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 № 2

вариант № 4

Название Проектирование цифровых устройств 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 сведений, полученных приизучении методики проектирования цифровых устройств на основе программируемыхлогических интегральных схем (ПЛИС), получение необходимых навыков работы с системойавтоматизированного проектирования ISEWebPack устройств на основе ПЛИС фирмы Xilinx, изучение аппаратных и программных средств моделирования, макетирования иотладки устройств на основе ПЛИС.

Задание

В ходе выполнения лабораторной работы выполнить проектирование счётчика нажатий на кнопку, следуя заданиям:

1. Выполнить кодирование состояний автомата в соответствии с индивидуальным вариантом;
2. Разработать текстовое описание модуля в соответствии с полученнымифункциями DLY_EN, CNT, SN(0), SN(1) на основе шаблона;
3. В интегрированном редакторе тестов САПР Xilinx ISE разработать тестдля полученного устройства и выполнить моделирование его работы в симуляторе Modelsim;
4. Разработать устройство управления, принимающее 16-разрядное слово Q[0..15] и управляющее их последовательной выдачей по шинеD[0..3] на декодер 7-сегментных индикаторов;

5. Разработать поведенческое VHDL описание схемы преобразованиячетырехразрядного информационного кода D[0..3] в код активизации 7-сегментногоиндикатора LED[0..7];

6. В редакторе схем САПР ISE добавить исходное описание, заменить пропущенные сигналы;

7. В программе XilinxPACE создать файл ограничений *.ucf или добавьте впроект имеющийся main_xc3s200.ucf;

8. В САПР ISE выполнить автоматический синтез технологической схемы,размещение и трассировку полученного устройства на кристалле Spartan3 XC3S200 ft256, сгенерировать файл конфигурации ПЛИС (*.bin);

9. Выполнить программирование макетной ПЛИС Spartan3 отладочногонабора XC3S200.

Индивидуальный вариант

Вариант	Набор	Двоичный код состояния S(1), S(0)			
		State0	State1	State2	State3
4	XC3S200	00	10	11	01

1 Кодирование автомата

Выполнение лабораторной работы начинается с проектирования схемы подавления дребезга кнопки. При нажатии и отжатии кнопка на плате дребезжит, из-за чего можно наблюдать лишние срабатывания устройства. Для подавления дребезга необходимо после нажатия кнопки подождать небольшой интервал времени, в течение которого состояние кнопки будет игнорироваться. То же самое необходимо произвести и после отжатия кнопки. Пока кнопка зажата, система должна выдавать соответствующий сигнал. Систему подавления дребезга можно реализовать в виде конечного автомата, диаграмма состояний которого приведена на рисунке 1.

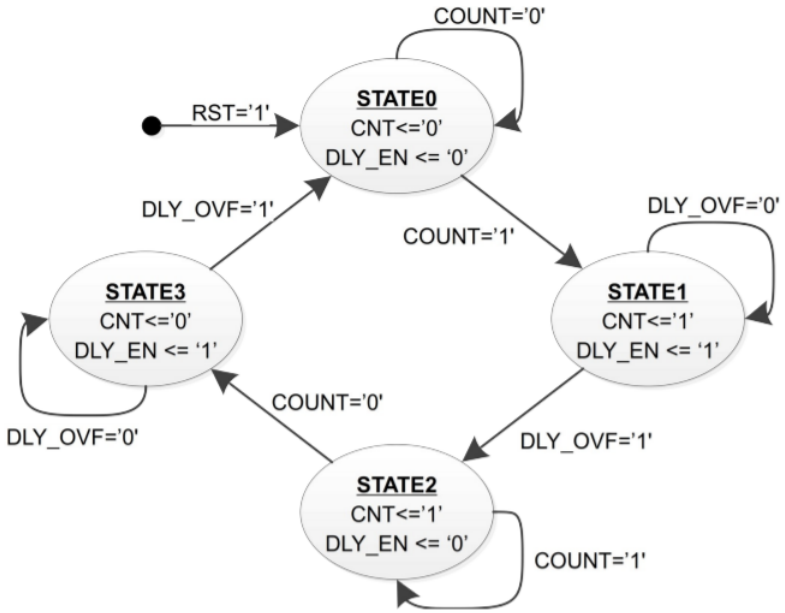

Рисунок 1 – Диаграмма состояния автомата подавления дребезга

В соответствии с индивидуальным вариантом была составлена таблица выходов автомата, зависящих от состояния. Таблица приведена на рисунке 2.

