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 лабораторной работе № 3

вариант № 67

Название Проектирование устройств управления на основе ПЛИС

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: закрепление на практике теоретических знаний о способах реализации устройств управления, исследование способов организации узлов ЭВМ, освоение принципов проектирования цифровых устройств на основе ПЛИС.

Задание

В ходе выполнения лабораторной работы необходимо разработать устройство управления схемного типа, обрабатывающее входное командное слово $C = ABCDEF$ и выдающее сигналы управления $M = M_0,\ldots,M_{k-1}$ операционному блоку в соответствии с приведенной в индивидуальном задании логикой работы.

Задание выполняется на основе выполненного домашнего задания, в рамках которого было разработано устройство управления. В лабораторной работе необходимо разработанное устройство подготовить для работы на ПЛИС и прошить плату.

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

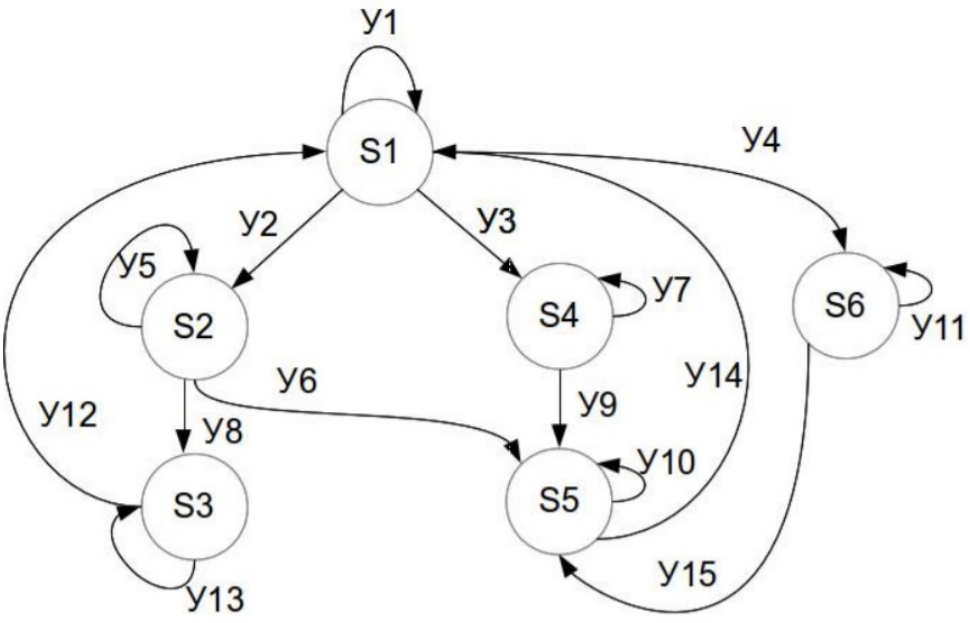

Рисунок 1 – Общая диаграмма переходов

1 Предварительные сведения об автомате

В домашнем задании было установлено, что заданный автомат является автоматом смешанного типа. Диаграмма переходов его состояний приведена на рисунке **??**. В соответствии с вариантом домашнего задания была составлена диаграмма переходов состояний автомата, приведённая на рисунке **??**. Автомат имеет асинхронные входы 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 0 ) );
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<=(0=>'1', others=>'0');
                        if (c(1)='1' and c(3)='0') then
                                next_state<=s3;
                        elsif (c(4)='1' and c(2)='0') 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;
```

2 Схема подавления дребезга

Для того, чтобы было удобно контролировать переходы автомата, в качестве тактирующего сигнала clk было решено подавать не сигналы генератора, а сигналы нажатия кнопки. Однако физическая кнопка имеет дребезг, который необходимо устранить. Для этого решено задействовать схему подавления дребезга, разработанную в лабораторной работе №2. Код устройства приведён ниже.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity delay_module is
         PORT (
                 RST: IN STD_LOGIC;
                 CLK: IN STD_LOGIC;
                 COUNT: IN STD_LOGIC;
                 CNT: OUT STD_LOGIC
         );
end delay_module;

architecture Behavioral of delay_module is
        --
        CONSTANT STATE0: STD_LOGIC_VECTOR (1 downto 0) := ”00”;
        CONSTANT STATE1: STD_LOGIC_VECTOR (1 downto 0) := ”10”;
        CONSTANT STATE2: STD_LOGIC_VECTOR (1 downto 0) := ”11”;
        CONSTANT STATE3: STD_LOGIC_VECTOR (1 downto 0) := ”01”;
        --      t
        SIGNAL S: STD_LOGIC_VECTOR (1 downto 0);
        --      t+1
        SIGNAL SN: STD_LOGIC_VECTOR (1 downto 0);
        --  2^20
        SIGNAL COUNTER: INTEGER;
        --  ” ”
        SIGNAL DLY_OVF: STD_LOGIC;
         --
         SIGNAL DLY_EN: STD_LOGIC;
begin
        --
        FSM_STATE_inst: PROCESS (CLK)
        BEGIN
        IF (CLK='1' and CLK'event) THEN
        IF (RST='1') THEN
        S <= STATE0;
        ELSE
        S <= SN;
        END IF;
        END IF;
        END PROCESS;
        --      CNT  DLY_EN (  )
        CNT <= S(1);
        DLY_EN <= S(0) xor S(1);
        --      (  )
        SN(0) <= (S(1)and S(0)) or (DLY_OVF and S(1)) or ( (not
         ↪  DLY_OVF) and S(0));
        SN(1) <= (S(1) and (not S(0))) or (COUNT and S(1)) or
         ↪  (COUNT and (not S(0)));
        --
        COUNTER_inst: PROCESS (CLK)
        BEGIN
                IF (CLK='1' and CLK'event) THEN
```

```
                        IF (RST='1' or DLY_EN = '0') THEN
                                COUNTER <= 0;
                        ELSE
                                COUNTER <= COUNTER + 1;
                        END IF;
                END IF;
        END PROCESS;
        DLY_OVF <= '1' WHEN COUNTER = 2**21-1 ELSE '0';
end Behavioral;
```

