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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 2

Название: Проектирование цифровых устройств на основе ПЛИС

Дисциплина: Основы проектирования устройств ЭВМ

Студент	ИУ6-62Б		И.С. Марчук
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

Вариант 18

Введение

Цель работы: закрепление на практике теоретических сведений, полученных при изучении методики проектирования цифровых устройств на основе программируемых логических интегральных схем (ПЛИС), получение необходимых навыков работы с системой автоматизированного проектирования ISE WebPack устройств на основе ПЛИС фирмы Xilinx, изучение аппаратных и программных средств моделирования, макетирования и отладки устройств на основе ПЛИС.

Ход работы

Условие по варианту показаны в таблице 1:

Таблица 1 – условия по варианту

Вариант	Набор	State0	State1	State2	State3
18	Nexys 2-500E	10	11	01	00

Задание 1

Выполнить кодирование состояний автомата в соответствии с индивидуальным вариантом.

Функциональная схема разрабатываемого устройства показана на рисунке 1.

Рисунок 1 - функциональная схема разрабатываемого устройства

Диаграмма состояний автомата подавления дребезга представлена на рисунке 2.

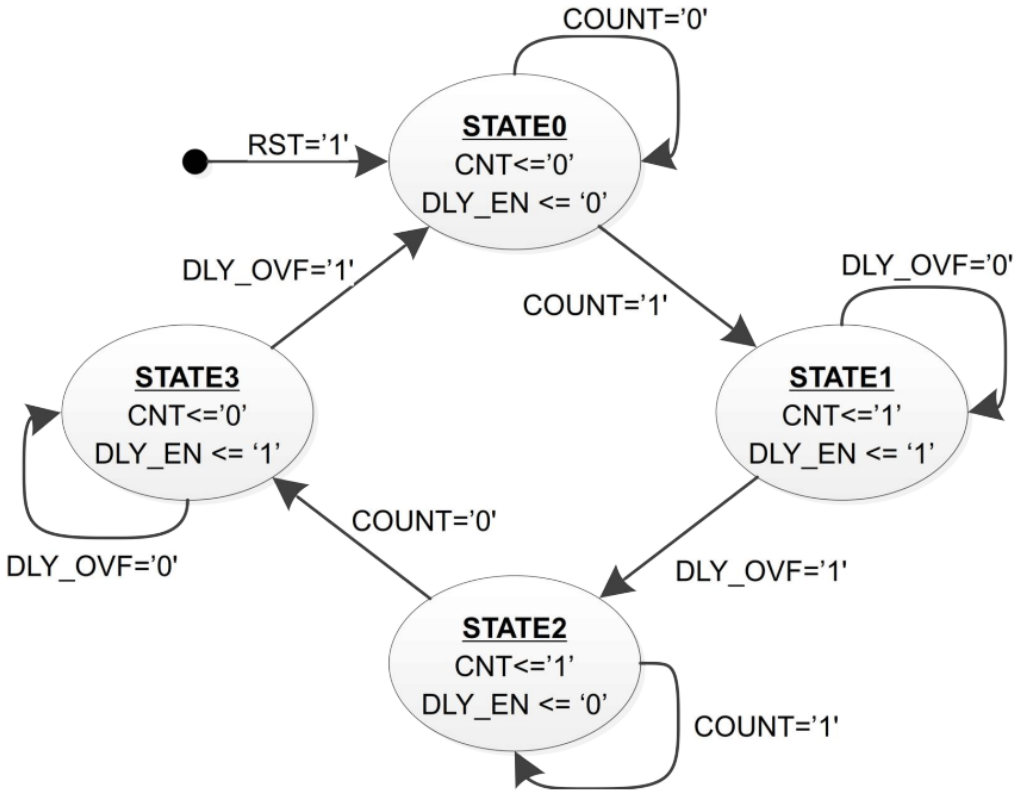

Рисунок 2 - диаграмма состояний автомата подавления дребезга

Функциональная схема устройства показана на рисунке 3.

Рисунок 3 – функциональная схема устройства

В таблице 2 представлены состояния выходов в зависимости от состояния автомата.

Таблица 2 – таблица состояний выходов

Состояние	State0	State1	State2	State3
Двоичный код состояния S(1),S(0)	10	11	01	00
CNT	0	1	1	0
DLY_EN	0	1	0	1

Из таблицы можно получить функции, задающие CNT и DLY_EN:

CNT = S(0)

DLY_EN = S(0) * S(1) OR -S(0) * -S(1)

Затем составим таблицу состояний SN.

