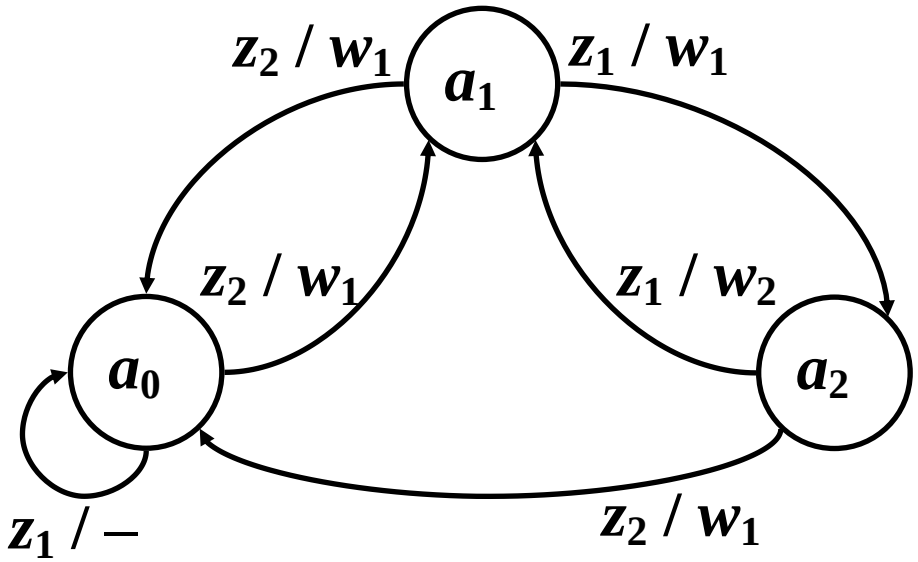
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное
образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КОНЕЧНЫЕ АВТОМАТЫ

Практикум



Санкт-Петербург
2021

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Целью выполнения работ является закрепление теоретического материала по курсу «Теория автоматов» и получение практических навыков анализа и синтеза конечных автоматов, а также машины Тьюринга.

По результатам выполнения работы студентом оформляется отчет.

Отчет должен содержать титульный лист, соответствующий образцу, размещенному на сайте университета.

Текст отчета в соответствии с требованиями ГОСТ 7.32 – 2017 набирается шрифтом Times New Roman кеглем не менее 12, строчным, без выделения, с выравниванием по ширине; абзацный отступ должен быть одинаковым и равным по всему тексту 1,25 см; строки разделяются полуторным интервалом; поля страницы: верхнее и нижнее – 20 мм, левое – 30 мм, правое – 15 мм; полужирный шрифт применяется только для заголовков разделов и подразделов; разрешается использовать компьютерные возможности акцентирования внимания, применяя шрифты разной гарнитуры.

Основную часть отчета следует делить на разделы и подразделы в соответствии с пунктом «Содержание отчета». Разделы и подразделы должны иметь порядковую нумерацию в пределах всего текста. Номер подраздела включает номер раздела и порядковый номер подраздела, разделенные точкой, после номера раздела и подраздела в тексте точку не ставят. Разделы и подразделы должны иметь заголовки; заголовки разделов и подразделов следует печатать с абзацного отступа с прописной буквы, полужирным шрифтом, без точки в конце, не подчеркивая; если заголовок состоит из двух предложений, их разделяют точкой; переносы слов в заголовках не допускаются. Каждый раздел основной части отчета начинают с новой страницы.

Страницы отчета следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту; титульный лист включают в общую нумерацию страниц; номер страницы на титульном листе не проставляют; номер страницы проставляют в центре нижней части листа без точки.

На все рисунки должны быть ссылки в тексте (например, …в соответствии с рисунком 1); рисунки следует нумеровать арабскими цифрами сквозной нумерацией; рисунки могут иметь наименование и пояснительные данные (подрисуночный текст).

На все таблицы должны быть ссылки в тексте; таблицы, следует нумеровать арабскими цифрами сквозной нумерацией; наименование таблицы следует помещать над таблицей слева, без абзацного отступа.

Контрольная работа 1
МАШИНА ТЬЮРИНГА

Цель работы: ознакомление с устройством и принципом работы машины Тьюринга.

1. Основные теоретические сведения

В первой половине *XX* века английский ученый Алан Тьюринг впервые определил понятие *алгоритма* как некоторой процедуры, которая может быть выполнена механически (без творческого вмешательства). Он показал, как эту идею можно воплотить в виде модели вычислительного процесса. Полученная модель вычислений, в которой каждый алгоритм разбивается на последовательность простых элементарных шагов, и была логической конструкцией, названной впоследствии *машиной Тьюринга* (МТ).

Машину Тьюринга невозможно реализовать на практике, однако она широко используется для теоретического анализа алгоритмов.

1.1. Схема машины Тьюринга. Назначение компонентов

МТ включает в себя бесконечную *рабочую ленту*, с которой считываются и на которую записываются символы, и *устройство управления* (УУ), оснащенное *головкой считывания-записи* (ГСЗ), которая может двигаться по рабочей ленте вправо или влево (рис. 1).

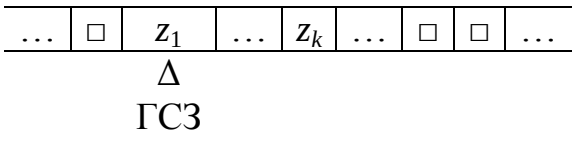


Рис. 1. Машина Тьюринга

Устройство управления в каждый дискретный момент времени $t = 0, 1, 2, \ldots$ может находиться в одном из состояний, образующих конечное множество $Q = \{q_0, q_1, q_2, \ldots, q_m\}$, входящее во *внутренний алфавит* МТ. Элементы алфавита Q называются *внутренними состояниями* МТ. Отметим, что с каждым состоянием

связывается действие, отличное от действий, выполняемых МТ в других состояниях. Среди состояний МТ выделяются начальное состояние q_0 и заключительное q_z (z – мнемонический знак конца, а не номер). В состоянии q_0 МТ находится перед началом работы, а попав в состояние q_z, останавливается.

Лента представляет собой бесконечное множество ячеек, расположенных последовательно одна за другой (рис. 1). В каждой ячейке может быть записан строго один символ z_j из конечного алфавита $Z = \{□, z_1, z_2, \ldots, z_n\}$, называемого *внешним алфавитом* МТ.

На ленте в виде слов в алфавите Z записываются исходные данные и результаты вычислений. Предполагается, что в начальный момент времени $t = 0$ часть ячеек ленты занята символами входного слова (исходными данными), а в остальные ячейки записаны пустые символы (в *JFLAP* обозначаются □).

Множества Z и Q могут быть любыми при условии, что они не имеют общих символов, то есть $Z \cap Q = \varnothing$.

Устройство управления вместе с ГСЗ представляют собой логический блок МТ.

Лента интерпретируется как *внешняя память*, в которой записываются исходные данные и окончательные результаты (слова, представленные в алфавите Z).

Внутренняя память МТ может быть представлена в виде двух ячеек:

1) ячейки состояния $q_i \in Q$, отображающей текущее состояние МТ;
2) ячейки сдвига, в которую заносится знак (направление) сдвига $d_k \in D = \{L, R, S\}$, где L – на одну ячейку влево, R – на одну ячейку вправо, S – отсутствие сдвига.

Множество $A = D \cup Q$ называется *внутренним алфавитом* МТ.

В каждый дискретный момент времени t МТ выполняет следующую последовательность элементарных действий:

- считывает символ из ячейки, обозреваемой ГСЗ;
- записывает в ячейку, обозреваемую ГСЗ, новый символ;
- сдвигает ГСЗ на одну ячейку вправо или влево или оставляет ГСЗ на месте;

– переводит УУ в новое состояние.

1.2. Способы задания логической функции МТ

Работа управляющего устройства определяется алгоритмом, который требуется реализовать на МТ, он задает так называемую *логическую функцию машины Тьюринга*.

Логическая функция позволяет по текущему состоянию q_i и обозреваемому головкой символу z_j на ленте однозначно определить:

– следующее состояние УУ $q_i' \in Q$;

– символ $z_j' \in Z$, который должен быть записан в обозреваемую ячейку (предыдущий символ на этом месте стирается);

– направление сдвига головки $d_k \in D$.

Таким образом, логическая функция МТ определяет отображение $(q_i, z_j) \rightarrow (q_i', z_j', d_k)$ (знак «$\rightarrow$» читается «влечет за собой» или «приводит к …»). Отображение вида $(q_i, z_j) \rightarrow (q_i', z_j', d_k)$ носит название *Тьюринговой команды* (ТК). Каждая команда выполняется в течение одного такта дискретного времени.

Логическая функция МТ может быть описана двумя способами: табличным и графическим.

Табличный способ сводится к построению таблицы переходов МТ (иногда называемой функциональной схемой) (табл. 1), столбцы которой соответствуют состояниям q_i , строки – входным символам z_j, а на пересечении столбца $q_i \in Q$ и строки $z_j \in Z$ ставится тройка (q_i', z_j', d_k).

Таблица 1

	q_0	…	q_i	…	q_m
z_0					
…			…		
z_j		…	q_i', z_j', d_k	…	
…			…		
z_n					

В таблице переходов должно иметь место отображение вида $(q_i, z_j) \rightarrow (q_z, z_j', S)$, означающее, что выполнение алгоритма завершено и машина останавливается.

Возможно упрощенное представление таблицы переходов. Если при выполнении отображения исходное состояние или обозреваемый символ не меняется ($q_i' = q_i$ или $z_j' = z_j$), они, соответственно, не записываются в клетку таблицы. Пример сокращенной таблицы приведен в разделе 1.3.

Графический способ задания логической функции МТ предусматривает использование графа (диаграммы) переходов, в котором состояниям МТ соответствуют вершины, а команде вида $(q_i, z_j) \rightarrow (q_i', z_j', d_k)$ соответствует ребро, ведущее из вершины q_i в вершину q_i', которое снабжается отметкой: $z_j \rightarrow z_j', d_k$ (рис. 2).

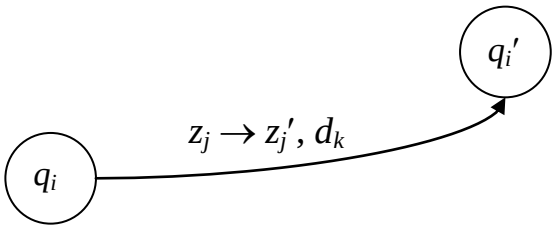


Рис. 2. Фрагмент графа переходов

Более подробно рассмотрим разработку машины Тьюринга на конкретном примере.

1.3. Пример программирования машины Тьюринга

Рассмотрим пример реализации с помощью машины Тьюринга алгоритма для проверки скобочных выражений.

Задача. На ленте записана последовательность открывающих и закрывающих круглых скобок (например, как на рис. 3). ГСЗ в начальный момент времени расположена над первым символом последовательности. Правильной считается последовательность, в которой каждой открывающей скобке «(» соответствует закрывающая «)». Необходимо определить, является ли заданная последовательность правильной.

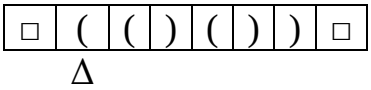


Рис. 3. Машина Тьюринга в начальный момент времени

Метод решения задачи и описание состояний

Алгоритм работы МТ для решения поставленной задачи можно описать следующим образом.

Ищется первая скобка «)», затем первая скобка «(», ей парная, и обе заменяются символом *X*. Вычеркивание парных скобок продолжается до тех пор, пока не произойдет одно из следующих событий.

Если ГСЗ, продвигаясь влево, не находит парного символа «(», то при достижении пустого символа она на его месте записывает символ 0 (т.е. последовательность скобок неправильная) и останавливается.

Если ГСЗ, продвигаясь вправо, не находит ни одного символа «)» и достигает пустого символа, то она начинает движение влево и проверяет, не остался ли на ленте символ «(». Если такой символ «(» найден, то на месте пустого символа записывается 0. Если нет символа «(», то вместо □ записывается 1 (т.е. последовательность скобок правильная).

Каждому из описанных действий соответствует отдельное состояние МТ.

Состояние q_0 предписывает движение вправо (*R*) для поиска «)».

Состояние q_1 предписывает движение влево (*L*) для поиска парной «(».

Состояние q_2 предписывает движение влево (*L*), когда не найдено ни одной «)». Если символ «(» не встретится, ГСЗ приходит к пустому символу и заменяет его на 1. Если символ «(» будет найден, МТ переходит в q_3 для того, чтобы, продолжая движение влево, поставить вместо пустого символа 0.

Формирование логической функции МТ

В соответствии с изложенным методом будем изображать ленту МТ для последовательности, приведенной на рис. 3, в характерные моменты дискретного времени, когда происходит смена состояний МТ, и указывать выполняемые в эти

моменты Тьюринговы команды (ТК). Далее, используя эти команды, заполним таблицу переходов МТ.

Отметим, что положение ГСЗ на ленте будем указывать с помощью специального символа Δ. Изображение Δ (q_i) означает, что в данный момент МТ находится в состоянии q_i.

В состоянии q_0 ищем первую скобку «)». В процессе поиска не меняются ни состояние МТ, ни содержимое ячеек ленты, через которые проходит ГСЗ (рис. 4).

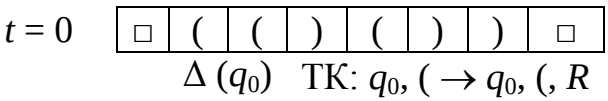


Рис. 4. Машина Тьюринга в начальный момент времени

Лента не изображена в такте 1, так как в этом такте выполняется такая же команда, что и в такте 0, и ее изображение очевидно.

В такте 2 (рис. 5) скобка «)» найдена, она заменяется на знак *X*, и осуществляется переход МТ в другое состояние, т.к. в следующем такте будет выполняться другая команда (поиск парной скобки «(») с движением влево.

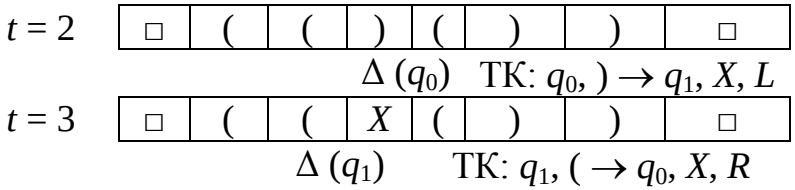


Рис. 5. Машина Тьюринга в тактах 2 и 3

В такте 3 скобка «(» найдена, она заменяется на знак *X*, после чего осуществляется переход МТ в состояние q_0, так как будет выполняться поиск следующей скобки «)» (рис. 6).

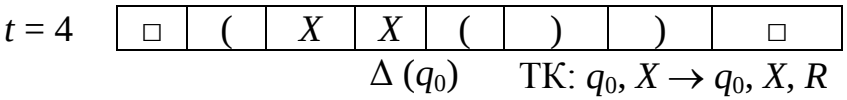


Рис. 6. Машина Тьюринга в такте 4

В рассматриваемом примере процесс чередования состояний q_0 и q_1 с заменой скобок на знаки *X* продолжается до тех пор, пока ГСЗ не попадет на

ячейку с пустым символом (рис. 7). При перемещениях ГСЗ на этой стадии она может проходить ячейки ленты с символом X. В этих случаях будут выполняться ТК: $q_0, X \to q_0, X, R$ и $q_1, X \to q_1, X, L$.

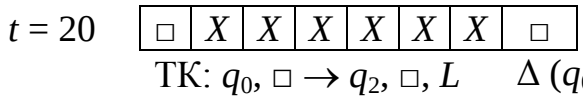


Рис. 7. Машина Тьюринга в такте 20

В такте 20 осуществляется переход МТ в новое состояние q_2, так как в следующем такте будет выполняться другое действие (поиск лишней скобки «(»). В состоянии q_2 в течение нескольких тактов МТ выполняет ТК $q_2, X \to q_2, X, L$. Отметим, что поиск «(» осуществляется и в состоянии q_1, однако команды МТ, выполняемые при ее обнаружении в состояниях q_1 и q_2 будут отличаться (см. метод решения задачи).

Скобка «(» не найдена, поэтому последовательность из скобок является правильной и в ячейке ленты на месте □ записывается 1 (рис. 8).

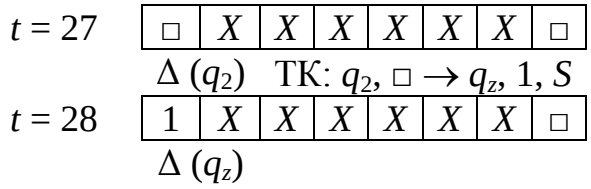


Рис. 8. Машина Тьюринга в тактах 27 и 28

Перейдем к составлению таблицы переходов МТ. Как было отмечено в методе решения задачи, МТ имеет пять состояний $q_0, \ldots, q_3, q_z$. Внешний алфавит разрабатываемой МТ будет содержать шесть символов: $Z = \{□, (,), X, 0, 1\}$, причем во время работы МТ может встретить только четыре символа: □, (,) и X. Этим определяется число строк и столбцов таблицы переходов (табл. 2).

Заполним таблицу на основании приведенных выше ТК, полученных при обработке верной последовательности скобок.

Для завершения формирования таблицы переходов МТ рассмотрим случаи, когда последовательность скобок на ленте неверна.

Таблица 2

	q_0	q_1	q_2	q_3
)	q_1, X, L			
(	$q_0, (, R$	q_0, X, R		
□	$q_2, □, L$		$q_z, 1, S$	
X	q_0, X, R	q_1, X, L	q_2, X, L	

Пусть левых скобок больше, чем правых. Это означает, что при поиске скобки «(» в состоянии q_2 ГСЗ ее находит. Тогда ГСЗ должна перейти в состояние q_3 и искать слева пустой символ, на месте которого должен быть записан 0. Выполняемые при этом команды определяют содержимое столбца таблицы переходов с заголовком q_3 (табл.3).

Таблица 3

	q_0	q_1	q_2	q_3
)	q_1, X, L			
(	$q_0, (, R$	q_0, X, R	$q_3, (, L$	$q_3, (, L$
□	$q_2, □, L$		$q_z, 1, S$	$q_z, 0, S$
X	q_0, X, R	q_1, X, L	q_2, X, L	q_3, X, L

Рассмотрим случай, когда правых скобок больше, чем левых. В этом случае в некотором такте ГСЗ в состоянии q_1 не найдет левой скобки, при достижении левого пустого символа ГСЗ должна записать на его место 0 и остановиться (табл. 4).

Так как незаполненные клетки табл. 4 соответствуют ситуациям, которые не встретятся при реализации алгоритма, построение таблицы переходов завершено.

Таблица 4

	q_0	q_1	q_2	q_3
)	q_1, X, L			
(	$q_0, (, R$	q_0, X, R	$q_3, (, L$	$q_3, (, L$
□	$q_2, □, L$	$q_z, 0, S$	$q_z, 1, S$	$q_z, 0, S$
X	q_0, X, R	q_1, X, L	q_2, X, L	q_3, X, L

Приведем упрощенное представление таблицы переходов, в котором в клетки заносятся только изменяющиеся значения состояний и символов (табл. 5). Например, в клетку с координатами (q_0, X) может быть записано лишь значение R.

Таблица 5

	q_0	q_1	q_2	q_3
)	q_1, X, L			
(	R	q_0, X, R	q_3, L	L
□	q_2, L	$q_z, 0, S$	$q_z, 1, S$	$q_z, 0, S$
X	R	L	L	L

При формировании графа переходов для моделирования МТ по таблице переходов для каждой непустой клетки таблицы должна быть построена дуга с отметкой, включающей все необходимые компоненты (см. раздел 1.2). Например, клетке с координатами (q_1, X) должна соответствовать дуга в виде петли с отметкой: $X \to X, R$. Граф переходов полученной МТ представлен на рис. 9.

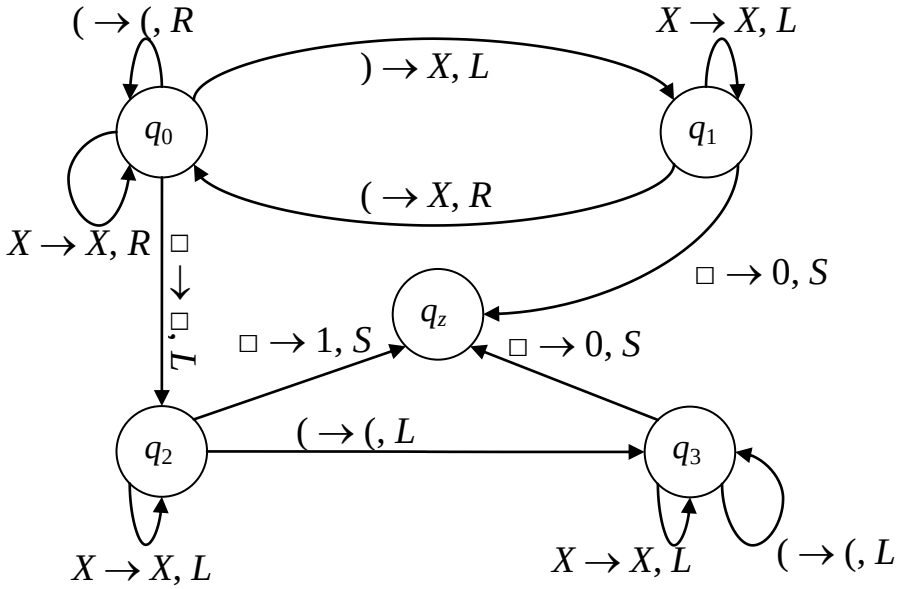


Рис. 9. Граф переходов МТ

При построении графа в пакете *JFLAP* вместо знака «→» используется символ «;». Граф переходов полученной МТ, построенный в *JFLAP*, приведен на рис. 10.

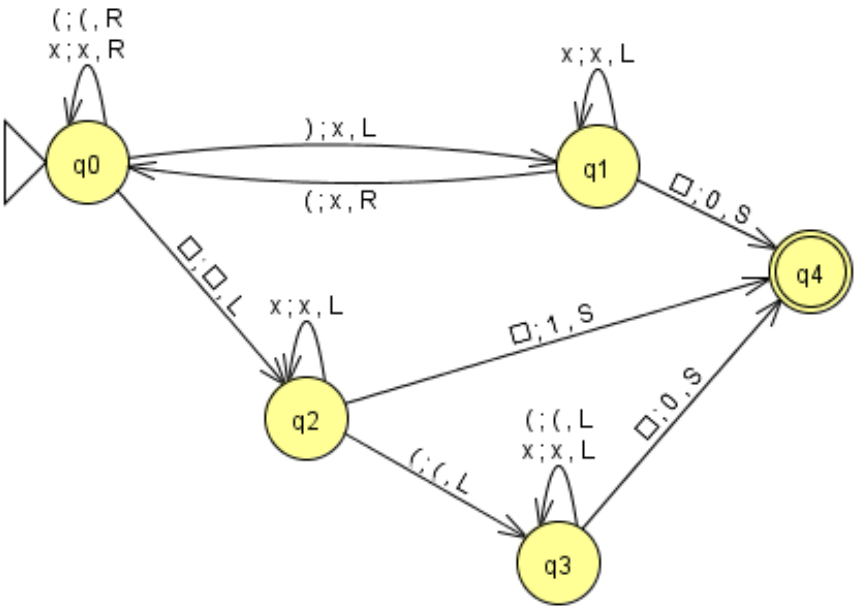


Рис. 10. Граф переходов МТ в JFLAP

Для того чтобы убедиться в правильности построения МТ необходимо проверить ее работу на тестовых примерах.

2. Задание по работе

В работе требуется построить машину Тьюринга для решения указанной задачи (см. раздел «Варианты заданий»). Приведенный в разделе «Варианты заданий» метод решения задачи носит рекомендательный характер, студентом может быть предложен другой метод.

Во всех вариантах предполагается, что в начальном положении ГСЗ обозревает ячейку ленты с первым символом входной последовательности. В конечный момент времени ГСЗ должна обозревать ячейку с первым символом результата.

Проверку корректности работы спроектированной МТ необходимо провести в пакете *JFLAP*.

3. Порядок выполнения работы

Выполнение работы состоит из двух частей: проектирования МТ и проверки корректности спроектированной МТ путем моделирования в пакете *JFLAP*.

Для проектирования МТ, необходимо:

1) ознакомиться с формулировкой задачи и методом ее решения (см. раздел «Варианты заданий»);
2) выполнить изображения ленты МТ в характерные для задачи такты дискретного времени (см. пример в разделе 1.3) и записать соответствующие этим тактам Тьюринговы команды;
3) на основании п. 3 выполнить описание состояний МТ и заполнить таблицу переходов МТ.

Для проверки корректности работы спроектированной МТ в пакете *JFLAP* необходимо:

1) запустить пакет *JFLAP* и выбрать режим *Turing Machine*;
2) в рабочем поле ввести граф переходов МТ, указать начальное и конечное состояние;
3) с помощью меню *Test: Highlight Nondeterminism* проверить введенный граф на отсутствие неопределенностей;
4) выбрать меню *Input: Step…* и в появившемся окне набрать входное слово, после чего нажать *OK*; внизу появится окно моделирования с изображением ленты;
5) нажимая на клавишу *Step* в нижней части окна, проследить за изменениями содержимого ячеек ленты до момента остановки ГСЗ;
6) по содержимому ячеек ленты установить правильность решения задачи, в противном случае внести необходимые коррективы.

Для проверки работы МТ также можно использовать меню *Input: Multiple Run (Transducer)*. При использовании этого режима в столбце *Input* задается входное слово (можно задать несколько слов). Затем следует нажать кнопку *Run*

Inputs. Содержимое ленты, полученное после окончания обработки каждого входного слова, можно посмотреть, нажав кнопку *View Trace*.

Подробная инструкция по построению и моделированию МТ в *JFLAP* приведена в Приложении.

4. Содержание отчета

Отчет по работе должен содержать следующие разделы.

1. Цель работы.
2. Задание по работе.

   Раздел должен включать в себя формулировку задания и данные варианта.
3. Разработка машины Тьюринга.

   В данном разделе необходимо привести описание состояний и таблицу переходов машины Тьюринга.
4. Моделирование машины Тьюринга.

   В данном разделе необходимо привести граф машины Тьюринга, построенный в JFLAP, и скриншоты с результатами моделирования работы МТ для максимально возможного количества контрольных последовательностей.
5. Вывод.

   Пример вывода: «В результате выполнения работы создана логическая функция машины Тьюринга для определения корректности последовательности скобок. Проверка работоспособности машины произведена в среде JFLAP. Изучено устройство и принцип работы машины Тьюринга».

5. Варианты заданий

Вариант 1

Удаление букв b

На ленте расположено слово из букв *a, b* и *c*, например: *abcbca*. Требуется удалить из исходного слова все буквы *b* (для приведенного примера запись на ленте должна принять вид: *acca*).

Метод решения задачи. Будем формировать результат справа от исходного слова, используя в качестве разделителя между словами знак >. Для этого МТ осуществляет поиск правого пустого символа □, заменяет его на символ > и снова возвращается в начало слова.

Теперь наша задача – перенести в цикле все символы входного слова, кроме *b*, вправо за знак > в формируемое выходное слово. Для этого анализируем первый символ входного слова. Если это *b*, тогда стираем его и переходим к следующему символу, не меняя состояния. Если же первый символ – это *a* или *c*, тогда стираем его, запоминаем его (путем дальнейшего перехода в различные состояния для различных символов) и двигаемся вправо до первой пустой клетки, куда и записываем стертый символ. Снова движемся влево к символу, который стал первым во входном слове, и повторяем те же самые действия.

Этот цикл завершается, когда во входном слове не останется символов. До завершения работы МТ должна стереть знак >.

Вариант 2

Декремент натурального числа

На ленте расположение натуральное число в десятичной системе счисления. Разработать машину Тьюринга, которая уменьшала бы заданное число на 1. При этом в выходном слове не должно быть нулей в незначащих разрядах (т.е., например, если входным словом было 100, то выходным словом должно быть 99, а не 099).

Метод решения задачи. Сначала необходимо найти младшую цифру числа, а затем уменьшить ее на 1. Если цифра была больше 1, то после уменьшения необходимо вернуться к началу слова и завершить работу МТ, если же младшая цифра равна 0, то вместо нее пишем 9, смещаемся влево и вновь выполняем вычитание. Если уменьшаемая цифра равна 1, то вместо нее пишем 0. После записи 0 в каком-либо разряде, кроме последнего, нужно проанализировать, не является ли этот ноль старшей незначащей цифрой (т.е. не стоит ли слева от него в записи выходного слова □). Если записанный 0 является старшей незначащей цифрой, то его надо удалить из записи выходного слова.

Вариант 3

Добавление двойки к троичному числу

На ленте расположено число в троичной системе счисления, например 102121. Требуется прибавить к данному числу 2 (для приведенного примера результат: 102200).

Метод решения задачи. Находим младший разряд числа (для этого, двигаясь вправо, находим □, а затем сдвигаем ГСЗ влево). Если последний разряд равен 0, заменяем его на 2, возвращаемся к началу слова и завершаем работу МТ. Если последний разряд равен 1 или 2, заменяем его на 0 или 1, соответственно, а затем смещаемся влево, чтобы увеличить предыдущий разряд числа на 1. Если в предыдущем разряде находится символ □, следует заменить его на 1 и остановиться.

Вариант 4

Сравнение символов и стирание слова

На ленте расположено слово из букв *a*, *b* и *c*, например: *abcbca*. Требуется стереть слово (заменить его пустыми символами □), если первый и последний символы слова различны.

Метод решения задачи. Основная идея решения этой задачи – запомнить первый символ входного слова, не стирая его. Для этого, прочитав первый символ

слова, нужно выполнить переход в новое состояние, если это *a* – в одно, если *b* – в другое, если *c* – в третье. Во всех этих трех состояниях ГСЗ перемещается вправо и ищет последний символ, чтобы сравнить его с первым символом. Если первый и последний символы входного слова совпали, МТ останавливается. В противном случае осуществляется переход МТ в состояние, в котором выполняется стирание любого символа входного слова (замена его на □). В такте, в котором ГСЗ обозревает пустой символ □, МТ останавливается.

Вариант 5

Редактирование слова

На ленте расположено слово из букв *a*, *b* и *c*, например: *abcbca*. Требуется, если первый и последний символы слова одинаковы, заменить все слово этими символами (для приведенного примера результат: *aaaaaa*).

Метод решения задачи. Основная идея решения этой задачи – запомнить первый символ входного слова, не стирая его. Для этого, прочитав первый символ слова, нужно выполнить переход в новое состояние, если это *a* – в одно, если *b* – в другое, если *c* – в третье. Во всех этих трех состояниях ГСЗ перемещается вправо и ищет последний символ, чтобы сравнить его с первым символом. Если первый и последний символы входного слова не совпали, МТ останавливается. В противном случае МТ выполняет замену каждого символа входного слова на начальный символ.

Вариант 6

Удвоение слова

На ленте расположено слово из букв *a* и *b*, например: *abba*. Требуется поставить после слова знак = и повторить исходное слово (для приведенного примера конечная запись на ленте имеет вид: *abba=abba*).

Метод решения задачи. Перед копированием необходимо найти ячейку ленты справа от правого крайнего символа входного слова, записать в нее знак =

и вернуться к началу слова. Затем необходимо все символы слова скопировать в пустые клетки справа от знака =.

При копировании очередного символа необходимо решить две проблемы: как запомнить копируемый символ и как запомнить его позицию. Первая проблема решается путем использования различных состояний, в которые переходит МТ при чтении символа: в одно, если это символ *a*, и в другое, если – *b*. Для решения второй проблемы на место копируемого символа заносится символ-двойник: А вместо *a*, *B* вместо *b*.

Затем необходимо записать соответствующий символ в конец строки. Чтобы после этого вернуться к следующему нескопированному символу, нужно отыскать на ленте клетку, где находится символ *A* или *B*. Затем в данной клетке нужно восстановить прежний символ.

Когда мы скопируем последний символ входного слова и вернёмся к его двойнику, после сдвига на одну позицию вправо мы попадём на знак =. Это сигнал о том, что входное слово полностью скопировано.

Вариант 7

Удаление меньшего числа

На ленте машины Тьюринга записаны два числа в унарном коде, разделенные знаком * (например, 111*11). Требуется стереть меньшее число и знак * (для приведенного примера результат будет иметь вид: 111). Если числа равны, стереть любое из двух чисел и знак *.

Метод решения задачи. Будем заменять первую и последнюю единицу исходной строки на символ *x*, пока в каком-либо числе не закончатся единицы.

Если при поиске единицы в первом числе МТ обозревает ячейку с *, это означает, что единицы первого числа иссякли. Следовательно, первое число меньше или равно второму числу. Тогда машина проходит ячейку с * и во всех последующих ячейках справа до символа □ проставляет единицы. Затем МТ начинает движение влево, стирая символы *x* и *.

Если же при поиске единицы во втором числе машина обнаруживает *, это означает, что иссякли единицы второго числа. Следовательно, большим оказывается первое число.

Вариант 8

Вставка символа в слово

На ленте расположено слово из букв *a, b* и *c*, например: *abcbca*. Если слово непустое, требуется после первого его символа вставить символ *a* (для приведенного примера результат: *aabcbca*).

Метод решения задачи. Очевидно, что между первым и вторым символами слова надо освободить клетку для нового символа *a*. Для этого надо заменить символ □, стоящий перед словом, на первый символ слова. При этом МТ читает первый символ и, для того, чтобы запомнить его, переходит в одно состояние, если это символ *a*, в друге – если *b*, в третье – если *c*. После копирования первого символа МТ заменяет первый символ исходного слова на символ *a*.

Вариант 9

Обмен символов слова местами

На ленте расположено слово из букв *a, b* и *c*, например: *abcbc*. Если слово непустое, требуется поменять местами его первый и последний символы (для приведенного примера результат: *cbcba*).

Метод решения задачи. Первый символ слова нужно запомнить и стереть (на его место записать □). Проблема запоминания решается путем использования различных состояний, в которое переходит МТ в зависимости от стертого символа. Далее необходимо двигаться вправо до символа □, а затем запомнить и заменить последний символ слова на стертый. Запоминание последнего символа также осуществляется за счет перехода МТ в различные состояния для каждого символа (*a, b* или *c*). Теперь необходимо вернуться к началу слова и записать запомненный символ вместо □.

Вариант 10

Сколько звездочек?

На ленте расположена последовательность звездочек, например **********. Требуется записать количество звездочек в последовательности, а сами звездочки стереть (для приведенного примера результат: 12).

Метод решения задачи. Будем удалять (заменять на □) звездочки, начиная с конца последовательности. Для поиска конца последовательности будем двигаться вправо, а при обнаружении □, сдвинем ГСЗ влево, при этом ГСЗ окажется над последней звездочкой. При удалении каждой звездочки будем увеличивать на 1 число, расположенное в начале последовательности. Для этого после замены * на □ будем двигаться влево, пока не обнаружим одну из десятичных цифр или символ □. Символ □ заменим на 1, цифры от 0 до 8, увеличим на 1, а затем возвратимся к концу последовательности звездочек для удаления следующей звездочки. Если цифра равна 9, заменим ее на 0 и сдвинемся влево для увеличения на 1 предыдущего разряда числа.

Вариант 11

Сравнение числа букв

На ленте расположено слово из букв *a* и *b*, например: *abbab*. Требуется определить, какой символ повторяется чаще, и записать его через одну ячейку после окончания слова. Исходное слово сохранять не обязательно. Для приведенного примера результат может иметь вид: *xxxxx*□*b*. Если количество букв *a* и *b* в слове одинаково, в указанную ячейку ничего не записывается.

Метод решения задачи. МТ заменяет первый символ слова на символ *x*. Если удален символ *b*, МТ ищет и удаляет (заменяет на символ *x*) первый в слове символ *a*. Если удален символ *a*, МТ ищет и удаляет первый в слове символ *b*. Затем МТ возвращается к началу слова и вновь удаляет первый из оставшихся символов *a* или *b*, а затем ищет и удаляет парный ему.

Получившийся таким образом цикл повторяется, пока ГСЗ при поиске очередного символа для удаления не дойдет до пустого символа □. Состояние, в

котором будет обнаружен □, укажет, каких символов больше (или на то, что их равное количество).

Вариант 12

Перевод числа из унарной системы счисления в двоичную

На ленте расположено число в унарной системе счисления, например 111111. Требуется записать это число в двоичной системе счисления, исходную запись числа необходимо стереть (для приведенного примера результат: 110).

Метод решения задачи. Будем формировать двоичную запись числа перед исходным числом, поставив между этими двумя числами разделитель >. Для этого сдвинем ГСЗ влево и заменим □ на >. Затем будем удалять (заменять на □) по одной единице из унарного числа, начиная с младшего разряда. Для поиска младшего разряда будем двигаться вправо, а при обнаружении □, сдвинем ГСЗ влево, при этом ГСЗ окажется над последней единицей. При удалении каждой единицы будем увеличивать на 1 формируемое двоичное число. Для этого после замены 1 на □ будем двигаться влево, пока не обнаружим символ >. При обнаружении > нужно сдвинуть ГСЗ влево и заменить младший разряд двоичного числа. Если младший разряд равен □ или 0, заменяем его на 1, а затем снова переходим к удалению единиц исходного числа. Если младший разряд двоичного числа равен 1, заменяем его на 0 и сдвигаемся влево для увеличения на 1 предыдущего разряда.

По окончании перевода символ > необходимо стереть.

Вариант 13

Удвоение букв в слове

На ленте расположено слово из букв *a*, *b* и *c*, например *abcab*. Требуется удвоить каждую букву этого слова (для приведенного примера результат: *aabbccaabb*).

Метод решения задачи. Поставим в конце заданного слова знак >, после которого будем формировать результат. Для этого будем двигаться вправо, а при

обнаружении □, заменим этот символ на >. Затем вернемся в начало заданного слова и будем поочередно удалять его буквы, начиная с первой. После удаления каждой буквы, будем переходить в конец формируемого слова и дописывать удаленную букву дважды. При этом для того, чтобы «запомнить», какой символ был удален, МТ будет переходить в одно состояние при удалении *a*, в другое – при удалении *b*, в третье – при удалении *c*. По окончании формирования результата необходимо стереть символ >.

Вариант 14

Перенести в начало все a

На ленте расположено слово из символов *a*, *b* и *c*, например *babcab*. Требуется переместить все символы *a* в начало слова (для приведенного примера результат: *aabbcb*).

Метод решения задачи. Поставим в конце заданного слова знак >, после которого будем формировать результат. Для этого будем двигаться вправо, а при обнаружении □, заменим этот символ на >. Затем вернемся в начало заданного слова и будем переносить за знак > все символы *a*, заменяя их в исходном слове на знак *x*. Затем последовательно перенесем оставшиеся символы, также заменяя их в исходном слове на знак *x*. При этом для того, чтобы «запомнить», какой символ был удален, МТ будет переходить в одно состояние при удалении *b*, в другое – при удалении *c*. По окончании переноса удалим знаки > и *x*.

Вариант 15

Удаление половины слова

На ленте расположено слово из символов *a*, *b* и *c*, например *acabcbab*. Если слово содержит четное число символов, требуется стереть левую половину слова (для приведенного примера результат: *cbab*).

Метод решения задачи. Будем читать слово слева направо, используя два состояния МТ: одно – для чтения четного символа, второе – для чтения нечетного

символа. Состояние, в котором окажется МТ по окончании чтения слова, укажет четность числа символов в слове.

Если число символов оказалось четным, вернемся к началу слова и удалим первый символ. Затем перейдем в конец слова и изменим регистр последнего символа (заменим *a* на *A*, *b* на *B*, *c* на *C*). Снова перейдем в начало слова и удалим следующий символ. Затем перейдем к измененному символу в конце слова, вернем его в исходное состояние и изменим регистр символа, стоящего перед ним. Таким образом, когда в начале слова окажется символ *A*, *B* или *C*, половина слова будет удалена.

Вариант 16

Замена a на bb

На ленте расположено слово из символов *a*, *b* и *c* (например, *abacbcab*). Требуется заменить последний символ *a* на *bb* (для приведенного примера конечная запись на ленте будет иметь вид: *abacbcbbb*).

Метод решения задачи. Переместим ГСЗ в конец слова. Для этого будем сдвигать ГСЗ вправо, пока не будет обнаружен пустой символ. Затем будем двигаться влево. При обнаружении символа *a* заменим его на *b*. Теперь необходимо переписать все находящиеся справа от ГСЗ символы на одну ячейку вправо, а в освободившуюся ячейку записать *b*. Для этого удобно использовать два состояния МТ: одно – для записи *b*, другое – для записи *c*.

Вариант 17

Слово abc?

На ленте расположено слово из символов *a*, *b* и *c*. Требуется оставить на ленте только символ *a*, если это слово *abc*. Если на ленте записано любое другое слово, необходимо удалить его целиком.

Метод решения задачи. Если первый символ слова равен *a*, проверим второй символ. Если второй символ равен *b*, то проверим третий символ. Если третий символ равен *c*, то проверим четвертый символ. Если четвертый символ слова

пустой, удалим символы *c* и *b*, сдвинемся влево и завершим работу МТ. Если же первый символ слова оказался не равен *a*, или второй символ не равен *b*, или третий символ не равен *c*, или четвертый символ слова непустой, то перейдем к началу слова, сотрем все его символы и завершим работу МТ.

Вариант 18

Определение остатка

На ленте записано число в унарной системе счисления (т.е. оно содержит количество единиц, равное значению числа), например 111111. Требуется определить остаток от деления исходного числа на 3. Остаток (0, 1 или 2) записывается через одну ячейку после числа (для приведенного примера конечная запись на ленте будет иметь вид: 111111□0).

Метод решения задачи. Если ГСЗ в начальный момент обозревает пустой символ □, то МТ записывает в нужную ячейку 0 (это означает, что слово пустое). Если ГСЗ в начальный момент обозревает 1, МТ сдвигается вправо. Если при этом обозревается □, то МТ записывает в нужную ячейку 1, иначе – снова сдвигается вправо. Если в этом случае обозревается □, то МТ записывает в нужную ячейку 2, иначе – снова происходит сдвиг вправо.

Описанная последовательность действий повторяется, пока МТ не обнаружит □.

