Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

ОТЧЕТ

по домашнему заданию № 1

вариант № 67

Название Проектирование устройств управления с жесткой логикой

Дисциплина Основы проектирования устройств ЭВМ

Студент гр. ИУ6-64Б _____ М.А.Гейне
(Подпись, дата) (И.О.Фамилия)

Преподаватель _____ А.Ю.Попов
(Подпись, дата) (И.О.Фамилия)

Москва, 2021

Цель работы: изучить методы проектирования устройств управления с жесткой логикой.

Задание

В ходе выполнения домашнего задания необходимо разработать устройство управления схемного типа, обрабатывающий входное командное слово $C = ABCDEF$ и выдающий сигналы управления $M = M_0,\ldots,M_{k-1}$ операционному блоку в соответствии с приведенной в индивидуальном задании логикой работы. Домашнее задание выполняется в несколько этапов.

1. А По диаграмме переходов автомата (Приложение 1) и описанию условий переходов и активных сигналов (дополнительный файл варианты.pdf), определить тип управляющего автомата (автомат Мили или Мура, смешанный). Выбор обосновать.

   В Произвести кодирование состояний управляющего автомата. Составить схему переходов/состояний полученного автомата. Схему представить в отчете.

2. Разработать описание устройства управления на языке VHDL, для чего использовать приведенные в Приложении 2 шаблоны для автоматов Мили и Мура. Разработать тестовое описание для устройства, представляющее собой генератор входных сигналов (см. Приложение 3). Тестовое описание должно обеспечивать проверку всех ветвей автомата

3. А Установить ПО ModelSim PE (или аналогичный продукт: Xilinx ISE, Altera Quartus)

   В Выполнить моделирование полученного теста в ПО ModelSim PE. Результаты моделирования представить в отчете.

Вариант 67

Таблица 1 – Варианты диаграмм и активных сигналов

Вариант	Диаграмма переходов	Активные сигналы M в состоянии					
		S1	S2	S3	S4	S5	S6
67	3	2	0	1,7	5,6	3	4

Таблица 2 – Условия переходов и наименование отладочной платы

Вариант	**Название отладочной платы**	**Активные сигналы C в состоянии**				
		У1	**У2**	**У3**	**У4**	**У5**
67	Spartan3	@	E_F	CD	_A_C	@
		У6	**У7**	**У8**	**У9**	**У10**
		D_B	@	A_C	C_D	@
		У11	**У12**	**У13**	**У14**	**У15**
		A+C	_B	@	DF	@

Таблица 3 – Активные сигналы для переходов

Вариант	**Активные сигналы M в состоянии**				
	У1	**У2**	**У3**	**У4**	**У5**
67					
	У6	**У7**	**У8**	**У9**	**У10**
				5, 6	5, 7
	У11	**У12**	**У13**	**У14**	**У15**
		4			

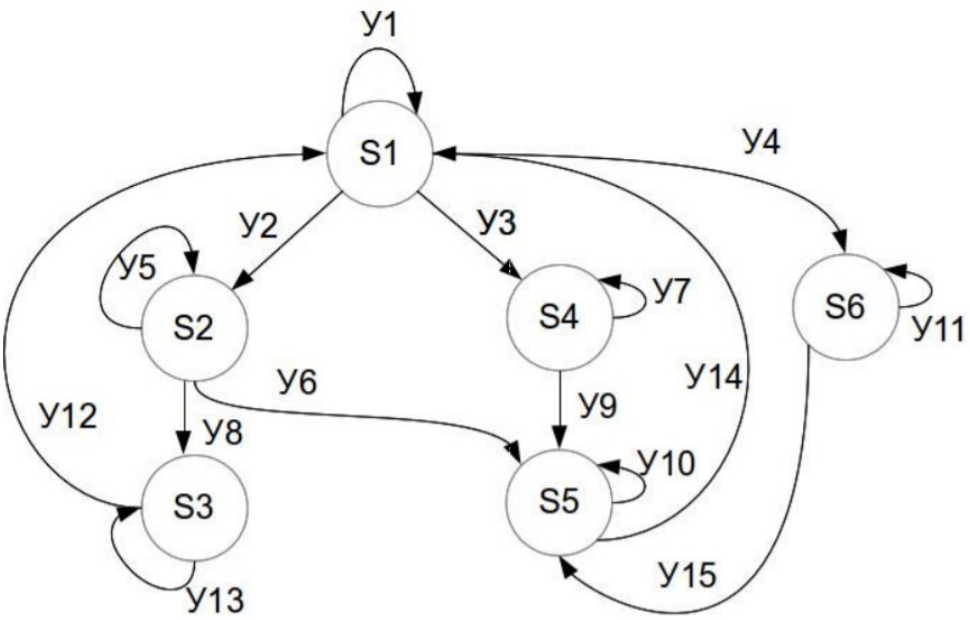

Рисунок 1 – Общая диаграмма переходов

1 Определение заданного автомата

В соответствии с вариантом домашнего задания была составлена диаграмма переходов состояний автомата, приведённая на рисунке 2.