Таблица 3 – таблица состояний SN

COUNT	DLY_OVF	S1(t)	S0(t)	S1(t+1)	S0(t+1)	SN(1)	SN(0)	Описание события
0	X	1	0	1	0	1	0	Ожидание нажатия кнопки
1	X	1	0	1	1	1	1	Нажатие кнопки
X	0	1	1	1	1	1	1	Ожидание окончания счета
X	1	1	1	0	1	0	1	Конец счета
1	X	0	1	0	1	0	1	Ожидание отпускания
0	X	0	1	0	0	0	0	Отпускание кнопки
X	0	0	0	0	0	0	0	Ожидание окончания счета
X	1	0	0	1	0	1	0	Конец счета

Составим карты Карно для SN(1) и SN(0). Они представлены таблицами 3 и 4.

Таблица 3 – карта Карно для SN(0).

SN(0)		S(1)(t), S(0)(t)			
		00	01	11	10
count, dly_ovf	00	0	0	1	0
	01	0	0	1	0
	11	0	1	**1**	**1**
	10	0	1	**1**	**1**

SN(0) = (S1 * S0) OR (S0 * COUNT) v (S1 * COUNT)

Таблица 4 – карта Карно для SN(1).

SN(1)		S(1)(t), S(0)(t)			
		00	01	11	10
count, dly_ovf	00	0	0	1	1
	01	1	0	0	1
	11	1	0	0	1
	10	0	0	1	1

SN(1) = (-S0 * DLY_OVF) v (S1 * -DLY_OVF)

Задание 2

Разработать текстовое описание модуля в соответствии с полученными функциями DLY_EN, CNT, SN(0), SN(1). Собрать на основе полученного описания проект в САПР Xilinx ISE.

Код программы по варианту:

```
-- Пример модуля подавления дребезга 10 мс.

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity lab2_example IS
 PORT (
 RST: IN STD_LOGIC; --Системный сигнал сброса
 CLK: IN STD_LOGIC; --Сигнал синхронизации
 COUNT: IN STD_LOGIC; --Сигнал кнопки с дребезгом
 CNT: OUT STD_LOGIC --Сигнал кнопки, очищенный от дребезга
 );
END lab2_example;
ARCHITECTURE behavioral OF lab2_example IS
-- Кодируем состояния в соответствии с вариантом
CONSTANT STATE0: STD_LOGIC_VECTOR (1 downto 0) := "10";
CONSTANT STATE1: STD_LOGIC_VECTOR (1 downto 0) := "11";
CONSTANT STATE2: STD_LOGIC_VECTOR (1 downto 0) := "01";
CONSTANT STATE3: STD_LOGIC_VECTOR (1 downto 0) := "00";
-- Состояние автомата в момент времени t
SIGNAL S: STD_LOGIC_VECTOR (1 downto 0);
-- Состояние автомата в момент времени t+1
SIGNAL SN: STD_LOGIC_VECTOR (1 downto 0);
SIGNAL COUNTER: integer; -- Счетчик 2^20
SIGNAL DLY_OVF: STD_LOGIC; -- Сигнал "Завершение счета"
 SIGNAL DLY_EN: STD_LOGIC; -- Сигнал разрешения работы счетчика
BEGIN
-- Память состояний
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;
-- Комбинационная схема для выработки сигналов CNT и DLY_EN (по индивидуальному варианту)
CNT <= S(0);
```

```
DLY_EN <= (S(0) and S(1)) or ( not S(0) and not S(1));
--Комбинационные схемы для определения следующего состояния (по индивидуальному варианту)
SN(0) <= ((not S(1) and S(0)) or (S(0) and COUNT) or (S(1) and COUNT));
SN(1) <= ((not S(0) and DLY_OVF) or (S(1) and not DLY_OVF));
-- Описание счетчика
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**20-1 ELSE '0'; --Длительность задержки
END Behavioral;
```

Задание 3

В интегрированном редакторе тестов САПР Xilinx ISE разработать тест для полученного устройства и выполнить моделирование его работы в симуляторе Modelsim.

На рисунке 4 показаны входные исходные для теста в графическом представлении.

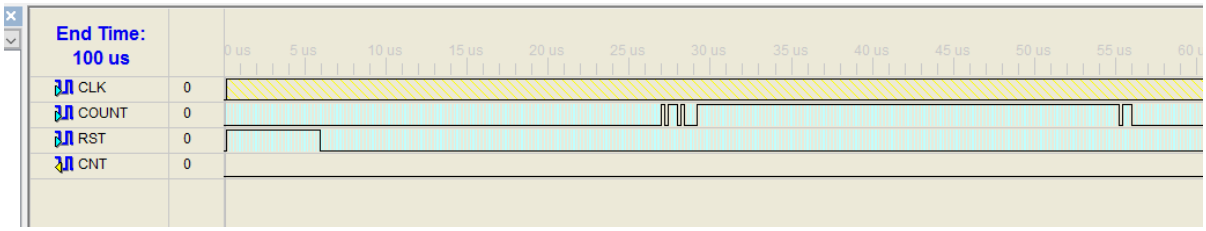

Рисунок 4 – исходные данные теста

Результаты теста показаны на рисунке 5.