Вариант 19

Определение частного

На ленте записано число в унарной системе счисления (т.е. оно содержит количество единиц, равное значению числа), например 111111. Требуется определить частное от деления исходного числа на 3 и записать его также в унарной системе счисления через одну ячейку после числа. Для приведенного примера конечная запись на ленте будет иметь вид: 111111□11.

Метод решения задачи. Если ГСЗ в начальный момент обозревает пустой символ □ (т.е. число пустое), то МТ останавливается, иначе ГСЗ сдвигается

вправо. Если при этом обозревается □, то МТ останавливается, иначе ГСЗ сдвигается вправо. Если после этого сдвига встретилась 1, то это последняя единица тройки, МТ заменяет ее на символ *X*, чтобы запомнить тем самым ее позицию, ищет символ □ справа от конца исходного слова и ставит очередную единицу в следующую ячейку. Затем ГСЗ ищет символ *X*, заменяет его на 1, после чего цикл повторяется.

Вариант 20

Удалить два последних a

На ленте расположено слово из букв *a* и *b*, например: *baabba*. Требуется удалить из слова две последние буквы *a* (для приведенного примера результат: *babb*).

Метод решения задачи. Переместим ГСЗ в конец слова, а затем, двигаясь справа налево, заменим две последние буквы *a* на *. В первую пустую ячейку справа от слова запишем знак > и переместим оставшиеся символы *a* и *b* исходного слова за знак >. Для этого перейдем к началу слова, удалим первый символ и запишем его после знака > (чтобы осуществить запись, МТ после удаления *a* должна перейти в одно состояние, а после удаления *b* – в другое). Знак * будем просто удалять, не записывая его. По окончании удаления символов исходного слова удалим знак >.

Вариант 21

Умножение чисел в унарной системе счисления

На ленте расположена запись вида 111*11=, где * – знак умножения, 111 и 11 – числа в унарном коде. Требуется после знака = разместить результат умножения также в унарном коде (для приведенного примера результат: 111*11=111111).

Метод решения задачи. Очередную единицу (начиная с первой) правого сомножителя заменяем на символ *x* и переносим весь левый сомножитель за знак равенства. Перенос производится путём поочерёдной замены каждой единицы левого сомножителя на *x* и дописывания единицы в конец строки. После того, как все единицы левого сомножителя оказались в конце строки, заменяем все *x* в

левом сомножителе обратно на единицы. Затем следующую единицу правого сомножителя заменяем буквой *x* и снова переносим единицы левого сомножителя в конец строки. Если при поиске единиц в правом сомножителе оказывается, что их нет, умножение завершено. Остается только восстановить единицы в правом сомножителе.

Вариант 22

Удалить второе a

На ленте расположено слово из букв *a, b* и *c*, например: *abcbcabbc*. Требуется удалить из исходного слова вторую букву *a* (для приведенного примера запись на ленте должна принять вид: *abcbcbbc*).

Метод решения задачи. Будем смещать символы исходного слова на один вправо, пока не встретим вторую букву *a*. Для этого сотрем первую букву слова (заменим ее на □) и запишем ее в следующую ячейку ленты, используя одно из трех состояний МТ: одно – для записи *a*, второе – для записи *b*, третье – для записи *c* (в зависимости от стертой буквы). Используя эти три состояния, будем смещать буквы исходного слова вправо, пока не запишем *a*. Теперь необходимо записывать только буквы *b* и *c*, пока одна из этих букв не будет записана вместо *a* или слово не закончится. Для этого необходимо использовать два других состояния МТ (одно – для записи *b*, другое – для записи *c*).

Вариант 23

Зеркальное отображение заданного слова в алфавите {*a, b*}

На ленте расположена запись вида *aabab*, где *aabab* – заданное слово в алфавите {*a, b*}. Требуется записать на ленте зеркальное отображение заданного слова, стерев исходное (для приведенного примера результат: *babaa*).

Метод решения задачи. Первый символ заданного слова заменяем на знак *x*. Для того чтобы «запомнить» стертый символ, МТ переходит в различные состояния в зависимости от стертого символа: если стерт символ *a* – в одно, если символ *b* – в другое. Далее ГСЗ начинает движение влево и ищет первый пустой символ. Найдя его, ГСЗ записывает на его месте стертый символ и затем движется

вправо до первого знака *x*, а найдя его, сдвигается еще раз вправо к следующей ячейке и заменяет очередной символ слова на знак *x*. Если при этом МТ в ячейке обнаруживает □, то это означает, что все символы проанализированы, и зеркальное отображение построено. По окончании отображения символы *x* необходимо заменить на □.

Вариант 24

Удаление парных скобок

На ленте расположена последовательность открывающих и закрывающих скобок, ограниченная пустыми символами, например)(()(). Требуется удалить из последовательности все парные скобки, т.е. для приведенного примера результат будет иметь вид:)*xxx*((*x*.

Метод решения задачи. Двигаясь вправо, ищем первый символ (. Если символ (найден, то сдвигам ГСЗ вправо и переходим в следующее состояние, в котором, двигаясь вправо, ищем парную закрывающую скобку. Если символ) найден, заменяем его на знак *x* и ищем начало входного слова, после чего заменяем на *x* первый в слове символ (. Затем ищем следующую пару скобок, для этого вновь переходим к поиску символа (. Если МТ при поиске) или (находит □, удаление скобок завершено.

Вариант 25

Сортировка букв a и b

На ленте расположена строка из символов *a* и *b*, ограниченная пустыми символами, например *aabbbaab*. Требуется переместить все символы *a* в левую, а символы *b* – в правую часть строки (для приведенного примера результат: *aaaabbbb*).

Метод решения задачи. В начальный момент времени ГСЗ обозревает крайний левый символ последовательности. В зависимости от того *a* это или *b*, МТ переходит в разные состояния (состояние движения по цепочке из букв *a* или состояние движения по цепочке из *b*). Если при движении по любой цепочке, встречается символ □ – сортировка завершена. Если при движении по цепочке из

букв *a* встречается *b*, МТ переходит в состояние движения по цепочке из *b*. Если при движении по цепочке из букв *b* встречается *a*, МТ заменяет ее на *b* и переходит в новое состояние, в котором перемещается влево, в начало цепочки из *b* и заменяет первое *b* на *a*. Затем снова переходит в состояние движения по цепочке из *a* вправо.

Вариант 26

Вычитание чисел в унарной системе счисления

На ленте расположена запись вида 1111-11, где 1111 и 11 – уменьшаемое и вычитаемое в унарном коде (уменьшаемое всегда не меньше вычитаемого). Требуется заменить исходную запись разностью в унарном коде (для приведенного примера конечная запись на ленте будет иметь вид: 11).

Метод решения задачи. Будем поочередно заменять на пустой символ самую левую единицу в уменьшаемом и самую правую единицу в вычитаемом. Когда в вычитаемом не останется единиц, заменим знак минуса на пустой символ и переместимся к началу результата.

Вариант 27

Равны ли два числа?

На ленте расположена запись вида 1111?11, где 1111 и 11 – два числа в унарном коде. Требуется заменить ? на символ =, если числа равны, и на символ !, если числа разные (для приведенного примера и предлагаемого ниже метода решения задачи конечная запись на ленте будет иметь вид: 1!).

Метод решения задачи. Будем поочередно заменять на пустой символ самую первую и самую последнюю единицу во входном слове. Если при поиске очередной единицы в первом числе, оно окажется пустым, проверим, остались ли единицы во втором числе, заменим ? на соответствующий знак и завершим работу. Если же после удаления очередной единицы из первого числа, второе число окажется пустым, числа не равны, заменим ? на ! и завершим работу.

Вариант 28

Какое число меньше?

На ленте расположена запись вида 1111??11, где 1111 и 11 – два числа в унарном коде. Требуется заменить ?? на символ <, если первое число меньше второго, и символы >= в противном случае (для приведенного примера и предлагаемого ниже метода решения конечная запись на ленте будет иметь вид: 1>=).

Метод решения задачи. Будем поочередно заменять на пустой символ самую первую и самую последнюю единицу во входном слове. Когда в одном из чисел не останется единиц, проверим, есть ли единицы во втором числе, заменим ?? на соответствующий знак, переместим ГСЗ к началу результата и завершим работу.

Вариант 29

Сравнение чисел в унарной системе счисления

На ленте расположена запись вида 1111?11, где 1111 и 11 – два числа в унарном коде. Требуется заменить ? на соответствующий символ: <, > или = (для приведенного примера конечная запись на ленте будет иметь вид: 1111>11).

Метод решения задачи. Будем поочередно заменять на символ *x* первую и последнюю единицу во входном слове, затем – вторую и предпоследнюю. При попытке удалить очередную единицу мы должны встретить знак ?. Если ? обнаружен при попытке удалить единицу из первого числа, проверим, есть ли единицы во втором числе и сделаем вывод о знаке, который нужно поставить. Если ? обнаружен при попытке удалить единицу из второго числа, значит, первое число больше второго. Перед окончанием работы необходимо восстановить удаленные ранее единицы.

Вариант 30

Определение кратности числа

На ленте машины Тьюринга находится неотрицательное целое число в десятичной системе счисления (например, запись имеет вид: 10). Определить,

делится ли это число на 5 без остатка. Если делится, то записать справа от числа слово ДА, иначе – НЕТ (для указанного примера результат будет иметь вид: 10ДА).

Метод решения задачи. Необходимо найти последний символ числа и проанализировать его. Если последний символ числа равен 0 и в старших разрядах есть ненулевые цифры или если последний символ числа равен 5, то переходим к записи ДА, иначе – переходим к записи НЕТ.

Контрольная работа 2
МИНИМИЗАЦИЯ ПОЛНОСТЬЮ ОПРЕДЕЛЕННЫХ АВТОМАТОВ

Цель работы: знакомство с понятием абстрактного автомата, способами задания абстрактных автоматов, освоение метода минимизации полностью определенных автоматов расщеплением классов эквивалентных состояний.

1. Основные теоретические сведения

Алгоритм работы любого цифрового устройства может быть описан математически в виде так называемого конечного абстрактного автомата.

1.1. Формальное описание абстрактного автомата

Абстрактный автомат (рис. 1) определяется шестеркой объектов:

$$S = (A, Z, W, \delta, \lambda, a_0),$$

где $A = \{a_0, a_1, \ldots, a_m, \ldots, a_M\}$ – множество состояний (алфавит состояний);

$Z = \{z_1, \ldots, z_f, \ldots, z_F\}$ – множество входных символов (входной алфавит);

$W = \{w_1, \ldots, w_g, \ldots, w_G\}$ – множество выходных символов (выходной алфавит);

δ: $A \times Z \to A$ – функция переходов; каждой паре (a_m, z_f) она ставит в соответствие состояние перехода a_s, т. е. $a_s = \delta(a_m, z_f)$;

λ: $A \times Z \to W$ – функция выходов; каждой паре (a_m, z_f) она ставит в соответствие выходной символ w_g, т. е. $w_g = \lambda(a_m, z_f)$;

$a_0 \in A$ – начальное состояние автомата.

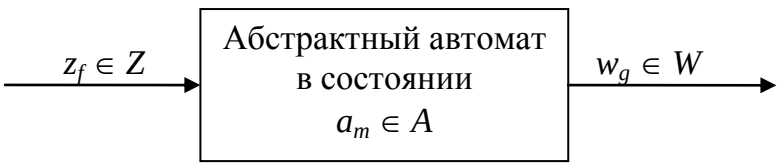


Рис. 1. Абстрактный автомат как «черный ящик»

Под алфавитом понимается непустое множество попарно различных символов. Конечная упорядоченная последовательность символов называется словом в данном алфавите.

Здесь и в дальнейшем будем рассматривать *конечные абстрактные автоматы*, то есть автоматы, для которых входной и выходной алфавиты, а также множество состояний содержат конечное множество элементов.

В теории автоматов используется модель дискретного времени, в соответствии с которой временная ось разбивается на временные отсчеты (дискреты) через равные, как правило, интервалы времени $t = 0, 1, \ldots$.

При $t = 0$ конечный автомат находится в начальном состоянии $a(0) = a_0$.

В момент дискретного времени t автомат, находясь в состоянии $a_m = a(t)$, которое будем для данного момента называть *исходным состоянием*, производит следующие действия:

- воспринимает подаваемый на вход символ входного алфавита $z_f = z(t)$;
- в соответствии с функцией выходов λ выдает символ выходного алфавита $w_g = w(t) = \lambda(a(t), z(t))$;
- в соответствии с функцией δ переходит в следующее состояние $a_s = a(t+1) = \delta(a(t), z(t))$; это следующее состояние в дальнейшем будем называть *состоянием перехода*.

Таким образом, если на вход автомата, находящегося в состоянии $a(0) = a_0$, подавать по очереди символы входного слова, то на выходе автомата будут последовательно появляться символы выходного слова. Следовательно, автомат реализует некоторое отображение множества слов входного алфавита во множество слов выходного алфавита.

На абстрактном уровне понятие «работа автомата» означает именно преобразование входных слов в выходные слова. При этом мы отвлекаемся от структуры автомата, рассматривая его как «черный ящик» (это принятый в кибернетике подход, когда основное внимание уделяется поведению системы относительно внешней среды). Оказывается, такой уровень позволяет решить ряд

сложных проблем, которые на структурном уровне скрываются за множеством деталей, несущественных с точки зрения поведения системы в целом.

Понятие состояния введено в определение автомата в связи с тем, что возникает необходимость в описании систем, информация на выходах которых зависит не только от информации на входах в данный момент времени, но и от сигналов, которые поступали на входы системы ранее. Состояния автомата как раз и соответствуют своеобразной памяти о прошлом, и позволяют устранить время как явную переменную в логических выражениях для выходных сигналов.

1.2. Способы задания абстрактных автоматов

Наибольшее распространение получили конечные автоматы моделей Мили и Мура. Они отличаются друг от друга определением функции выходов.

Функции переходов и выходов автомата Мили задаются в виде

$$a(t+1) = \delta(a(t), z(t)),$$

$$w(t) = \lambda(a(t), z(t)), \quad t = 0, 1, 2,\ldots$$

Функции переходов и выходов автомата Мура задаются в виде

$$a(t+1) = \delta(a(t), z(t)),$$

$$w(t+1) = \lambda(a(t+1)), \; t = 0, 1, 2,\ldots$$

Таким образом, выходной символ в автомате Мили зависит как от исходного состояния, так и от входного символа в рассматриваемый момент времени, а в автомате Мура выходной символ зависит только от состояния перехода.

Обычно конечный автомат задается одним из двух способов: с помощью таблиц или с помощью графа.

Функции переходов δ и выходов λ конечного автомата являются решетчатыми функциями, поэтому их удобно задавать в табличной форме. Пример *таблично* заданных функций переходов и выходов для автомата Мили приведен в табл. 1 и 2 соответственно.

Таблица 1

	a_0	a_1	a_2	a_3
z_1	a_3	a_2	a_1	a_0
z_2	a_0	a_3	a_2	a_1
z_3	a_1	a_0	a_3	a_2

Таблица 2

	a_0	a_1	a_2	a_3
z_1	w_1	w_2	w_3	w_4
z_2	w_4	w_1	w_2	w_3
z_3	w_3	w_4	w_1	w_2

Поскольку области определения функций δ и λ для автомата Мили совпадают, таблица переходов и таблица выходов могут быть объединены в одну совмещенную таблицу переходов-выходов (СТПВ) (табл. 3). В клетку совмещенной таблицы на пересечении i-ой строки и j-го столбца ($i \in \{1, 2, \ldots, F\}$, $j \in \{0, 1, \ldots, M\}$) записываются состояние перехода автомата $a(t+1)$ и выходной символ $w(t)$, вырабатываемый при переходе в это состояние.

Таблица 3

	a_0	a_1	a_2	a_3
z_1	a_3/w_1	a_2/w_2	a_1/w_3	a_0/w_4
z_2	a_0/w_4	a_3/w_1	a_2/w_2	a_1/w_3
z_3	a_1/w_3	a_0/w_4	a_3/w_1	a_2/w_2

В табл. 4 и 5 приведен пример таблично заданных функций переходов δ и выходов λ автомата Мура.

Таблица 4

	a_0	a_1	a_2	a_3
z_1	a_3	a_2	a_1	a_0
z_2	a_0	a_3	a_2	a_1
z_3	a_1	a_0	a_3	a_2

Таблица 5

$a(t)$	a_0	a_1	a_2	a_3
$w(t)$	w_4	w_3	w_2	w_1

Поскольку функция выходов λ для автомата Мура зависит только от одного параметра $a(t)$, при совмещении таблиц переходов и выходов достаточно в таблицу переходов добавить строку выходных символов из таблицы выходов. Полученная таблица называется отмеченной таблицей переходов (ОТП) автомата Мура (табл. 6).

Другим способом задания автоматов является *графический способ*. В этом случае поведение автомата описывается с помощью ориентированного графа. На

рис. 2 изображен граф переходов-выходов автомата Мили, функции переходов и выходов которого приводились в табл. 3.

Таблица 6

	w_4	w_3	w_2	w_1
	a_0	a_1	a_2	a_3
z_1	a_3	a_2	a_1	a_0
z_2	a_0	a_3	a_2	a_1
z_3	a_1	a_0	a_3	a_2

Каждому состоянию автомата соответствует вершина графа, в которую записывается это состояние. Если имеется переход из состояния a_i в состояние a_j, вершины с этими состояниями соединяются дугой, направленной к a_j. На дуге ставится отметка, состоящая из двух значений, разделенных косой чертой: *входной символ*, под воздействием которого осуществляется переход, и *выходной символ*, формируемый при этом переходе.

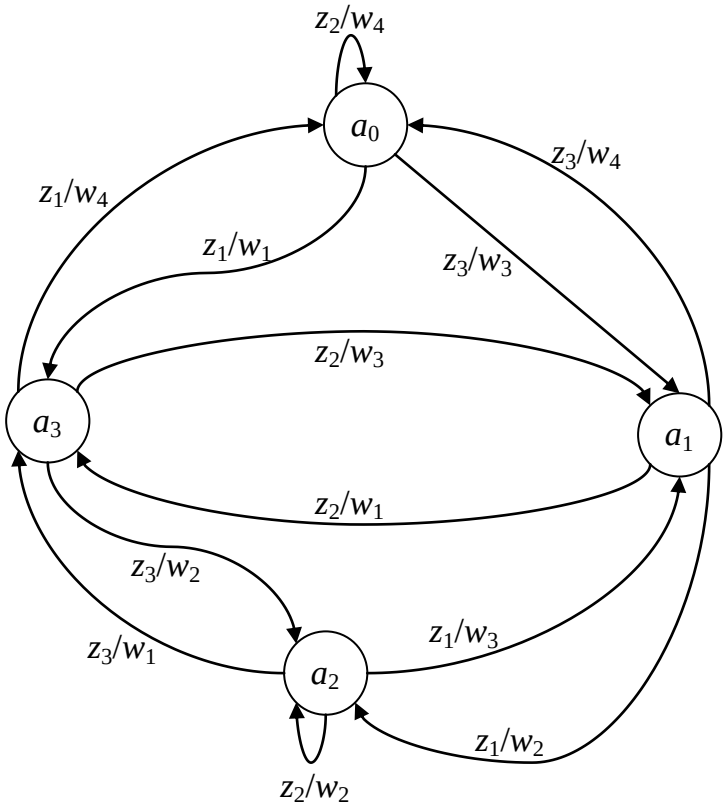


Рис. 2. Граф автомата Мили (табл. 3)

На рис. 3 изображен граф переходов-выходов автомата Мура, построенный по таблице переходов-выходов (табл. 6). В данном случае, поскольку выходной

символ в автомате Мура зависит только от состояния, в вершину графа записывается и состояние, и соответствующий ему выходной символ. Кроме того, каждая дуга графа снабжается только одной отметкой с входным символом, под воздействием которого осуществляется данный переход.

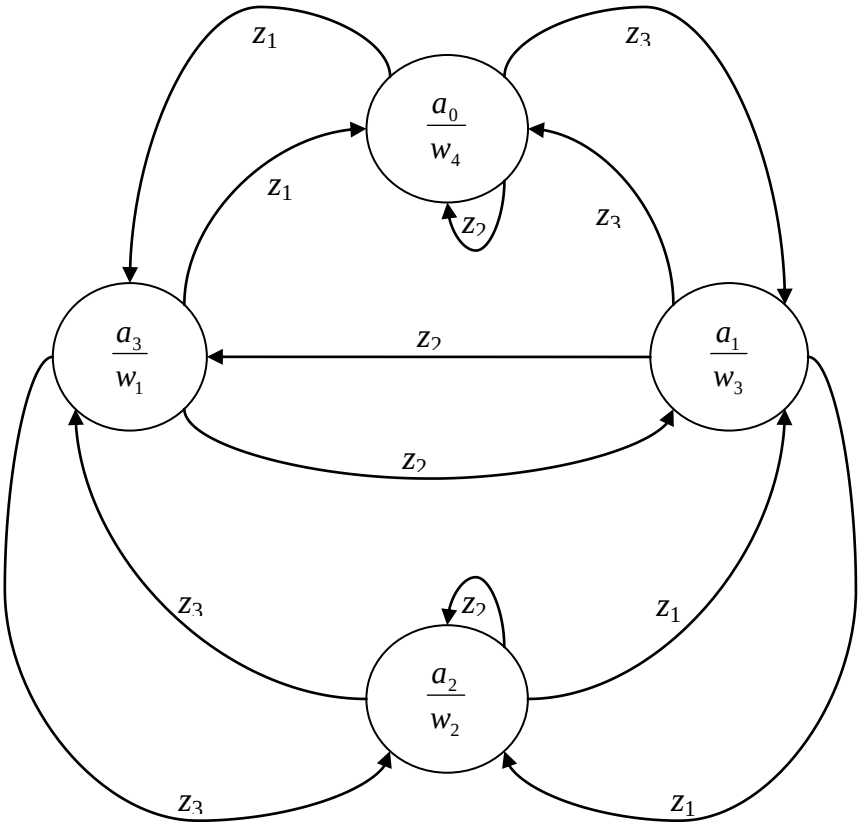


Рис. 3. Граф автомата Мура (табл. 6)

В приведенных примерах автоматов функции δ и λ определены при всех значениях их аргументов $a(t)$ и $z(t)$. Такие автоматы называются *полностью определенными*. Если функции δ и λ определены не на всех парах $a(t)$ и $z(t)$, такой автомат называется *частично определенным*. Для таких автоматов в таблицах на месте неопределенных состояний перехода или выходных символов ставятся прочерки. Частично определенные автоматы будут рассмотрены в работе 3.

1.3. Понятие автоматной ленты

Выше упоминалось, что основной функцией автомата является преобразование входных слов в выходные слова. Для формирования выходных слов по заданным входным словам строят *автоматные ленты*, содержащие три

строки. В первой строке содержится последовательность входных символов, во второй – соответствующая последовательность состояний, в третьей – последовательность выходных символов. Основой для построения автоматной ленты служит граф или таблицы переходов и выходов автомата. Пример автоматной ленты для автомата Мили (табл. 3) для входной последовательности z_3 z_1 z_2 z_3 z_3 z_1 z_2 z_2 z_3 z_1 приведен на рис. 4 (строка с номерами тактов является необязательной).

Такты	0	1	2	3	4	5	6	7	8	9	10
Входной символ	z_3	z_1	z_2	z_3	z_3	z_1	z_2	z_2	z_3	z_1	
Состояние	a_0	a_1	a_2	a_2	a_3	a_2	a_1	a_3	a_1	a_0	a_3
Выходной символ	w_3	w_2	w_2	w_1	w_2	w_3	w_1	w_3	w_4	w_1	

Рис. 4. Автоматная лента для автомата Мили

Покажем, как формируются первые три символа строки состояний и строки выходных символов. Пусть $a(0) = a_0$. Из табл. 3 следует, что

$$w(0) = \lambda(a(0), z(0)) = \lambda(a_0, z_3) = w_3;$$

$$a(1) = \delta(a(0), z(0)) = \delta(a_0, z_3) = a_1.$$

Для того чтобы лучше представить себе процесс обработки автоматом входного слова, рекомендуется сначала записывать символ w_3, а затем – символ a_1. Такой же порядок записи (сначала выходной символ, а затем – символ состояния перехода) рекомендуется соблюдать и для остальных тактов.

В такте $t = 1$:

$$w(1) = \lambda(a(1), z(1)) = \lambda(a_1, z_1) = w_2;$$

$$a(2) = \delta(a(1), z(1)) = \delta(a_1, z_1) = a_2.$$

В такте $t = 2$:

$$w(2) = \lambda(a(2), z(2)) = \lambda(a_2, z_2) = w_2;$$

$$a(3) = \delta(a(2), z(2)) = \delta(a_2, z_2) = a_2.$$

Таким же образом формируются и остальные символы строки состояний и строки выходных символов автоматной ленты.

Для такого же входного слова построим автоматную ленту (рис. 5) для автомата Мура по его ОТП (табл. 6). Так же, как и для автомата Мили, лента

содержит три строки: строку входных символов, строку состояний и строку выходных символов (строка с номерами тактов является необязательной).

Здесь следует учесть, что поскольку выходной символ в автомате Мура зависит только от состояния перехода, в такте $t = 0$ он не формируется, первый выходной символ $w(1) = \lambda(a(1))$.

Кроме того, для того чтобы яснее представить процесс обработки автоматом входного слова, при заполнении строк автоматной ленты для автомата Мура в каждом такте сначала рекомендуется записывать состояние перехода, а затем – выходной символ.

Такты	0	1	2	3	4	5	6	7	8	9	10
Входной символ	z_3	z_1	z_2	z_3	z_3	z_1	z_2	z_2	z_3	z_1	
Состояние	a_0	a_1	a_2	a_2	a_3	a_2	a_1	a_3	a_1	a_0	a_3
Выходной символ	-	w_3	w_2	w_2	w_1	w_2	w_3	w_1	w_3	w_4	w_1

Рис. 5. Автоматная лента для автомата Мура

По виду автоматной ленты можно сделать вывод: в отличие от автомата Мили автомат Мура работает с задержкой на один такт.

1.4. Минимизация полностью определенных АА расщеплением классов эквивалентных состояний

Основной задачей абстрактного синтеза автомата является переход от задания автомата на исходном языке (например, в виде оператора соответствия (см. работу 3)) к описанию на формализованном языке (обычно к таблицам переходов-выходов либо графам).

Исходя из одного исходного задания, можно построить множество различных соответствующих этому описанию абстрактных автоматов. Поэтому на этапе абстрактного синтеза можно поставить дополнительную задачу, например, построения автомата с минимально возможным числом состояний, реализующего исходное задание (очевидно, что автомат с большим числом состояний в дальнейшем будет реализовать сложнее, чем автомат с меньшим числом состояний). Такая задача называется *минимизацией числа состояний*

абстрактного автомата. Для решения ее необходимо ввести понятие об эквивалентности состояний автоматов и об эквивалентности автоматов.

Состояние a_i автомата A_1 *эквивалентно* состоянию a_j автомата A_2, если автомат A_1 в состоянии a_i и автомат A_2 в состоянии a_j под воздействием любой последовательности входных символов вырабатывают одинаковые выходные последовательности. В частности, состояния a_i и a_j могут принадлежать одному автомату (т. е. $A_1 = A_2$).

Два автомата A_1 и A_2 *эквивалентны*, если каждому состоянию a_i автомата A_1 эквивалентно по крайней мере одно состояние автомата A_2 и если при этом каждому состоянию a_j автомата A_2 эквивалентно по крайней мере одно состояние автомата A_1.

Группа эквивалентных состояний, принадлежащих одному автомату, может заменяться одним состоянием. В результате такой замены можно получить новый автомат, эквивалентный заданному, но с меньшим числом состояний. Таким образом, задача минимизации числа состояний абстрактного автомата сводится к отысканию автомата, эквивалентного заданному, с минимально возможным числом состояний.

При минимизации все состояния исходного автомата разбиваются на попарно непересекающиеся группы эквивалентных состояний, затем каждая группа эквивалентных состояний замещается одним состоянием. Получающийся минимальный автомат имеет столько же состояний, на сколько групп разбиваются состояния исходного автомата.

Существуют различные методы минимизации АА. Рассмотрим минимизацию расщеплением классов эквивалентных состояний [1].

1.4.1. Минимизация автоматов Мили

Два состояния a_i и a_j абстрактного автомата называются *одноэквивалентными*, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: в состоянии a_i при подаче входного символа z_k автомат выработает такой же выходной символ, как и при подаче входного символа z_k в состоянии a_j.

На первом этапе минимизации составляются группы одноэквивалентных состояний автомата. Совокупность всех групп одноэквивалентных состояний абстрактного автомата образует *первый класс эквивалентности*.

Два одноэквивалентных состояния a_i и a_j называются *двухэквивалентными*, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: из a_i при подаче входного символа z_k автомат переходит в состояние, находящееся в той же группе одноэквивалентности, что и состояние, в которое переходит автомат под действием z_k из a_j.

На втором этапе минимизации составляются группы двухэквивалентных состояний. Совокупность всех групп двухэквивалентных состояний образует *второй класс эквивалентности*.

Можно распространить приведенные выше определения до n-эквивалентных состояний и n-го класса эквивалентности.

Два (n-1)-эквивалентных состояния a_i и a_j называются *n-эквивалентными*, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: из a_i при подаче входного символа z_k автомат переходит в состояние, находящееся в той же группе (n-1)-эквивалентности, что и состояние, в которое переходит автомат под действием z_k из a_j.

Важно отметить, что n-эквивалентные состояния – это эквивалентные состояния для входных слов длины n.

Если i-ый класс эквивалентности совпадает с (i+1)-ым классом, то он совпадает и со всеми последующими до бесконечного классами и называется бесконечным или *финальным классом эквивалентности*.

Требуемое разбиение состояний абстрактного автомата на группы эквивалентности определяется финальным классом эквивалентности. Существенно, что финальный класс эквивалентности можно получить за конечное число шагов.

Таким образом, **на третьем и последующих этапах** минимизации строятся 3-ий и последующие классы эквивалентности до тех пор, пока не будет получено два совпадающих класса.

На конечном этапе минимизации каждая группа состояний из финального класса заменяется одним состоянием.

Пример 1. Полностью определенный автомат Мили задан совмещенной таблицей переходов-выходов (табл. 7). Необходимо получить автомат, эквивалентный заданному, с минимальным числом состояний.

Таблица 7

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_3/w_1	a_2/w_1	a_1/w_2	a_5/w_1	a_1/w_1	a_3/w_2	a_7/w_2	a_0/w_1
z_2	a_2/w_2	a_4/w_2	a_6/w_1	a_0/w_2	a_5/w_2	a_6/w_1	a_4/w_1	a_2/w_2

На этапе 1 выпишем разбиение на группы одноэквивалентных состояний, которое выглядит следующим образом: $b_0 = \{a_0, a_1, a_3, a_4, a_7\}$, $b_1 = \{a_2, a_5, a_6\}$. Получили первый класс эквивалентности.

На этапе 2 найдем разбиение на группы двухэквивалентных состояний. С этой целью на основе исходной СТПВ (табл. 7) составим вспомогательную таблицу (табл. 8).

Таблица 8

	b_0					**b_1**		
	a_0	a_1	a_3	a_4	a_7	a_2	a_5	a_6
z_1	b_0	b_1	b_1	b_0	b_0	b_0	b_0	b_0
z_2	b_1	b_0	b_0	b_1	b_1	b_1	b_1	b_0

В таблице 8 в заголовках столбцов сначала перечислены состояния из первой группы одноэквивалентных состояний, затем – из второй. В столбце для каждого состояния a_i указаны группы b_i одноэквивалентных состояний, в которые осуществляется переход из a_i под действием символов z_1 и z_2. На основе таблицы 8 удобно выполнить формирование групп двухэквивалентных состояний путем расщепления групп одноэквивалентных состояний:

$$c_0 = \{a_0, a_4, a_7\},\ c_1 = \{a_1, a_3\},\ c_2 = \{a_2, a_5\},\ c_3 = \{a_6\}.$$

Получили второй класс эквивалентности.

Поскольку второй класс эквивалентности не совпал с первым классом, переходим к **этапу 3**, на котором будем искать группы трехэквивалентных состояний. Предварительно на основе полученного на этапе 2 разбиения

состояний на группы и на основе исходной СТПВ (табл. 7) построим вспомогательную таблицу (табл. 9) аналогичную таблице 8.

Таблица 9

	c_0			c_1		c_2		c_3
	a_0	a_4	a_7	a_1	a_3	a_2	a_5	a_6
z_1	c_1	c_1	c_0	c_2	c_2	c_1	c_1	c_0
z_2	c_2	c_2	c_2	c_0	c_0	c_3	c_3	c_0

Из таблицы 9 выписываем группы трехэквивалентных состояний:

$$d_0 = \{a_0, a_4\},\ d_1 = \{a_7\},\ d_2 = \{a_1, a_3\},\ d_3 = \{a_2, a_5\},\ d_4 = \{a_6\}.$$

Получили третий класс эквивалентности.

Поскольку третий класс эквивалентности не совпал со вторым классом, переходим к **этапу 4**, на котором будем искать группы четырехэквивалентных состояний. Для этого снова построим вспомогательную таблицу (табл. 10).

Таблица 10

	d_0		d_1	d_2		d_3		d_4
	a_0	a_4	a_7	a_1	a_3	a_2	a_5	a_6
z_1	d_2	d_2	d_0	d_3	d_3	d_2	d_2	d_1
z_2	d_3	d_3	d_3	d_0	d_0	d_4	d_4	d_0

Группы четырехэквивалентных состояний имеют следующий вид: $e_0 = \{a_0, a_4\}$, $e_1 = \{a_7\}$, $e_2=\{a_1, a_3\}$, $e_3 = \{a_2, a_5\}$, $e_4 = \{a_6\}$. Получили четвертый класс эквивалентности.

Четвертый класс эквивалентности совпал с третьим классом эквивалентности, следовательно, он является финальным классом. Требуемое разбиение состояний абстрактного автомата на группы эквивалентности получено. Минимальный автомат имеет пять состояний: d_0, d_1, d_2, d_3, d_4.

Исходя из СТПВ исходного автомата (табл. 7) и полученного объединения состояний в финальные группы построим СТПВ минимального автомата (табл. 11). Один из способов построения такой СТПВ состоит в следующем: нужно в табл. 7 заменить имена состояний a_i на имена групп d_j, в которых они оказались, а затем удалить повторяющиеся столбцы.

Таким образом, мы получили автомат Мили, эквивалентный заданному, имеющий пять состояний, вместо исходных восьми.

Таблица 11

	d_0	d_1	d_2	d_3	d_4
z_1	d_2/w_1	d_0/w_1	d_3/w_1	d_2/w_2	d_1/w_2
z_2	d_3/w_2	d_3/w_2	d_0/w_2	d_4/w_1	d_0/w_1

Для проверки правильности выполненной минимизации АА построим автоматные ленты для исходного и минимального АА с одним и тем же входным словом (рис. 6). При этом входное слово должно быть составлено так, чтобы при его обработке исходный автомат выполнил все возможные переходы (то есть путь по графу должен проходить по всем дугам).

В случае если выходные слова автоматов не совпадут – минимизация проведена неверно.

а)

Входной символ	z_1	z_1	z_2	z_2	z_1	z_2	z_2	z_1	z_2
Состояние	a_0	a_3	a_5	a_6	a_4	a_1	a_4	a_5	a_3
Выходной символ	w_1	w_1	w_1	w_1	w_1	w_2	w_2	w_2	w_2

Входной символ	z_2	z_1	z_1	z_2	z_1	z_2	z_2	z_1	z_1	
Состояние	a_0	a_2	a_1	a_2	a_6	a_7	a_2	a_6	a_7	a_0
Выходной символ	w_2	w_2	w_1	w_1	w_2	w_2	w_1	w_2	w_2	

б)

Входной символ	z_1	z_1	z_2	z_2	z_1	z_2	z_2	z_1	z_2
Состояние	d_0	d_2	d_3	d_4	d_0	d_2	d_0	d_3	d_2
Выходной символ	w_1	w_1	w_1	w_1	w_1	w_2	w_2	w_2	w_2

Входной символ	z_2	z_1	z_1	z_2	z_1	z_2	z_2	z_1	z_1	
Состояние	d_0	d_3	d_2	d_3	d_4	d_1	d_3	d_4	d_1	d_0
Выходной символ	w_2	w_2	w_1	w_1	w_2	w_2	w_1	w_2	w_2	

Рис. 6. Автоматные ленты

для исходного (а) и минимизированного (б) автоматов Мили

Выходные слова совпали, следовательно, полученный минимизированный автомат эквивалентен исходному.

1.4.2. Минимизация автоматов Мура

При минимизации автоматов Мура вводится понятие нуль-эквивалентных состояний и нулевого класса эквивалентности.

Два состояния автомата Мура называются *нуль-эквивалентными*, если они отмечены одинаковыми выходными символами.

На нулевом этапе минимизации составляются группы нуль-эквивалентных состояний автомата. Совокупность всех групп нуль-эквивалентных состояний абстрактного автомата образует *нулевой класс эквивалентности*.

Два нуль-эквивалентных состояния a_i и a_j автомата Мура называются *одноэквивалентными*, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: из a_i при подаче входного символа z_k автомат переходит в состояние, находящееся в той же группе нуль-эквивалентности, что и состояние, в которое переходит автомат под действием z_k из a_j.

На первом этапе минимизации составляются группы одноэквивалентных состояний автомата. Совокупность всех групп одноэквивалентных состояний абстрактного автомата образует *первый класс эквивалентности*.

Все дальнейшие классы эквивалентности состояний, а также этапы минимизации для автомата Мура определяются так же, как для автоматов Мили.

Пример 2. Автомат Мура задан следующей отмеченной таблицей переходов (табл. 12). Необходимо построить автомат, эквивалентный заданному, имеющий минимальное число состояний.

Таблица 12

	w_1	w_2	w_1	w_1	w_2	w_2	w_1	w_2	w_3	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
z_1	a_6	a_2	a_3	a_0	a_2	a_9	a_3	a_2	a_1	a_0
z_2	a_2	a_7	a_9	a_9	a_5	a_1	a_2	a_4	a_0	a_2
z_3	a_4	a_0	a_8	a_4	a_3	a_6	a_1	a_3	a_8	a_8

На этапе 0 выпишем разбиение на группы нуль-эквивалентных состояний, которое выглядит следующим образом: $b_0 = \{a_0, a_2, a_3, a_6, a_9\}$, $b_1 = \{a_1, a_4, a_5, a_7\}$, $b_2 = \{a_8\}$. Получили нулевой класс эквивалентности.

На этапе 1 найдем разбиение на группы одноэквивалентных состояний. С этой целью на основе исходной ОТП (табл. 12) и полученного разбиения на группы нуль-эквивалентных состояний составим вспомогательную таблицу (табл. 13). Эта таблица составляется по тому же принципу, что и вспомогательная таблица (табл. 8) для автомата Мили.

Таблица 13

	$\boldsymbol{b_0}$					$\boldsymbol{b_1}$				$\boldsymbol{b_2}$
	a_0	a_2	a_3	a_6	a_9	a_1	a_4	a_5	a_7	a_8
z_1	b_0	b_0	b_0	b_0	b_0	b_0	b_0	b_0	b_0	b_1
z_2	b_0	b_0	b_0	b_0	b_0	b_1	b_1	b_1	b_1	b_0
z_3	b_1	b_2	b_1	b_1	b_2	b_0	b_0	b_0	b_0	b_2

С помощью таблицы 13 удобно выполнить формирование групп одноэквивалентных состояний путем расщепления групп нуль-эквивалентных состояний:

$$c_0 = \{a_0, a_3, a_6\},\ c_1 = \{a_2, a_9\},\ c_2 = \{a_1, a_4, a_5, a_7\},\ c_3 = \{a_8\}.$$

Получили первый класс эквивалентности.

Поскольку первый класс эквивалентности не совпал с нулевым классом, переходим **к этапу 2**, в котором будем искать группы двухэквивалентных состояний. Предварительно на основе исходной ОТП (табл. 12) и полученного на этапе 1 разбиения построим вспомогательную таблицу (табл. 14).

Таблица 14

	$\boldsymbol{c_0}$			$\boldsymbol{c_1}$		$\boldsymbol{c_2}$				$\boldsymbol{c_3}$
	a_0	a_3	a_6	a_2	a_9	a_1	a_4	a_5	a_7	a_8
z_1	c_0	c_0	c_0	c_0	c_0	c_1	c_1	c_1	c_1	c_2
z_2	c_1	c_1	c_1	c_1	c_1	c_2	c_2	c_2	c_2	c_0
z_3	c_2	c_2	c_2	c_3	c_3	c_0	c_0	c_0	c_0	c_3

Группы двухэквивалентных состояний имеют вид: $d_0 = \{a_0, a_3, a_6\}$, $d_1 = \{a_2, a_9\}$, $d_2 = \{a_1, a_4, a_5, a_7\}$, $d_3 = \{a_8\}$. Второй класс эквивалентности совпал с первым классом эквивалентности, следовательно, он является финальным классом.

Требуемое разбиение состояний абстрактного автомата на группы эквивалентности получено. Минимальный автомат имеет четыре состояния: c_0, c_1, c_2, c_3.

На основе ОТП исходного автомата и полученного разбиения его состояний на группы эквивалентности построим ОТП минимального автомата (табл. 15).

Таблица 15

	w_1	w_1	w_2	w_3
	d_0	d_1	d_2	d_3
z_1	d_0	d_0	d_1	d_2
z_2	d_1	d_1	d_2	d_0
z_3	d_2	d_3	d_0	d_3

Таким образом, мы получили автомат, эквивалентный заданному, имеющий всего четыре состояния вместо исходных десяти.