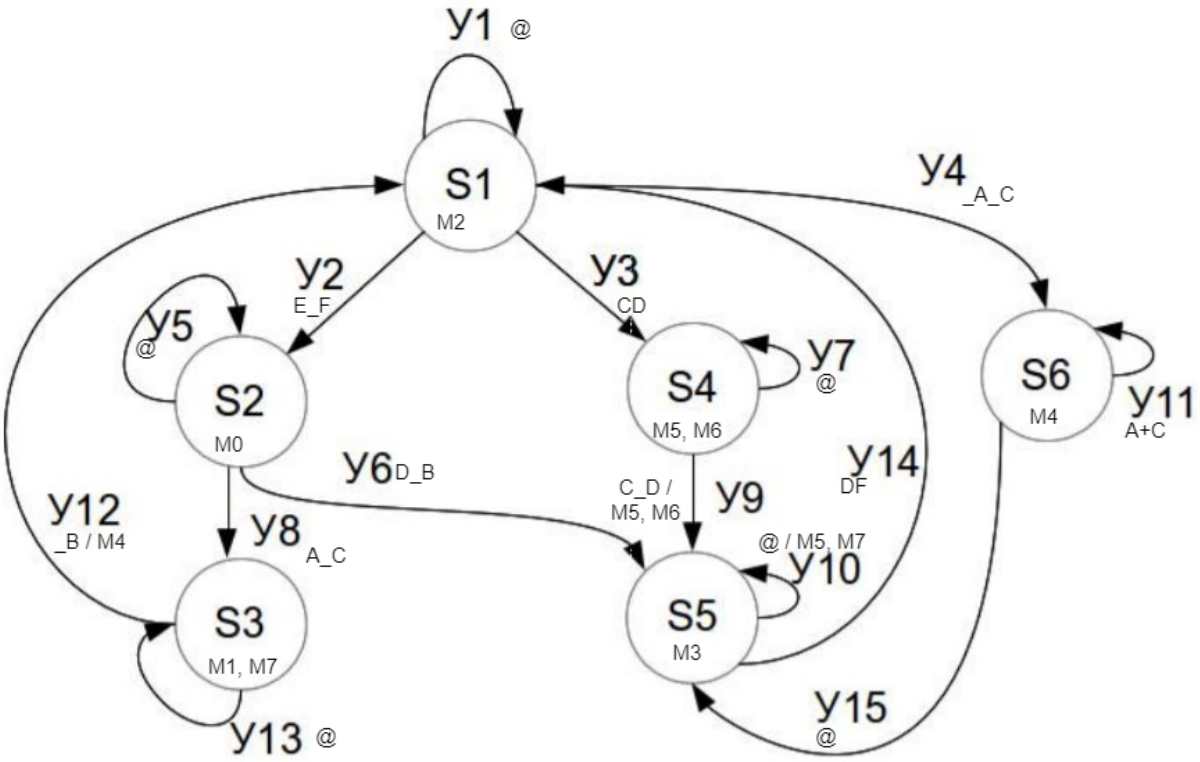

Рисунок 2 – Диаграмма переходов состояний заданного автомата

Так как выходные сигналы определены в некоторых случаях только состояниями (как в состоянии S2), а в некоторых случаях определены ещё и входными сигналами (переход У9), то можно сказать, что заданный автомат является автоматом смешанного типа.

2 Разработка устройства

После того, как был определён тип автомата, на языке VHDL был описан заданный автомат. Автомат имеет асинхронные вхо-

ды и выходы. Код устройства приведён ниже.

```
----------------------------------------------------------------------
 ↪  ------------------
-- Company:
-- Engineer:
--
-- Create Date:    17:40:21 05/09/2021
-- Design Name:
-- Module Name:    state_machine - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
 ↪  ------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity state_machine is
PORT(        c : IN std_logic_vector ( 6 DOWNTO 1 );
                clk : IN std_logic;
                rst : IN std_logic;
                m : OUT std_logic_vector ( 7 DOWNTO 1 ) );
end state_machine;

architecture Behavioral of state_machine is
        TYPE STATE_TYPE IS (s1, s2, s3, s4, s5, s6);
        SIGNAL current_state, next_state : STATE_TYPE;
begin
        clocked_proc: PROCESS(clk)
        BEGIN
                IF(rising_edge(clk)) THEN
                        IF (rst='1') THEN
                        current_state <= s1;
                        ELSE
```

```
                current_state <= next_state;
                END IF;
        END IF;
END PROCESS clocked_proc;

comb_proc : PROCESS (current_state,C)
BEGIN
        CASE current_state IS
        when s1=>
                m <= (2 =>'1', others => '0');
                if (c(5)='1' and c(6)='0') then
                        next_state<=s2;
                elsif (c(3)='1' and c(4)='1') then
                        next_state<=s4;
                elsif (c(1)='0' and c(3)='0') then
                        next_state<=s6;
                else
                        next_state<=s1;
                end if;
        when s2 =>
                m<=(1=>'1', others=>'0');
                if (c(1)='1' and c(3)='0') then
                        next_state<=s3;
                elsif (c(4)='1' and c(2)='1') then
                        next_state<=s5;
                else
                        next_state<=s2;
                end if;
        when s3=>
                m<=(1=>'1',7=>'1',others=>'0');
                if (c(2)='0') then
                        m(4)<='1';
                        next_state<=s1;
                else
                        next_state<=s3;
                end if;
        when s4 =>
                m<=(5=>'1',6=>'1', others=>'0');
                if (c(3)='1' and c(4)='0') then
                        m(5)<='1';
                        m(6)<='1';
                        next_state<=s5;
                else
                        next_state<=s4;
                end if;
        when s5 =>
                m<=(3=>'1', others=>'0');
                if (c(4)='1' and c(6)='1') then
                        next_state<=s1;
                else
                        m(5)<='1';
                        m(7)<='1';
```

```
                        end if;
                when s6 =>
                        m<=(4=>'1', others=>'0');
                        if (c(1)='1' or c(3)='1') then
                                next_state<=s6;
                        else
                                next_state<=s5;
                        end if;
                when others =>
                        next_state <= s1;
                end case;
        end process comb_proc;
end Behavioral;
```