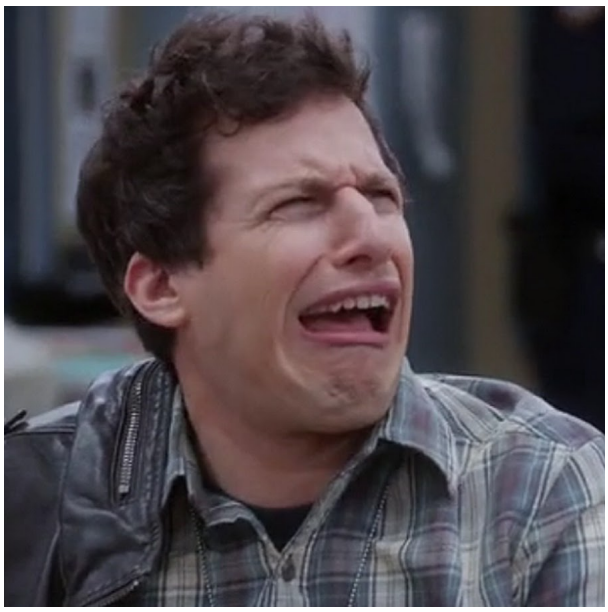

Рисунок 2 – Таблица выходов

Далее была составлена таблица переходов состояний автомата, приведённая на рисунке 3.

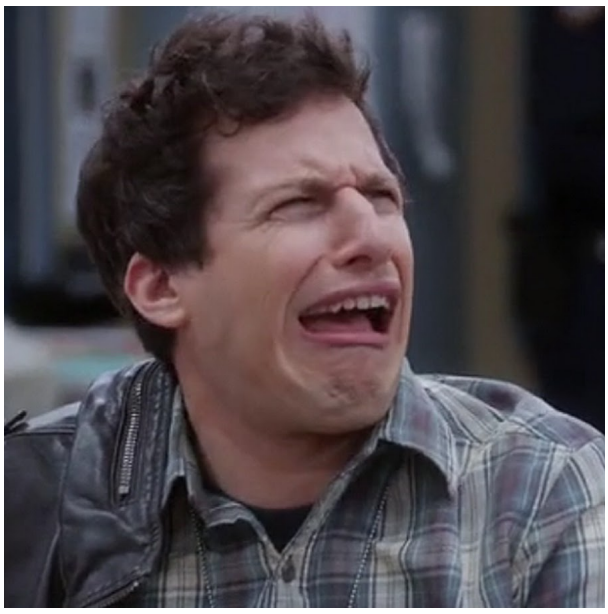

Рисунок 3 – Таблица переходов состояний автомата

На основе данной таблицы были составлены функции следующих управляющих сигналов: $SN(1) =$ и $SN(0) =$.