а)

Входной символ	z_3	z_1	z_2	z_2	z_3	z_2	z_1	z_3	z_2	z_1	z_1	z_2	z_3	z_3
Состояние	a_0	a_4	a_2	a_9	a_2	a_8	a_0	a_6	a_1	a_7	a_2	a_3	a_9	a_8
Выходной символ	-	w_2	w_1	w_1	w_1	w_3	w_1	w_1	w_2	w_2	w_1	w_1	w_1	w_3

Входной символ	z_1	z_1	z_1	z_3	z_2	z_3	z_3	z_2	z_2	z_3	z_1	z_2	z_1	z_3
Состояние	a_8	a_1	a_2	a_3	a_4	a_5	a_6	a_1	a_7	a_4	a_3	a_0	a_2	a_3
Выходной символ	w_3	w_2	w_1	w_1	w_2	w_2	w_1	w_2	w_2	w_2	w_1	w_1	w_1	w_1

Входной символ	z_2	z_1	z_1	z_1	z_2	z_3	z_1	z_2	z_3	z_3	z_2	z_2	z_3	
Состояние	a_4	a_5	a_9	a_0	a_6	a_2	a_8	a_1	a_7	a_3	a_4	a_5	a_1	a_0
Выходной символ	w_2	w_2	w_1	w_1	w_1	w_1	w_3	w_2	w_2	w_1	w_2	w_2	w_2	w_1

б)

Входной символ	z_3	z_1	z_2	z_2	z_3	z_2	z_1	z_3	z_2	z_1	z_1	z_2	z_3	z_3
Состояние	d_0	d_2	d_1	d_1	d_1	d_3	d_0	d_0	d_2	d_2	d_1	d_0	d_1	d_3
Выходной символ	-	w_2	w_1	w_1	w_1	w_3	w_1	w_1	w_2	w_2	w_1	w_1	w_1	w_3

Входной символ	z_1	z_1	z_1	z_3	z_2	z_3	z_3	z_2	z_2	z_3	z_1	z_2	z_1	z_3
Состояние	d_3	d_2	d_1	d_0	d_2	d_2	d_0	d_2	d_2	d_2	d_0	d_0	d_1	d_0
Выходной символ	w_3	w_2	w_1	w_1	w_2	w_2	w_1	w_2	w_2	w_2	w_1	w_1	w_1	w_1

Входной символ	z_2	z_1	z_1	z_1	z_2	z_3	z_1	z_2	z_3	z_3	z_2	z_2	z_3	
Состояние	d_2	d_2	d_1	d_0	d_0	d_1	d_3	d_2	d_2	d_0	d_2	d_2	d_2	d_0
Выходной символ	w_2	w_2	w_1	w_1	w_1	w_1	w_3	w_2	w_2	w_1	w_2	w_2	w_2	w_1

Рис. 7. Автоматные ленты

для исходного (а) и минимизированного (б) автоматов Мура

Для проверки правильности выполненной минимизации АА будем подавать на вход исходного (табл. 12) и минимального (табл. 15) АА одну и ту же последовательность входных символов и построим автоматные ленты (рис. 7). При этом входное слово должно быть составлено так, чтобы при его обработке исходный автомат выполнил все возможные переходы.

Выходные последовательности исходного и минимального АА совпали, следовательно, минимизация проведена верно и полученный автомат эквивалентен исходному.

1.5. Минимизация полностью определенных АА с использованием треугольной таблицы

Другой метод минимизации числа состояний АА основан на использовании треугольной таблицы. Процесс минимизации АА указанным методом можно разделить на несколько этапов:

- построение треугольной таблицы и ее заполнение,
- проверка полученных условий объединения состояний в группу на непротиворечивость,
- составление классов,
- построение минимизированного автомата.

1.5.1. Построение треугольной таблицы и ее использование для минимизации АА модели Мили

Рассмотрим процесс минимизации полностью определенного абстрактного автомата модели Мили.

Таблица для минимизации абстрактного автомата содержит n–1 строку и n–1 столбец, где n – число состояний абстрактного автомата. Строки таблицы соответствуют состояниям a_1, a_2, ..., a_{n-1}, а столбцы – состояниям a_0, a_1, a_2, ..., a_{n-2}. В клетку таблицы на пересечении строки a_i и столбца a_j записывается условие объединения состояний a_i и a_j в одну группу. Поскольку содержимое клеток таблицы с парой заголовков (a_i, a_j) и (a_j, a_i) идентично по определению, половину

таблицы можно исключить, после чего прямоугольная таблица приобретает вид треугольной таблицы (см. табл. 17 ниже).

Заполнение таблицы осуществляется за два этапа.

На первом этапе все клетки треугольной таблицы заполняются условиями объединения соответствующих состояний в группы. В клетку, находящуюся на пересечении столбца с заголовком a_j и строки с заголовком a_i (эти заголовки будем называть координатами клетки), заносятся условия объединения в группу состояний a_j и a_i.

При заполнении клетки с координатами (a_j, a_i) следует руководствоваться следующими правилами.

Если состояния a_j и a_i не являются одноэквивалентными, это означает, что данные состояния не могут быть в одной группе *ни при каких условиях* и в соответствующую клетку таблицы заносится знак ×.

Если состояния a_j и a_i являются одноэквивалентными в соответствующую клетку таблицы заносятся условия объединения их в группу, которые состоят в следующем.

Два одноэквивалентных состояния a_i и a_j могут быть объединены в одну группу, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: из a_i при подаче входного символа z_k автомат переходит в состояние, находящееся в той же группе, что и состояние, в которое переходит автомат под действием z_k из a_j.

Поскольку на момент заполнения клетки информация относительно попадания состояний перехода в одну группу является неизвестной, пары состояний перехода для каждого входного символа записываются в клетку таблицы в качестве *условия* объединения в группу состояний a_j и a_i. Таким образом, число пар состояний перехода, записываемых в клетку таблицы, в общем случае равно числу символов во входном алфавите автомата.

Два одноэквивалентных состояния a_i и a_j могут быть объединены в одну группу *безусловно*, если для каждого символа z_k из множества входных символов автомата справедливо утверждение: из a_i при подаче входного символа z_k автомат

переходит в то же состояние, в которое переходит автомат под действием z_k из a_j. В этом случае в соответствующую клетку таблицы заносится знак $\vee$.

На втором этапе выполняется анализ условий объединения, записанных в клетках таблицы, на непротиворечивость с заменой их на знак $\vee$, если все условия, записанные в клетке, непротиворечивы, или на знак $\times$, если хотя бы одно условие в клетке оказалось противоречивым. Пример проверки условий на непротиворечивость приведен ниже.

После того как треугольная таблица будет окончательно заполнена, приступают к объединению состояний АА в группы. Для этого по клеткам треугольной таблицы, заполненным знаком $\vee$, сначала выписываются все пары эквивалентных состояний исходного АА. Таким образом, получаем группы, каждая из которых состоит из двух состояний.

Далее делается попытка сформировать на основе получившихся групп группы из большего числа состояний так, чтобы все состояния, оказавшиеся в одной группе, были попарно эквивалентны. Например, чтобы состояния a, b, c, d попали в одну группу, должны иметь место пары: (a, b), (a, c), (a, d), (b, c), (b, d), (c, d). Так как в конечном итоге каждая группа будет заменена одним состоянием, для минимизации числа состояний АА необходимо, чтобы групп состояний было как можно меньше, а значит, сами группы должны быть как можно более крупными.

Если какое-либо из состояний АА не вошло ни в одну группу, для него необходимо создать отдельную группу, состоящую из одного состояния.

На конечном этапе каждая группа заменяется одним состоянием, которое и является состоянием минимального автомата. На основании состава групп и исходной таблицы переходов-выходов строится таблица переходов-выходов минимального автомата.

Описанный выше процесс минимизации проиллюстрируем на примере автомата, СТПВ которого приведена в табл. 16.

Таблица 16

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_3/w_1	a_7/w_1	a_1/w_2	a_3/w_1	a_0/w_1	a_1/w_2	a_7/w_2	a_1/w_1
z_2	a_4/w_2	a_5/w_2	a_5/w_1	a_4/w_2	a_7/w_2	a_6/w_1	a_2/w_1	a_5/w_2

Для заданного автомата построим треугольную таблицу (табл. 17). Таблица будет иметь 7 строк и 7 столбцов.

Выполним первый этап заполнения треугольной таблицы. При заполнении клеток будем заносить в них индексы состояний вместо полного их обозначения, например, вместо a_3, a_7 запишем: 3,7.

На втором этапе проверяем условия, записанные в клетках табл. 17, на непротиворечивость.

Рассмотрим клетку с координатами a_0 и a_1. В ней содержатся два условия. Из табл. 17 видно, что условие (4, 5) не выполняется (см. клетку с координатами a_4 и a_5). Поэтому и состояния a_0 и a_1 не могут находиться в одной группе и в данную клетку заносим знак ×. Рассуждая подобным образом, запишем знак × в клетки с координатами (a_0, a_7), (a_1, a_3), (a_1, a_4), (a_3, a_7), (a_4, a_7).

Таблица 17

a_1	3,7 4,5 ×						
a_2	×	×					
a_3	∨	3,7 4,5 ×	×				
a_4	0,3 4,7 ×	0,7 5,7 ×	×	0,3 4,7 ×			
a_5	×	×	5,6 ∨	×	×		
a_6	×	×	1,7 2,5 ∨	×	×	1,7 2,6 ∨	
a_7	1,3 4,5 ×	1,7 ∨	×	1,3 4,5 ×	0,1 5,7 ×	×	×
	a_0	a_1	a_2	a_3	a_4	a_5	a_6

Примечание к табл. 17: знаки ∨ и × в клетках, в которых записаны условия, на первом этапе отсутствуют; они будут внесены в эти клетки на втором этапе.

Клетки с координатами (a_0, a_4) и (a_3, a_4) содержат условие (4,7), которое оказалось противоречивым, поэтому в эти клетки заносим ×.

Очевидно непротиворечивым является условие (1,7), находящееся в клетке с координатами (a_1, a_7).

При рассмотрении клетки (a_2, a_5) образуется достаточно сложная цепочка условий. В этой клетке содержится условие (5,6). В клетке с координатами (a_5, a_6) вновь содержатся условия: (1,7), которое является непротиворечивым, и (2,6). В клетке с координатами (a_2, a_6) вновь содержится непротиворечивое условие (1,7) и условие (2, 5). Все множество условий, необходимых для объединения состояний a_2 и a_5 в группу, удобно изобразить в виде дерева (рис. 8).

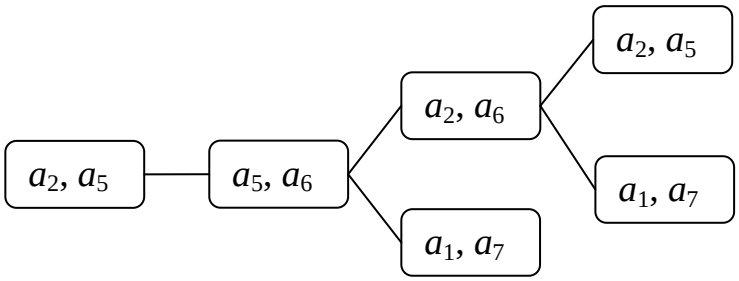


Рис. 8. Дерево условий для объединения состояний a_2 и a_5 в группу.

Мы получили замкнутую цепочку условий, среди которых нет противоречивых, поэтому в клетках, участвующих в этой цепочке (т.е. в клетках с координатами (a_2, a_5), (a_5, a_6) и (a_2, a_6)), поставим знак ∨.

После того как в каждой клетке треугольной таблицы окажется «крестик» или «галочка», можно приступать к объединению состояний в группы.

По клеткам, содержащим знаки ∨, выпишем пары эквивалентных состояний, а для состояния a_4, не попавшего ни в одну группу, создадим отдельную группу:

$$(a_0, a_3), (a_1, a_7), (a_2, a_5), (a_2, a_6), (a_5, a_6), (a_4).$$

Укрупним полученные группы объединив попарно эквивалентные состояния a_2, a_5, a_6, и заменим каждую группу одним состоянием минимального автомата:

$$b_0 = (a_0, a_3),\ b_1 = (a_1, a_7),\ b_2 = (a_2, a_5, a_6),\ b_3 = (a_4).$$

На основании табл. 16 и состава групп построим совмещенную таблицу переходов-выходов минимального автомата (табл. 18).

Таблица 18

	b_0	b_1	b_2	b_3
z_1	b_0/w_1	b_1/w_1	b_1/w_2	b_0/w_1
z_2	b_3/w_2	b_2/w_2	b_2/w_1	b_1/w_2

а)

Входной символ	z_1	z_1	z_2	z_1	z_2	z_2	z_1	z_1	z_2	z_1
Состояние	a_0	a_3	a_3	a_4	a_0	a_4	a_7	a_1	a_7	a_5
Выходной символ	w_1	w_1	w_2	w_1	w_2	w_2	w_1	w_1	w_2	w_2

Входной символ	z_2	z_2	z_2	z_2	z_2	z_2	z_1	z_2	z_2	z_1	
Состояние	a_1	a_5	a_6	a_2	a_5	a_6	a_2	a_1	a_5	a_6	a_7
Выходной символ	w_2	w_1	w_1	w_1	w_1	w_1	w_2	w_2	w_1	w_2	

б)

Входной символ	z_1	z_1	z_2	z_1	z_2	z_2	z_1	z_1	z_2	z_1
Состояние	b_0	b_0	b_0	b_3	b_0	b_3	b_1	b_1	b_1	b_2
Выходной символ	w_1	w_1	w_2	w_1	w_2	w_2	w_1	w_1	w_2	w_2

Входной символ	z_2	z_2	z_2	z_2	z_2	z_2	z_1	z_2	z_2	z_1	
Состояние	b_1	b_2	b_2	b_2	b_2	b_2	b_2	b_1	b_2	b_2	b_1
Выходной символ	w_2	w_1	w_1	w_1	w_1	w_1	w_2	w_2	w_1	w_2	

Рис. 9. Автоматные ленты

для исходного (а) и минимизированного (б) автоматов Мили

Для проверки эквивалентности исходного и минимального автоматов составим для них автоматные ленты с одним и тем же входным словом, при трассировке которого исходный автомат выполняет все возможные для него переходы (рис. 9).

Выходные слова совпали, следовательно, полученный минимальный автомат эквивалентен исходному, минимизация выполнена верно.

1.5.2. Построение треугольной таблицы и ее использование для минимизации АА модели Мура

Принципы минимизации и правила работы с треугольной таблицей для автомата Мура те же, что и для автомата Мили (см. раздел 1.5.1). Единственное отличие состоит в следующем.

При заполнении треугольной таблицы для автомата Мили используется понятие одноэквивалентных состояний. Для автомата Мура следует вместо этого использовать понятие нуль-эквивалентных состояний (см. раздел 1.4). Если состояния a_j и a_i не являются нуль-эквивалентными, в клетку, для которой эти состояния являются заголовками (координатами), заносится знак ×. В противном случае в клетку записываются условия объединения состояний a_j и a_i в группу, так же как для автомата Мили.

С учетом этих замечаний для минимизации автомата Мура можно полностью использовать материал из раздела 1.5.1.

Рассмотрим пример. Пусть автомат модели Мура задан отмеченной таблицей переходов (табл. 19).

Таблица 19

	w_1	w_2	w_3	w_1	w_2	w_2	w_3	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_4	a_3	a_5	a_2	a_3	a_1	a_5	a_4
z_2	a_6	a_7	a_3	a_1	a_0	a_2	a_3	a_2

Построим и заполним треугольную таблицу, которая для данного примера будет иметь 7 строк и 7 столбцов (табл. 20).

На первом этапе заполнения таблицы, так же как и в примере для автомата Мили, в данном случае в клетку может быть записано две пары состояний перехода (по числу различных входных символов).

Таблица 20

a_1	×					
a_2	×	×				
a_3	2,4 1,6 ×	×	×			
a_4	×	0,7 ∨	×	×		
a_5	×	1,3 2,7 ×	×	×	1,3 0,2 ×	
a_6	×	×	∨	×	×	×
a_7	2,6 ∨	×	×	2,4 1,2 ×	×	×
	a_0	a_1	a_2	a_3	a_4	a_5	a_6

Примечание к табл. 20: знаки ∨ и × в клетках, в которых записаны условия, на первом этапе отсутствуют; они будут внесены в эти клетки на втором этапе.

На втором этапе заполнения таблицы условия, записанные в клетках, проверяются на непротиворечивость. В те клетки, которые содержат хотя бы одно противоречивое условие, заносится знак ×.

В данном примере два условия оказались непротиворечивыми. В клетку с координатами (a_0, a_7) занесем знак ∨, так как в клетке с координатами (a_2, a_6) стоит знак ∨. Таким образом, получаем непротиворечивое условие (0, 7) для объединения пары (a_1, a_4), поэтому в клетку с координатами (a_1, a_4) также занесем знак ∨.

В результате второго этапа заполнения треугольной таблицы в клетках должны содержатся знаки × либо ∨.

По клеткам, содержащим знак ∨, выпишем группы состояний, а для состояний, не вошедших ни в одну пару, создадим отдельные группы:

$$(a_0, a_7), (a_1, a_4), (a_2, a_6), (a_3), (a_5).$$

Так как каждое состояние исходного автомата входит только в одну группу, укрупнение групп невозможно.

Каждую группу заменим состоянием минимального автомата:

$b_0 = (a_0, a_7)$, $b_1 = (a_1, a_4)$, $b_2 = (a_2, a_6)$, $b_3 = (a_3)$, $b_4 = (a_5)$.

Процедура минимизации завершается построением ОТП минимального автомата (табл. 21).

Таблица 21

	w_1	w_2	w_3	w_1	w_2
	b_0	b_1	b_2	b_3	b_4
z_1	b_1	b_3	b_4	b_2	b_1
z_2	b_2	b_0	b_3	b_1	b_2

Для проверки эквивалентности исходного и минимального автоматов составим для них автоматные ленты с одним и тем же входным словом, при трассировке которого исходный автомат совершает все возможные переходы (рис. 10).

а)

Входной символ	z_2	z_1	z_1	z_2	z_1	z_1	z_2	z_1	z_1	z_1	z_2	z_2
Состояние	a_0	a_6	a_5	a_1	a_7	a_4	a_3	a_1	a_3	a_2	a_5	a_2
Выходной символ	-	w_3	w_2	w_2	w_1	w_2	w_1	w_2	w_1	w_3	w_2	w_3

Входной символ	z_2	z_2	z_2	z_1	z_1	z_2	z_1	z_2	z_1	z_2	z_2	z_2	
Состояние	a_3	a_1	a_7	a_2	a_5	a_1	a_7	a_4	a_0	a_4	a_0	a_6	a_3
Выходной символ	w_1	w_2	w_1	w_3	w_2	w_2	w_1	w_2	w_1	w_2	w_1	w_3	w_1

б)

Входной символ	z_2	z_1	z_1	z_2	z_1	z_1	z_2	z_1	z_1	z_1	z_2	z_2
Состояние	b_0	b_2	b_4	b_1	b_0	b_1	b_3	b_1	b_3	b_2	b_4	b_2
Выходной символ	-	w_3	w_2	w_2	w_1	w_2	w_1	w_2	w_1	w_3	w_2	w_3

Входной символ	z_2	z_2	z_2	z_1	z_1	z_2	z_1	z_2	z_1	z_2	z_2	z_2	
Состояние	b_3	b_1	b_0	b_2	b_4	b_1	b_0	b_1	b_0	b_1	b_0	b_2	b_3
Выходной символ	w_1	w_2	w_1	w_3	w_2	w_2	w_1	w_2	w_1	w_2	w_1	w_3	w_1

Рис. 10. Автоматные ленты

для исходного (а) и минимизированного (б) автоматов Мура

Выходные слова совпали, следовательно, минимизированный автомат эквивалентен исходному, минимизация выполнена верно.

2. Задание по работе

Абстрактный автомат задан совмещенной таблицей переходов-выходов или отмеченной таблицей переходов (см. раздел «Варианты заданий»). Требуется, используя метод минимизации (расщеплением классов эквивалентных состояний – для четных вариантов; с помощью треугольной таблицы – для нечетных вариантов), построить автомат, эквивалентный заданному, содержащий минимальное число состояний. Эквивалентность исходного и минимизированного автоматов необходимо проверить с помощью автоматной ленты, учитывающей все возможные переходы исходного автомата, в пакете *JFLAP*.

3. Порядок выполнения работы

Выполнение работы состоит из двух частей: минимизации АА и проверки корректности минимизации путем моделирования в пакете *JFLAP*.

При минимизации заданного АА необходимо:

1) записать исходную ОТП или СТПВ автомата;
2) составить автоматную ленту, учитывающую все возможные переходы автомата;
3) выполнить этапы минимизации АА.

Для проверки эквивалентности исходного и минимизированного автоматов в пакете *JFLAP* необходимо:

1)набрать граф исходного АА и задать начальное состояние;

2)с помощью меню *Inputs*: *Multiple Run* ввести входную последовательность символов автоматной ленты, учитывающей все возможные переходы исходного автомата; убедиться, что при трассировке ленты автомат осуществляет все возможные переходы;

3)набрать граф минимального АА и задать начальное состояние;

4)с помощью меню *Inputs*: *Multiple Run* ввести входную последовательность символов из п. 2, убедиться, что полученная последовательность выходных

символов совпадает с последовательностью выходных символов исходного автомата.

Подробные методические указания по работе в пакете *JFLAP* приведены в Приложении.

4. Содержание отчета

Отчет по работе должен содержать следующие разделы.

1. Цель работы.
2. Задание по работе.

   Раздел должен включать в себя формулировку задания и данные варианта.
3. Минимизация автомата.

   В данном разделе необходимо привести описание этапов минимизации и таблицу (СТПВ или ОТП) минимизированного автомата.
4. Проверка эквивалентности автоматов.

   В данном разделе необходимо привести: графы исходного и минимизированного автоматов, построенные в JFLAP; автоматную ленту (ы), учитывающую все возможные переходы исходного автомата; скриншоты с результатами моделирования работы исходного и минимизированного автоматов для входной последовательности из автоматной ленты.
5. Вывод.

   Пример вывода: «В результате выполнения работы произведена минимизация заданного автомата модели Мура методом расщепления классов эквивалентных состояний. Число состояний автомата сократилось с 10 до 5. Проверка эквивалентности исходного и минимизированного автоматов произведена в среде JFLAP. Изучено понятие абстрактного автомата и способы задания абстрактных автоматов, освоен метод минимизации полностью определенных автоматов расщеплением классов эквивалентных состояний».

5. Варианты заданий

Вариант 1

	w_1	w_2	w_1	w_1	w_1	w_2	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_5	a_5	a_3	a_6	a_0
z_2	a_6	a_0	a_0	a_4	a_1	a_0	a_4
z_3	a_1	a_2	a_2	a_2	a_6	a_5	a_3

Вариант 2

	w_1	w_1	w_1	w_2	w_1	w_2	w_2	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_2	a_5	a_2	a_2	a_0	a_4	a_5	a_2
z_2	a_3	a_1	a_7	a_3	a_5	a_5	a_1	a_7
z_3	a_1	a_6	a_1	a_1	a_1	a_1	a_6	a_1

Вариант 3

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_2/w_2	a_3/w_1	a_0/w_2	a_6/w_1	a_8/w_2	a_2/w_2	a_3/w_1	a_3/w_2	a_4/w_2
z_2	a_5/w_1	a_7/w_2	a_3/w_1	a_8/w_2	a_0/w_1	a_4/w_1	a_2/w_2	a_1/w_1	a_6/w_1

Вариант 4

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_5/w_1	a_7/w_2	a_3/w_2	a_4/w_1	a_7/w_1	a_6/w_1	a_5/w_2	a_1/w_1
z_2	a_1/w_2	a_2/w_1	a_0/w_1	a_6/w_2	a_6/w_2	a_0/w_2	a_2/w_1	a_4/w_2

Вариант 5

	w_2	w_3	w_1	w_2	w_2	w_3	w_1	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_2	a_4	a_1	a_2	a_0	a_4	a_3	a_3
z_2	a_6	a_6	a_0	a_7	a_1	a_7	a_1	a_5

Вариант 6

	w_1	w_2	w_3	w_1	w_2	w_2	w_3
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_4	a_4	a_1	a_1	a_1	a_4	a_5
z_2	a_6	a_0	a_3	a_2	a_0	a_3	a_3

Вариант 7

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_6/w_1	a_0/w_2	a_3/w_1	a_2/w_1	a_1/w_1	a_7/w_1	a_4/w_2	a_5/w_1
z_2	a_5/w_1	a_4/w_1	a_7/w_1	a_1/w_2	a_3/w_1	a_6/w_2	a_0/w_1	a_2/w_1

Вариант 8

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_6/w_1	a_7/w_1	a_1/w_2	a_4/w_1	a_3/w_1	a_2/w_1	a_0/w_1	a_5/w_2
z_2	a_3/w_1	a_6/w_1	a_5/w_1	a_0/w_1	a_2/w_2	a_4/w_1	a_7/w_2	a_1/w_1

Вариант 9

	w_1	w_1	w_2	w_2	w_1	w_2	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_1	a_0	a_2	a_3	a_6	a_0	a_4
z_2	a_3	a_2	a_4	a_4	a_2	a_1	a_3
z_3	a_5	a_4	a_1	a_1	a_4	a_5	a_5

Вариант 10

	w_1	w_1	w_1	w_2	w_1	w_2	w_2	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_3	a_6	a_3	a_3	a_5	a_5	a_6	a_3
z_2	a_2	a_2	a_0	a_4	a_2	a_6	a_5	a_1
z_3	a_0	a_7	a_2	a_2	a_7	a_2	a_2	a_2

Вариант 11

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_3/w_2	a_8/w_1	a_1/w_2	a_7/w_1	a_0/w_2	a_3/w_2	a_4/w_1	a_5/w_2	a_2/w_2
z_2	a_6/w_1	a_0/w_2	a_2/w_1	a_0/w_2	a_1/w_1	a_5/w_1	a_3/w_2	a_8/w_1	a_7/w_1

Вариант 12

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_4/w_1	a_3/w_2	a_5/w_1	a_7/w_1	a_3/w_1	a_2/w_1	a_5/w_2	a_3/w_1
z_2	a_7/w_1	a_7/w_1	a_1/w_1	a_1/w_2	a_0/w_1	a_6/w_2	a_2/w_1	a_6/w_1

Вариант 13

	w_1	w_2	w_3	w_1	w_2	w_2	w_3	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_5	a_4	a_6	a_3	a_7	a_2	a_2	a_5
z_2	a_6	a_0	a_2	a_2	a_1	a_3	a_6	a_2

Вариант 14

	w_1	w_2	w_3	w_1	w_2	w_2	w_3
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_5	a_0	a_4	a_5	a_3	a_2	a_1
z_2	a_4	a_6	a_1	a_1	a_6	a_3	a_4

Вариант 15

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_5/w_1	a_0/w_1	a_5/w_2	a_7/w_1	a_3/w_1	a_2/w_2	a_1/w_2	a_2/w_1
z_2	a_4/w_2	a_5/w_2	a_0/w_1	a_2/w_2	a_6/w_2	a_0/w_1	a_6/w_1	a_4/w_2

Вариант 16

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_5/w_2	a_6/w_2	a_4/w_2	a_5/w_1	a_2/w_2	a_8/w_1	a_1/w_2	a_4/w_2	a_5/w_1
z_2	a_3/w_1	a_8/w_1	a_7/w_1	a_0/w_2	a_5/w_1	a_1/w_2	a_2/w_1	a_6/w_1	a_4/w_2

Вариант 17

	w_1	w_1	w_2	w_2	w_1	w_2	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_2	a_0	a_3	a_4	a_0	a_1	a_2
z_2	a_1	a_5	a_4	a_1	a_3	a_4	a_4
z_3	a_6	a_3	a_2	a_0	a_5	a_6	a_6

Вариант 18

	w_2	w_1	w_1	w_1	w_2	w_1	w_2	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_3	a_3	a_6	a_3	a_3	a_1	a_5	a_6
z_2	a_0	a_4	a_2	a_0	a_4	a_6	a_6	a_2
z_3	a_2	a_2	a_7	a_2	a_2	a_2	a_2	a_7

Вариант 19

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_6/w_2	a_7/w_1	a_0/w_1	a_2/w_2	a_5/w_1	a_4/w_1	a_3/w_1	a_1/w_1
z_2	a_2/w_1	a_4/w_1	a_7/w_1	a_6/w_1	a_1/w_1	a_3/w_2	a_5/w_1	a_0/w_2

Вариант 20

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_6/w_2	a_5/w_2	a_0/w_1	a_4/w_2	a_3/w_2	a_6/w_1	a_5/w_1	a_3/w_2
z_2	a_5/w_1	a_7/w_1	a_1/w_2	a_5/w_1	a_6/w_1	a_4/w_2	a_3/w_2	a_2/w_1

Вариант 21

	w_1	w_2	w_3	w_1	w_2	w_2	w_3	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_6	a_2	a_7	a_4	a_3	a_6	a_0	a_2
z_2	a_7	a_5	a_5	a_3	a_2	a_1	a_1	a_0

Вариант 22

	w_1	w_2	w_3	w_1	w_2	w_2	w_3
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_6	a_4	a_6	a_2	a_1	a_0	a_2
z_2	a_3	a_2	a_5	a_0	a_6	a_1	a_5

Вариант 23

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_2/w_1	a_6/w_1	a_5/w_2	a_0/w_1	a_5/w_1	a_7/w_2	a_3/w_2	a_0/w_1
z_2	a_6/w_2	a_1/w_2	a_2/w_1	a_5/w_2	a_1/w_2	a_4/w_1	a_1/w_1	a_6/w_2

Вариант 24

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_6/w_2	a_0/w_1	a_4/w_2	a_8/w_1	a_3/w_2	a_6/w_2	a_7/w_1	a_7/w_2	a_8/w_2
z_2	a_1/w_1	a_2/w_2	a_7/w_1	a_6/w_2	a_7/w_1	a_8/w_1	a_3/w_2	a_5/w_1	a_4/w_1

Вариант 25

	w_1	w_1	w_1	w_2	w_2	w_1	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_3	a_1	a_4	a_5	a_1	a_2
z_2	a_2	a_5	a_6	a_5	a_2	a_4	a_5
z_3	a_0	a_0	a_4	a_3	a_1	a_6	a_0

Вариант 26

	w_1	w_1	w_1	w_2	w_1	w_2	w_2	w_2
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_5	a_3	a_5	a_5	a_7	a_7	a_0	a_5
z_2	a_0	a_4	a_2	a_6	a_0	a_0	a_4	a_2
z_3	a_4	a_1	a_4	a_4	a_4	a_4	a_1	a_4

Вариант 27

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_1/w_2	a_6/w_1	a_6/w_2	a_5/w_2	a_2/w_2	a_3/w_2	a_1/w_1	a_0/w_2
z_2	a_0/w_2	a_2/w_2	a_2/w_2	a_4/w_1	a_6/w_2	a_7/w_1	a_2/w_2	a_1/w_2

Вариант 28

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_5/w_1	a_6/w_2	a_3/w_1	a_0/w_2	a_2/w_1	a_5/w_1	a_6/w_2	a_8/w_1	a_7/w_1
z_2	a_8/w_2	a_4/w_1	a_6/w_2	a_2/w_1	a_3/w_2	a_7/w_2	a_1/w_1	a_5/w_2	a_0/w_2

Вариант 29

	w_1	w_2	w_3	w_1	w_2	w_2	w_3	w_1
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_7	a_2	a_1	a_5	a_6	a_6	a_5	a_7
z_2	a_1	a_0	a_6	a_4	a_3	a_7	a_2	a_5

Вариант 30

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_1/w_1	a_4/w_1	a_3/w_1	a_2/w_2	a_6/w_1	a_2/w_1	a_4/w_2	a_0/w_2
z_2	a_3/w_2	a_3/w_2	a_5/w_2	a_7/w_1	a_1/w_2	a_6/w_2	a_7/w_1	a_5/w_1

Контрольная работа 3
СИНТЕЗ КОНЕЧНЫХ АВТОМАТОВ ПО ОПЕРАТОРУ СООТВЕТСТВИЯ

Цель работы: приобретение практических навыков составления формального описания АА, заданного в виде оператора соответствия.

1. Основные теоретические сведения

Одной из форм задания абстрактного автомата является так называемый оператор соответствия (ОС), представляющий собой набор входных и соответствующих им выходных слов данного автомата [1].

1.1. Оператор соответствия

Оператор соответствия оформляется в виде таблицы. Левая часть таблицы содержит перечень входных слов, которые могут быть поданы на вход АА во время его работы. В правой части таблицы на одной строке с входным словом помещается соответствующее выходное слово, которое АА формирует при обработке этого входного слова. Пример ОС приведен на рис. 1.

$z(0)$	$z(1)$	$z(2)$	$w(0)$	$w(1)$	$w(2)$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	1	0	0	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	1	1

Рис. 1. Оператор соответствия

(в скобках указаны такты дискретного времени)

Определим условия работы АА, заданного с помощью ОС.

1. Перед поступлением на вход АА первого символа любого входного слова автомат находится в начальном состоянии a_0.

2. После поступления на вход АА последнего символа любого входного слова автомат возвращается в начальное состояние a_0.

Первым действием при работе с оператором соответствия является переход

от исходного задания АА к формализованному заданию, на основании которого можно представить АА в одном из двух видов формального описания: в табличном или в графическом. Это действие назовем приведением ОС к автоматному виду.

Выделим два условия, выполнение которых определяет автоматный вид оператора соответствия.

1. *Однозначность отображения*. Если два каких-либо входных слова имеют совпадающие начальные отрезки длиной *n* символов, соответствующие этим словам выходные слова также должны иметь совпадающие начальные отрезки длиной *n* символов.

2. *Равенство длин*. Длины входного слова и соответствующего ему выходного слова должны быть равны.

Если ОС не удовлетворяет приведенным двум условиям, выполняют *приведение ОС к автоматному виду* путем добавления к словам «пустых» символов α и β. В большинстве случаев α приписывают к входному слову справа, а β – к выходному слову слева.

Проверим выполнение условий автоматности для ОС, показанного на рис. 1, и если условия не выполняются, приведем ОС к автоматному виду.

Сначала рассмотрим начальные участки длиной один символ (первые символы входных и выходных слов). В ОС три входных слова начинаются с символа 0 (это первые три входных слова ОС), первые символы соответствующих этим словам выходных слов также одинаковы (символы 0, 0 и 0). С символа 1 начинаются три последние входные слова ОС. Первые символы соответствующих этим словам выходных слов различны (символы 1, 1 и 0), что является нарушением условия 1.

Чтобы устранить это нарушение, припишем символ β слева к каждому из последних трех выходных слов, а чтобы устранить нарушение этим действием условия 2, припишем символ α справа к каждому из последних трех входных слов (рис. 2). Теперь можно сказать, что все первые символы входных слов, равные 0, оператор отображает в выходные символы 0, а первые символы

входных слов, равные 1, оператор отображает в выходные символы β.

Затем рассмотрим начальные участки длиной два (первые два символа входных и выходных слов). Одинаковые начальные участки 00 имеют первые два входных слова, оператором они отображаются в различные начальные участки выходных слов (01 и 00), что является нарушением условия 1. Чтобы устранить это нарушение, припишем символ β слева к каждому из первых двух выходных слов, а чтобы устранить нарушение этим действием условия 2, припишем символ α справа к каждому из первых двух входных слов.

Одинаковые начальные участки 11 имеют два последних входных слова, оператором они отображаются в разные начальные участки выходных слов (β1 и β0), т.е. нарушено условие 1. Чтобы устранить это нарушение, припишем символ β слева к каждому из последних двух выходных слов, а чтобы устранить нарушение этим действием условия 2, припишем символ α справа к каждому из последних двух входных слов.

После выполнения указанных действий все начальные отрезки входных слов, равные 00, оператор отображает в отрезки β0 выходных слов, а начальные отрезки входных слов, равные 11, оператор отображает в отрезки ββ выходных слов.

Других совпадающих начальных отрезков длиной два символа во входных словах в ОС нет. Также, очевидно, во входных словах ОС нет ни одной пары совпадающих отрезков длиной три символа (т.к. все входные слова ОС различны).

0	0	0	α			β	0	1	1
0	0	1	α			β	0	0	0
0	1	1					0	0	1
1	0	1	α			β	1	0	0
1	1	0	α	α	β	β	1	1	0
1	1	1	α	α	β	β	0	1	1

Рис. 2. Приведение ОС к автоматному виду.

После дописывания символов при проверке начальных отрезков длины два и больше, может возникнуть нарушение условия 1 для начальных отрезков меньшей длины. В связи с этим в заключение необходимо снова проверить выполнение условия 1 автоматности для полученного ОС, начиная с начальных

отрезков длиной один символ.

Вновь рассмотрим начальные отрезки во входных словах, равные 0 (первые три строки ОС на рис. 2). В двух строках входному символу 0 соответствует символ β, а в третьей – символ 0, то есть оказывается нарушенным условие 1. Чтобы устранить это нарушение, припишем символ β слева к третьему выходному слову, а чтобы устранить нарушение этим действием условия 2, припишем символ α справа к третьему входному слову (рис. 3).

Дальнейшая проверка (для начальных отрезков входных слов, равных 1, а также для начальных отрезков длиной два символа) не обнаруживает нарушений условий автоматности. Таким образом, получаем ОС, приведенный к автоматному виду (рис. 3).

$z(0)$	$z(1)$	$z(2)$	$z(3)$	$z(4)$	$w(0)$	$w(0\|1)$	$w(1\|2)$	$w(2\|3)$	$w(3\|4)$
0	0	0	α			β	0	1	1
0	0	1	α			β	0	0	0
0	1	1	α			β	0	0	1
1	0	1	α			β	1	0	0
1	1	0	α	α	β	β	1	1	0
1	1	1	α	α	β	β	0	1	1

Рис. 3. ОС, приведенный к автоматному виду.

Примечание к рисунку 3. Заголовок столбца $w(1|2)$ означает, что для первых четырех выходных слов выходной символ формируется в такте 1, для последних двух – в такте 2.

ОС в автоматном виде является основой для построения формального описания АА.

1.2. Построение формального описания АА по оператору соответствия

Рассмотрим процесс построения графа и таблиц переходов и выходов автоматов модели Мили и модели Мура, заданных оператором соответствия.

1.2.1. Построение графа переходов и СТПВ для автомата модели Мили

Из условий работы АА, приведенных в разделе 1.1, следует, что в такте 0 автомат находится в состоянии a_0, т.е. $a(0) = a_0$. Кроме того, под воздействием последнего входного символа автомат должен возвратиться в состояние a_0.

Построим граф переходов автомата Мили по ОС, приведенному на рис. 3.

Из ОС видно, что в момент 0 на вход автомата может поступить один из двух символов: 0 или 1, поэтому из вершины с состоянием a_0 исходит две дуги: одна помечена входным символом 0, другая – 1 (рис. 4).

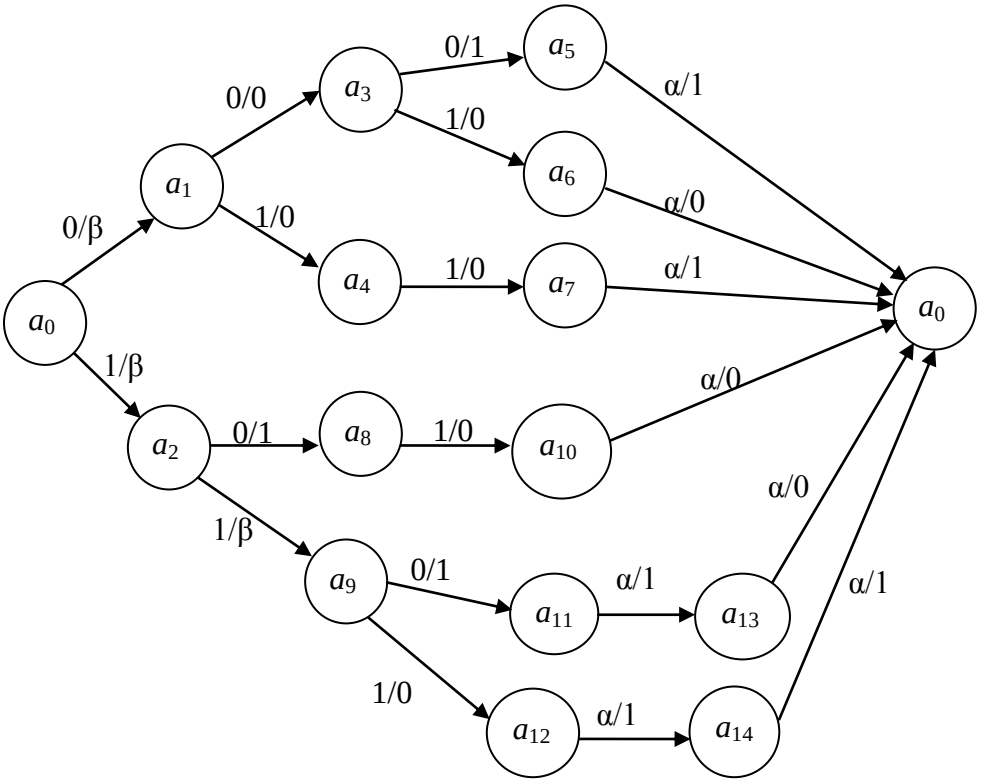


Рис. 4. Граф переходов автомата Мили.

После поступления на вход символа 0 автомат перейдет из a_0 в следующее состояние, назовем его a_1. При этом согласно ОС будет выработан выходной символ β, что отмечено на графе.

Когда автомат находится в состоянии a_1, на вход также может поступить один из двух символов: 0 (если поступает первое или второе входное слово) или 1 (если поступает третье входное слово). Поэтому из вершины с состоянием a_1 также исходит две дуги: в состояние, которое мы обозначим a_3, и в состояние a_4.

Когда автомат находится в состоянии a_4, на его вход может поступить только символ 1, поэтому из вершины с состоянием a_4 исходит лишь одна дуга, помеченная входным символом 1. Рассуждая подобным образом, можно произвести построение графа, который изображен на рис. 4.

При построении графа мы получили множество состояний автомата {a_0, …,

a_{14}}. Наряду с графом другой формой описания АА модели Мили является совмещенная таблица переходов-выходов (СТПВ).

Напомним, что заголовками строк совмещенной таблицы переходов-выходов являются символы входного алфавита автомата, заголовками столбцов – состояния. На пересечении строки z_i и столбца a_j записывается состояние, в которое переходит автомат из a_j по входному символу z_i, и выходной символ, который при этом вырабатывается. Если из некоторого состояния отсутствует переход по какому-либо входному символу, в соответствующей клетке таблицы ставится прочерк.