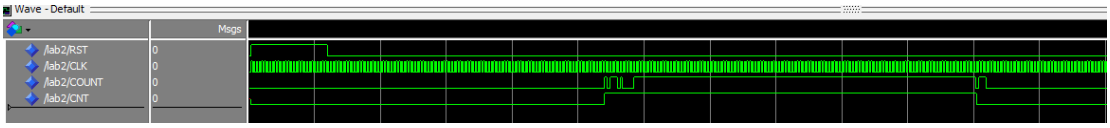

Рисунок 5 – результаты теста

Как видно из результатов теста – устройство работает корректно.

Задание 4

Разработать устройство управления, принимающее 16-разрядное информационное слово Q[0..15] и управляющее их последовательной выдачей по шине D[0..3] на декодер 7-сегментных индикаторов.

Исходный код модуля:

```
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;
```

На рисунке 6 показаны временные диаграммы, полученные при тестировании данного модуля.

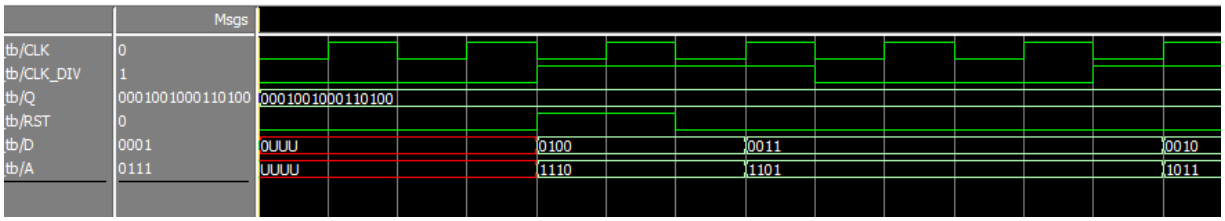

Рисунок 6 – тестирование модуля управления

Задние 5

Разработать поведенческое VHDL описание схемы преобразования четырехразрядного информационного кода D[0..3] в код активизации 7-сегментного индикатора LED[0..7].

Исходный код:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY led_decode IS
PORT (
DH: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
SEG_DATA: OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END led_decode;

ARCHITECTURE Behavioral OF led_decode IS
BEGIN
        PROCESS (DH)
        BEGIN
                CASE DH IS
                        WHEN "0000" => SEG_DATA <= "10000001";
                        WHEN "0001" => SEG_DATA <= "11001111";
                        WHEN "0010" => SEG_DATA <= "10010010";
                        WHEN "0011" => SEG_DATA <= "10000110";
```