2 Текстовое описание автомата

После того, как были выведены необходимые логические функции, было создано текстовое описание модуля на языке VHDL. Текст модуля 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**24-1 ELSE '0';
end Behavioral;
```

---

3 Тестирование автомата

Для созданного автомата был разработан testbench средствами ISE. Временная диаграмма тестирования приведена на рисунке **??**.

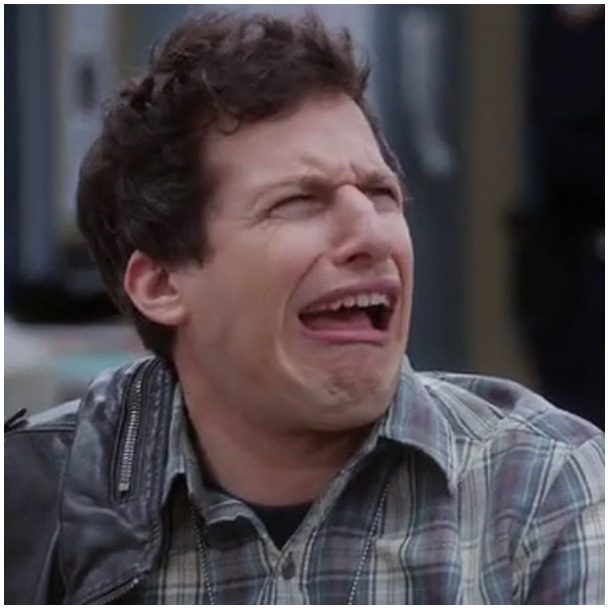

Рисунок 4 – Временная диаграмма теста автомата

Как видно на временной диаграмме, на входе автомата сигнал clk имеет несколько фронтов и спадов в малом временном промежутке, что симулирует дребезг кнопки. На выходе автомата cnt дребезг отсутствует, сигнал стабильный.

4 Разработка устройства управления индикаторами

Задача данного устройства состоит в том, чтобы вывести на 4 семи-сегментных индикатора платы текущее значение счётчика. В счётчике хранится двоичное число, тетрады которого возможно представить в виде шестнадцатиричных чисел. Всего в счёт-

чике четыре тетрады. Схема управления последовательно выделяет тетрады и подаёт их на свой выход. Кроме того, необходимо обеспечить едновременное включение всех 4 индикаторов. Для этого на каждый из индикаторов последовательно выводится код очередной цифры, переключение текущего индикатора производится с большой скоростью.

Таким образом, схема управления производит выбор индикатора, на который производится вывод, и выбор тетрады из счётчика для вывода на индикатор. Код устройства приведён ниже.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY Seven_Segment_Driver IS
  PORT(
        CLK                    : IN              std_logic;
        CLK_DIV                  : IN
         ↪                    std_logic;
        Q                        : IN
         ↪           std_logic_vector(15 DOWNTO 0);
        RST                      : IN               std_logic;
        D                      : OUT
         ↪           std_logic_vector(3 DOWNTO 0);
        A                        : OUT
         ↪           std_logic_vector(3 DOWNTO 0));
```

```
END ENTITY Seven_Segment_Driver;

ARCHITECTURE Struct OF Seven_Segment_Driver IS
        --Internal Anode
   SIGNAL        A_int : std_logic_vector(3 DOWNTO 0);
BEGIN

--Output Anode
A <= A_int;

A_drive: PROCESS (CLK,RST)
BEGIN
  IF (RST = '1') THEN
     A_int<=”1110”;
  ELSIF (CLK'EVENT AND CLK='1') THEN
        IF (CLK_DIV='1') THEN
                    A_int(3)<=A_int(2);
                    A_int(2)<=A_int(1);
                    A_int(1)<=A_int(0);
                    A_int(0)<=A_int(3);
        END IF;
  END IF;
END PROCESS A_drive;

D(0)  <= (Q(0)  AND NOT(A_int(0)))
            OR (Q(4)  AND NOT(A_int(1)))
```

```
            OR (Q(8) AND NOT(A_int(2)))
            OR (Q(12) AND NOT(A_int(3)));

D(1)  <= (Q(1)  AND NOT(A_int(0)))
            OR (Q(5)  AND NOT(A_int(1)))
            OR (Q(9) AND NOT(A_int(2)))
            OR (Q(13) AND NOT(A_int(3)));

D(2)  <= (Q(2)  AND NOT(A_int(0)))
            OR (Q(6)  AND NOT(A_int(1)))
            OR (Q(10) AND NOT(A_int(2)))
            OR (Q(14) AND NOT(A_int(3)));

D(3)  <= (Q(3)  AND NOT(A_int(0)))
            OR (Q(7)  AND NOT(A_int(1)))
            OR (Q(11) AND NOT(A_int(2)))
            OR (Q(15) AND NOT(A_int(3)));

END ARCHITECTURE Struct;
```

Для проверки правильности работы устройства был разработан testbench, временная диагрмма с которого приведена на рисунке **??**.

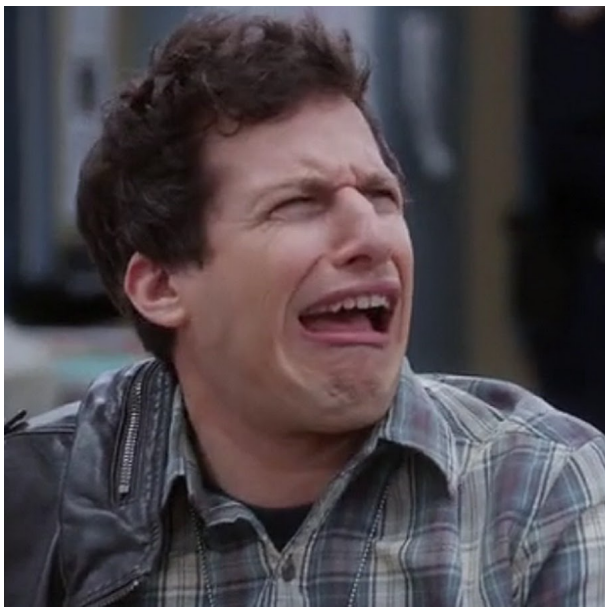

Рисунок 5 – Временная диаграмма теста устройства управления