СТПВ, составленная по графу (рис. 4), представлена в табл. 1.

Таблица 1

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
0	a_1/β	a_3/0	a_8/1	a_5/1		-	-	-	-	a_{11}/1	-	-	-	-	-
1	a_2/β	a_4/0	a_9/β	a_6/0	a_7/0	-	-	-	a_{10}/0	a_{12}/0	-	-	-	-	-
α	-	-	-	-	-	a_0/1	a_0/0	a_0/1	-	-	a_0/0	a_{13}/1	a_{14}/1	a_0/0	a_0/1

Рассмотрим состояния a_5, a_7, a_{14}. Из каждого из этих состояний исходит одна дуга, помеченная одинаковым входным символом α, дуга в каждом случае входит в одну и ту же вершину a_0 с выдачей одинакового выходного символа. Следовательно, указанные состояния эквивалентны и могут быть заменены одним состоянием. То же можно сказать относительно состояний a_6, a_{10} и a_{13}. Заменив каждую группу одним состоянием, получим автомат, имеющий 11 состояний:

$b_0 = a_0$; $b_1 = a_1$; $b_2 = a_2$; $b_3 = a_3$; $b_4 = a_4$; $b_5 = (a_5, a_7, a_{14})$; $b_6 = (a_6, a_{10}, a_{13})$; $b_7 = a_8$; $b_8 = a_9$; $b_9 = a_{11}$; $b_{10} = a_{12}$.

СТПВ нового автомата представлена в табл. 2.

Таблица 2

	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
0	b_1/β	b_3/0	b_7/1	b_5/1	-	-	-	-	b_9/1	-	-
1	b_2/β	b_4/0	b_8/β	b_6/0	b_5/0	-	-	b_6/0	b_{10}/0	-	-
α	-	-	-	-	-	b_0/1	b_0/0	-	-	b_6/1	b_5/1

Таким образом, мы получили два различных описания АА Мили для данного

ОС (рис. 3): в виде графа и в виде СТПВ.

1.2.2. Построение графа переходов и ОТП для автомата модели Мура

Принцип построения графа переходов автомата Мура тот же, что и для графа переходов автомата Мили. Однако есть два отличия в оформлении графа.

Первое отличие состоит в том, что выходные символы записываются в вершинах графа вместе с состояниями, так как выходной символ автомата Мура определяется только состоянием и не зависит от входного символа.

Второе отличие – *в числе начальных состояний*. Так, автомат Мили всегда имеет одно начальное состояние. Количество начальных состояний автомата Мура равно количеству различных последних символов в выходных словах ОС. Это объясняется, во-первых, условием работы АА (см. раздел 1.1), согласно которому после поступления на вход АА последнего символа любого входного слова автомат переходит в начальное состояние, а во-вторых, тем, что выходной символ автомата Мура определяется состоянием перехода.

Для данного оператора соответствия автомат Мура будет иметь два начальных состояния (т.к. в выходных словах ОС (рис. 3) имеют место два различных последних выходных символа: 0 и 1), обозначим эти начальные состояния a_0', a_0".

Все начальные состояния автомата Мура являются равноправными в том смысле, что из них под воздействием одинаковых входных символов происходят переходы в одинаковые состояния.

Граф переходов автомата Мура для ОС (рис. 3) изображен на рис. 5.

Так же, как и для автомата Мили, при построении графа мы получили множество состояний автомата $\{a_0', a_0'', \ldots, a_{14}\}$. Другой формой описания АА модели Мура является отмеченная таблица переходов (ОТП).

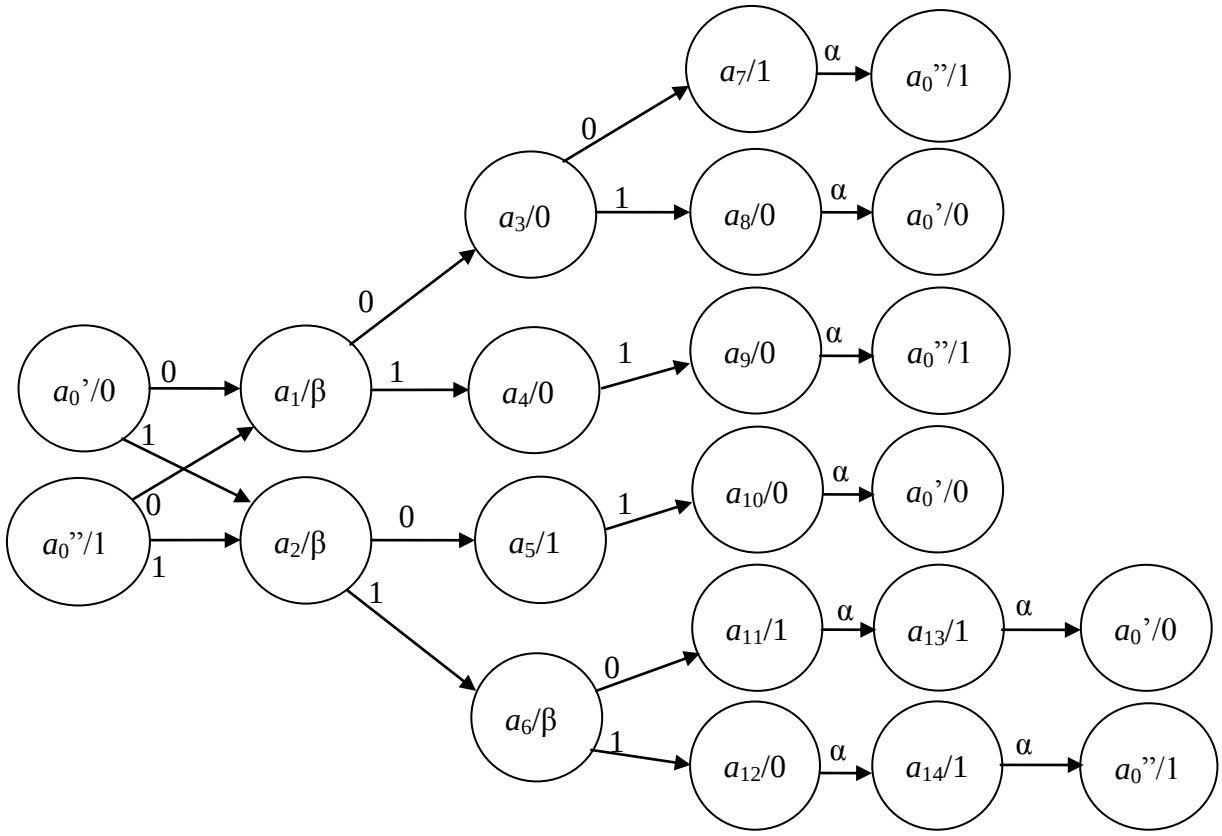


Рис. 5. Граф переходов автомата модели Мура.

ОТП, составленная по графу, изображенному на рис. 5, представлена в табл. 3.

Таблица 3

	0	1	β	β	0	0	1	β	1	0	0	0	1	0	1	1
	a_0’	a_0”	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
0	a_1	a_1	a_3	a_5	a_7	-	-	a_{11}	-	-	-	-	-	-	-	-
1	a_2	a_2	a_4	a_6	a_8	a_9	a_{10}	a_{12}	-	-	-	-	-	-	-	-
α	-	-	-	-	-	-	-	-	a_0”	a_0’	a_0”	a_0’	a_{13}	a_{14}	a_0’	a_0”

Аналогично тому, как это было сделано для автомата модели Мили, можно выполнить первый простейший этап минимизации автомата Мура.

В данном случае можно сформировать две группы эквивалентных состояний: (a_7, a_{14}) и (a_8, a_{10}). Состояния в каждой группе, во-первых, отмечены одинаковыми выходными символами, во-вторых, из них исходит по одной дуге, отмеченной одним и тем же символом α и, наконец, переход из состояний каждой группы осуществляется в одно и то же состояние.

Каждую группу эквивалентных состояний заменим одним состоянием. Тогда

мы получим новый автомат, имеющий четырнадцать состояний:

$b_0' = a_0'$; $b_0'' = a_0''$; $b_1 = a_1$; $b_2 = a_2$; $b_3 = a_3$; $b_4 = a_4$; $b_5 = a_5$; $b_6 = a_6$; $b_7 = (a_7, a_{14})$; $b_8 = (a_8, a_{10})$; $b_9 = a_9$; $b_{10} = a_{11}$; $b_{11} = a_{12}$; $b_{12} = a_{13}$.

ОТП нового автомата представлена в табл. 4.

Таблица 4

	0	1	β	β	0	0	1	β	1	0	0	1	0	1
	b_0'	b_0''	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}
0	b_1	b_1	b_3	b_5	b_7	-	-	b_{10}	-	-	-	-	-	-
1	b_2	b_2	b_4	b_6	b_8	b_9	b_8	b_{11}	-	-	-	-	-	-
α	-	-	-	-	-	-	-	-	b_0''	b_0'	b_0''	b_{12}	b_7	b_0'

Таким образом, мы получили два различных описания АА Мура для данного ОС (рис. 3): в виде графа и в виде ОТП.

1.3. Минимизация частично определенных автоматов

Абстрактный автомат, в котором не для всех пар вида (a_m, z_f) (см. описание работы 2) определено состояние перехода a_s или выходной символ w_g, называется *частично определенным*. В таблицах переходов-выходов таких автоматов в соответствующих клетках присутствуют прочерки. Как правило, автомат, заданный оператором соответствия, является частично определенным.

Для минимизации частично определенного АА можно предварительно доопределить прочерки любым образом, а затем произвести минимизацию полученного полностью определенного АА. Однако такой подход не гарантирует получения максимально возможной степени минимизации АА. В [1] рассмотрен простой пример, иллюстрирующий этот вывод. Там же рассматривается другой подход, который и предлагается использовать в настоящей работе.

Для минимизации частичных автоматов так же, как и для минимизации полностью определенных автоматов, можно использовать треугольную таблицу.

1.3.1. Минимизация числа состояний АА модели Мили

Построение треугольной таблицы и ее заполнение для частично определенных автоматов производится по такому же принципу, как и для

полностью определенных автоматов. Есть лишь ряд особенностей, которые отметим в этом разделе.

В табл. 5 приведен фрагмент таблицы переходов-выходов некоторого АА модели Мили.

Таблица 5

	…	a_5	…	a_{10}	…
0	…	a_5/w_2	…	-	…
1	…	a_{15}/w_1	…	-	…
α		-		a_2/w_1	…

Пусть необходимо заполнить клетку треугольной таблицы с координатами (a_5, a_{10}). Для этого используем то обстоятельство, что в табл. 5 в клетках с прочерком может быть занесено любое значение пары состояние / выходной символ. Целесообразным при минимизации представляется заполнить клетки с прочерками таким образом, чтобы столбцы с заголовками a_5 и a_{10} были идентичны, так, как это показано в табл. 6.

Таблица 6

	…	a_5	…	a_{10}	…
0	…	a_5/w_2	…	a_5/w_2	…
1	…	a_{15}/w_1	…	a_{15}/w_1	…
α		a_2/w_1		a_2/w_1	…

В результате доопределения АА, находящийся в состоянии a_5, при подаче любого входного символа будет вести себя так же, как если бы он находился в состоянии a_{10}, следовательно, эти состояния могут быть включены в одну группу безусловно и в клетку с координатами (a_5, a_{10}) ставится знак $\vee$.

При минимизации частично определенных АА, так же как и для полностью определенных АА, по клеткам треугольной таблицы, заполненным знаком $\vee$, выписываются все возможные пары эквивалентных состояний исходного АА, а затем делается попытка укрупнения полученных групп эквивалентности.

Отличительной особенностью минимизации частично определенных автоматов является необходимость проверки полученных групп на выполнение

двух условий:

1) *условие полноты*: каждое из состояний исходного автомата должно входить хотя бы в одну группу;

2) *условие замкнутости*: для любой пары состояний (a_i, a_j), находящихся в одной группе, для каждого символа z_k из множества входных символов автомата должно быть справедливо утверждение: из a_i при подаче входного символа z_k автомат переходит в состояние, находящееся в той же группе, что и состояние, в которое переходит автомат под действием z_k из a_j.

Другими словами, условие замкнутости означает, что в одну группу эквивалентных состояний могут быть включены лишь *двухэквивалентные* состояния в рамках выбранного варианта распределения состояний по группам.

Необходимость проверки условия замкнутости объясняется возможностью получения нескольких вариантов разбиения состояний исходного частично определенного автомата на группы. В конкретном выбранном варианте условия объединения для некоторых пар состояний могут быть нарушены.

Если для какой-либо пары состояний условие замкнутости оказывается нарушенным, необходимо выбрать другой вариант распределения состояний по группам.

После формирования групп и успешной их проверки на выполнение условий полноты и замкнутости каждая группа заменяется одним состоянием, и строится таблица переходов-выходов минимизированного автомата.

Указанные выше действия покажем на примере автомата Мили (табл. 2).

Построим и заполним треугольную таблицу (табл. 7). При записи условий для указания состояний будем использовать их индексы.

Таблица 7

b_1	×								
b_2	×	×							
b_3	×	×	×						
b_4	×	4,5 ∨	×	5,6 ×					
b_5	∨	∨	∨	∨	∨				
b_6	∨	∨	∨	∨	∨	×			
b_7	×	4,6 ∨	×	∨	5,6 ×	∨	∨		
b_8	×	×	×	5,9 × 6,10	5,10 ∨	∨	∨	6,10 ×	
b_9	∨	∨	∨	∨	∨	0,6 ∨	×	∨	∨
b_{10}	∨	∨	∨	∨	∨	0,5 ∨	×	∨	∨	5,6 ×
	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9

Выписывая из треугольной таблицы пары эквивалентных состояний, получим:

(0, 5)	(1, 4)	(2, 5)	(3, 5)	(4, 5)	(5, 7)	(6, 7)	(7, 9)	(8, 9)
(0, 6)	(1, 5)	(2, 6)	(3, 6)	(4, 6)	(5, 8)	(6, 8)	(7, 10)	(8, 10)
(0, 9)	(1, 6)	(2, 9)	(3, 7)	(4, 8)	(5, 9)			
(0, 10)	(1, 7)	(2, 10)	(3, 9)	(4, 9)	(5, 10)			
	(1, 9)		(3, 10)	(4, 10)				
	(1, 10)							

Выполним, насколько это будет возможно, укрупнение групп и определим один из возможных вариантов распределения состояний по группам:

(0, 5, 9), (1, 4, 6), (2, 10), (3, 7), (8).

Теперь необходимо проверить полученные группы на полноту и замкнутость.

Условие полноты соблюдается, т.к. каждое состояние присутствует хотя бы в одной группе.

Перейдем к проверке каждой группы на замкнутость. Рассмотрим пары (0, 5) и (0, 9) в первой группе. Они удовлетворяют условию замкнутости. Что касается состояний b_5 и b_9 (третья пара в группе), то из табл. 7 видно, что условием их объединения в группу является присутствие в одной группе состояний b_0 и b_6. Однако при выбранном варианте распределения состояний по группам состояния

b_0 и b_6 не входят в одну группу. Следовательно, и состояния b_5 и b_9 не могут находиться в одной группе (они не являются двухэквивалентными), их надо распределить по разным группам.

Изменим состав групп:

(0, 5, 10), (1, 4, 6), (2, 9), (3, 7), (8).

Теперь все пары первой группы удовлетворяют условию замкнутости, так как условием объединения в группу состояний b_5 и b_{10} является присутствие в одной группе состояний b_0 и b_5 (табл. 7).

Рассмотрим пару (1, 4) во второй группе. Из табл. 7 видно, что условием объединения состояний b_1 и b_4 является присутствие в одной группе состояний b_4 и b_5, это условие при выбранном варианте распределения состояний по группам не выполняется. Следовательно, состояния b_1 и b_4 не могут находиться в одной группе. Вновь изменим состав групп:

(0, 5, 10), (1, 6), (2, 9), (3, 7), (4, 8).

Группы (b_1, b_6), (b_2, b_9), и (b_3, b_7) удовлетворяют условию замкнутости, так как состояния, входящие в них, могут быть объединены безусловно. Для пятой группы условием объединения состояний b_4 и b_8 является присутствие в одной группе состояний b_5 и b_{10}. Оба эти состояния входят в первую группу, следовательно, выполняется замкнутость пятой группы.

Таким образом, выбранное распределение состояний по группам удовлетворяет условиям полноты и замкнутости.

Обозначим состояния рассматриваемого минимального автомата c_i:

$c_0 = (b_0, b_5, b_{10})$, $c_1 = (b_1, b_6)$, $c_2 = (b_2, b_9)$, $c_3 = (b_3, b_7)$, $c_4 = (b_4, b_8)$.

Составим СТПВ минимального автомата (табл. 8). Для этого будем использовать СТПВ исходного автомата (табл. 2). На примере столбца c_0 покажем ее заполнение. Состояние c_0 объединяет в себе состояния b_0, b_5 и b_{10}, поэтому сначала используем для входных символов 0 и 1 столбец b_0 из табл. 2, заменяя состояния b_1и b_2 на c_1 и c_2 соответственно. Для входного символа α используем любой из столбцов b_5, b_{10}.

Таблица 8

	c_0	c_1	c_2	c_3	c_4
0	c_1/β	c_3/0	c_3/1	c_0/1	c_2/1
1	c_2/β	c_4/0	c_4/β	c_1/0	c_0/0
α	c_0/1	c_0/0	c_1/1	-	-

Наконец, необходимо проверить корректность выполненной минимизации. Так как автомат Мили (табл. 2) построен по оператору соответствия (рис. 3), проверка выполняется построением автоматных лент для каждого входного слова из ОС. Пример автоматной ленты минимального автомата для входного слова: 1 1 1 α α приведен на рис. 6. Лента заполнена на основе СТПВ (табл. 8) с учетом того, что в начальный момент времени автомат находится в начальном состоянии c_0.

Входной символ	1	1	1	α	α	
Состояние	c_0	c_2	c_4	c_0	c_0	c_0
Выходной символ	β	β	0	1	1	

Рис. 6. Автоматная лента минимального автомата Мили

Полученное выходное слово совпало со словом, приведенным в ОС. Для полной проверки автомата необходимо построить ленту для всех входных слов ОС.

1.3.2. Минимизация числа состояний АА модели Мура

Принципы минимизации и правила работы с треугольной таблицей для частично определенного автомата Мура те же, что и изложенные в разделе 1.3.1 для автомата Мили. Так же, как и в случае минимизации полностью определенного автомата, единственное отличие состоит в следующем.

При заполнении треугольной таблицы для автомата Мили используется понятие одноэквивалентных состояний. Для автомата Мура следует вместо этого использовать понятие нуль-эквивалентных состояний (см. описание работы 2).

В качестве примера рассмотрим минимизацию АА модели Мура (табл. 4).

Для минимизации автомата построим треугольную таблицу (табл. 9).

Таблица 9

	b_0’	b_0”	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
b_0”	×												
b_1	×	×											
b_2	×	×	3,5 × 4,6										
b_3	1,7 × 2,8	×	×	×									
b_4	2,9 ×	×	×	×	8,9 ×								
b_5	×	2,8 ×	×	×	×	×							
b_6	×	×	3,10 × 4,11	5,10 × 6,11	×	×	×						
b_7	×	∨	×	×	×	×	∨	×					
b_8	∨	×	×	×	∨	∨	×	×	×				
b_9	∨	×	×	×	∨	∨	×	×	×	×			
b_{10}	×	∨	×	×	×	×	∨	×	0”,12 ∨	×	×		
b_{11}	∨	×	×	×	∨	∨	×	×	×	0’,7 ×	0”,7 ∨	×	
b_{12}	×	∨	×	×	×	×	∨	×	×	×	×	0’,12 ×	×

Далее выпишем всевозможные пары эквивалентных состояний. Для указания состояний будем использовать их индексы.

(0’, 8) (0”, 7) (3, 8) (4, 8) (5, 7) (7, 10) (9, 11)

(0’, 9) (0”, 10) (3, 9) (4, 9) (5, 10)

(0’, 11) (0”, 12) (3, 11) (4, 11) (5, 12)

Попытаемся укрупнить группы и выпишем один из вариантов распределения состояний по группам:

(0’, 8), (0”, 7, 10), (1), (2), (3, 9), (4, 11), (5, 12), (6).

Далее следует проверить выполнение условий полноты и замкнутости.

Условие полноты выполняется, поскольку каждое из состояний присутствует хотя бы в одной группе.

Теперь проверим условие замкнутости. Рассмотрим первую группу. Пара (0’, 8) удовлетворяет условию замкнутости, т.к. в клетке табл. 9 с координатами (0’, 8) стоит знак ∨.

Условие замкнутости нарушено во второй группе для пары (7, 10), т.к. в клетке таблицы для этой пары стоит условие (0”, 12), а в выбранном варианте

распределения состояний по группам состояния b_0” и b_{12} входят в разные группы.

Поменяем состояния 7, 10 и 12 местами:

(0’, 8), (0”, 12), (1), (2), (3, 9), (4, 11), (5, 7, 10), (6).

Пары (0”, 12), (3, 9), (4, 11), (5, 7) и (5, 10) могут быть объединены безусловно, так как в соответствующих им клетках треугольной таблицы стоит знак $\vee$. Условие для объединения пары (7, 10) также выполняется, так как состояния b_0” и b_{12} после перегруппировки входят в одну группу.

Обозначим состояния минимального автомата c_i:

c_0’ = (0’, 8), c_0” = (0”, 12), c_1 = (1), c_2 = (2), c_3 = (3, 9), c_4 = (4, 11), c_5 = (5, 7, 10), c_6 = (6).

ОТП минимального автомата строится так же, как и СТПВ для автомата Мили, и приведена в табл. 10.

Таблица 10

	0	1	β	β	0	0	1	β
	c_0’	c_0”	c_1	c_2	c_3	c_4	c_5	c_6
0	c_1	c_1	c_3	c_5	c_5	-	-	c_5
1	c_2	c_2	c_4	c_6	c_0’	c_3	c_0’	c_4
α	c_0’	c_0’	-	-	c_0”	c_5	c_0”	-

Для проверки корректности выполненной минимизации необходимо построить автоматные ленты для каждого входного слова из ОС (рис. 3). При этом следует считать, что перед подачей каждого слова автомат находится в одном из начальных состояний c_0’ или c_0”.

Пример автоматной ленты для входного слова 1 1 1 α α, построенной по ОТП минимального автомата (табл. 10), приведен на рис. 7.

Входной символ	1	1	1	α	α	
Состояние	c_0’	c_2	c_6	c_4	c_5	c_0”
Выходной символ	-	β	β	0	1	1

Рис. 7. Автоматная лента минимального автомата Мура

Полученное выходное слово совпадает с выходным словом, приведенном в ОС. Отметим, что в отличие от ленты автомата Мили (рис. 6), лента автомата Мура в последней строке начинается с прочерка. Это связано с тем, что символ на

выходе автомата Мура появляется только после перехода его в новое состояние, а в автомате Мили – при подаче входного символа.

2. Задание по работе

Автомат задан оператором соответствия (см. раздел «Варианты заданий»). Требуется построить граф переходов и таблицу переходов и выходов соответствующего автомата модели Мили (для четных вариантов) или Мура (для нечетных вариантов). Выполнить минимизацию полученного автомата.

Проверку корректности работы полученного автомата для заданного ОС необходимо провести в пакете *JFLAP*.

3. Порядок выполнения работы

Выполнение работы состоит из трех частей: построения АА, минимизации и проверки корректности построения путем моделирования в пакете *JFLAP*.

Для построения АА по заданному ОС необходимо:

1) привести заданный ОС к автоматному виду;
2) для полученного ОС построить граф переходов АА модели Мили (для четных вариантов) или Мура (для нечетных вариантов);
3) составить СТПВ для АА модели Мили, либо ОТП для АА модели Мура;
4) выполнить первый этап минимизации полученного АА: составить группы эквивалентных состояний и таблицу переходов минимизированного автомата.

При минимизации АА, необходимо выполнить все этапы минимизации АА с помощью треугольной таблицы для АА, полученного в результате выполнения первого этапа минимизации.

Для проверки корректности работы полученного автомата в пакете *JFLAP* необходимо:

1) набрать граф АА, полученного в результате минимизации, и задать начальное состояние;

2) выбрать меню *Inputs*: *Multiple Run* и ввести входные слова оператора соответствия;
3) нажать кнопку *Run Inputs* и убедиться, что полученные выходные слова совпадают с выходными словами оператора соответствия.

Подробные методические указания по работе в пакете *JFLAP* приведены в Приложении.

4. Содержание отчета

Отчет по работе должен содержать следующие разделы.

1. Цель работы.
2. Задание по работе.

   Раздел должен включать в себя формулировку задания и данные варианта.
3. Приведение оператора соответствия к автоматному виду.

   В данном разделе необходимо привести оператор соответствия в автоматном виде.
4. Синтез автомата по оператору соответствия.

   В данном разделе необходимо привести граф переходов АА, составленный по оператору соответствия и таблицу (СТПВ или ОТП) синтезированного автомата.
5. Первый этап минимизации автомата.

   В данном разделе необходимо привести группы эквивалентных состояний, таблицу (СТПВ или ОТП) минимизированного автомата и список соответствия состояний исходного и минимизированного автомата.
6. Минимизация автомата с использованием треугольной таблицы.

   В данном разделе необходимо привести описание этапов минимизации и таблицу (СТПВ или ОТП) минимизированного автомата.
7. Проверка корректности синтеза.

В данном разделе необходимо привести граф минимизированного автомата, построенный в JFLAP, и скриншот с результатами моделирования работы минимизированного автомата для всех входных слов оператора соответствия.

8. Вывод.

Пример вывода: «В результате выполнения работы произведен синтез автомата модели Мура по заданному оператору соответствия. Произведен первый этап минимизации автомата, в результате чего число состояний автомата сократилось с 17 до 15. Затем произведена минимизация полученного автомата с помощью треугольной таблицы. В результате минимизации число состояний автомата сократилось с 15 до 7. Проверка соответствия минимизированного автомата заданному оператору произведена в среде JFLAP».

5. Варианты заданий

Вариант 1

входные слова			выходные слова		
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	1	0	1	1

Вариант 2

входные слова			выходные слова		
0	0	1	0	1	0
0	1	0	0	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	1	0	0

Вариант 3

входные слова			выходные слова		
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	0
1	1	0	0	1	1
1	1	1	1	0	1

Вариант 4

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	1	1
0	0	1	0	0	1

Вариант 5

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	0	1	0	0
1	1	0	1	1	0
1	1	1	0	1	1
1	0	1	1	1	1

Вариант 6

входные слова			выходные слова		
0	0	1	0	1	1
0	1	1	0	0	1
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	1	1
0	0	0	0	1	1

Вариант 7

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
0	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	0
1	0	1	0	0	0

Вариант 8

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	0	1	0	0
0	0	1	1	1	1
1	1	1	0	1	1

Вариант 9

входные слова			выходные слова		
0	1	0	0	1	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	0	0	0	0	1
1	1	1	0	1	1

Вариант 10

входные слова			выходные слова		
0	0	0	0	0	1
0	1	1	1	0	1
1	0	1	0	0	0
1	1	0	1	1	0
0	1	0	1	0	0
1	1	1	0	1	0

Вариант 11

входные слова			выходные слова		
0	0	0	0	1	0
0	1	1	1	0	0
0	0	1	0	0	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	1	0

Вариант 12

входные слова			выходные слова		
0	0	1	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
0	0	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	1

Вариант 13

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	1	1

Вариант 14

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	1	0
1	1	1	0	1	1

Вариант 15

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
0	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	1

Вариант 16

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
0	1	0	1	0	0
0	0	1	0	0	0
1	1	0	1	1	0
1	1	1	1	1	1

Вариант 17

входные слова			выходные слова		
0	0	0	1	1	1
0	1	1	1	0	1
1	0	1	1	0	0
0	0	1	1	1	0
1	0	0	1	1	0
1	1	1	1	1	1

Вариант 18

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
0	1	0	0	1	1
1	1	0	1	1	0
1	0	0	0	1	1

Вариант 19

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
0	0	1	1	1	0
1	0	1	1	0	0
0	1	0	1	1	0
1	1	1	0	1	1

Вариант 20

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	0	1	1	0
0	0	1	1	1	0
1	1	1	0	1	1

Вариант 21

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	0	0	0
1	0	0	1	1	0
1	1	0	1	0	0
1	1	1	0	1	1

Вариант 22

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	1	1
0	1	0	0	0	1

Вариант 23

входные слова			выходные слова		
0	0	0	0	1	1
0	0	1	1	0	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	1	1
0	1	1	1	1	1

Вариант 24

входные слова			выходные слова		
0	0	0	0	1	0
0	1	1	1	0	1
0	1	0	1	0	0
1	1	0	0	1	0
1	1	1	0	1	1
1	0	1	1	0	1

Вариант 25

входные слова			выходные слова		
0	0	0	0	0	1
0	1	1	1	0	1
1	0	1	1	0	0
1	0	0	1	1	0
1	1	1	1	1	1
1	1	0	0	0	0

Вариант 26

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	0	0	1	1	0
1	1	1	0	1	1
1	1	0	0	1	1

Вариант 27

входные слова			выходные слова		
0	0	0	0	0	1
0	1	1	1	0	1
0	1	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	0
1	1	1	0	0	1

Вариант 28

входные слова			выходные слова		
0	0	0	0	0	1
0	1	1	1	0	1
1	0	1	0	0	0
0	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	0	1

Вариант 29

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	1	0	0	1
1	1	0	1	1	0
1	0	0	0	1	1

Вариант 30

входные слова			выходные слова		
0	0	0	0	1	1
0	1	1	1	1	1
1	0	1	1	0	0
0	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	1	1

Лабораторная работа 1
СТРУКТУРНЫЙ СИНТЕЗ КОНЕЧНЫХ АВТОМАТОВ

Цель работы: изучение основ канонического метода структурного синтеза конечных автоматов; получение навыков построения структурных схем конечных автоматов.

1. Основные теоретические сведения

Процесс проектирования конечных автоматов (КА) включает выполнение двух основных этапов: абстрактного синтеза и структурного синтеза.

На этапе *абстрактного синтеза* КА рассматривается как устройство с неизвестной физической структурой (черный ящик), имеющее один обобщенный вход, с которым связывается множество входных символов (входной алфавит $Z = \{z_1, \ldots, z_f, \ldots, z_F\}$), и один обобщенный выход, с которым связывается множество выходных символов (выходной алфавит $W = \{w_1, \ldots, w_g, \ldots, w_G\}$). Автомат имеет множество состояний (алфавит состояний $A = \{a_0, a_1, \ldots, a_m, \ldots, a_M\}$), его поведение описывается функцией переходов δ и функцией выходов λ, которые задаются, в частности, в виде таблиц переходов-выходов [1]. Основной задачей, решаемой на этом этапе, является нахождение автомата, эквивалентного заданному, но имеющего возможно меньшее число состояний. При этом предполагается, что при меньшем числе состояний техническая реализация автомата будет проще.

На этапе *структурного синтеза* выбирается способ представления входных и выходных сигналов, а также способ представления состояний автомата. Структурный автомат, полученный в результате этого этапа, содержит несколько структурных входных и структурных выходных каналов, по которым передаются сигналы в структурном алфавите (входном и выходном соответственно). В настоящее время наиболее распространенным структурным алфавитом является двоичный алфавит, что объясняется простотой его представления в цифровых узлах и приборах. Кроме того, для двоичного алфавита хорошо развит аппарат

булевой алгебры, позволяющий производить многие операции над схемой формально.

В случае двоичного алфавита каждый входной z_f и выходной w_g символы абстрактного автомата (АА) могут быть закодированы двоичными векторами длины L и N соответственно. Это означает, что число структурных входных каналов будет равно L, а число структурных выходных каналов – N:

$$z_f \Leftrightarrow \mathbf{X}_f = (x_{f1} \ldots x_{fl} \ldots x_{fL}),$$

где $f = 1, \ldots, F$; F – мощность входного алфавита (число различных символов во входном алфавите), $x_{fl} = 0, 1$;

$$w_g \Leftrightarrow \mathbf{Y}_g = (y_{g1} \ldots y_{gn} \ldots y_{gN}),$$

где $g = 1, \ldots, G$; G – мощность выходного алфавита, $y_{gn} = 0, 1$.

Каждому состоянию a_m абстрактного автомата ставится в соответствие код – набор состояний некоторых элементарных автоматов [1]. Переход автомата из одного состояния в другое заключается в изменении состояний элементарных автоматов.

В качестве элементарных автоматов обычно выбираются полные автоматы Мура (независимо от того, строится ли все устройство в виде автомата Мили или Мура). Известно [1], что триггер, обладающий двумя устойчивыми состояниями (0 и 1), является полным автоматом Мура и может быть использован в качестве элементарного автомата.

По аналогии с входными и выходными символами каждое состояние автомата a_m может быть закодировано двоичным вектором длины P. Значение P определяет количество триггеров в структурной схеме автомата.

$$a_m \Leftrightarrow \mathbf{Q}_m = (Q_{m1} \ldots Q_{mp} \ldots Q_{mP}),$$

где $m = 0, \ldots, M$; $M+1$ – количество состояний абстрактного автомата, $Q_{mp} = 0, 1$.

Очевидно, что

$$L =]\log_2 F[, \quad (1)$$

$$N =]\log_2 G[, \quad (2)$$

$$P =]\log_2 (M+1)[. \quad (3)$$

Здесь]*[– ближайшее целое число, большее или равное *.

Например, пусть абстрактный автомат имеет 1000 состояний. Сколько потребуется триггеров для хранения такого количества состояний? Известно, что $2^{10} = 1024$, то есть $\log_2 1024 = 10$, $2^9 = 512$, то есть $\log_2 512 = 9$. Следовательно, $9 < \log_2 1000 < 10$ и ответ: $P =]\log_2 1000[= 10$.

Естественно допустить, что структурный автомат будет отражать поведение исходного АА, если в процессе синтеза структурного автомата используются таблицы переходов-выходов исходного АА, в которых входные, выходные символы, а также символы состояний АА заменены их структурными эквивалентами.

В результате структурного синтеза должна быть получена структурная схема автомата в виде композиции трех составляющих (рис. 1): *совокупности элементарных автоматов (триггеров)*, принадлежащих к заранее заданному числу типов, и *двух комбинационных схем*: КС1, на выходах которой формируются функции возбуждения (ФВ) для триггеров, и КС2, на выходах которой формируются выходные сигналы автомата.

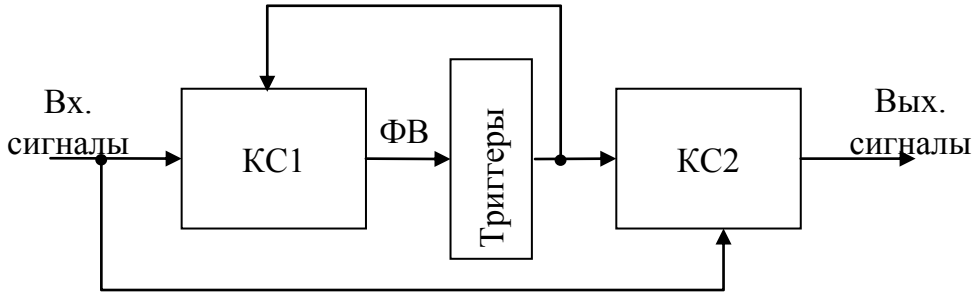


Рис. 1. Обобщенная структурная схема конечного автомата

КС1 обеспечивает переходы автомата из состояния в состояние (в соответствии с таблицей переходов абстрактного автомата, поведение которого должен реализовать структурный автомат), КС2 обеспечивает выдачу выходных сигналов автомата при каждом переходе (в соответствии с таблицей выходов абстрактного автомата).

Отметим, что в дальнейшем будем рассматривать три типа триггеров с *синхронным* способом управления: *JK*-триггер, *T*-триггер и *D*-триггер. Триггеры с синхронным управлением кроме информационных входов (*J* и *K*, *T*, *D*, соответственно), имеют специальный вход для синхросигналов *C* (рис. 2), на

который поступает сигнал от внешнего генератора тактовых импульсов. Переключение таких триггеров из одного состояния в другое происходит только в момент изменения (например, из 0 в 1) значения синхросигнала на входе *C*. Такой способ управления обеспечивает одновременное переключение всех триггеров и предотвращает тем самым возможность появления сбойных ситуаций.

Как показано на рис. 2, *T*-триггер может быть реализован на базе *JK*-триггера. Для этого необходимо подать функцию возбуждения *T*-триггера одновременно на оба входа *J* и *K*.

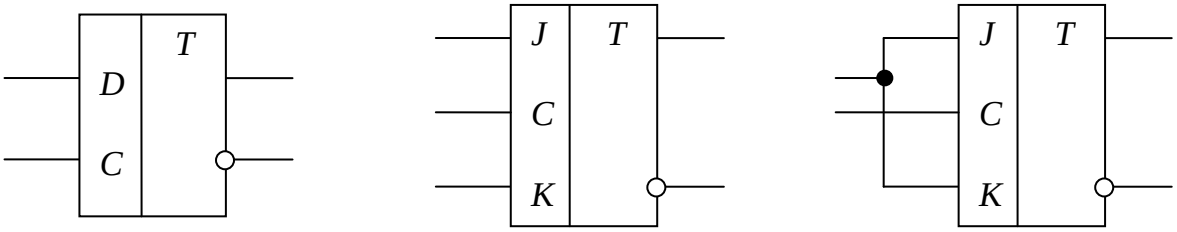


Рис. 2. Условные графические обозначения, слева направо: D-триггера, JK-триггера и T-триггера на базе JK

Исходной информацией для выполнения структурного синтеза является либо совмещенная таблица переходов-выходов (СТПВ) (при синтезе минимального автомата модели Мили), либо ОТП (при синтезе минимального автомата модели Мура). При проектировании структурного автомата применяется канонический метод структурного синтеза автоматов, подробно описанный в [1, раздел 1.5].

Рассмотрим методику структурного синтеза, построенную на использовании канонического метода. В ней можно выделить ряд этапов.

1.1. Автомат модели Мили

СТПВ автомата представлена в табл. 1.

Таблица 1

	a_0	a_1	a_2	a_3
z_1	a_1/w_3	a_0/w_1	a_1/w_1	a_2/w_1
z_2	a_0/w_2	a_1/w_2	a_2/w_2	a_1/w_1
z_3	a_3/w_1	a_0/w_2	a_0/w_3	a_0/w_1

Этап 1. Кодирование входных символов, выходных символов и состояний.

На этом этапе определяется количество структурных входных каналов, структурных выходных каналов и элементарных автоматов (триггеров) для хранения состояний структурного автомата. Для автомата Мили (табл. 1) $F = 3$ (z_1, z_2, z_3); $G = 3$ (w_1, w_2, w_3); $M+1 = 4$ (a_0, a_1, a_2, a_3).

На основании формул (1 – 3) устанавливаем, что создаваемый структурный автомат должен иметь два входных канала, два выходных канала и содержать два триггера. Пусть x_2, x_1 – переменные, связанные с входами автомата, y_2, y_1 – переменные, связанные с выходами автомата, Q_2, Q_1 – выходные сигналы триггеров.

Закодируем входные, выходные символы и состояния абстрактного автомата их структурными эквивалентами (табл. 2 – 4).

Таблица 2

	x_2	x_1
z_1	0	0
z_2	0	1
z_3	1	0

Таблица 3

	y_2	y_1
w_1	0	0
w_2	0	1
w_3	1	0

Таблица 4

	Q_2	Q_1
a_0	0	0
a_1	0	1
a_2	1	0
a_3	1	1

Построение КС1

Этап 2. Формирование кодированной таблицы переходов структурного автомата (КТП).

КТП формируется из таблицы переходов (ТП) исходного автомата заменой в ТП абстрактных символов их структурными эквивалентами (табл. 2 – 4). ТП автомата Мили образуется, если из клеток СТПВ исходного автомата исключить выходные символы.

Пример ТП, полученной из табл. 1, приведен в табл. 5.

Таблица 5

	a_0	a_1	a_2	a_3
z_1	a_1	a_0	a_1	a_2
z_2	a_0	a_1	a_2	a_1
z_3	a_3	a_0	a_0	a_0

КТП для этого примера, полученная в соответствии с табл. 2 и 4, представлена в табл. 6.

Таблица 6

x_2x_1 \ Q_2Q_1	00	01	10	11
00	01	00	01	10
01	00	01	10	01
10	11	00	00	00

Этап 3. Формирование кодированной таблицы функций возбуждения структурного автомата (КТФВ).

На данном этапе необходимо указать тип используемых триггеров. Выбор осуществляется из трех типов: *JK*-триггер, *T*-триггер со счетным входом и *D*-триггер. Табл. 7, 8 и 9 соответственно представляют собой таблицы переходов этих триггеров.

Таблица 7

JK \ *Q*	0	1
00	0	1
01	0	0
10	1	1
11	1	0

Таблица 8

T \ *Q*	0	1
0	0	1
1	1	0

Таблица 9

D \ *Q*	0	1
0	0	0
1	1	1

КТФВ автомата строится на основе его КТП (табл. 6). Напомним, что в заголовках столбцов КТП указаны выходные сигналы триггеров, определяющие исходное состояние автомата, то есть $Q_2(t)$, $Q_1(t)$, а в клетках КТП содержатся выходные сигналы триггеров, определяющие состояние перехода, то есть $Q_2(t+1)$, $Q_1(t+1)$.

Анализ любого столбца КТП показывает, что переход автомата из состояния в состояние осуществляется путем переключения триггеров. Рассмотрим, например, столбец с заголовком $Q_2(t)Q_1(t) = 00$. Первый элемент этого столбца: $Q_2(t+1)Q_1(t+1) = 01$. Это означает, что при переходе автомата из состояния 00 в состояние 01 второй триггер не изменил своего состояния, а первый триггер переключился в противоположное состояние.