3 Сборка устройства

В целях упрощения объединения элементов был создан schematic-файл, в котором были добавлены разработанные ранее устройства, обозначены входные и выходные порты, установлены соединения между элементами. Схема устройства приведена на рисунке 2

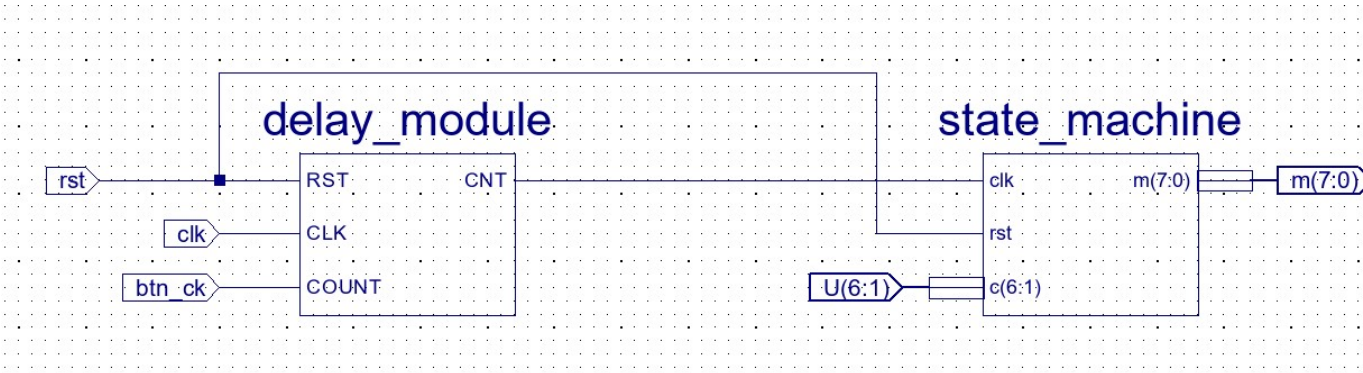

Рисунок 2 – Схема устройства

Для работы устройства необходимо назначить портам устройства корректные места подключения на плате. Так, для порта U необходимо подключить переключатели на плате, к порту btn_ck подключить кнопку, а к выходу автомата подключить светодиоды. Файл ограничений, реализующий это, приведён ниже.

```
#PACE: Start of Constraints generated by PACE
```

```
#PACE: Start of PACE I/O Pin Assignments
NET "clk"  LOC = "T9"  ;
NET "btn_ck"  LOC = "M13"  ;
NET "rst"  LOC = "L14"  ;
NET "u<1>" LOC = "F12" ;
NET "u<2>" LOC = "G12" ;
NET "u<3>" LOC = "H14" ;
NET "u<4>" LOC = "H13" ;
NET "u<5>" LOC = "J14" ;
NET "u<6>" LOC = "J13" ;
NET "m<0>" LOC = "K12";
NET "m<1>" LOC = "P14";
NET "m<2>" LOC = "L12";
NET "m<3>" LOC = "N14";
NET "m<4>" LOC = "P13";
NET "m<5>" LOC = "N12";
NET "m<6>" LOC = "P12";
NET "m<7>" LOC = "P11";
#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

После этого был сгенерирован файл для программирования, плата была прошита. Корректность работы автомата была проверена вручную.

На переключателях платы задавались входные сигналы $C = ABCDEF$. При нажатии на кнопку выполнялся переход в соответствии с диаграммой. При этом было проверено, что входы и выходы автомата являются асинхронными: в некоторых состояниях наблюдалось, как при изменении состояния переключателей менялось состояние светодиодов. При этом кнопка на плате не нажималась.

Выводы

Изучены принципы проектирования устройств управлнения с жесткой логикой.

Изучены способы задания автоматов Мили и Мура на языке VHDL.

Изучены устройства с синхронными и асинхронными входами и выходами.

Изучены методы описания устройств в виде схем в среде ISE.

Разработано устройство, демонстрирующее работу устройства управления с жесткой логикой. Входные сигналы задавались на переключателях платы, выходные наблюдались на светодиодах. Переходы состояний производились в момент нажатия на кнопку.

Проведено программирование разработанного проекта на ПЛИС.

Проведено ручное тестирование разработанного устройства на плате. По результатам тестирования было установлено, что разработанное устройство работает корректно.