Для каждого вида переключения триггера (из 0 в 0, из 0 в 1, из 1 в 0 и из 1 в 1) имеется набор функций возбуждения (входных сигналов триггера), обеспечивающий необходимое переключение. Кодированная таблица функций возбуждения строится из КТП путем замены в клетках этой таблицы сигналов состояния перехода соответствующими входными сигналами триггеров, которые обеспечивают переход в это состояние из состояния, находящегося в заголовке столбца.

Для определения значений ФВ, обеспечивающих нужный переход, используется дополнительная таблица (табл. 10), которую нетрудно вывести из таблиц переходов триггеров указанных типов (табл. 7 – 9).

Таблица 10

$Q(t)$	$Q(t+1)$	*JK*-триггер		*T*-триггер	*D*-триггер
		$J(t)$	$K(t)$	T(t)	$D(t)$
0	0	0	-	0	0
0	1	1	-	1	1
1	0	-	1	1	0
1	1	-	0	0	1

Для рассматриваемого примера выберем *JK*-триггер. Построим КТФВ (табл. 11) по КТП (табл. 6), используя табл. 10.

Таблица 11

Q_2Q_1 / x_2x_1	00				01				10				11			
	J_2	K_2	J_1	K_1	J_2	K_2	J_1	K_1	J_2	K_2	J_1	K_1	J_2	K_2	J_1	K_1
00	0	-	1	-	0	-	-	1	-	1	1	-	-	0	-	1
01	0	-	0	-	0	-	-	0	-	0	0	-	-	1	-	0
10	1	-	1	-	0	-	-	1	-	1	0	-	-	1	-	1

Этап 4. Построение диаграмм Вейча и логических выражений (ЛВ) для функций возбуждения, представление ЛВ в заданном базисе, построение КС1.

КТФВ следует рассматривать как двумерную таблицу истинности для функций возбуждения. В нашем примере J_2, K_2, J_1, K_1 являются булевыми функциями от четырех аргументов: $Q_2(t)$, $Q_1(t)$, $x_2(t)$, $x_1(t)$ (в дальнейшем аргумент t может быть опущен при записи ЛВ). Для каждой функции возбуждения строится диаграмма Вейча, из которой выписывается минимальное выражение для функции.

Диаграммы Вейча и выписанные по ним функции возбуждения для рассматриваемого примера, приведены на рис. 3.

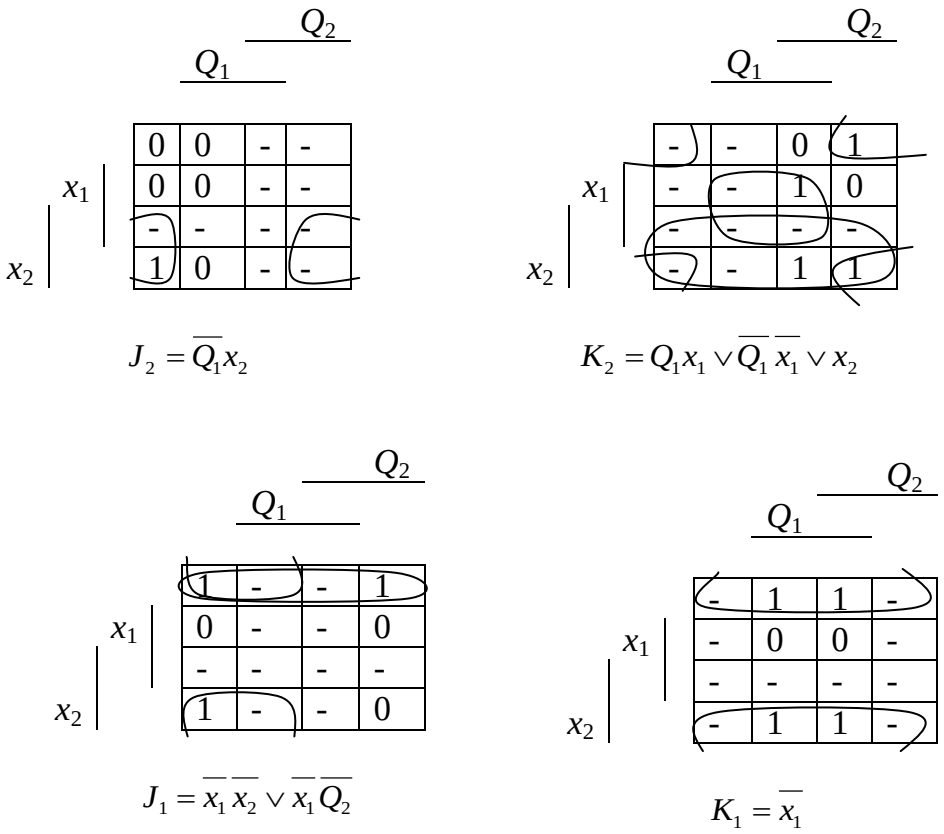


Рис. 3. Диаграммы Вейча для функций возбуждения

Полученные функции необходимо перевести в требуемый базис. КС1 составляется как совокупность схем для функций J_2, K_2, J_1, K_1. Процедуры представления булевых функций в заданном базисе и построения логической схемы описаны в разделе 1.3.

Построение КС2

Этап 5. Построение кодированной таблицы выходов (КТВ).

КТВ формируется из таблицы выходов (ТВ) исходного автомата заменой в ТВ абстрактных символов их структурными эквивалентами (табл. 2 – 4). ТВ образуется, если из клеток СТПВ исходного автомата исключить символы состояний. ТВ, составленная по табл. 1, приведена в табл. 12.

Таблица 12

	a_0	a_1	a_2	a_3
z_1	w_3	w_1	w_1	w_1
z_2	w_2	w_2	w_2	w_1
z_3	w_1	w_2	w_3	w_1

В рассматриваемом примере КТВ имеет следующий вид (табл. 13).

Таблица 13

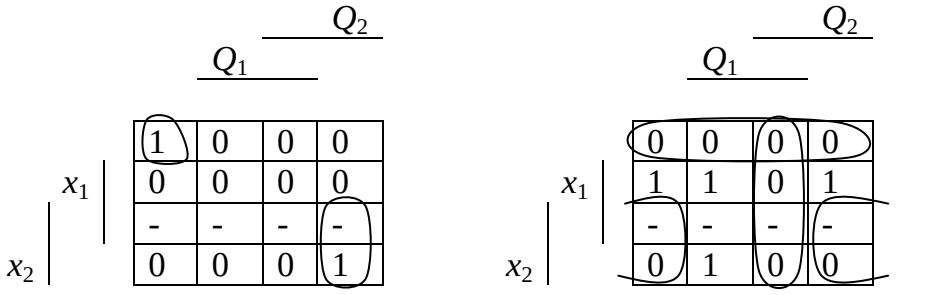
Q_2Q_1 / x_2x_1	00	01	10	11
00	10	00	00	00
01	01	01	01	00
10	00	01	10	00

Примечание: в каждой клетке таблицы цифра слева – y_2, цифра справа – y_1.

Этап 6. Построение диаграмм Вейча и ЛВ для выходных сигналов, представление ЛВ в заданном базисе, построение КС2.

КТВ представляет собой двумерную таблицу истинности для выходных сигналов. В рассматриваемом примере y_2 и y_1 являются, так же как и функции возбуждения триггеров, булевыми функциями от четырех аргументов: $Q_2(t)$, $Q_1(t)$, $x_2(t)$, $x_1(t)$.

Построим диаграммы Вейча для y_2, y_1 и выпишем минимальные выражения (рис. 4).



$$y_2 = \overline{Q_1}\,\overline{Q_2}\,\overline{x_1}\,\overline{x_2} \vee \overline{Q_1} Q_2 x_2 \qquad y_1 = (\overline{Q_1} \vee \overline{Q_2})(Q_1 \vee \overline{x_2})(x_1 \vee x_2)$$

Рис. 4. Диаграммы Вейча для выходных сигналов автомата Мили

Полученные логические выражения необходимо перевести в требуемый базис и построить схемы (см. раздел 1.3). КС2 составляется как совокупность схем для функций y_2, y_1.

1.2. Автомат модели Мура

Напомним, что автомат модели Мура задается своей отмеченной таблицей переходов. Рассмотрим пример ОТП (табл. 14).

Таблица 14

	w_1	w_3	w_4	w_1	w_2
	a_0	a_1	a_2	a_3	a_4
z_1	a_1	a_4	a_0	a_3	a_0
z_2	a_3	a_0	a_3	a_2	a_1
z_3	a_3	a_2	a_0	a_1	a_4

Для автомата модели Мура первые четыре этапа (построение КС1) структурного синтеза выполняются так же, как для автомата модели Мили. Отличия касаются последних двух этапов (построение КС2). Рассмотрим их подробнее.

Построение КС2

Этап 5. Построение кодированной таблицы выходов.

КТВ формируется из ТВ исходного автомата заменой в ТВ абстрактных символов их структурными эквивалентами (табл. 2 – 4).

Пусть на этапе 1 были составлены таблицы соответствия 15 и 16.

Таблица 15

	y_2	y_1
w_1	0	0
w_2	0	1
w_3	1	0
w_4	1	1

Таблица 16

	Q_3	Q_2	Q_1
a_0	0	0	0
a_1	0	0	1
a_2	0	1	0
a_3	0	1	1
a_4	1	0	0

ТВ для рассматриваемого примера приведена в табл. 17.

Таблица 17

w_1	w_3	w_4	w_1	w_2
a_0	a_1	a_2	a_3	a_4

В табл. 18 представлена кодированная таблица выходов.

Таблица 18

$y_2\, y_1$	00	10	11	00	01
$Q_3\, Q_2\, Q_1$	000	001	010	011	100

Этап 6. Построение диаграмм Вейча и логических выражений для выходных сигналов, представление ЛВ в заданном базисе, построение КС2.

Для рассматриваемого примера на этом этапе необходимо построить две диаграммы Вейча для y_2, y_1 (рис. 5).

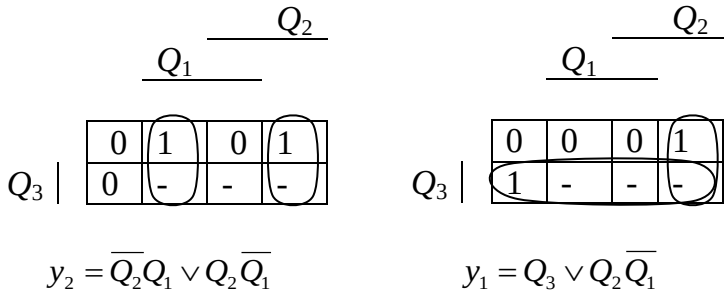


Рис. 5. Диаграммы Вейча для выходных сигналов автомата Мура

Полученные логические выражения необходимо перевести в требуемый базис и построить схемы (см. раздел 1.3). КС2 составляется как совокупность логических схем для функций y_2, y_1.

1.3. Проектирование логических схем

1.3.1 Основы построения логических схем

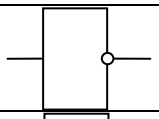
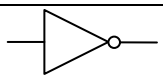
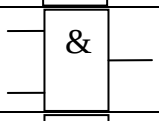
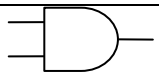
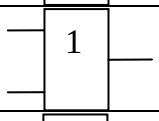
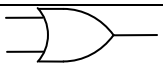
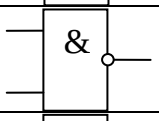
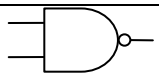
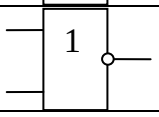
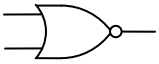
Техническим аналогом любого ЛВ для БФ является логическая схема (ЛС). При этом переменные, от которых зависит БФ, связываются с внешними входами этой схемы, значение БФ формируется на внешнем выходе схемы, а каждая логическая операция в ЛВ реализуется логическим элементом. Таким образом, для каждого набора входных сигналов на выходе ЛС формируется сигнал, соответствующий значению БФ на данном наборе переменных (в дальнейшем будем использовать такое соглашение: 0 – низкий уровень сигнала, 1 – высокий уровень сигнала).

При построении логических схем будем считать, что переменные, от которых зависит БФ, подаются на вход в парафазном коде (то есть доступно и прямое, и инверсное значение переменной).

В табл. 19 приведены условные графические обозначения некоторых логических элементов по ГОСТ 2.743-91, а также их зарубежные аналоги

(используемые, например, в системе автоматизированного проектирования *Quartus*).

Таблица 19

Название элемента (операции)	Российский стандарт	Зарубежный стандарт
Инвертор (НЕ)		
Конъюнктор (И)		
Дизъюнктор (ИЛИ)		
Штрих Шеффера (И-НЕ)		
Стрелка Пирса (ИЛИ-НЕ)		

Кроме элементов, реализующих три операции булевой алгебры (И, ИЛИ, НЕ), в табл. 19 приведены и элементы, реализующие операции производные от основных:

– И-НЕ – отрицание логического умножения, также называется *штрих Шеффера* (обозначается |)

$$\overline{x_1 x_2} = x_1 \mid x_2;$$

– ИЛИ-НЕ – отрицание логического сложения, также называется *стрелка Пирса* (обозначается ↓)

$$\overline{x_1 \vee x_2} = x_1 \downarrow x_2.$$

Последовательно соединяя логические элементы между собой, можно реализовать любую булеву функцию.

Пример 1. Построим комбинационную схему ДНФ от четырех аргументов:

$$F(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C} \vee \overline{B}\,\overline{D}.$$

Для реализации схемы нам понадобятся два конъюнктора: один с тремя и один с двумя входами; а также двухвходовой дизъюнктор. Соединяя эти элементы в порядке выполнения логических операций. Получим схему, представленную на рис. 6.

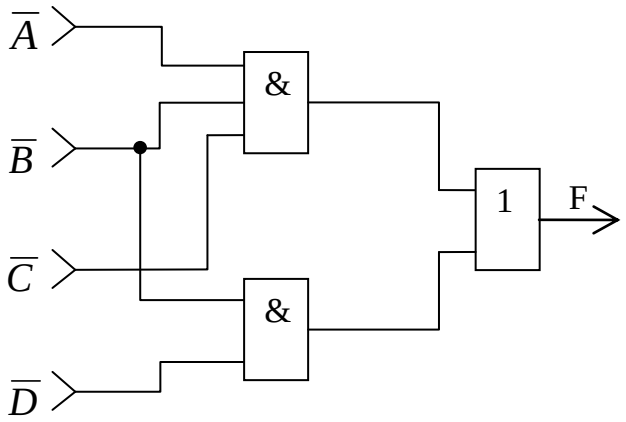


Рис. 6. Комбинационная схема ДНФ из примера 1

1.3.2 Представление БФ в универсальном функциональном базисе

Под *набором булевых функций* будем понимать множество элементарных операций, используемых для представления БФ. Например, набор функций, используемых в ДНФ и КНФ, содержит три операции: {И, ИЛИ, НЕ}. Набор БФ называется *функционально полным*, если с его помощью может быть представлена любая БФ. Очевидно, что набор {И, ИЛИ, НЕ} является функционально полным. Функционально полный набор называется *базисным* (или *базисом*), если при исключении из него любой элементарной операции теряется свойство полноты.

Будем называть базис *универсальным*, если он включает лишь одну логическую операцию.

Известны два универсальных базиса:

– *базис Шеффера*, содержащий логическую операцию И-НЕ;

– *базис Пирса*, содержащий логическую операцию ИЛИ-НЕ.

Поставим задачу преобразования БФ из исходного представления в виде минимальной ДНФ (КНФ) в представление в универсальном базисе. В исходном представлении БФ присутствуют логические операции И и ИЛИ, одна из которых при представлении БФ в любом универсальном базисе является нежелательной. Для ее замены на требуемую логическую операцию будем использовать правило де Моргана.

Пример 2. Исходную БФ от тех переменных, заданную в виде ДНФ, необходимо представить в базисе ИЛИ-НЕ:

$$F = \overline{x_3} x_2 \vee x_3 \overline{x_1} \vee \overline{x_3}\,\overline{x_2}\,\overline{x_1}.$$

Нежелательной в этом выражении является операция И. Чтобы устранить ее, к каждой элементарной конъюнкции применим закон де Моргана:

$$F = \left(\overline{x_3 \vee \overline{x_2}}\right) \vee \left(\overline{\overline{x_3} \vee x_1}\right) \vee \left(\overline{x_3 \vee x_2 \vee x_1}\right).$$

Данная форма не является окончательной, т.к. выражения в скобках объединены с помощью операции ИЛИ, а не ИЛИ-НЕ. Чтобы получить логическое выражение заданной БФ в окончательном виде, необходимо дважды инвертировать правую часть последнего выражения:

$$F = \overline{\overline{\overline{x_3 \vee \overline{x_2}} \vee \overline{\overline{x_3} \vee x_1} \vee \overline{x_3 \vee x_2 \vee x_1}}}.$$

На основе этой формулы, построим схему БФ на элементах Пирса (рис. 7).

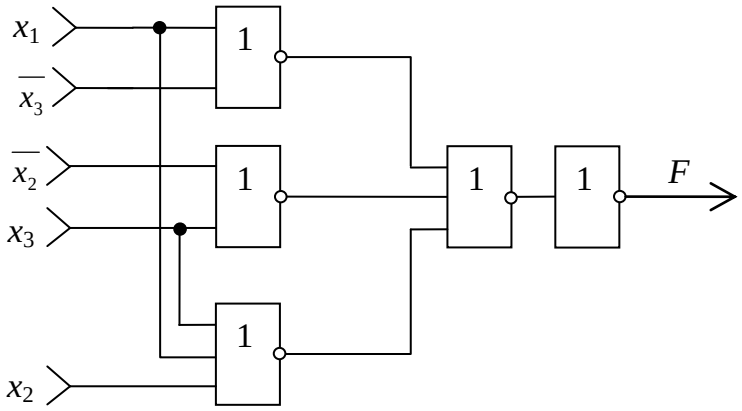


Рис. 7. Комбинационная схема БФ на элементах Пирса

Отметим, что полученная схема содержит элементы с различным числом входов.

2. Задание по работе

Требуется выполнить структурный синтез автомата, построенного при выполнении контрольной работы 3 с использованием триггеров и логических элементов, тип которых указан в разделе «Варианты заданий».

Проверку корректности работы спроектированной функциональной схемы автомата необходимо провести в пакете *Quartus*.

3. Порядок выполнения работы

Выполнение работы состоит из двух частей: структурного синтеза автомата и проверки корректности спроектированной функциональной схемы путем моделирования в пакете *Quartus*.

При выполнении структурного синтеза автомата необходимо

1) последовательно выполнить пункты методики, описанной в разделах 1.1 и 1.2;

2) составить структурные аналоги автоматных лент для каждого из слов оператора соответствия для проведения моделирования.

Пример автоматной ленты для СТПВ, приведенной в табл. 1, представлен ниже.

Входной символ	z_1	z_2	z_1	z_3	z_2	z_3	z_2	z_3	z_1
Состояние	a_0	a_1	a_1	a_0	a_3	a_1	a_0	a_0	a_3
Выходной символ	w_3	w_2	w_1	w_1	w_1	w_2	w_2	w_1	w_1

Входной символ	z_2	z_3	z_3	z_1	z_1	z_3	z_3	z_3	
Состояние	a_2	a_2	a_0	a_3	a_2	a_1	a_0	a_3	a_0
Выходной символ	w_2	w_3	w_1	w_1	w_1	w_2	w_1	w_1	

Структурный аналог этой автоматной ленты в соответствии с табл. 2-4 будет иметь следующий вид.

Входной символ (x_2x_1)	00	01	00	10	01	10	01	10	00
Состояние (Q_2Q_1)	00	01	01	00	11	01	00	00	11
Выходной символ (y_2y_1)	10	01	00	00	00	01	01	00	00

Входной символ (x_2x_1)	01	10	10	00	00	10	10	10	
Состояние (Q_2Q_1)	10	10	00	11	10	01	00	11	00
Выходной символ (y_2y_1)	01	10	00	00	00	01	00	00	

При выполнении работы в пакете *Quartus* необходимо выполнить следующие действия.

1) создать проект и построить схему автомата, полученную в результате структурного синтеза, на логических элементах заданного базиса и триггерах (рекомендуется использовать триггеры из библиотеки *primitives/storage*: *jkff* для *JK*-триггеров и *T*-триггеров, *dff* для *D*-триггеров);

2) провести компиляцию проекта;
3) создать файл временных диаграмм, задать входные сигналы в соответствии с построенной автоматной лентой (см. рекомендации ниже);
4) провести функциональное моделирование и сравнить результат с автоматной лентой.

Ниже приведены рекомендации по подготовке и проверке временных диаграмм.

Целесообразно установить длительность одного такта (*Grid Size***)** равную 5 *ns*. Тогда время моделирования *End Time* следует установить равным 5 × (число столбцов в автоматной ленте). Для приведенного выше примера *End Time* = = 5 × 18 = 90 *ns*. Из первой строки структурной автоматной ленты следует, что временные диаграммы входных сигналов должны иметь вид:

x_2: 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1;

x_1: 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0.

Для установки значений входных сигналов в поле *Name* нужно выделить имя x_2 и на панели слева выбрать пиктограмму 0. Затем в строке x_2 нужно выделить нужные интервалы с помощью клавиши *CTRL* и установить значение 1. Аналогично формируется временная диаграмма для сигнала x_1. Для удобства проверки входные сигналы можно объединить в шину.

Временная диаграмма синхросигналов формируется с помощью пиктограммы с изображением часов. В появившемся окне необходимо установить длительность синхроимпульса (*Period*) равную длительности такта (*Grid Size*), в нашем случае это 5 *ns***.**

Проверка правильности работы автомата по временным диаграммам осуществляется на основе автоматной ленты. Последовательности значений выходных сигналов на диаграмме и на ленте должны совпадать.

Внимание! Моменты регистрации выходных сигналов в каждом машинном такте различны для автомата модели Мили и для автомата модели Мура.

Триггеры из библиотеки *primitives* в *Quartus* срабатывают по переднему фронту, поэтому в автомате модели Мили выходные сигналы должны

регистрироваться до переднего фронта синхросигнала (когда синхросигнал равен нулю). Это связано с тем, что выходной сигнал автомата Мили зависит от исходного состояния автомата и входного сигнала.

В автомате модели Мура выходные сигналы должны регистрироваться после переднего фронта синхросигнала, так как они зависят только от состояния перехода.

4. Содержание отчета

Отчет по лабораторной работе должен содержать следующие разделы.

1. Цель работы.
2. Задание по работе.

Раздел должен включать в себя формулировку задания и данные варианта.

3. Структурный синтез конечного автомата.

В данном разделе необходимо привести описание этапов структурного синтеза автомата с указанием номеров и названий этапов в соответствии с приведенным выше примером.

4. Моделирование работы структурного автомата.

В данном разделе необходимо привести схему структурного автомата, построенную в QUARTUS, автоматные ленты для проверки переходов структурного автомата и скриншот временных диаграмм с результатами моделирования автоматной ленты.

5. Вывод.

Пример вывода: «В результате выполнения работы произведен структурный синтез автомата модели Мура. Автомат имеет 3 структурных входа и 2 структурных выхода. Для реализации автомата потребовалось 4 триггера типа D и 15 элементов И-НЕ. Моделирование работы структурного автомата произведено в пакете QUARTUS. Изучен канонический метод структурного синтеза конечных автоматов; получены навыки построения структурных схем конечных автоматов».

5. Контрольные вопросы

1. Назовите три составляющих структурной схемы автомата, синтезируемого по исходному АА.

2. Как выполнить *T*-триггер на базе *JK*-триггера?

3. Исходный АА работает во входном алфавите из 20 символов, в выходном алфавите из 12 символов и имеет 35 состояний. Определить количество структурных входных и выходных каналов, а также элементов памяти (триггеров) для хранения состояний в структурном автомате, который будет синтезирован по данному абстрактному автомату.

4. Дана таблица переходов автомата Мили **(**табл. 19). Составить КТП, произведя предварительно кодирование входных символов и состояний.

Таблица 19

	a_0	a_1	a_2
z_1	a_1	a_0	a_0
z_2	a_0	a_2	a_2
z_3	a_0	a_0	a_0

Таблица 20

	a_0	a_1	a_2
z_1	w_1	w_1	w_2
z_2	w_2	w_1	w_1
z_3	w_3	w_3	w_3

5. Дана таблица выходов автомата Мили (табл. 20). Составить КТВ, произведя предварительно кодирование входных, выходных символов и состояний.

6. Дана ОТП автомата Мура **(**табл. 21). Составить КТП, произведя предварительно кодирование входных символов и состояний.

Таблица 21

	w_2	w_3	w_1
	a_0	a_1	a_2
z_1	a_1	a_0	a_2
z_2	a_0	a_2	a_1
z_3	a_1	a_0	a_0

Таблица 22

Q_2Q_1 / x_2x_1	00	01	10	11
00	01	10	10	10
01	00	01	01	00
10	10	00	01	00

7. Дана ОТП автомата Мура **(**табл. 21). Составить КТВ, произведя предварительно кодирование выходных символов и состояний.

8. Дана КТП **(**табл. 22). Составить КТФВ для триггеров типа a) *T*; b) *JK*; c) *D*.

9. Выпишите минимальное выражение для булевой функции из диаграммы Вейча а) по единицам; б) по нулям.

			$\overline{C}$				
			$\overline{B}$				
		$\overline{A}$			$\overline{\quad}$		
		1	-	-	-	1	1	0	1
	D	1	1	-	0	1	0	1	1
		1	-	-	1	0	0	-	1
E		1	0	0	1	-	0	0	-

10. Задана ОТП:

a)

	w_1	w_2
	a_1	a_2
z_1	a_1	a_1
z_2	a_2	a_2

b)

	w_1	w_2
	a_1	a_2
z_1	a_1	a_2
z_2	a_2	a_1

c)

	w_1	w_2
	a_1	a_2
z_1	a_1	a_2
z_2	a_1	a_1
z_3	a_2	a_2
z_4	a_2	a_1

Постройте структурный автомат Мура с использованием любых логических элементов и триггеров типа

d) *D*; e) *T*; f) *JK*.

6. Варианты заданий

Номер варианта	Тип триггера	Элементный базис	Номер варианта	Тип триггера	Элементный базис
1	*D*	И-НЕ	**16**	*JK*	ИЛИ-НЕ
2	*T*	ИЛИ-НЕ	**17**	*D*	И-НЕ
3	*JK*	И-НЕ	**18**	*T*	ИЛИ-НЕ
4	*D*	ИЛИ-НЕ	**19**	*D*	ИЛИ-НЕ
5	*T*	И-НЕ	**20**	*T*	И-НЕ
6	*JK*	ИЛИ-НЕ	**21**	*JK*	ИЛИ-НЕ
7	*T*	ИЛИ-НЕ	**22**	*D*	И-НЕ
8	*JK*	И-НЕ	**23**	*T*	ИЛИ-НЕ
9	*D*	ИЛИ-НЕ	**24**	*JK*	И-НЕ
10	*T*	И-НЕ	**25**	*T*	И-НЕ
11	*JK*	ИЛИ-НЕ	**26**	*JK*	ИЛИ-НЕ
12	*D*	И-НЕ	**27**	*D*	И-НЕ
13	*JK*	И-НЕ	**28**	*T*	ИЛИ-НЕ
14	*D*	ИЛИ-НЕ	**29**	*JK*	И-НЕ
15	*T*	И-НЕ	**30**	*D*	ИЛИ-НЕ

Лабораторная работа 2
СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ

Цель работы: изучение основ проектирования микропрограммных автоматов (МПА), приобретение навыков построения структурных схем МПА.

1. Основные теоретические сведения

1.1. Принцип микропрограммного управления

Для выполнения операций над информацией используются операционные устройства (ОУ): процессоры, каналы ввода-вывода, устройства управления внешними устройствами и т.д.

Функцией ОУ является выполнение заданного множества *операций* $F = \{f_1, ..., f_G\}$ над входными словами $D = \{d_1, ..., d_H\}$ с целью вычисления слов $R = \{r_1, ..., r_Q\}$, представляющих результаты операций ($R = f_g(D)$). Здесь G, H, Q – количество различных операций, выполняемых ОУ, число допустимых входных слов и число возможных различных выходных слов ОУ соответственно, $g = 1, .., G$.

Функциональная и структурная организация ОУ базируется на *принципе микропрограммного управления*, который состоит в следующем [3].

1. Любая операция f_g ($g = 1, .., G$), реализуемая устройством, рассматривается как сложное действие, которое разделяется на последовательность элементарных действий над словами информации. Эти элементарные действия называются *микрооперациями.* Например, к микрооперациям относятся: передача информации из одного регистра в другой, формирование обратного кода, сдвиг содержимого регистра и т.д.

2. Для управления порядком следования микроопераций используются логические условия, которые в зависимости от значений слов, преобразуемых микрооперациями, принимают значения "ложь" или "истина" (0 или 1). Примерами логических условий могут служить следующие: содержимое регистра равно нулю; содержимое регистра является четным числом.

3. Процесс выполнения операций в ОУ описывается в форме алгоритма, который представляется в терминах микроопераций и логических условий и называется *микропрограммой*. Микропрограмма определяет порядок проверки логических условий и следования микроопераций, необходимый для получения требуемых результатов.

4. Микропрограмма используется как форма представления функции ОУ, на основе микропрограммы определяется структура и порядок функционирования ОУ во времени.

Сказанное можно рассматривать как содержательное описание принципа микропрограммного управления (МПУ), из которого следует, что структура и порядок функционирования операционных устройств предопределяется алгоритмом выполнения операции из множества $F = \{f_1, ..., f_G\}$.

Схемные решения ОУ могут быть различны, но во всех случаях точка зрения на процесс функционирования ОУ как процесс реализации микроопераций и проверки логических условий, предопределяемый микропрограммой, является результативной, поскольку позволяет упорядочить и формализовать проектирование ОУ различного назначения.

1.2. Обобщенная структурная схема операционного устройства

В функциональном и структурном отношении ОУ разделяется на две части: операционный автомат (ОА) и управляющий автомат (УА) (рис. 1).

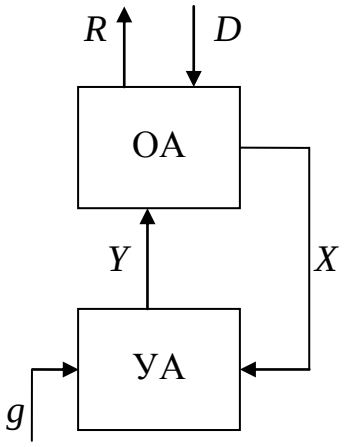


Рис. 1. Обобщенная структурная схема ОУ

Операционный автомат (ОА) служит для хранения слов информации, выполнения набора микроопераций и вычисления значений логических условий, т.е. операционный автомат является структурой, организованной для выполнения действий над информацией. Микрооперации, выполняемые ОА, задаются множеством *управляющих сигналов* $Y = \{y_1, ..., y_M\}$, с каждым из которых отождествляется определенная микрооперация в том смысле, что выполнение любой микрооперации инициируется определенным управляющим сигналом. Значения логических условий, вычисляемые в операционном автомате, отображаются множеством *осведомительных сигналов* $X = \{x_1, ..., x_L\}$, каждый из которых отождествляется с определенным логическим условием. M – число возможных микроопераций. L – число условий.

Управляющий автомат (УА) генерирует последовательность управляющих сигналов, предписанную микропрограммой и соответствующую значениям логическим условий. Иначе говоря, управляющий автомат задает порядок выполнения микроопераций в ОА, вытекающий из алгоритма выполнения операций. Наименование операции, которую необходимо выполнить в устройстве, определяется *кодом операции* g ($g = 1, \ldots, G$), поступающим в УА извне. Сигналы $g_1, ..., g_h$, представляющие собой разряды двоичного числа g ($g_i = 0, 1$; $i = 1, \ldots, h$; $h =]\log_2 G[$), и осведомительные сигналы $x_1, ..., x_L$, поступающие в УА от ОА, играют одинаковую роль, так как и те и другие влияют на порядок выработки управляющих сигналов Y (сначала $g_1, ..., g_h$ определяют алгоритм, который будет выполняться ОА, а затем $x_1, ..., x_L$ определяют порядок следования микроопераций в этом алгоритме). Поэтому сигналы $g_1, ..., g_h$ и $x_1, ..., x_L$ относятся к одному классу – к классу осведомительных сигналов, поступающих на вход УА.

Таким образом, любое операционное устройство – процессор, канал ввода-вывода и т. д. – является композицией операционного и управляющего автоматов. Операционный автомат, реализуя действия над словами информации, является исполнительной частью устройства. Работой ОА управляет УА, генерирующий необходимые последовательности управляющих сигналов. Поскольку УА

генерирует последовательность управляющих сигналов, предписанную микропрограммой, его часто называют *микропрограммным автоматом* (МПА).

Одной из форм задания функции МПА является *граф-схема алгоритма* (ГСА).

1.3. Граф-схемы алгоритмов

Граф-схема алгоритма – это ориентированный связный граф, содержащий вершины четырех типов: начальную, конечную, операторную и условную (рис. 2).

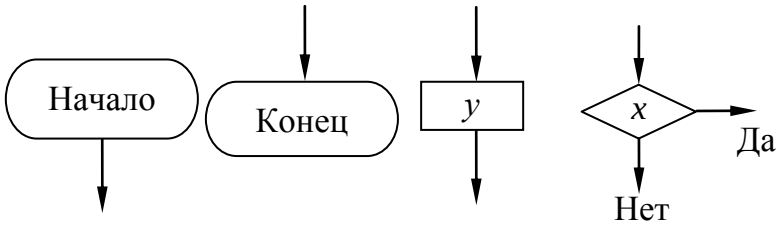


Рис. 2. Типы вершин граф-схемы алгоритма;
слева направо: начальная, конечная, операторная, условная

Начальная вершина имеет один выход и ни одного входа; конечная – один вход и ни одного выхода; операторная – один вход и один выход; условная – один вход и два выхода, помеченные словами да и нет (или 1 и 0 соответственно).

Граф-схема алгоритма удовлетворяет следующим условиям [2]:

1) содержит конечное число вершин, каждая из которых принадлежит одному из перечисленных выше типов;

2) имеет точно одну начальную и одну конечную вершину;

3) входы и выходы вершин соединяются друг с другом с помощью дуг, направленных всегда от выхода к входу;

4) каждый выход соединен точно с одним входом;

5) любой вход соединен, по крайней мере, с одним выходом;

6) для любой вершины графа существует, по крайней мере, один путь из этой вершины к конечной;

7) один из выходов условной вершины может соединяться с ее входом;

8) в каждой из условных вершин записывается один из элементов множества $X = \{x_1, ..., x_L\}$, называемого *множеством логических условий,* которые в виде осведомительных сигналов поступают из ОА в УА;

9) в каждой операторной вершине записывается произвольное подмножество множества $Y = \{y_1, ..., y_M\}$, называемого *множеством микроопераций*. С каждой микрооперацией связывается управляющий сигнал, инициирующий выполнение этой операции.

В качестве примера рассмотрим граф-схему алгоритма, представленную на рис. 3.

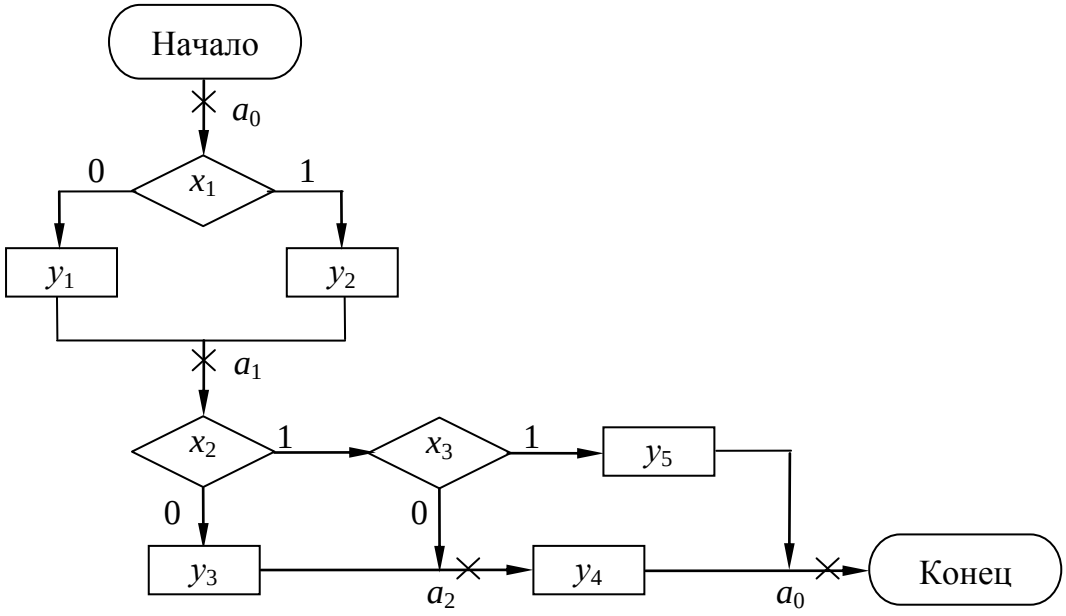


Рис. 3. Пример граф-схемы алгоритма

На рис. 3 множество логических условий содержит три элемента $X = \{x_1, x_2, x_3\}$, а множество микроопераций – пять элементов $Y = \{y_1, y_2, y_3, y_4, y_5\}$. Следовательно, МПА, поведение которого задается этой граф-схемой, должен иметь три входных и пять выходных каналов.

Для структурной реализации автомата будем использовать триггеры с синхронным способом управления. Напомним, что переход синхронных триггеров из состояния в состояние происходит только при изменении значения синхросигнала (например, 0 →1).

Поясним работу автомата, заданного ГСА, на примере ГСА (рис. 3).

Начальной вершине граф-схемы соответствует начальное состояние автомата, при котором на всех выходных каналах вырабатывается нулевой сигнал.

Если в начальном состоянии на вход автомата поступит сигнал $x_1 = 1$, то независимо от сигнала на входах x_2 и x_3 при подаче синхросигнала автомат перейдет в новое состояние, на выходном канале y_2 появится сигнал, равный единице, а на остальных – сигнал, равный нулю.

При появлении следующего тактового импульса входной сигнал $x_2 = 0$ независимо от значений остальных входных сигналов вызовет появление выходного сигнала y_3, и автомат перейдет в новое состояние. Если же $x_2 = 1$, то выходной сигнал будет зависеть от значения на входе x_3.

Отметим отдельно два случая расположения вершин в ГСА.

1. Подряд следуют несколько операторных вершин (например, y_3 и y_4 на рис. 3). В этом случае автомат будет поочередно вырабатывать выходные сигналы, соответствующие этим операторным вершинам, (по одному выходному сигналу за один такт, т.е. при подаче синхросигнала) независимо от значений входных сигналов.

2. Если после операторной вершины стоит возвратная условная вершина с логическим условием x_k (рис. 4), то далее во всех тактах все выходные сигналы будут равны нулю, пока не появится сигнал $x_k = 1$. С помощью возвратной условной вершины можно описывать ожидание в работе автомата.

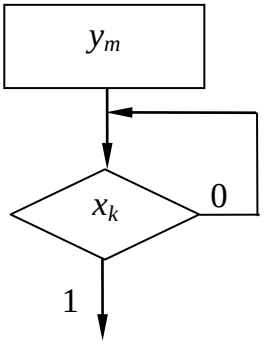


Рис. 4. Возвратная вершина

По завершении выполнения микропрограммы автомат должен вернуться в начальное состояние, поэтому переходу в конечную вершину ГСА соответствует переход автомата в начальное состояние.

1.4. Структурный синтез МПА

1.4.1. МПА модели Мили

Для синтеза микропрограммного автомата введем соответствие между элементами граф-схемы алгоритма и элементами автомата.

Предварительно выполним разметку ГСА символами a_0, **...**, a_h по правилам:

1) символом a_0 отмечается вход вершины, следующей за начальной, и вход конечной вершины;

2) вход каждой вершины, следующей за операторной, отмечается символом a_i;

3) если вход вершины соединен с выходами нескольких операторных вершин, то он отмечается лишь один раз.

Введем понятие *пути перехода* на граф-схеме от a_m к a_l как пути в направлении дуг графа, проходящего не более чем через одну операторную вершину:

$$a_m x(a_m,a_l) y(a_m,a_l) a_l,$$

где $x(a_m,a_l)$ – конъюнкция содержимого условных вершин на пути перехода, взятых в прямом виде, если путь выходит из вершины по стрелке да (1), или в инверсном, если путь выходит по стрелке нет (0); $y(a_m,a_l)$ – содержимое операторной вершины. Путь заканчивается в первой отметке после вершины $y(a_m,a_l)$.

Допустимы пути, ведущие в ту же самую отметку:

$$a_m x(a_m,a_m) y(a_m,a_m) a_m,$$

пути, не содержащие условных вершин:

$$a_m y(a_m,a_l) a_l,$$

и пути, не содержащие операторной вершины:

$$a_m x(a_m,a_l) a_l.$$

Для граф-схемы алгоритма (рис. 3) введем отметки a_0, a_1, a_2 и получим полное множество путей перехода

$$\{a_0 x_1 y_2 a_1, a_0 \overline{x}_1 y_1 a_1, a_1 \overline{x}_2 y_3 a_2, a_1 x_2 \overline{x}_3 a_2, a_1 x_2 \overline{x}_3 y_4 a_0, a_1 x_2 x_3 y_5 a_0, a_2 y_4 a_0\}. \quad (1)$$

После определения множества путей перехода микропрограммный автомат может быть представлен в стандартной графической или табличной форме.

Состояниям автомата ставим в соответствие отметки на граф-схеме алгоритма. За начальное состояние принимаем отметку a_0.Примем, что между состояниями автомата имеются переходы, если соответствующие отметки на граф-схеме связаны путем перехода. Входной сигнал, определяющий переход, полагаем равным $x(a_m,a_l)$ – конъюнкции содержимого условных вершин на пути перехода, а выходной сигнал равным $y(a_m,a_l)$ – содержимому операторной вершины на пути перехода. Для путей перехода вида $a_m y(a_m,a_l)a_l$ роль входного сигнала будет исполнять синхросигнал.

Для путей перехода вида $a_m x(a_m,a_l)a_l$ все выходные сигналы полагаем равными нулю.

Отметим, что учет путей перехода, не проходящих через операторную вершину, увеличивает время выполнения микропрограммы за счет наличия пустых тактов, хотя и снижает сложность комбинационных схем.

В дальнейшем будем рассматривать только те пути перехода, которые содержат операторную вершину.

По этой причине во множестве путей перехода (1) в дальнейшем не будет использоваться четвертый путь перехода.

В результате отождествления элементов ГСА с элементами автомата получаем автомат Мили, имеющий столько же состояний, сколько символов потребовалось для отметки вершин на граф-схеме алгоритма.

Конечный автомат, эквивалентный ГСА (рис. 3), можно представить в виде графа переходов (рис. 5).

Структурный синтез автомата можно производить, используя канонический метод, но в связи с большим объемом кодированных таблиц переходов и выходов (для шести входных сигналов таблица переходов будет содержать $2^6 = 64$ строки) удобнее пользоваться структурными таблицами, которые строятся на основе граф-схемы алгоритма или графа переходов МПА.

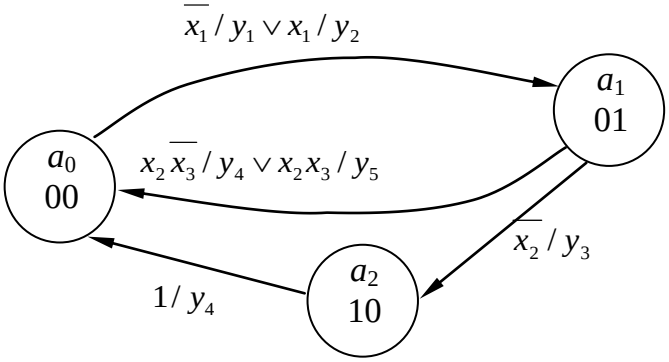


Рис. 5. Граф переходов МПА модели Мили

Примечания к рис. 5:

1)если между двумя состояниями имеется несколько путей перехода, допускается замена соответствующих дуг между этими состояниями одной дугой, причем отметка на этой дуге формируется как дизъюнкция отметок на заменяемых дугах;

2)в каждой вершине графа вместе с абстрактным символом состояния помещен его структурный эквивалент (см. указания к лабораторной работе 3), в дальнейшем символ состояния можно не указывать.

Примеры таких структурных таблиц для МПА модели Мили, граф переходов которого изображен на рис. 5, представлены в табл. 1 и 2.

Таблица 1

Исходное состояние	Код исходного состояния Q_2Q_1	Состояние перехода	Код состояния перехода Q_2Q_1	Входной сигнал	Выходной сигнал	Обязательная функция возбуждения		
						Триггер типа *JK*	Триггер типа *T*	Триггер типа *D*
a_0	00	a_1	01	x_1	y_1	J_1	T_1	D_1
		a_1	01	$\overline{x_1}$	y_2	J_1	T_1	D_1
a_1	01	a_0	00	$x_2\overline{x_3}$	y_4	K_1	T_1	
		a_0	00	x_2x_3	y_5	K_1	T_1	
a_1	01	a_2	10	$\overline{x_2}$	y_3	J_2, K_1	T_2, T_1	D_2
a_2	10	a_0	00	1	y_4	K_2	T_2	

Табл. 1 представляет собой *прямую структурную таблицу*. В первом столбце прямой таблицы последовательно перечисляются все состояния, начиная с нулевого. Коды этих состояний заносятся во второй столбец. В третьем и четвертом столбцах записываются состояния переходов и их коды. Если между двумя состояниями есть несколько путей перехода, даже содержащих один и тот же выходной сигнал, то для каждого пути отводится в таблице отдельная строчка.

Пятый и шестой столбцы содержат входные и выходные сигналы для соответствующего перехода.

В седьмом столбце таблицы перечисляются обязательные функции возбуждения, вырабатываемые на соответствующих переходах, то есть функции возбуждения, которые для осуществления данного перехода должны быть равны единице (см. таблицу функционирования триггеров в лабораторной работе 3).

Например, в первой строке табл. 1 описан переход автомата из состояния 00 в состояние 01. Это означает, что первый триггер должен переключиться из 0 в 1, а второй – остаться в состоянии 0. Для переключения *JK*-триггера из 0 в 1 нужно подать 1 на его вход *J*, поэтому в первой строке табл. 1 поставлен символ J_1.

В ряде случаев удобнее пользоваться *обратной структурной таблицей* автомата, которая отличается от прямой таблицы тем, что в ней последовательно перечисляются все переходы в первое состояние, затем во второе и так далее (табл. 2).

Таблица 2

Исходное состояние	Код исходного состояния Q_2Q_1	Состояние перехода	Код состояния перехода Q_2Q_1	Входной сигнал	Выходной сигнал	Обязательная функция возбуждения
a_1	01	a_0	00	$x_2\overline{x_3}$	y_4	K_1
a_1	01			x_2x_3	y_5	K_1
a_2	10			1	y_4	K_2
a_0	00	a_1	01	x_1	y_1	J_1
a_0	00			$\overline{x_1}$	y_2	J_1
a_1	01	a_2	10	$\overline{x_2}$	y_3	J_2, K_1

По структурным таблицам записываются ЛВ сигналов возбуждения при построении КС1 и ЛВ выходных сигналов автомата при построении КС2.

Построение КС1

Напомним, что структурная схема автомата, в том числе и МПА, содержит три составляющих:

1)КС1, на выходах которой вырабатываются функции возбуждения, таким образом она определяет переходы МПА из одного состояние в другое;

2)КС2, на выходах которой вырабатываются выходные сигналы МПА;

3)необходимое количество триггеров для представления состояний МПА.

Для построения КС1 из структурной таблицы выписываются логические выражения для функций возбуждения. С этой целью в таблице выбираются строки, содержащие одинаковые отметки в последнем столбце. Для каждой строки записывается конъюнкция исходного состояния и входного сигнала. Если строк несколько, полученные конъюнкции объединяются знаком дизъюнкции. Таким образом, образуются ДНФ функций возбуждения, по которым затем строится схема в обычном базисе (И, ИЛИ, НЕ).

Для рассматриваемого примера получаем следующие функции возбуждения *JK*-триггеров:

$$J_2 = \overline{Q_2} Q_1 \overline{x_2},\ K_2 = Q_2 \overline{Q_1},\ J_1 = \overline{Q_2}\,\overline{Q_1} x_1 \vee \overline{Q_2}\,\overline{Q_1}\,\overline{x_1} = \overline{Q_2}\,\overline{Q_1},$$
$$K_1 = \overline{Q_2} Q_1 x_2 \overline{x_3} \vee \overline{Q_2} Q_1 x_2 x_3 \vee \overline{Q_2} Q_1 \overline{x_2} = \overline{Q_2} Q_1. \qquad (2)$$

Здесь Q_2 и Q_1 – выходные сигналы второго и первого триггеров памяти (в данном примере использованы триггеры типа *JK*).

Рассмотрим возможность уменьшения ранга конъюнкций в ЛВ для функций возбуждения и, как следствие, сокращения числа входов на конъюнкторы в схеме. Она связана с вводом в состав схемы дешифратора с двумя входами и четырьмя выходами (рис. 6).

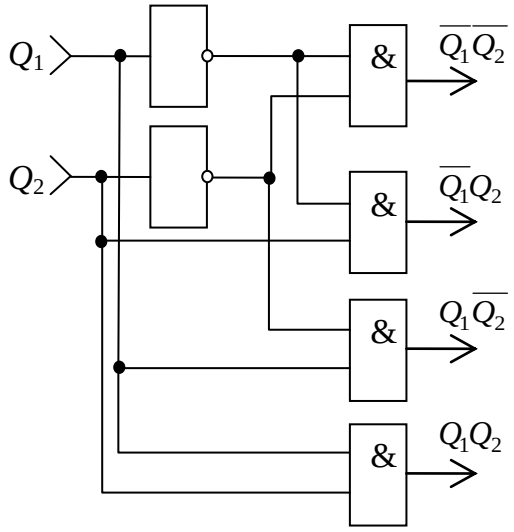


Рис. 6. Схема дешифратора 2×4

Условное графическое обозначение дешифратора изображено на рис. 7.

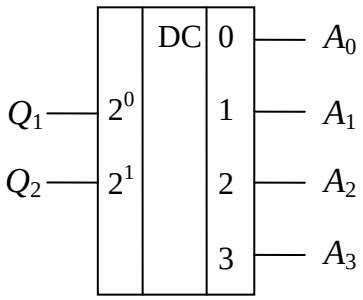


Рис. 7. Условное графическое обозначение дешифратора 2×4

Если к входам дешифратора подключить выходные сигналы триггеров, на его выходах будут формироваться сигналы, соответствующие состояниям автомата:

$$A_0 = \overline{Q_2}\,\overline{Q_1},\ A_1 = \overline{Q_2}Q_1,\ A_2 = Q_2\overline{Q_1},\ A_3 = Q_2Q_1.$$

Учитывая конкретный вид уравнений, систему (2) для функций возбуждения триггеров можно переписать в более простом виде:

$$J_2 = A_1\overline{x_2},\ K_2 = A_2,\ J_1 = A_0,\ K_1 = A_1.$$

Построение КС2

ЛВ для выходных сигналов записываются аналогично тому, как это делалось для функций возбуждения, при этом рассматриваются отметки выходных сигналов в столбце 6 структурной таблицы. Запишем эти выражения, предполагая наличие в схеме дешифратора:

$$y_1 = A_0x_1,\ y_2 = A_0\overline{x_1},\ y_3 = A_1\overline{x_2},\ y_4 = A_1x_2\overline{x_3} \vee A_2,\ y_5 = A_1x_2x_3.$$

По логическим выражениям, полученным для функций возбуждения и выходных сигналов, составляется функциональная схема автомата.

1.4.2. МПА модели Мура

Рассмотрим особенности синтеза МПА модели Мура. Напомним, что в автомате модели Мура выходные сигналы, в отличие от автомата Мили, зависят только от состояния. Поэтому правила разметки ГСА несколько отличаются от таковых для автомата Мили и состоят в следующем:

1)начальная и конечная вершины ГСА отмечаются одинаковой отметкой a_0;

2)каждая операторная вершина отмечается своей отметкой.

В связи с этим индексы состояний и соответствующих им выходных сигналов, как правило, совпадают.

Разметка ГСА, приведенной на рис. 3, для автомата Мура изображена на рис. 8.

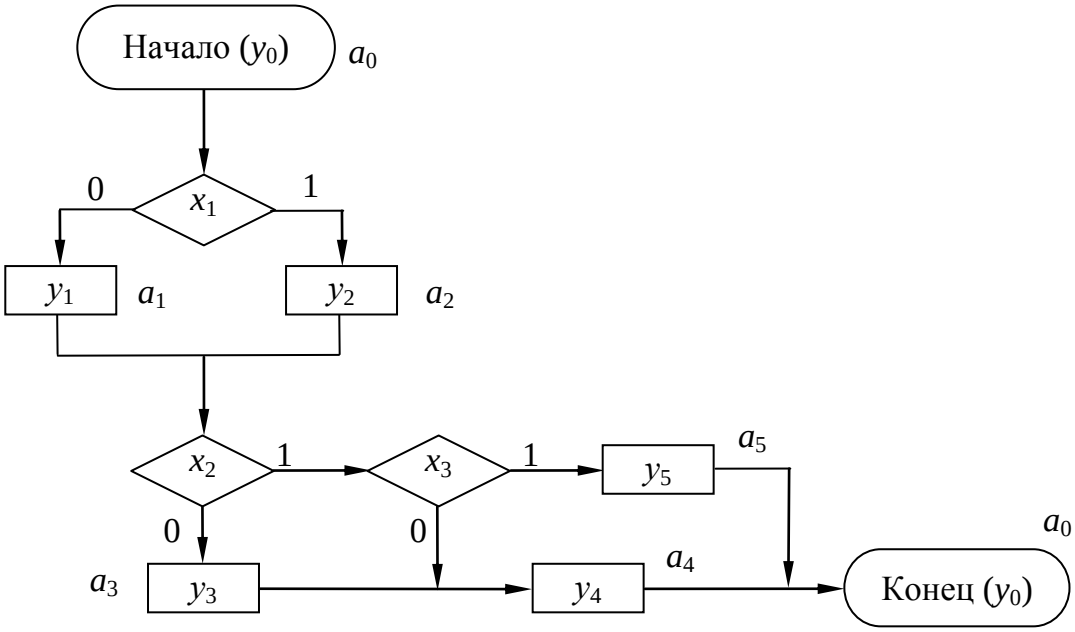


Рис. 8. Разметка ГСА МПА модели Мура

Путь перехода от a_m к a_l на граф-схеме МПА модели Мура определяется как путь в направлении дуг графа, проходящего от вершины a_m к вершине a_l**:**

$$a_m x(a_m,a_l) a_l y_l,$$

где $x(a_m,a_l)$ – конъюнкция содержимого условных вершин на пути перехода, взятых в прямом виде, если путь выходит из вершины по стрелке да (1), или в инверсном, если путь выходит по стрелке нет (0); y_l – содержимое операторной вершины a_l. Путь заканчивается в вершине a_l.

Приведем множество путей перехода для автомата Мура (рис. 8):

$$\{a_0\overline{x_1}a_1y_1, a_0x_1a_2y_2, a_1x_2x_3a_5y_5, a_1x_2\overline{x_3}a_4y_4, a_1\overline{x_2}a_3y_3, a_2x_2x_3a_5y_5,$$
$$a_2x_2\overline{x_3}a_4y_4, a_2\overline{x_2}a_3y_3, a_3a_4y_4, a_4a_0y_0, a_5a_0y_0\}.$$

При этом будем считать, что с начальной и конечной вершиной ГСА связан выходной сигнал y_0.Отметим, что этот сигнал в отличие от других выходных сигналов не инициирует выполнение какой-либо микрооперации, а выполняет функцию осведомительного сигнала, указывая либо на готовность МПА к работе, либо на завершение автоматом выполнения алгоритма.

По множеству путей перехода строится граф переходов МПА модели Мура. В данном случае он содержит шесть вершин по числу состояний (рис. 9). В каждую вершину записывается состояние и выходной сигнал, находящийся в операторной вершине ГСА, отмеченной данным состоянием (рис. 8). Между вершинами размещаются дуги, всегда направленные от вершины с исходным состоянием к вершине с состоянием перехода. Каждая дуга имеет отметку (помещается в начале дуги) в виде соответствующего входного сигнала (отметка 1 означает, что переход выполняется при подаче синхросигнала независимо от значений входных сигналов x_1, x_2, x_3).

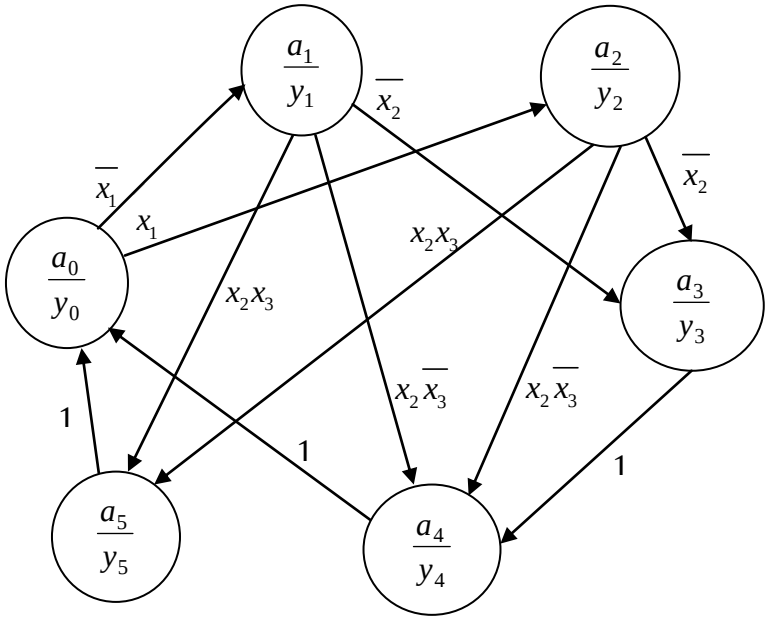


Рис. 9. Граф переходов МПА модели Мура

По графу переходов может быть построена прямая и обратная структурные таблицы (табл. 3 и 4). Правила их построения и заполнения такие же, как и для МПА модели Мили.

Внимание! В табл. 3 переставлены местами столбцы 5 и 6. Это сделано для того, чтобы подчеркнуть зависимость выходных сигналов в МПА модели Мура только от состояния.

Таблица 3

Исх. сост.	Код исх. сост. $Q_3Q_2Q_1$	Сост. перех.	Код сост. перех. $Q_3Q_2Q_1$	Выходной сигнал	Входной сигнал	Обязательная функция возбуждения		
						Триггер *JK*	Триггер *T*	Триггер *D*
a_0	000	a_1	001	y_1	$\overline{x_1}$	J_1	T_1	D_1
		a_2	010	y_2	x_1	J_2	T_2	D_2
a_1	001	a_3	011	y_3	$\overline{x_2}$	J_2	T_2	D_2, D_1
		a_4	100	y_4	$x_2\overline{x_3}$	J_3, K_1	T_3,T_1	D_3
		a_5	101	y_5	x_2x_3	J_3	T_3	D_3, D_1
a_2	010	a_3	011	y_3	$\overline{x_2}$	J_1	T_1	D_2, D_1
		a_4	100	y_4	$x_2\overline{x_3}$	J_3,K_2	T_3,T_2	D_3
		a_5	101	y_5	x_2x_3	J_3,K_2, J_1	T_3,T_2, T_1	D_3, D_1
a_3	011	a_4	100	y_4	1	J_3,K_2, K_1	T_3,T_2, T_1	D_3
a_4	100	a_0	000	y_0	1	K_3	T_3	
a_5	101	a_0	000	y_0	1	K_3, K_1	T_3,T_1	

Таблица 4

Исх. сост.	Код исх. сост. $Q_3Q_2Q_1$	Сост. перех.	Код сост. перех. $Q_3Q_2Q_1$	Выходной сигнал	Входной сигнал	Обязательная функция возбуждения		
						Триггер *JK*	Триггер *T*	Триггер *D*
a_4	100	a_0	000	y_0	1	K_3	T_3	
a_5	101				1	K_3, K_1	T_3,T_1	
a_0	000	a_1	001	y_1	$\overline{x_1}$	J_1	T_1	D_1
a_0	000	a_2	010	y_2	x_1	J_2	T_2	D_2
a_1	001	a_3	011	y_3	$\overline{x_2}$	J_2	T_2	D_2, D_1
a_2	010				$\overline{x_2}$	J_1	T_1	D_2, D_1
a_1	001	a_4	100	y_4	$x_2\overline{x_3}$	J_3, K_1	T_3,T_1	D_3
a_2	010				$x_2\overline{x_3}$	J_3,K_2	T_3,T_2	D_3
a_3	011				1	J_3,K_2, K_1	T_3,T_2, T_1	D_3
a_1	001	a_5	101	y_5	x_2x_3	J_3	T_3	D_3, D_1
a_2	010				x_2x_3	J_3,K_2, J_1	T_3,T_2, T_1	D_3, D_1

В структурной схеме МПА модели Мура КС2 будет содержать только дешифратор на три входа и восемь выходов. К его входам будут подключаться сигналы с выходов триггеров Q_3, Q_2 и Q_1**,**а на его выходах будут формироваться выходные сигналы МПА (в данном примере два выхода A_6 и A_7 не используются):

$$A_0 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1},\ A_1 = \overline{Q_3}\,\overline{Q_2}Q_1,\ A_2 = \overline{Q_3}Q_2\overline{Q_1},\ A_3 = \overline{Q_3}Q_2Q_1,\ A_4 = Q_3\overline{Q_2}\,\overline{Q_1},\ A_5 = Q_3\overline{Q_2}Q_1.$$

Также как и для автомата Мили, выходные сигналы дешифратора могут быть использованы в схеме КС1 для сокращения числа входов на конъюнкторы.

Отметим, что построение КС1 в структурной схеме МПА модели Мура и модели Мили осуществляется аналогично.

В качестве примера, приведем несколько логических выражений для функций возбуждения, по которым будут построены схемы, входящие в состав КС1. Пусть третий триггер – *JK*-типа, второй триггер – типа *T*, первый триггер – типа *D*. Тогда по табл. 4 с учетом использования выходных сигналов дешифратора A_i (i = 0, .., 5) определяем логические выражения для функций возбуждения:

$$J_3 = A_1 x_2 \overline{x_3} \vee A_2 x_2 \overline{x_3} \vee A_3 \vee A_1 x_2 x_3 \vee A_2 x_2 x_3 = A_1 x_2 \vee A_2 x_2 \vee A_3;$$

$$K_3 = A_4 \vee A_5;$$

$$T_2 = A_0 x_1 \vee A_1 \overline{x_2} \vee A_2 x_2 \overline{x_3} \vee A_3 \vee A_2 x_2 x_3 = A_0 x_1 \vee A_1 \overline{x_2} \vee A_2 x_2 \vee A_3;$$

$$D_1 = A_0 \overline{x_1} \vee A_1 \overline{x_2} \vee A_2 \overline{x_2} \vee A_1 x_2 x_3 \vee A_2 x_2 x_3 = A_0 \overline{x_1} \vee A_1 \overline{x_2} \vee A_2 \overline{x_2} \vee A_1 x_3 \vee A_2 x_3.$$

При преобразовании логических выражений для функций возбуждения были использованы следствия из законов булевой алгебры.

2. Задание по работе

Дана графическая схема алгоритма (см. раздел «Варианты заданий»). Необходимо построить функциональную схему МПА (для нечетных вариантов – модели Мили, для четных – модели Мура), работающего в соответствии с этим алгоритмом.

Проверку корректности работы спроектированной функциональной схемы автомата необходимо провести в пакете *Quartus*.

3. Порядок выполнения работы

Выполнение работы состоит из двух частей: синтеза автомата и проверки корректности спроектированной функциональной схемы путем моделирования в пакете *Quartus*.

Для выполнения синтеза автомата необходимо:

1)выполнить разметку заданной ГСА (для четных вариантов – для модели Мили, для нечетных – для модели Мура) (см. раздел 1.3);

2)построить граф переходов микропрограммного автомата;

3)заполнить структурную таблицу (прямую или обратную по заданию);

4)выписать из полученной структурной таблицы логические выражения для функций возбуждения триггеров;

5)по логическим выражениям, полученным в п. 4, построить схему КС1;

6)выписать из структурной таблицы логические выражения для выходных сигналов МПА;

7)по логическим выражениям, полученным в п. 6, построить схему КС2;

8)добавить к схемам КС1 и КС2 необходимое количество триггеров;

9) для проверки корректности полученной схемы по исходной ГСА составить автоматную ленту, учитывающую все возможные переходы автомата.

Для проверки корректности полученной функциональной схемы МПА в пакете *Quartus* необходимо выполнить следующие действия:

1) создать проект и построить схему на заданных логических элементах и триггерах; при наборе схемы рекомендуется использовать дешифраторы *dec*38 или 16*dmux* из каталога *others* (следует обратить внимание, что эти дешифраторы имеют инверсные выходы);

2) провести компиляцию проекта;

3) создать файл временных диаграмм, задать последовательность входных сигналов в соответствии с автоматной лентой;

4) провести функциональное моделирование и проверить правильность работы МПА.

Подробные методические указания по работе в пакете *Quartus* приведены в Приложении.

4. Содержание отчета

Отчет по лабораторной работе должен содержать следующие разделы.

1. Цель работы.
2. Задание по работе.

Раздел должен включать в себя формулировку задания и данные варианта.

3. Абстрактный синтез микропрограммного автомата.

В данном разделе необходимо привести ГСА с отметкой состояний автомата и граф переходов микропрограммного автомата.

4. Структурный синтез микропрограммного автомата.

В данном разделе необходимо привести структурную таблицу, логические выражения для функций возбуждения триггеров и выходных сигналов МПА.

5. Моделирование работы структурного автомата.

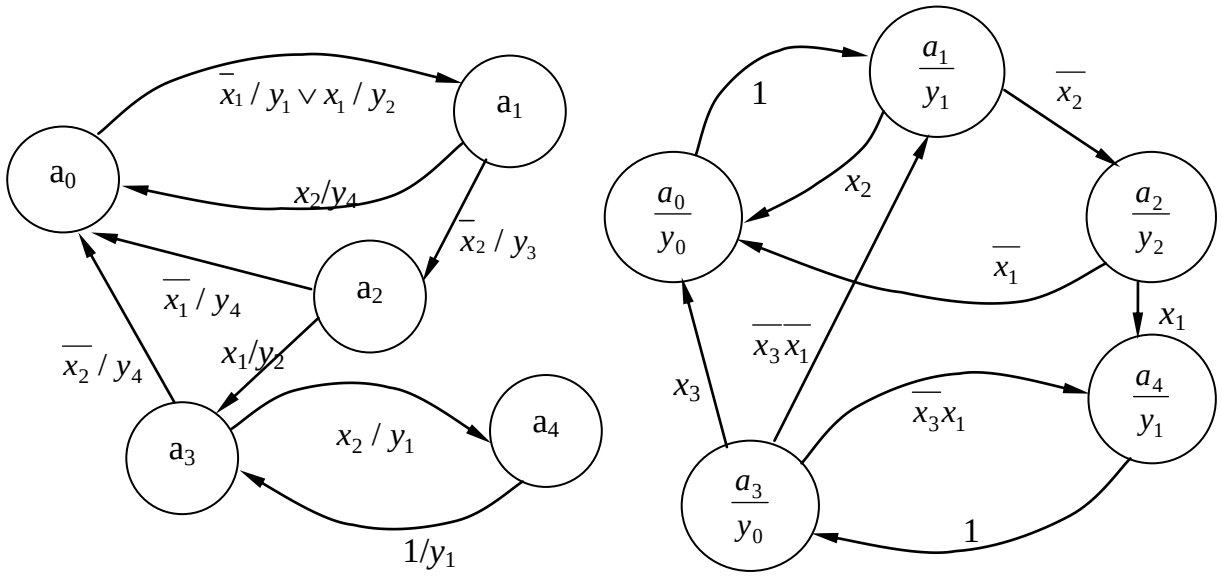
В данном разделе необходимо привести схему структурного автомата, построенную в QUARTUS, автоматные ленты для проверки корректности работы МПА и скриншот временных диаграмм с результатами моделирования автоматной ленты.

6. Вывод.

Пример вывода: «В результате выполнения работы произведен структурный синтез микропрограммного автомата модели Мура. Автомат имеет 3 структурных входа и 6 структурных выходов. Для реализации автомата потребовалось 3 триггера типа D, 4 элемента И, 6 элементов ИЛИ, 8 элементов НЕ и дешифратор 3×8. Автомат имеет 6 состояний. Моделирование работы микропрограммного автомата произведено в пакете QUARTUS. Изучены основы проектирования микропрограммных автоматов, приобретены навыки построения структурных схем МПА».

5. Контрольные вопросы

1. Опишите основные положения принципа микропрограммного управления.
2. Объясните смысл понятий микрооперация и микропрограмма.
3. Назовите примеры микроопераций, осведомительных сигналов.
4. Опишите назначение и функции узлов, входящих в состав операционного устройства.
5. Дайте определение пути перехода по ГСА, напишите множество путей перехода по заданной ГСА.
6. В чем отличие прямой структурной таблицы от обратной?
7. Объясните названия триггеров *D* (*Delay*), *T* (*Toggle*), *JK* (*Jump-Kill*).
8. Укажите, откуда на входы МПА поступают входные сигналы, а также назначение выходных сигналов МПА и куда они поступают.
9. По заданной ГСА выполните отметку вершин ГСА состояниями для автомата **а)** модели Мили, **б)** модели Мура; постройте граф переходов МПА.
10. Задан граф переходов (рис. 10).



а) б)

11.Рис. 10. Графы автоматов

– По заданному графу переходов постройте:

в) прямую структурную таблицу;

г) обратную структурную таблицу.

– Предусмотрите использование триггеров:

д) типа *JK*;

е) типа *T*;

ж) типа *D*.

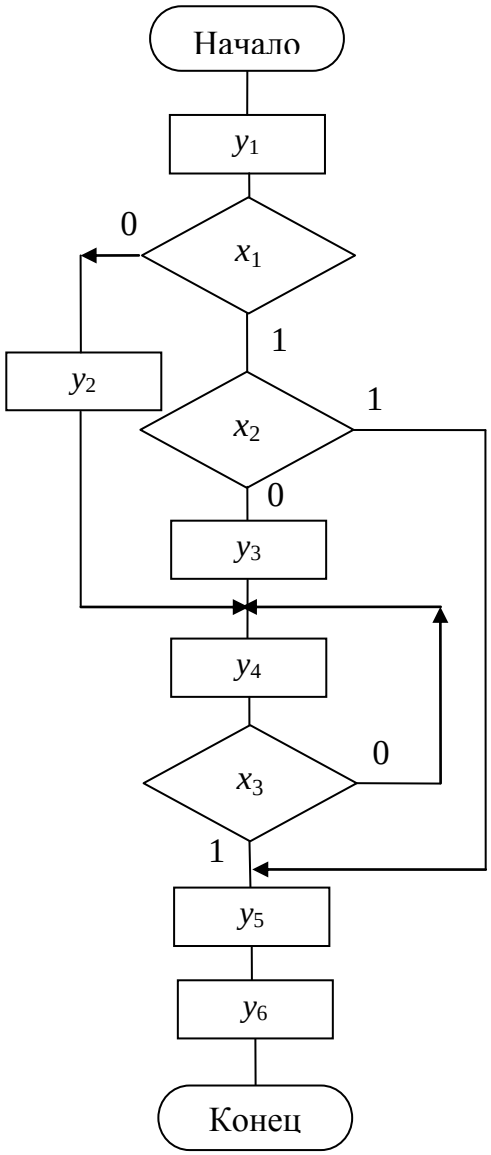
– Напишите необходимые логические выражения для построения:

з) КС1;

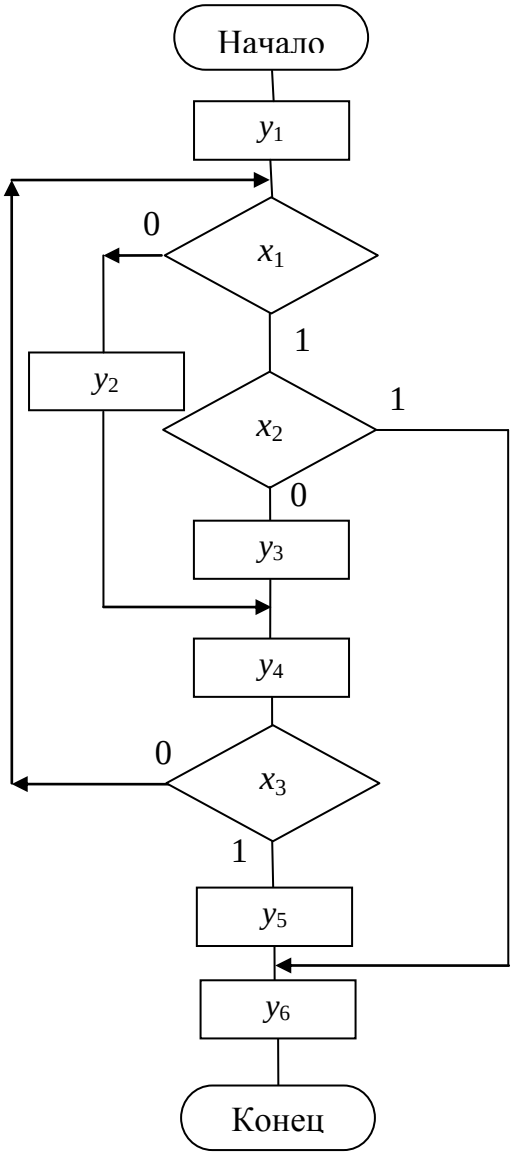
и) КС2.

6. Варианты заданий

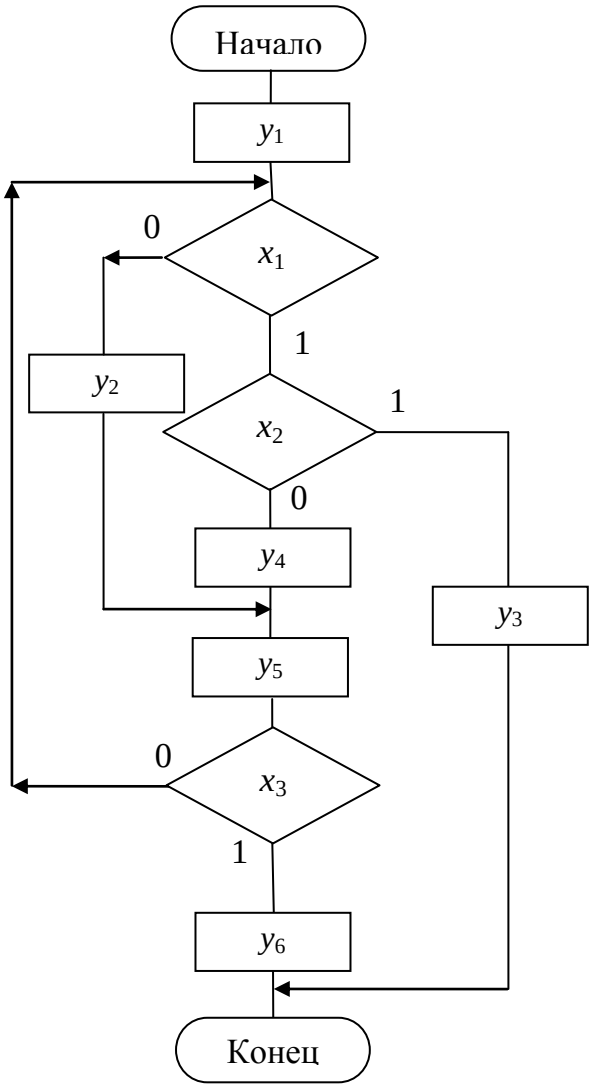
Номер вар-та	Алгоритм	Триггер	Таблица	Номер вар-та	Алгоритм	Триггер	Таблица
1	1	*D*	прямая	16	4	*JK*	обратная
2	2	*T*	обратная	17	5	*D*	прямая
3	3	*JK*	прямая	18	6	*T*	обратная
4	4	*D*	обратная	19	1	*D*	обратная
5	5	*T*	прямая	20	2	*T*	прямая
6	6	*JK*	обратная	21	3	*JK*	обратная
7	1	*T*	обратная	22	4	*D*	прямая
8	2	*JK*	прямая	23	5	*T*	обратная
9	3	*D*	обратная	24	6	*JK*	прямая
10	4	*T*	прямая	25	1	*T*	прямая
11	5	*JK*	обратная	26	2	*JK*	обратная
12	6	*D*	прямая	27	3	*D*	прямая
13	1	*JK*	прямая	28	4	*T*	обратная
14	2	*D*	обратная	29	5	*JK*	прямая
15	3	*T*	прямая	30	6	*D*	обратная



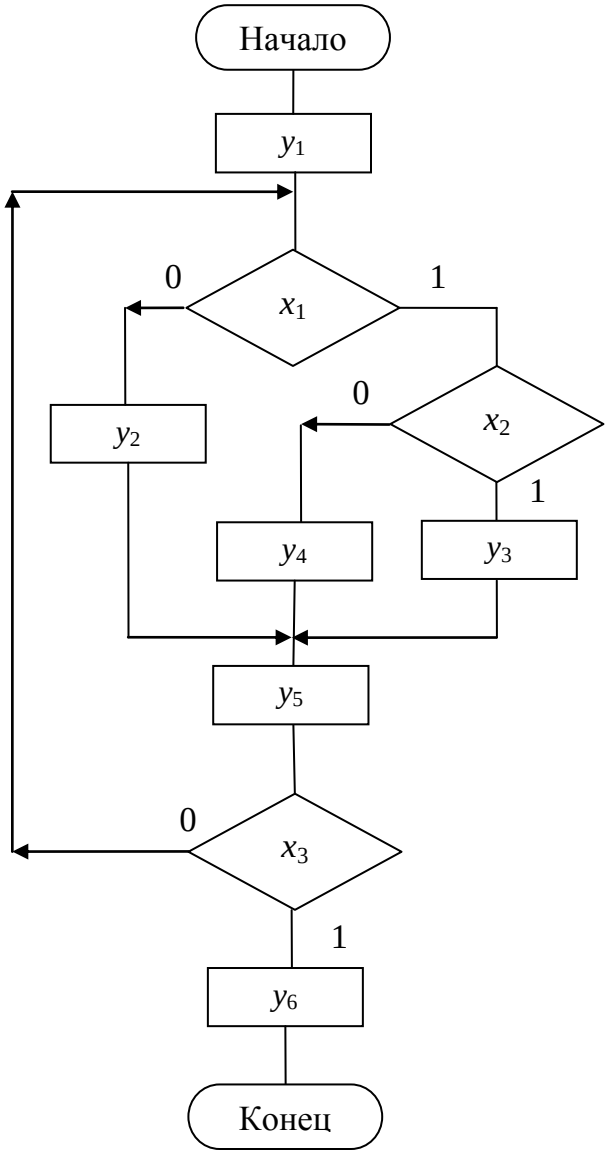
Алгоритм 1



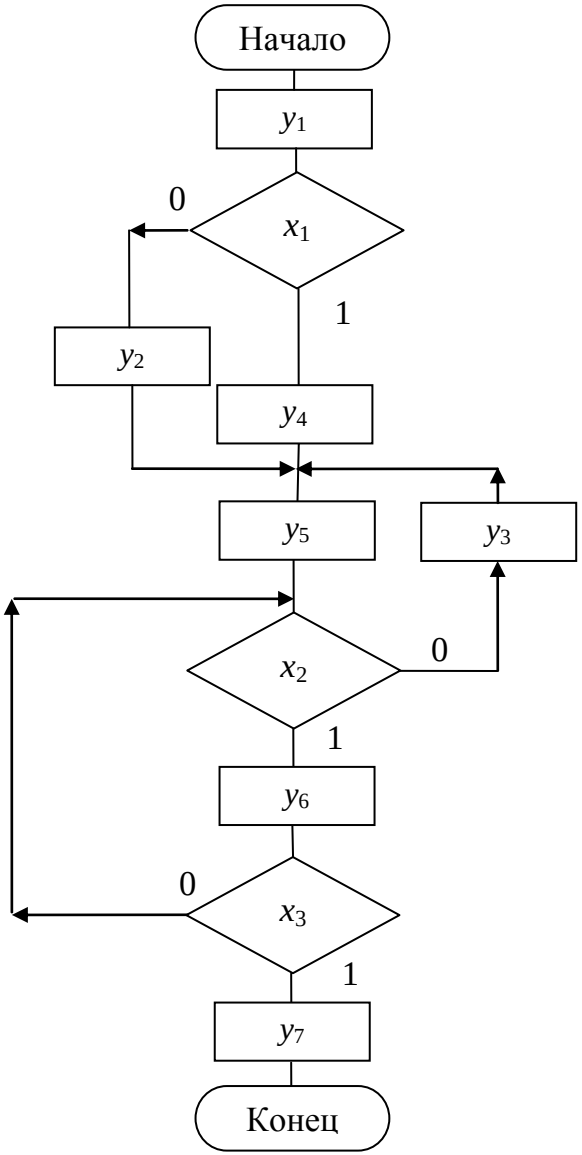
Алгоритм 2



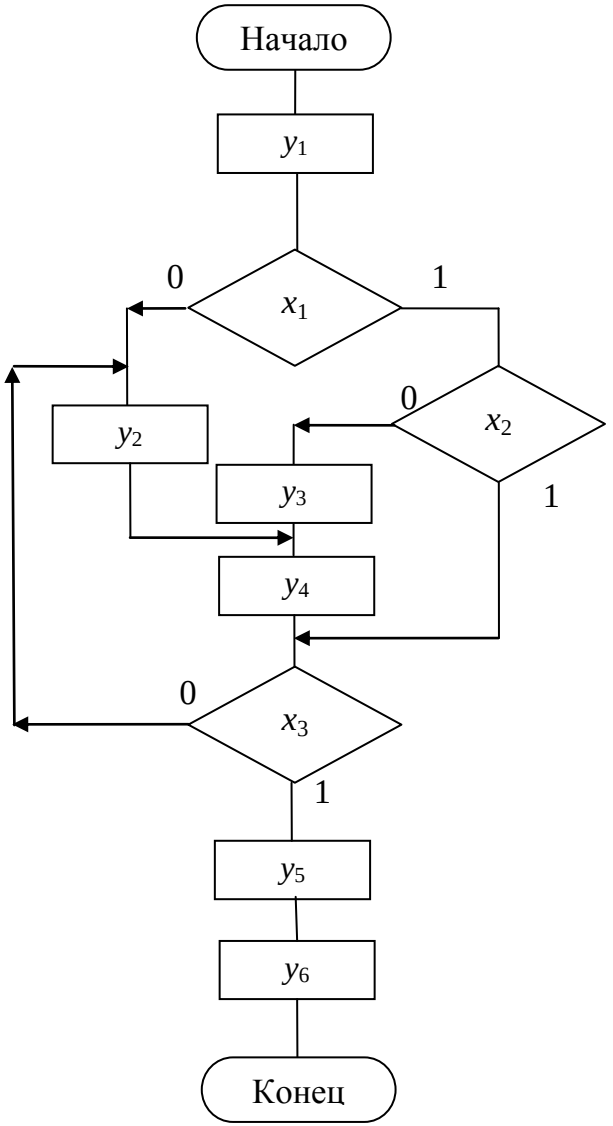
Алгоритм3



Алгоритм 4



Алгоритм 5



Алгоритм 6

ПРИЛОЖЕНИЕ А

ОСНОВЫ МОДЕЛИРОВАНИЯ В ПАКЕТЕ JFLAP

Одним из наиболее удобных средств моделирования работы абстрактных автоматов, машин Тьюринга и других объектов, встречающихся в теории формальных языков, является пакет *JFLAP*.

JFLAP (*Java Formal Languages and Automata Package*) разрабатывается с 1993 года Университетом Дьюка в образовательных и исследовательских целях при поддержке Национального научного фонда. Это бесплатное программное обеспечение, являющееся *Java*-приложением. Приложение и исходный код последней версии доступны на официальном сайте: http://www.jflap.org/.

К объектам регулярных языков, реализуемых в *JFLAP*, относятся конечные автоматы (*Finite Automation*), в том числе автоматы Мили (*Mealy Machine*) и автоматы Мура (*Moore Machine*); регулярные грамматики и регулярные выражения (*Regular Expression*). К объектам контекстно-свободных языков – магазинные автоматы (*Pushdown Automation*) и контекстно-свободные грамматики. К объектам рекурсивно перечислимых языков – машины Тьюринга (*Turing Machine*) (в том числе многоленточные) и неограниченные грамматики.

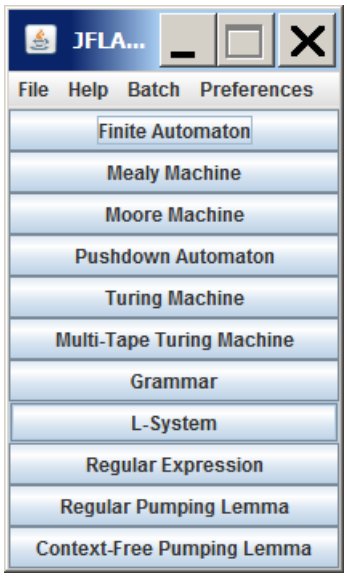


Рис. 1. Окно выбора задачи

Кроме того, *JFLAP* позволяет моделировать так называемые *L*-системы (*L-System*) и демонстрирует лемму о разрастании для регулярных (*Regular Pumping Lemma*) и контекстно-свободных (*Context-Free Pumping Lemma*) грамматик.

При запуске программы пользователю предлагается выбрать один из перечисленных объектов (рис. 1).

Рассмотрим методику моделирования с помощью пакета *JFLAP* машины Тьюринга, а также автоматов Мили и Мура.

1. Создание машины Тьюринга

Машина Тьюринга в *JFLAP* задается в виде графа переходов.

Для создания МТ необходимо после запуска программы в появившемся окне (рис. 1) выбрать пункт *Turing Machine*. При этом откроется окно редактора графа переходов (рис. 2).

Начать построение графа целесообразно с создания состояний. Для этого нужно выбрать на панели инструментов вторую кнопку (*State Creator*) и щелчками мышью по рабочему полю разместить состояния МТ.

Для удаления элементов графа (состояний или переходов) необходимо выбрать на панели инструментов кнопку с изображением пиратского флага (*Deleter*), после чего курсор мыши примет форму крестика и щелчок мышью по элементу приведет к его удалению.

Для задания переходов МТ нужно выбрать на панели инструментов третью кнопку (*Transition Creator*). Для создания перехода необходимо нажать мышью на исходное состояние, провести линию к состоянию перехода и отпустить кнопку мыши. В появившихся полях следует задать: обозреваемый символ (слева), записываемый символ (в центре) и направление сдвига головки (справа) (рис. 2).

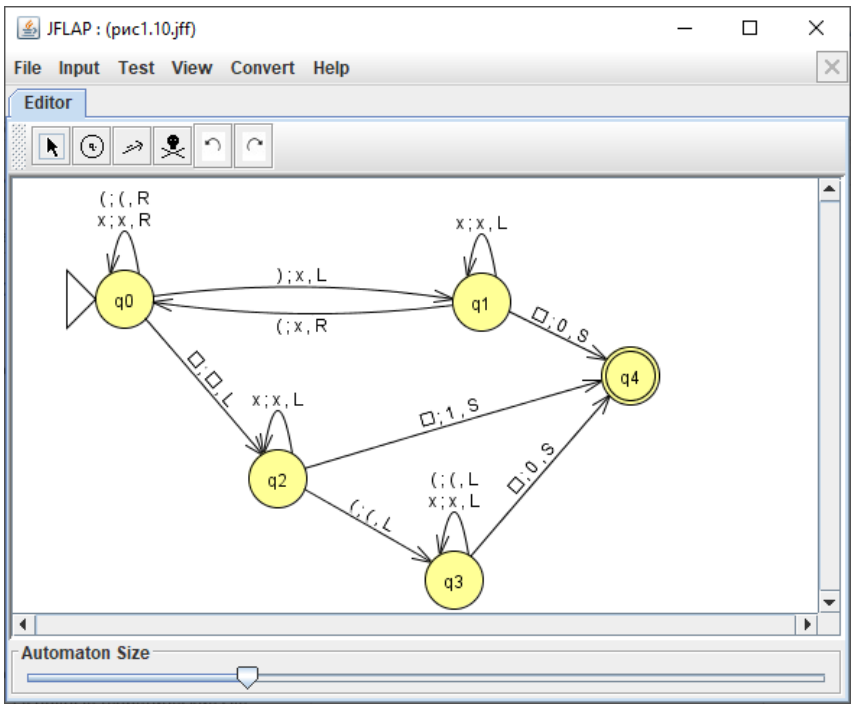


Рис. 2. Создание графа переходов Машины Тьюринга

Для редактирования элементов графа необходимо выбрать на панели инструментов первую кнопку (*Attribute Editor*). Для изменения имени состояния нужно, нажав правой кнопкой мыши на состоянии, выбрать в контекстном меню пункт *Set Name*. Для изменения перехода нужно дважды щелкнуть по этому переходу.

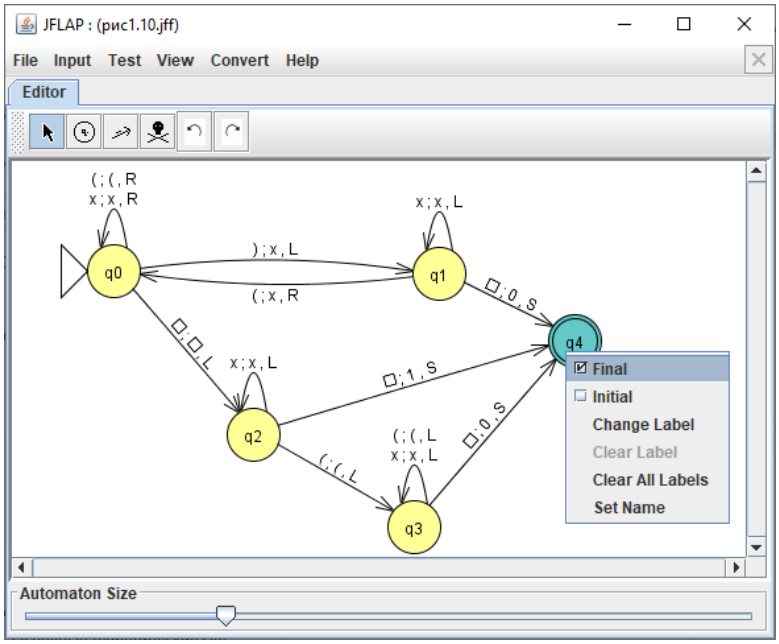


Рис. 3. Выбор конечного состояния МТ

По окончании построения графа необходимо указать начальное и конечное состояние. Для этого нужно нажать первую кнопку на панели инструментов (*Attribute Editor*), а затем щелкнуть правой кнопкой мыши на начальном (конечном) состоянии и выбрать пункт *Initial (Final)* (рис. 3).

На любом этапе проектирования граф МТ можно сохранить с помощью меню *File: Save*.

2. Моделирование работы машины Тьюринга

Для моделирования работы МТ служит группа команд меню *Input*.

Если при выборе команды из меню Input программа выдаст ошибку

,

это означает, что в спроектированном графе присутствуют неопределенности. Поможет найти ошибку команда меню Test: Highlight Nondeterminism, которая выделит состояния, в которых обнаружена неопределенность.

Если граф МТ не содержит неопределенностей, после выбора команды *Input: Step* откроется диалоговое окно, в котором необходимо ввести входное слово (рис. 4). Слово можно задать как непосредственным вводом, так и из файла.

После окончания ввода в окне (рис. 4) нужно нажать *OK*. При этом МТ перейдет в режим симуляции (рис. 5). В нижней части окна появится изображение ленты. В начале симуляции головка устанавливается над крайним левым символом входного слова.

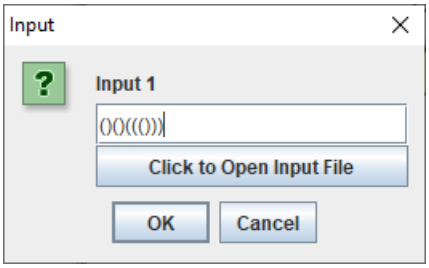


Рис. 4. Ввод входного слова МТ

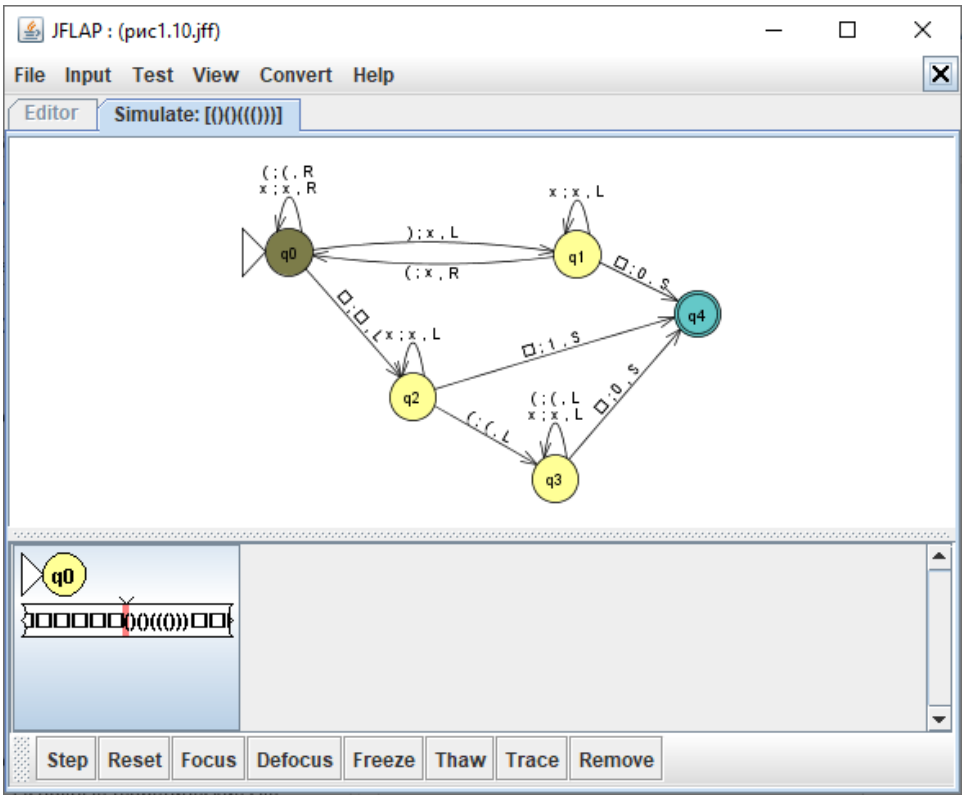


Рис. 5. Режим моделирования МТ

Нажимая кнопку *Step*, можно наблюдать за изменением состояний МТ и содержимого ленты.

Запустить моделирование сначала можно с помощью кнопки *Reset* в нижней части окна.

Для проверки работы МТ также можно использовать меню *Input: Multiple Run (Transducer)*. При использовании этого режима в столбце *Input* задается входное слово (можно задать несколько слов). Затем следует нажать кнопку *Run*

Inputs. Содержимое ленты, полученное после окончания обработки каждого входного слова, можно посмотреть, нажав кнопку *View Trace*.

3. Создание графа автомата Мили

Автоматы Мили и Мура в *JFLAP* задаются в виде графа.

Для создания автомата Мили необходимо после запуска программы в появившемся окне (рис. 1) выбрать пункт *Mealy Machine*. При этом откроется окно редактора графа автомата (рис. 6).

Начать построение графа целесообразно с создания состояний. Для этого нужно выбрать на панели инструментов вторую кнопку (*State Creator*).

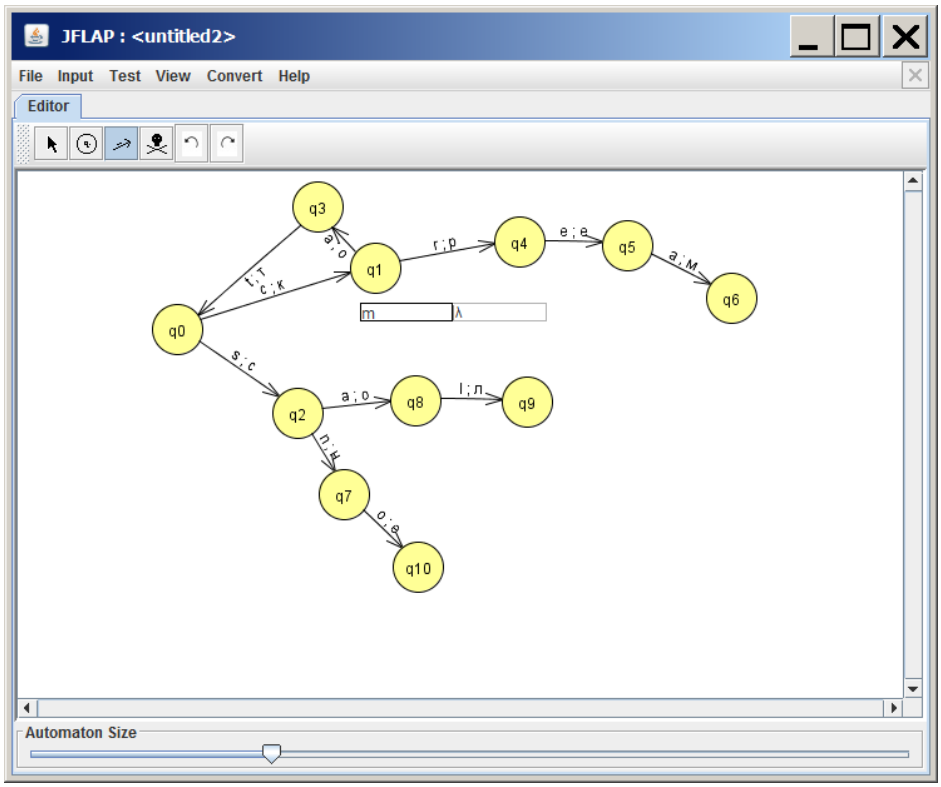


Рис. 6. Создание графа переходов автомата Мили

Для задания переходов автомата нужно выбрать на панели инструментов третью кнопку (*Transition Creator*). Для создания перехода необходимо нажать мышью на исходное состояние, провести линию к состоянию перехода и отпустить кнопку мыши. В появившихся полях следует задать входной символ (слева), выходной символ (справа) (рис. 6) и нажать *Enter*.

По окончании построения графа необходимо указать начальное состояние. Для этого нужно нажать первую кнопку на панели инструментов (*Attribute Editor*), а затем щелкнуть правой кнопкой мыши на начальном состоянии и выбрать пункт *Initial* (рис. 7), рядом с состоянием появится стрелка.

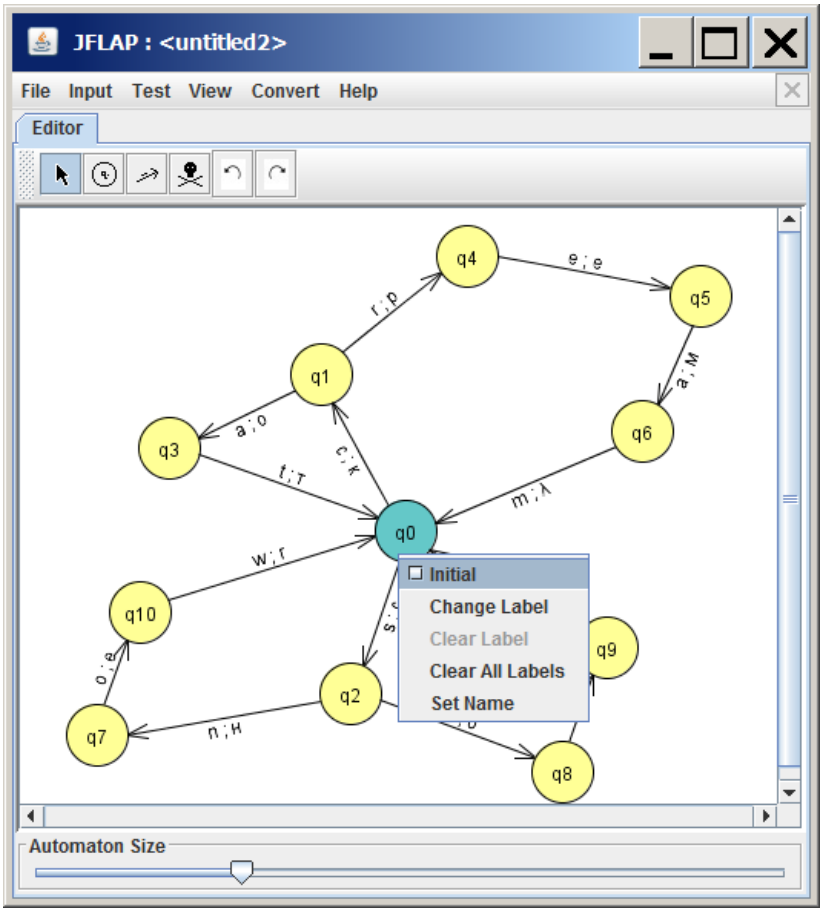


Рис. 7. Выбор начального состояния автомата

На любом этапе проектирования граф автомата можно сохранить с помощью меню *File: Save*.

4. Создание графа автомата Мура

Так же как и граф автомата Мили в *JFLAP* задается граф автомата Мура.

Для создания автомата Мура необходимо после запуска программы в появившемся окне (рис. 1) выбрать пункт *Moore Machine*.

Для создания графа сначала, выбрав на панели инструментов вторую кнопку (*State Creator*), нужно щелчками мышью разместить состояния автомата по

рабочему полю. При создании каждого состояния появляется диалоговое окно, в котором следует ввести выходной символ для этого состояния (рис. 8).

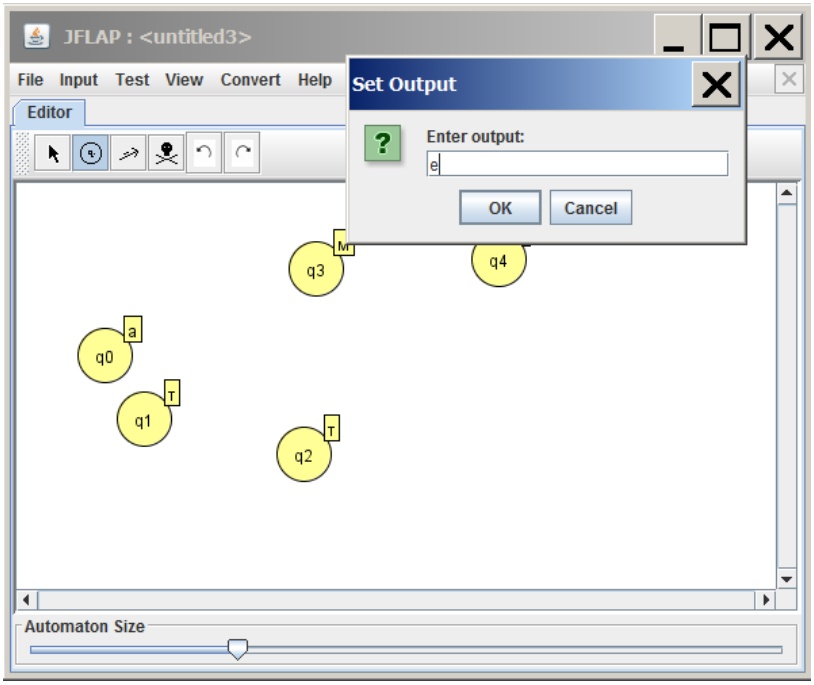


Рис. 8. Создание состояний автомата Мура

Для задания переходов и соответствующих им входных символов нужно выбрать на панели инструментов третью кнопку (*Transition Creator*). Для создания перехода необходимо нажать мышью на исходное состояние и провести линию к состоянию перехода. Затем в появившемся поле следует задать входной символ для этого перехода и нажать *Enter* (рис. 9).

После задания всех переходов автомата Мура, так же как и для автомата Мили, необходимо указать начальное состояние (рис. 7).

Во избежание неопределенности при моделировании в JFLAP для автомата Мура не предусмотрено наличия нескольких начальных состояний, поэтому в качестве начального следует указать любое из них.

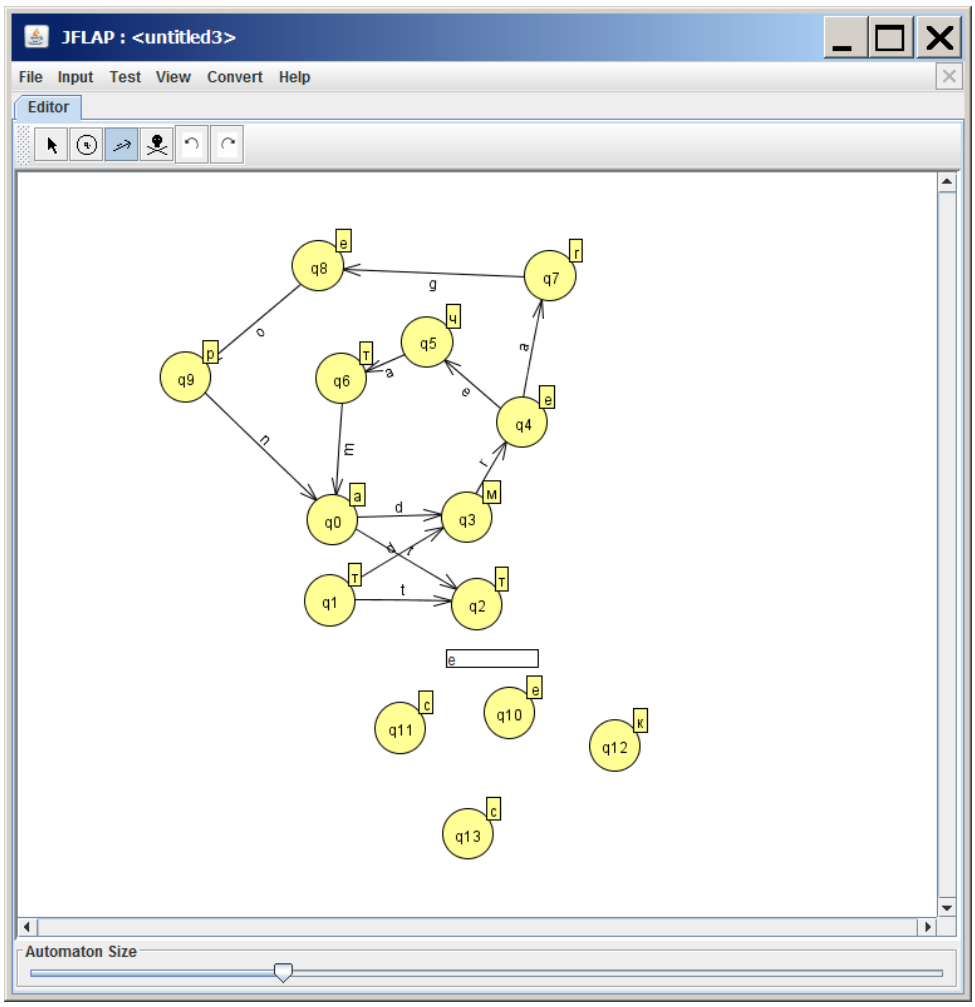


Рис. 9. Задание переходов автомата Мура

На любом этапе проектирования граф автомата можно сохранить с помощью меню *File: Save*.

5. Моделирование работы конечных автоматов

Для моделирования работы автоматов Мили или Мура предназначена группа команд меню *Input*.

Если при выборе команд меню Input программа выдаст ошибку

,

это означает, что в спроектированном автомате присутствуют неопределенности. Неопределенности могут возникать в случае, если для перехода не задан входной символ или из одного состояния имеется несколько переходов по одинаковым входным символам. Поможет найти ошибку команда меню Test: Highlight Nondeterminism, которая выделит состояние, в котором обнаружена неопределенность.

Рассмотрим возможности меню *Input*.

Команда *Input: Step* позволяет задать одно входное слово и по шагам проследить за его обработкой автоматом (рис. 10, слева).

Каждый раз при нажатии кнопки *Step* в нижней части окна автомат переходит в следующее состояние и добавляет соответствующий символ к выходному слову.

Для того чтобы сразу увидеть всю последовательность шагов при обработке автоматом заданного входного слова, используйте команду меню *Input: Fast Run* (рис. 10, справа).

Для проверки работы автомата сразу по нескольким входным словам (например, по оператору соответствия) следует использовать команду меню *Input: Multiple Run* (рис. 11).

В левом столбце появившейся таблицы следует ввести входные слова, а затем нажать кнопку *Run Inputs* в нижней части окна. В правом столбце появятся выходные слова, полученные автоматом.

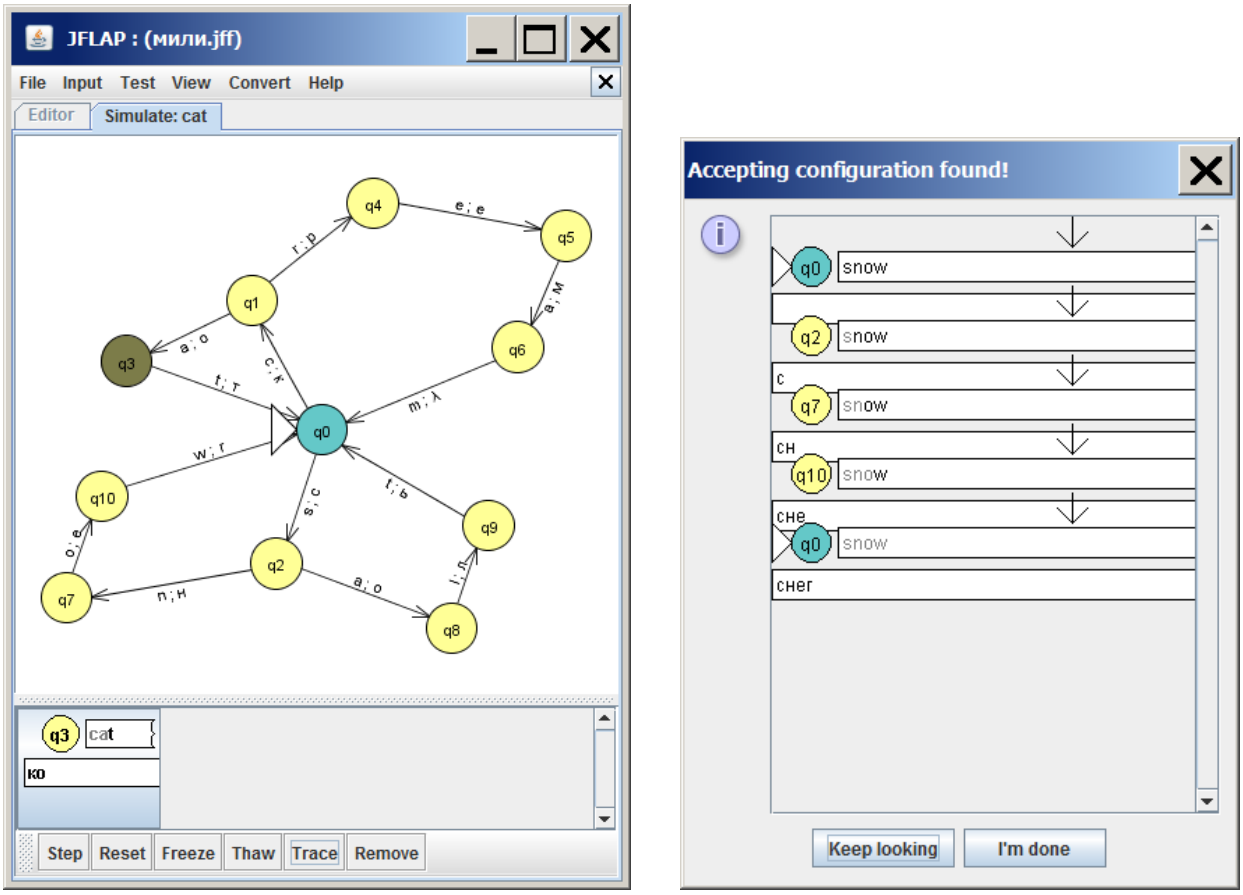


Рис. 10. Пошаговое моделирование работы автомата (слева) и быстрая обработка входного слова (справа)

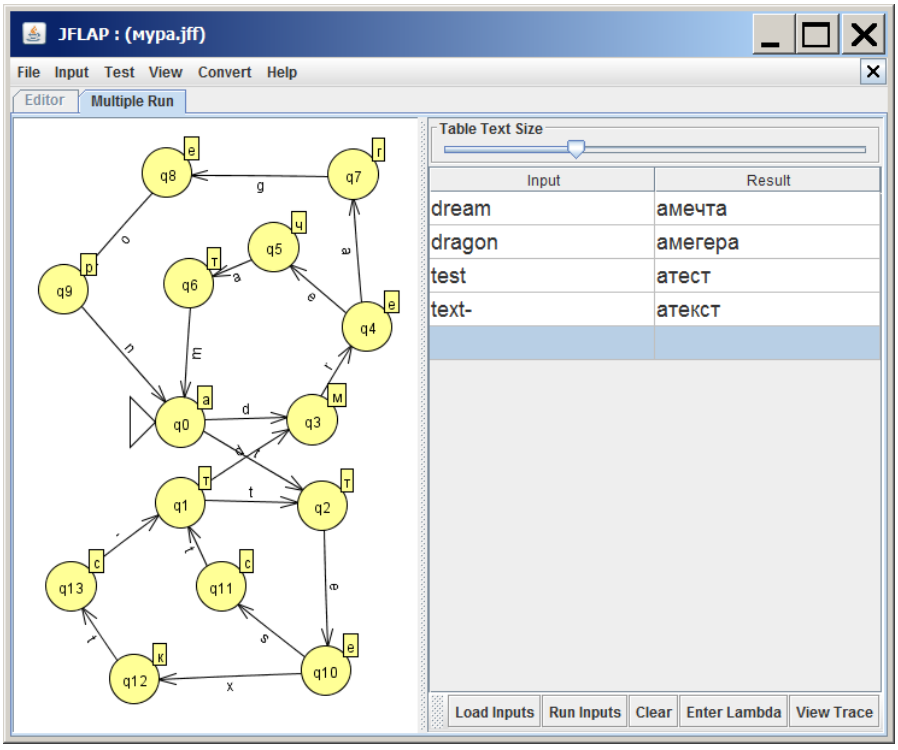


Рис. 11. Проверка автомата по оператору соответствия

Для того чтобы проследить за пошаговой обработкой одного из слов нужно выделить его в таблице и нажать кнопку *View Trace*.

При работе с автоматом Мура JFLAP всегда в качестве первого выходного символа выдает выходной символ начального состояния. При проверке автомата не следует принимать его во внимание.

ПРИЛОЖЕНИЕ Б

ОСНОВЫ ПРОЕКТИРОВАНИЯ ПРИНЦИПИАЛЬНЫХ СХЕМ В QUARTUS II

Пакет *Quartus II* разрабатывается фирмой *Altera* и предназначен для анализа и синтеза интегральных схем. *Quartus II Web Edition* представляет собой бесплатную версию *Quartus II*, она доступна для загрузки с официального сайта *Altera* (*http://www.altera.com*) после регистрации.

Рассмотрим кратко методику построения и симуляции (функционального моделирования) принципиальных схем цифровых устройств в пакете *Quartus* [4].

Внимание! встроенная функция симуляции отсутствует в пакетах Quartus версии 10.x, 11.x и 12.x. При установке Quartus 13 и выше отдельно требуется загрузить библиотеки элементов Altera и установить ModelSim.

---

1. Создание проекта

1. Запустить приложение *quartus.exe* (*Quartus II*).
2. Выбрать пункт меню *File*: *New Project Wizard*. При этом откроется окно мастера создания проекта.
3. В появившемся окне следует нажать *Next*. В результате открывается страница 1 мастера (рис. 1).

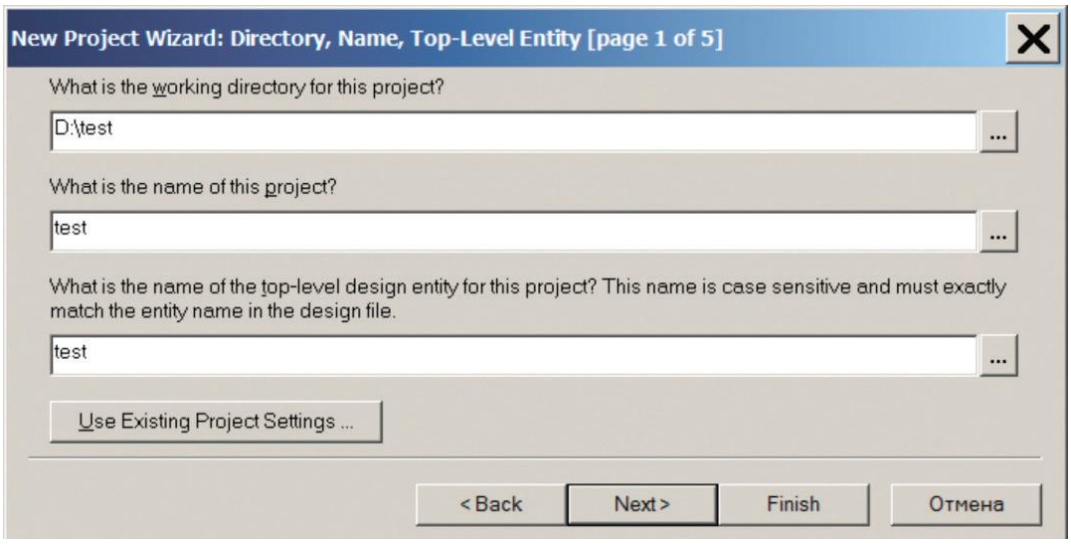


Рис. 1. Задание расположения и имени проекта

В верхнем поле открывшейся страницы следует ввести путь к папке проекта, ниже вводится его имя. Каждый проект должен иметь отдельную папку. Последнее поле служит для задания имени устройства, занимающего верхнюю позицию в иерархии проекта (если это многоуровневый проект), по умолчанию оно совпадает с именем проекта.

Замечание: *Quartus* не поддерживает кириллицу в имени проекта.

4. На этом шаге работу мастера можно завершить нажатием кнопки *Finish*. Проект создан. Теперь нужно добавить в него файлы, описывающие логику работы проектируемого устройства.

2. Создание принципиальной схемы устройства

1. Для создания файла, который будет содержать принципиальную схему устройства нужно выбрать команду меню *File*: *New*. В появившемся диалоговом окне в разделе *Design Files* следует выбрать тип файла *Block Diagram / Schematic File* и нажать *OK* (рис. 2).

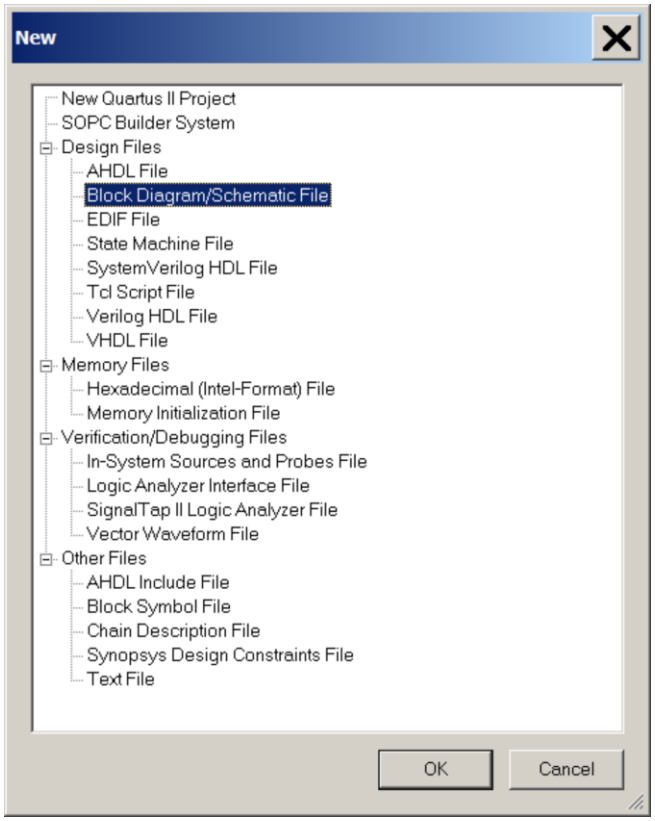


Рис. 2. Выбор типа файла – схема устройства

Эти действия приведут к открытию окна графического редактора с загруженным в него файлом с расширением *.bdf*.

2. Необходимо включить файл в проект: *File***:** *Save As***…** В появившемся диалоговом окне сохранения файла следует задать его имя и установить флажок *Add file to current project*. **Внимание! Имя файла схемы должно совпадать с именем устройства верхнего уровня, заданным при создании проекта (третье поле на рис. 1)**.

3. Ввести компоненты.

Создание блок-схемы проекта выполняется с использованием панели инструментов главного окна программы. Назначение различных инструментов панели приведено на рис. 3.

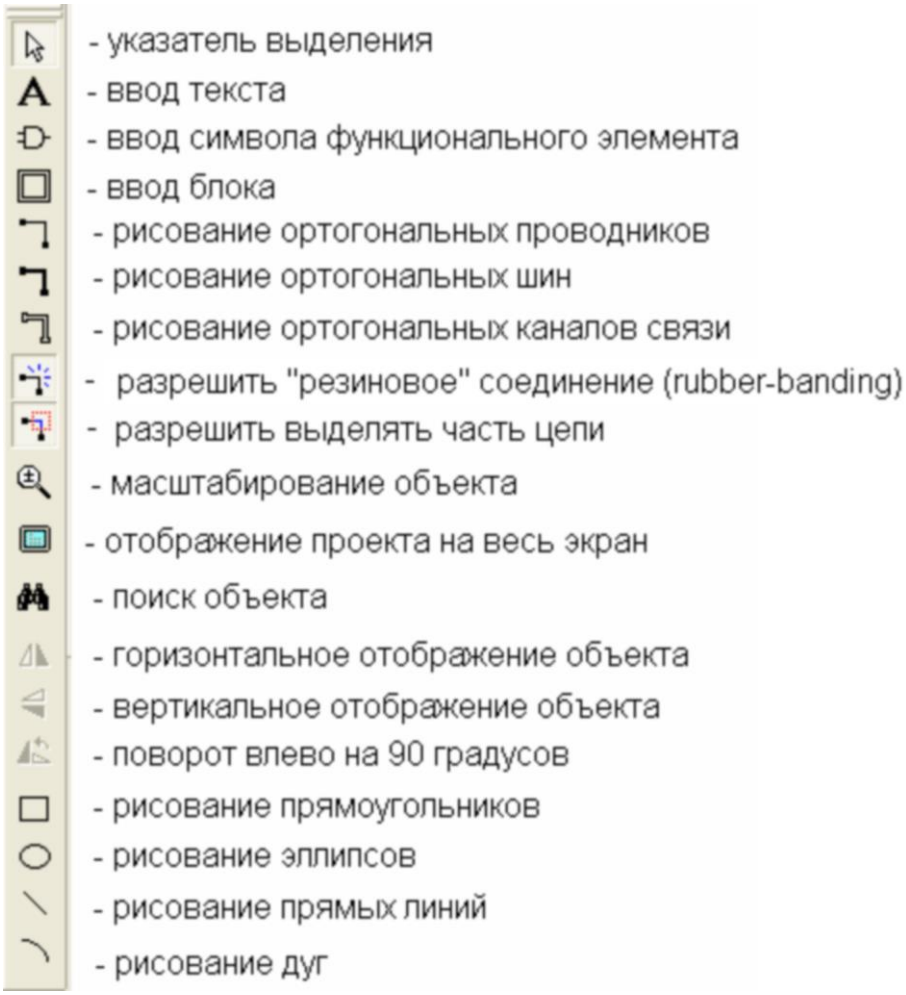


Рис. 3. Назначение кнопок панели инструментов редактора схем

Замечание: *Rubber-banding* – от англ. эластичное соединение, соединение резиновой нитью – при выбранной функции соединение (цепь, проводник или

шина) останется неразрывно связанным с элементом при его перемещении в другое место.

Выбор компонентов осуществляется двойным щелчком на созданном документе, или нажатием кнопки *Symbol Tool* на панели инструментов, или нажатием правой кнопки мыши на свободном участке рабочего поля и выбором в контекстном меню пункта *Insert***:** *Symbol***…**

Далее в диалоговом окне (рис. 4) нужно выбрать требуемый каталог и нужный компонент и нажать *OK*.

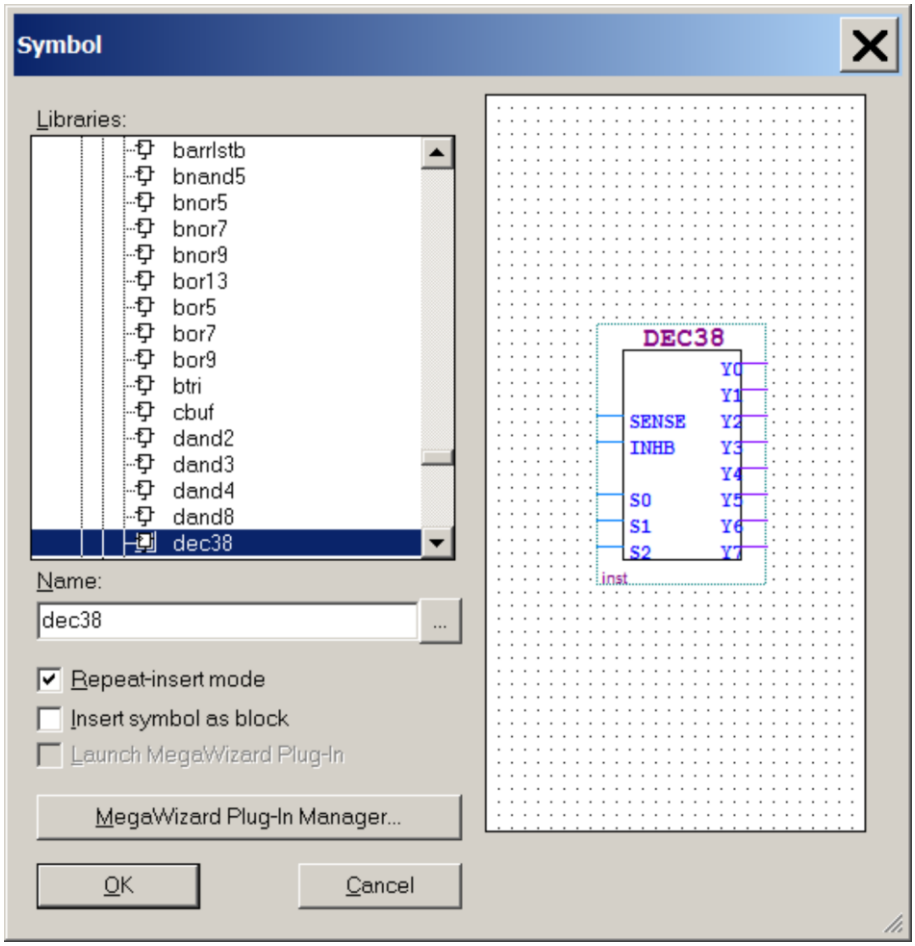


Рис. 4. Выбор компонента

Основные простейшие элементы для создания схем находятся в каталоге *primitives* стандартной библиотеки **...***/altera/***…***/libraries/*.

В разделе *primitives/logic* расположены основные логические элементы (НЕ (*not*), И (*and*), И-НЕ (*nand*), ИЛИ (*or*), ИЛИ-НЕ (*nor*)). В разделе *primitives/pin* – элементы для задания входных и выходных сигналов схемы. В разделе *primitives/storage* – триггеры.

Более сложные элементы (например, дешифратор 3 на 8 (*dec*38)[1]) находятся в каталоге *others*.

Для того чтобы не искать элементы в библиотеке вручную, можно указать имя нужного элемента в поле *Name* окна *Symbol*.

4. Соединить компоненты.

Соединение компонентов в проектируемой схеме можно осуществить двумя способами.

Первый способ состоит в непосредственном соединении точек схемы с помощью проводников (*node tool* на панели инструментов), он удобен для простых схем. Для этого нужно переместить курсор мыши в одну из тех двух точек схемы, которые нужно соединить между собой, нажать левую кнопку мыши и, не отпуская ее, перемещать курсор ко второй из соединяемых точек.

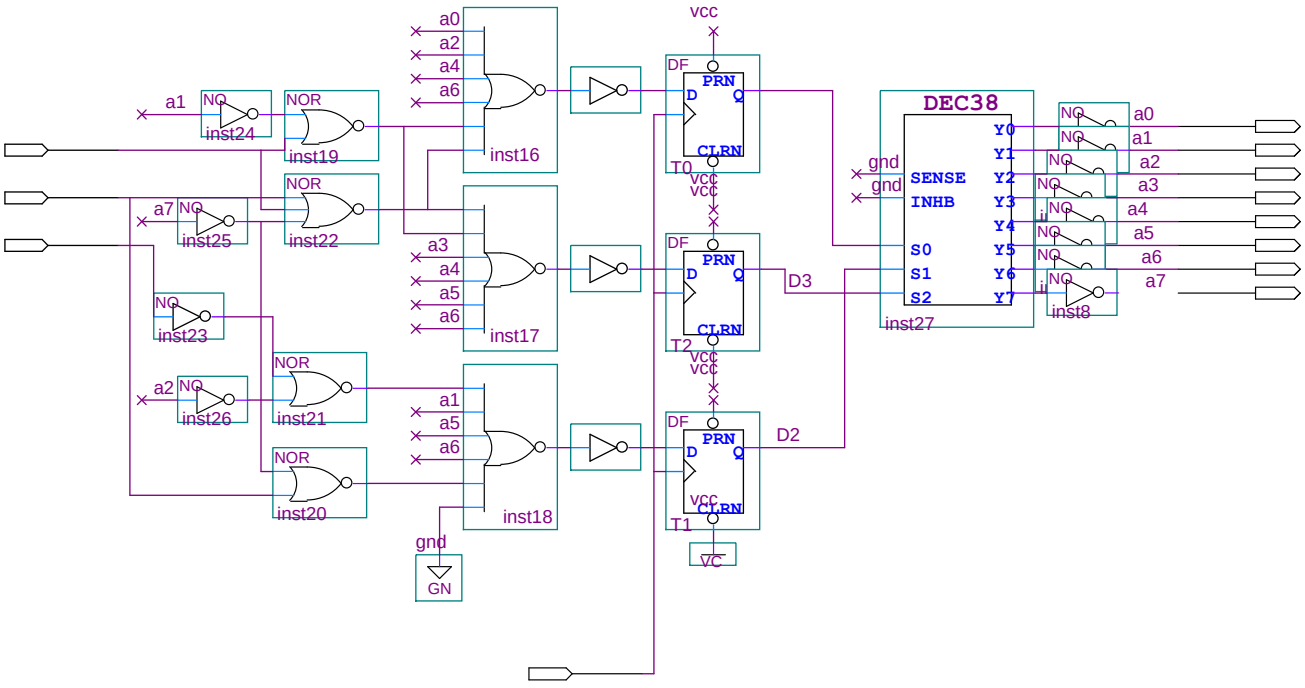


Рис. 5. Соединение элементов схемы

Второй способ основан на том, что несколько линий схемы (в том числе "оборванных"), которым присвоено одно и то же имя, считаются соединенными между собой. Например, присвоением одинакового имени a0 на рис. 5 первый

[1] Следует иметь в виду, что дешифратор *dec*38 имеет инверсные выходы.

вход элемента *inst*16 соединен с инвертированным выходом Y0 дешифратора *inst*27.

Для присвоения имени какой-либо линии нужно выделить линию щелчком мыши и ввести название (только латинскими буквами) или, выделив линию, нажать правую кнопку мыши и выбрать пункт *Properties*. В открывшемся окне свойств проводника на вкладке *General* можно изменить имя (рис. 6).

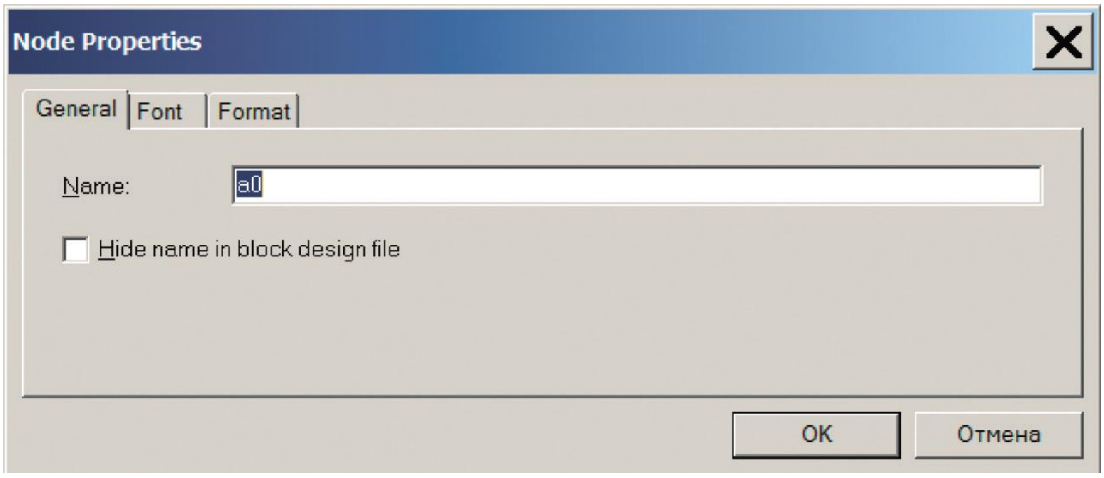


Рис. 6. Присвоение имени проводнику

Аналогичны способы соединения шинами.

5. Обеспечить постоянный уровень логической единицы в тех точках схемы, где он необходим.

Это обеспечивается соединением этих точек с примитивом *VCC* (библиотека *primitives*/*other*). Аналогично, использование примитива *GND* обеспечивает уровень логического нуля.

6. Отметить входы и выходы проектируемого устройства соответственно примитивами *INPUT* и *OUTPUT*.

7. Отредактировать названия соответствующих входов и выходов.

Для этого нужно дважды щелкнуть по элементу или, выделив его, нажать правую кнопку мыши и выбрать пункт *Properties*.

3. Компиляция проекта

После того как схема устройства будет введена средствами графического редактора, необходимо осуществить компиляцию проекта.

В простейшем случае для проверки корректности соединений и возможности дальнейшего функционального моделирования достаточно провести только первый этап компиляции – *Analysis & Synthesis*[2]. Запустить компиляцию можно сочетанием клавиш *Ctrl+K*, или с помощью кнопки *Start Analysis & Synthesis* на панели инструментов, или двойным щелчком на соответствующем пункте в окне *Tasks*, или с помощью меню *Processing*: *Start*: *Start Analysis & Synthesis* (рис. 7).

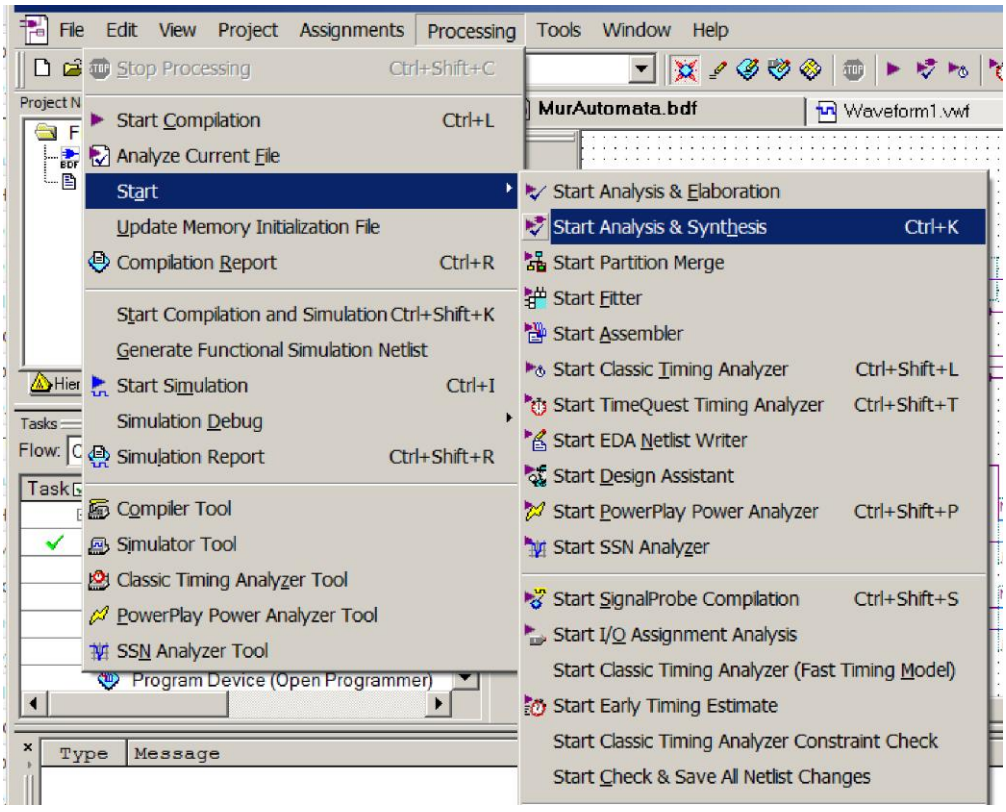


Рис. 7. Запуск компиляции

---

[2] Процесс компиляции в *Quartus* разбит на несколько этапов (как это можно увидеть в окне *Tasks*). Анализ проекта и синтез базы данных проекта является первым этапом. После его завершения возможно проведение функционального моделирования. Дальнейшие этапы относятся к реализации проекта на конкретном кристалле программируемой логики.

В случае обнаружения ошибки компилятор выдаст сообщение. Местоположение ошибки на схеме можно найти при помощи двойного щелчка на сообщении об ошибке в окне *Messages*. После двойного щелчка мышью на строке сообщения открывается файл, содержащий ошибку, и в нем выделяется то место, которое компилятор зафиксировал как ошибочное.

Если в проекте нет файла-схемы (или другого из раздела *Design Files*), имя которого, совпадает с именем устройства верхнего уровня, заданного при создании проекта (рис. 1), компилятор выдаст ошибку: *Top-Level design entity* "*file_name*" *is undefined*. В этом случае следует переименовать файл-схему, дав ей имя, указанное в сообщении компилятора.

4. Подготовка временных диаграмм

Следующим после компиляции этапом является верификация проекта, то есть моделирование (симуляция) и проверка правильности функционирования спроектированного устройства.

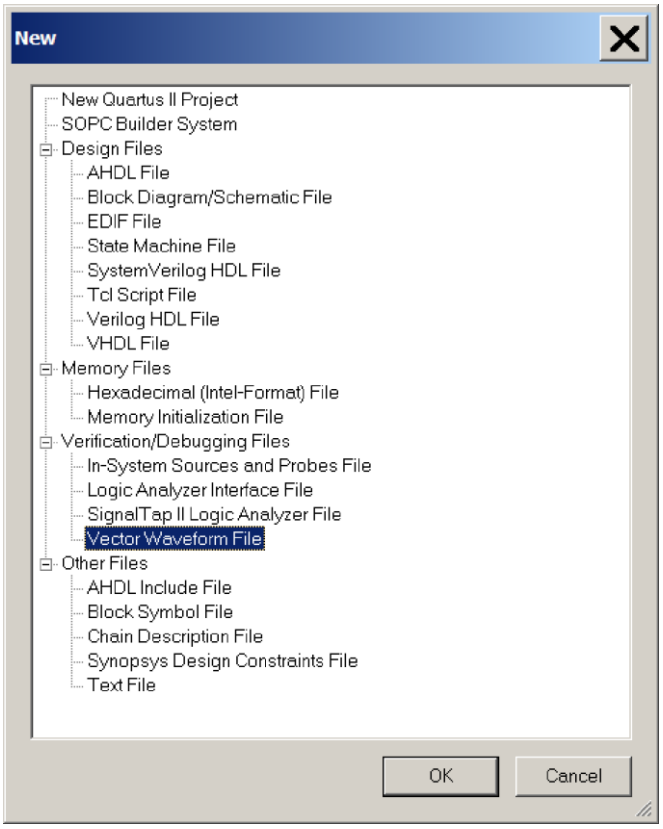


Рис. 8. Создание файла временных диаграмм

В *Quartus* можно проводить временное и функциональное моделирование. Функциональное моделирование позволяет проверить логику работы устройства без учета временных задержек.

1. Для моделирования работы схемы необходимо создать файл временных диаграмм (меню *File*: *New Vector Waveform File* из раздела *Verification/Debugging Files*) (рис. 8).

Замечание: В *Quartus* 13 вместо *Vector Waveform File – University Program VWF*.

При этом запустится сигнальный редактор с загруженным файлом с расширением **.vwf*. Созданный файл необходимо сохранить в проекте, имя файла может быть любым.

2. Для добавления сигналов на диаграмму нужно в открывшемся документе дважды щелкнуть в колонке поля *Name* или нажать правую кнопку мыши и выбрать *Insert*: *Insert Node or Bus*... (рис. 9).

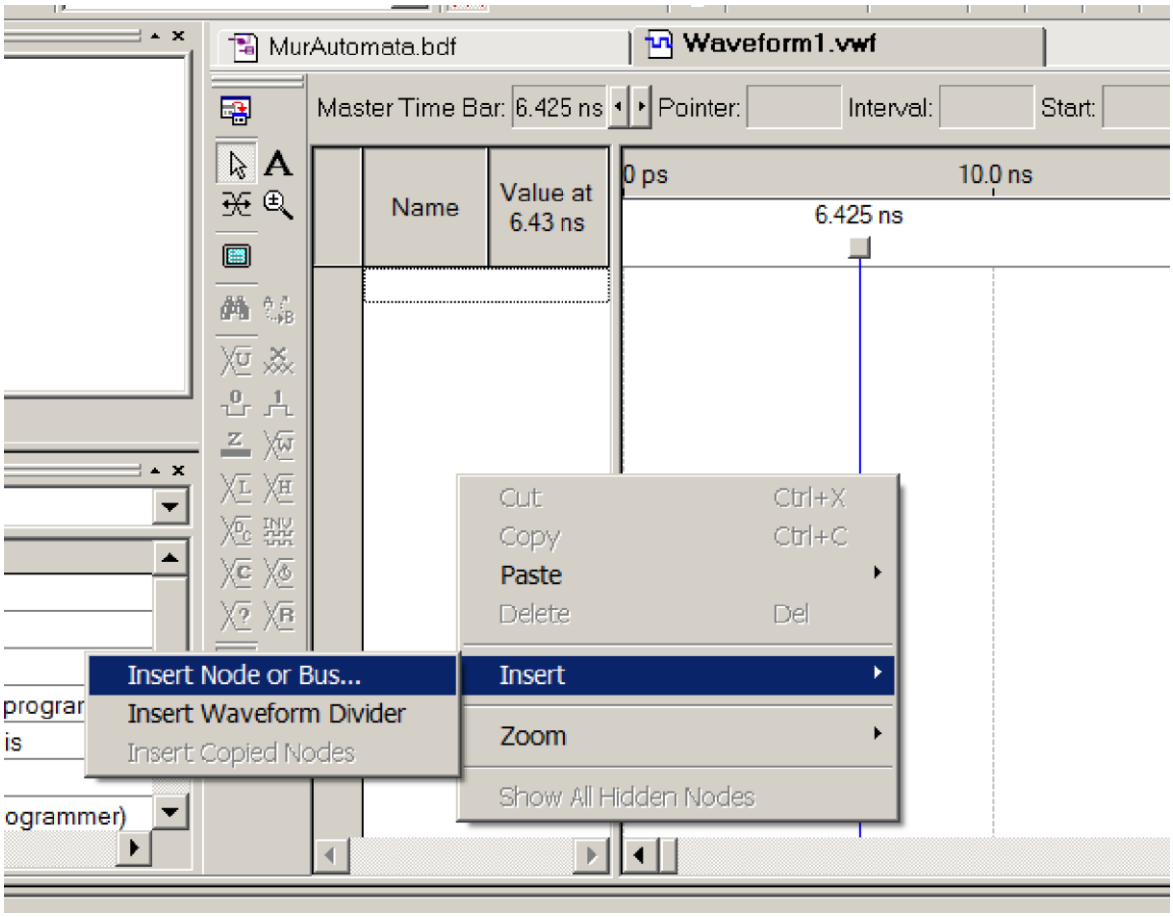


Рис. 9. Выбор данных

В появившемся окне (рис. 10) следует нажать кнопку поиска сигналов в проекте *Node Finder...*. После в поле *Filter* установить значение *Pins: all*, нажать кнопку *List*. Далее следует выбрать все нужные сигналы слева в таблице найденных сигналов *Nodes Found*, перенести их вправо в таблицу выбранных сигналов *Selected Nodes* и нажать *OK*.

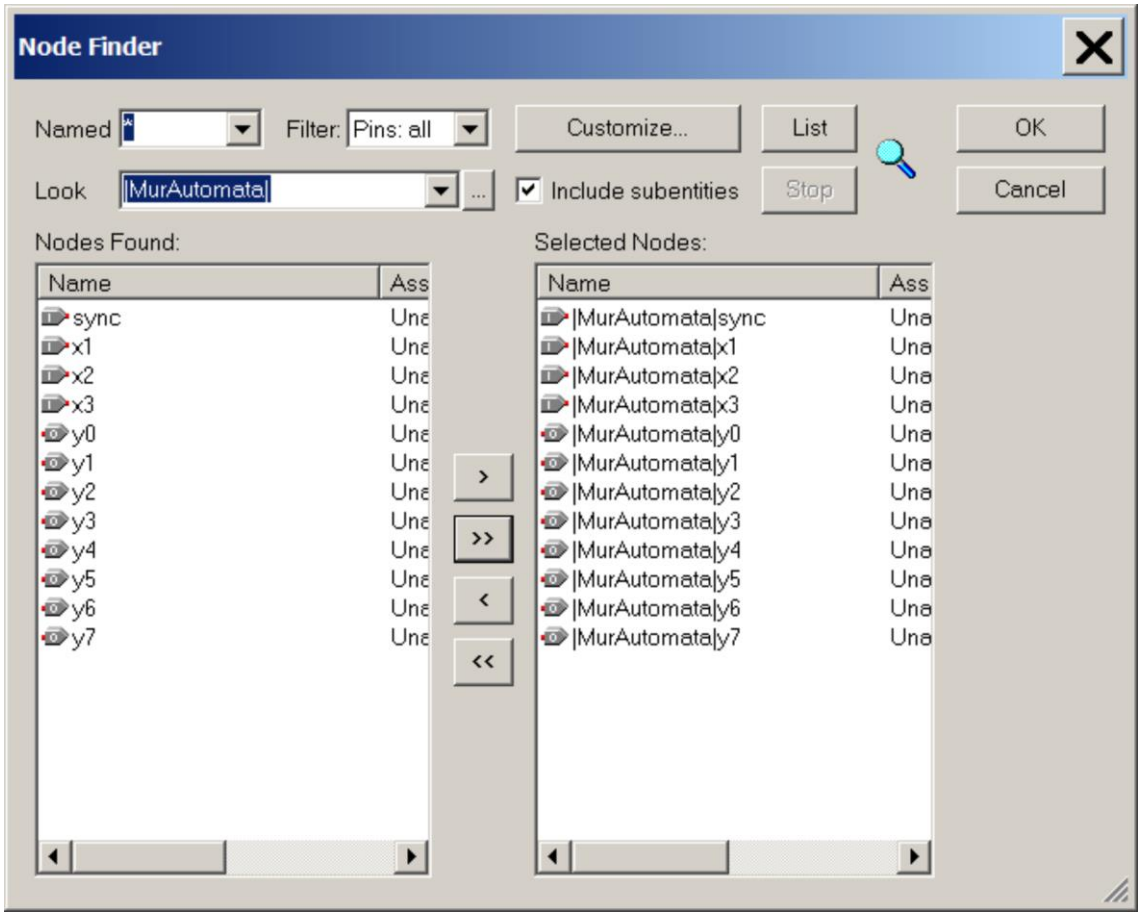


Рис. 10. Добавление сигналов на временную диаграмму

После нажатия кнопки *OK* информация о входах и выходах устройства, полученная в процессе компиляции, перенесется в файл с расширением **.vwf* и отобразится в рабочем окне сигнального редактора. В частности, в поле *Name* появятся названия всех входов и выходов устройства.

3. Для добавления внутренних сигналов (например, состояний триггеров) нужно выбрать вместо *Pins:all* фильтр *Design Entity* **(***all names***)**. Отметим, что при моделировании сложных проектов вы можете столкнуться с фактом, что нельзя найти некоторые внутренние сигналы. Это объясняется тем, что компилятор произвел оптимизацию и исключил сигналы как несущественные.

4. Прежде чем редактировать сигналы можно задать временную сетку. Для этого используется пункт меню *Edit***:** *Grid Size*. Для установки длительности симуляции служит пункт меню *Edit***:** *End Time*.

5. Затем следует установить значения входных сигналов схемы. Чтобы задать временную диаграмму сигнала на некотором входе, необходимо выделить этот сигнал целиком (нажатием левой кнопки мыши на поле *Value*) или выделить некоторый временной интервал редактируемого сигнала. Затем с помощью панели инструментов, расположенной в левой части окна редактора, задать требуемую форму сигнала.

Панель инструментов предоставляет, в частности, следующие возможности.

а) Щелчок мышью на пиктограмме с изображением нуля позволяет задать уровень логического нуля. Аналогично задается уровень логической единицы, неопределенное состояние (*X*) и третье состояние (*Z*). Под неопределенным состоянием понимается любое или неизвестное значение сигнала.

б) Щелчок мышью на пиктограмме *INV* приводит к инверсии сигнала на выделенном временном интервале.

в) Для формирования меандра (периодической последовательности импульсов) используется пиктограмма с изображением часов (рис. 11).

г) Пиктограмме *C* соответствует возможность формирования на входах или выходах последовательности чисел, образующих арифметическую прогрессию с заданной разностью (с заданным параметром *Increment By*). Абсолютная величина этого параметра может превышать единицу только в том случае, когда упомянутая последовательность формируется на шине, образованной несколькими линиями-проводниками.

На панели инструментов редактора временных диаграмм имеется кнопка привязки к временной сетке *Snap to Grid*, после ее нажатия границы выделяемого интервала будут привязываться к сетке. Для удобства анализа диаграмм можно добавлять временные метки (*Time Bar*).

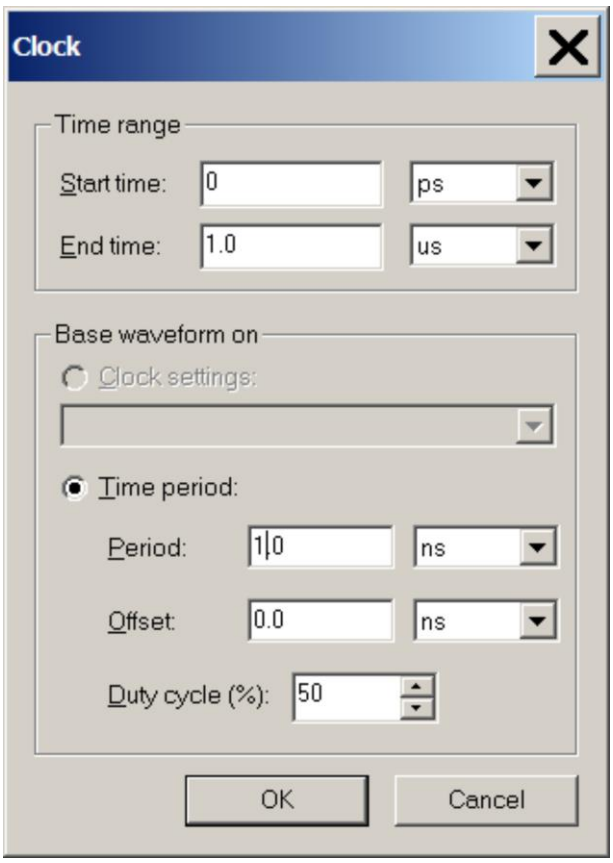


Рис. 11. Установка меандра

Группировка сигналов в шину для удобства моделирования осуществляется следующим образом. Прежде всего, в поле *Name* выделяются сигналы, подлежащие группировке. Затем нужно щелкнуть правой кнопкой мыши на выделенной области и в появившемся меню выбрать пункт *Group...*. Далее указывается система счисления для представления данных на шине: двоичная (*Binary*), восьмеричная (*Octal*), десятичная при числах со знаком (*Signed Decimal*) и при числах без знака (*Unsigned Decimal*) или шестнадцатеричная (*Hexadecimal*).

Обратная операция (разгруппировка шины) реализуется выделением требуемой шины в поле *Name* с последующим нажатием правой кнопки мыши и выбором в меню пункта *Ungroup*.

При проверке работы автомата удобно применять группировку выходных сигналов триггеров, при этом на шине образуется номер текущего состояния автомата. В качестве примера на рис. 12 приведены временные диаграммы работы автомата, причем состояния трех *T*-триггеров объединены в шину.

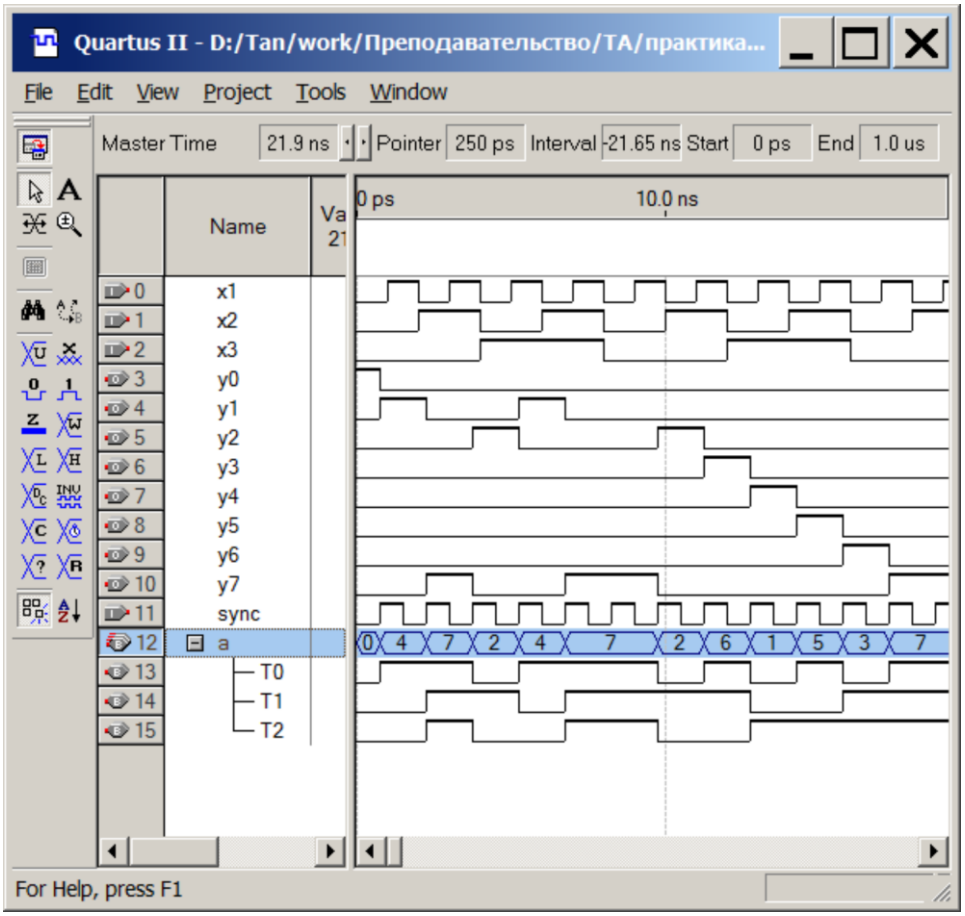


Рис. 12. Группировка сигналов в шину

После того, как будут установлены необходимые значения всех входные сигналов, можно приступать к моделированию поведения устройства во времени.

5. Функциональное моделирование проекта

После того, как будут заданы все необходимые сигналы, можно запустить симуляцию (моделирование) устройства.

Для запуска функционального моделирования в меню *Processing*: *Simulator tool* нужно выбрать *Simulation mode*: *Functional*, нажать кнопку *Generate Functional Simulation Netlist*, после сообщения о завершении создания списка цепей нажать кнопку *Start* (рис. 13).

Замечание: В *Quartus* 13 симуляция запускается кнопкой *Run Functional Simulation* на панели инструментов.

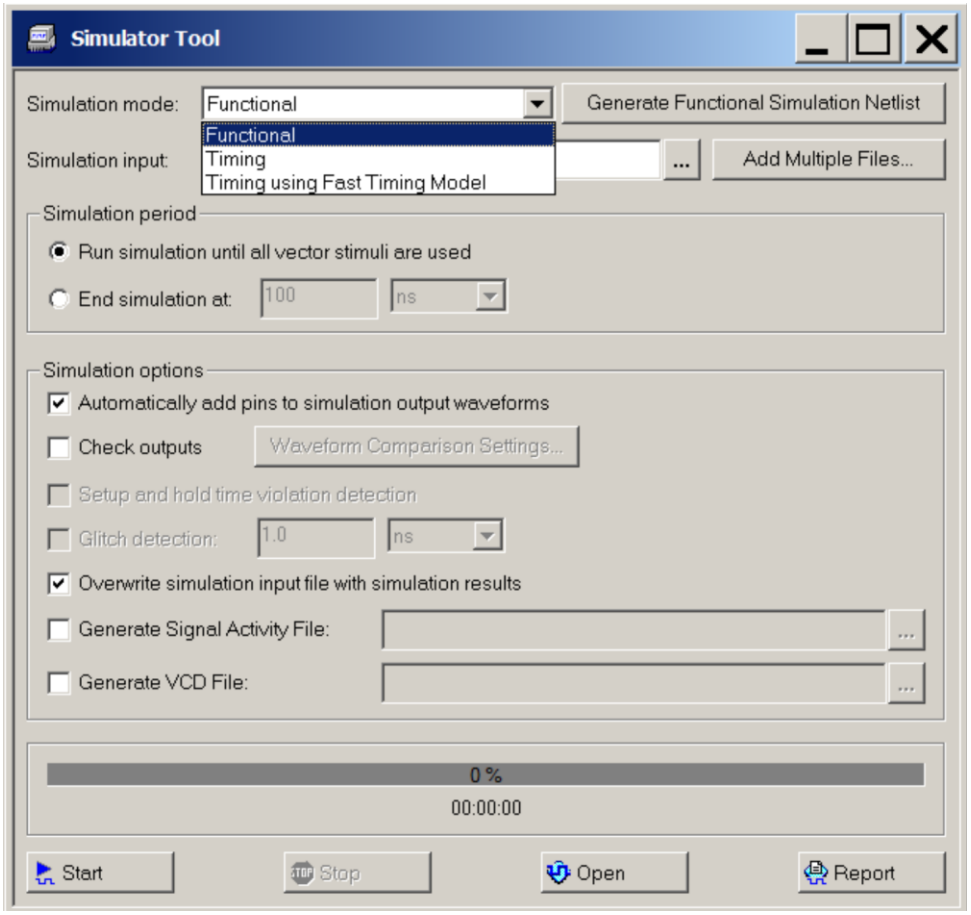


Рис. 13. Установка параметров моделирования

Результат моделирования можно увидеть, нажав кнопу *Report*. Временные диаграммы, полученные в результате симуляции, сохраняются в файл, имя которого имеет следующий вид имя_*vwf*_файла-*sim.vwf*, где имя_*vwf*_файла является именем файла временных диаграмм на входе устройства и, как правило, совпадает с именем проекта. Данный файл находится в подкаталоге *db* каталога проекта.

Для того чтобы результаты моделирования сохранялись в исходном файле необходимо в *Simulator Tool* поставить отметку в пункте *Overwrite simulation input file with simulation result*, тогда для просмотра результатов вместо *Report* можно нажимать кнопку *Open*.

6. Создание условного графического обозначения устройства

После успешной верификации спроектированного устройства для него можно создать символ (условное графическое обозначение). Для этого следует открыть файл-схему проекта и проделать путь по меню *File*: *Create / Update*: *Create Symbol Files for Current File* (рис. 14).

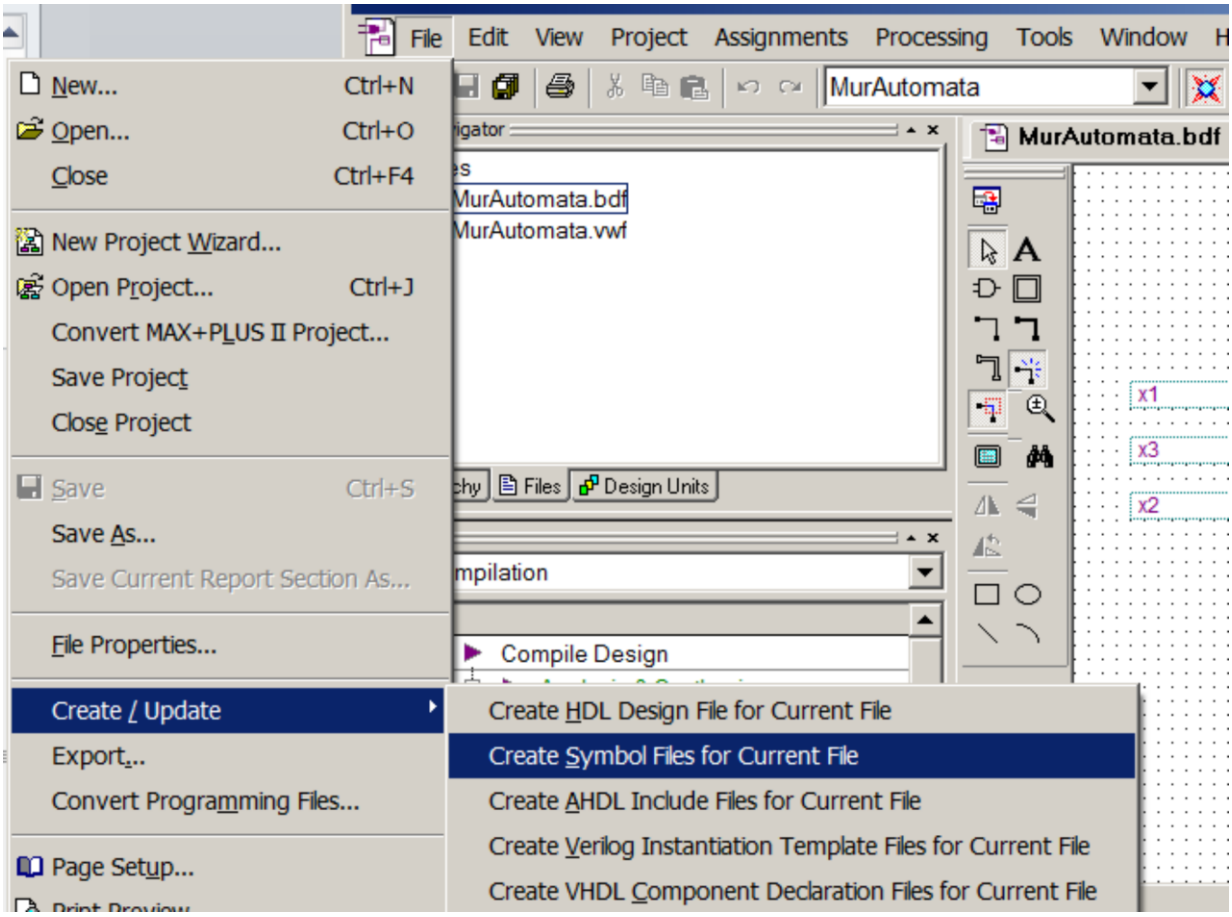


Рис. 14. Создание обозначения устройства.

В результате автоматически создается графическое обозначение для устройства (файл с расширением **.bsf*, имеющий то же имя, что и соответствующий файл-схема).

Для редактирования обозначения нужно открыть соответствующий файл **.bsf*. (рис. 15).

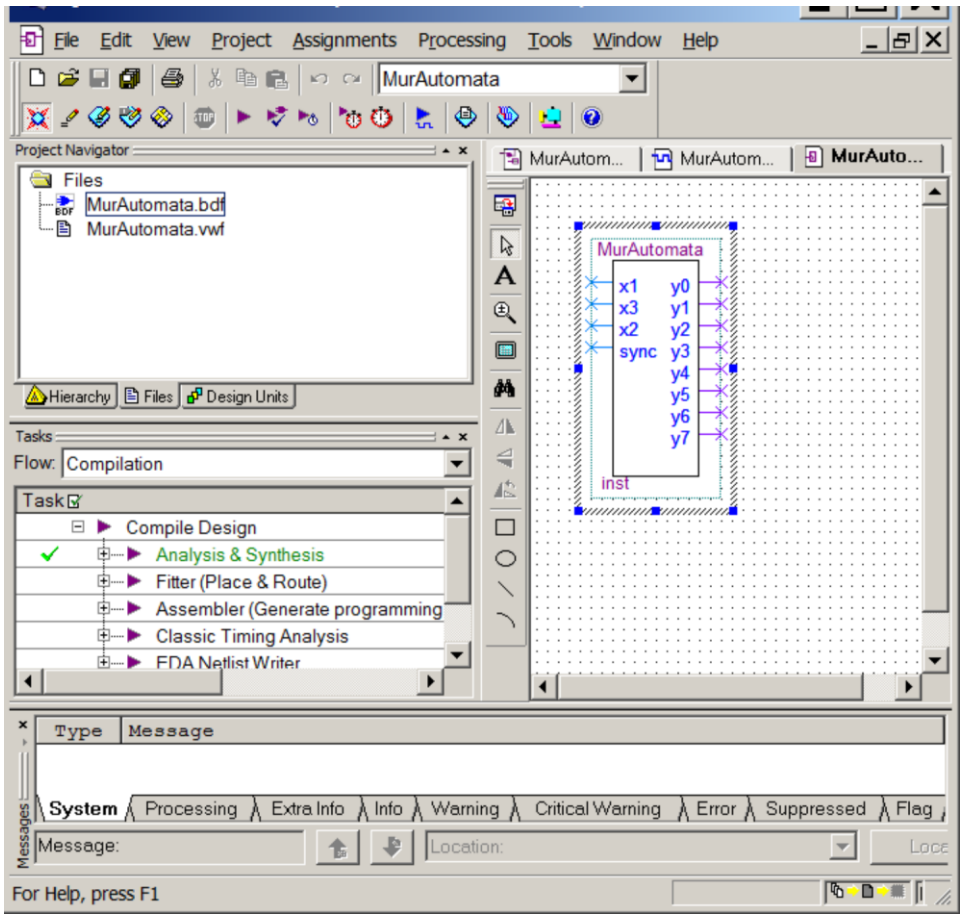


Рис. 15. Редактирование обозначения устройства

Таким образом, можно в виде отдельных блоков реализовывать часто используемые функции. Кроме того, для удобства проектирования можно разбивать на отдельные блоки слишком объемные схемы устройств.

Список литературы

1. *Ерош, И. Л.* Проектирование цифровых автоматов : Учебное пособие. Часть 1 / И. Л. Ерош, В. В. Михайлов. СПб.: ГУАП, 2009, 92 с.

2. *Лупал, А. М.* Теория автоматов : Учебное пособие / А. М. Лупал. СПб.: ГУАП, 2000, 119 с.

3. *Майоров, С. А.* Структура электронных вычислительных машин / С. А. Майоров, Г. И. Новиков. Л.: Машиностроение, 1979, 384 с.

4. *Комолов, Д. А.* Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II. Краткое описание и самоучитель / Д. А. Комолов, Р. А. Мяльк, А. А. Зобенко, А. С. Филиппов. М.: РадиоСофт, 2002, 360 с.