```
                    WHEN "0100" => SEG_DATA <= "11001100";
                    WHEN "0101" => SEG_DATA <= "10100100";
                    WHEN "0110" => SEG_DATA <= "10100000";
                    WHEN "0111" => SEG_DATA <= "10001111";
                    WHEN "1000" => SEG_DATA <= "10000000";
                    WHEN "1001" => SEG_DATA <= "10000100";
                    WHEN "1010" => SEG_DATA <= "10001000";
                    WHEN "1011" => SEG_DATA <= "11100000";
                    WHEN "1100" => SEG_DATA <= "10110001";
                    WHEN "1101" => SEG_DATA <= "11000010";
                    WHEN "1110" => SEG_DATA <= "10110000";
                    WHEN "1111" => SEG_DATA <= "10111000";
                    WHEN OTHERS => NULL;
            END CASE;
        END PROCESS;
END Behavioral;
```

Задание 6

В редакторе схем САПР ISE добавить описание основного модуля.

Исходный код:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY main IS
        PORT ( CLK : IN std_logic;
        COUNT : IN std_logic;
        RESET : IN std_logic;
        A : OUT std_logic_vector (3 DOWNTO 0);
        LED : OUT std_logic_vector (7 DOWNTO 0));
END main;

ARCHITECTURE Behavioral OF main IS
        COMPONENT Seven_Segment_Driver
        PORT ( CLK : IN std_logic;
CLK_DIV : IN std_logic;
        RST : IN std_logic;
        Q : IN std_logic_vector (15 DOWNTO 0);
        D : OUT std_logic_vector (3 DOWNTO 0);
        A : OUT std_logic_vector (3 DOWNTO 0));
 END COMPONENT;
```

```
 COMPONENT led_decode
        PORT ( DH : IN std_logic_vector (3 DOWNTO 0);
        SEG_DATA : OUT std_logic_vector (7 DOWNTO 0));
 END COMPONENT;

 COMPONENT lab2_example
        PORT ( RST : IN std_logic;
        CLK : IN std_logic;
        COUNT : IN std_logic;
        CNT : OUT std_logic);
 END COMPONENT;

SIGNAL CNT_int,CNT_ff,CNT_RISE:std_logic;
        SIGNAL COUNTER: integer;
        SIGNAL COUNTER_OVF: std_logic;
        -- Main counter
        SIGNAL MAIN_COUNTER: std_logic_vector(15 DOWNTO 0);
        SIGNAL D_int : std_logic_vector(3 DOWNTO 0);
BEGIN
        ssd_inst : Seven_Segment_Driver
        PORT MAP (CLK=>CLK,
                CLK_DIV=> COUNTER_OVF,
                Q(15 DOWNTO 0)=>MAIN_COUNTER,
                RST=>RESET,
                D(3 DOWNTO 0)=>D_INT,
                A(3 DOWNTO 0)=>A
        );

        led_decode_inst : led_decode
        PORT MAP (DH(3 DOWNTO 0)=>D_INT,
                SEG_DATA(7 DOWNTO 0)=>LED
        );

lab2_example_inst : lab2_example
 PORT MAP (CLK=>CLK,
        COUNT=>COUNT,
        RST=>RESET,
 CNT=>CNT_int);
 -- Описание делителя частоты
 COUNTER_inst: PROCESS (CLK)
 BEGIN
        IF (CLK='1' and CLK'event) THEN
                IF (RESET='1' or COUNTER_OVF='1') THEN
```

```
                COUNTER <= 0;
            ELSE
                COUNTER <= COUNTER + 1;
            END IF;
        END IF;
    END PROCESS;
    COUNTER_OVF <= '1' WHEN COUNTER = 2**16 ELSE '0';

     --Детектор фронта сигнала CNT
    CNT_RISE <= '1' WHEN CNT_int='1' and CNT_ff='0' ELSE '0';
    CNT_ff_inst: PROCESS (CLK)
    BEGIN
        IF (CLK='1' and CLK'event) THEN
            IF (RESET='1') THEN
                CNT_ff <= '0';
            ELSE
                CNT_ff <= CNT_int;
            END IF;
        END IF;
    END PROCESS;

     --Основной счетчик
    MAIN_COUNTER_inst: PROCESS (CLK)
    BEGIN
        IF (CLK='1' and CLK'event) THEN
            IF (RESET='1') THEN
                MAIN_COUNTER <= (others => '0');
            ELSIF (CNT_RISE = '1') THEN
                MAIN_COUNTER <= MAIN_COUNTER + 1;
            END IF;
        END IF;
    END PROCESS;
END BEHAVIORAL;
```

Задание 7

В программе Xilinx PACE создать файл ограничений *.ucf или добавьте в проект имеющийся main_xc3s200.ucf.

Исходный код:

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

#PACE: Start of PACE I/O Pin Assignments
NET "A<0>"  LOC = "D14"  ;
```

```
NET "A<1>"  LOC = "G14"  ;
NET "A<2>"  LOC = "F14"  ;
NET "A<3>"  LOC = "E13"  ;
NET "CLK"  LOC = "T9"  ;
NET "COUNT"  LOC = "M13"  ;
NET "LED<0>"  LOC = "N16"  ;
NET "LED<1>"  LOC = "F13"  ;
NET "LED<2>"  LOC = "R16"  ;
NET "LED<3>"  LOC = "P15"  ;
NET "LED<4>"  LOC = "N15"  ;
NET "LED<5>"  LOC = "G13"  ;
NET "LED<6>"  LOC = "E14"  ;
NET "LED<7>"  LOC = "P16"  ;
NET "RESET"  LOC = "L14"  ;

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

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

Задание 8-9

В САПР ISE выполнить автоматический синтез технологической схемы, размещение и трассировку полученного устройства на кристалле Spartan XC3S 500E, генерировать файл конфигурации ПЛИС (*.bin).

Выполнить программирование макетной ПЛИС Spartan3 отладочного набора Nexys2.

Провести тестирование разработанного устройства.

Тестирование было проведено успешно, устройство инкрементировало значение счетчика выводимого на дисплей по нажатию на кнопку, при этом отфильтровывая дребезг.

Вывод: Были закреплены на практике знания полученных при изучении методики проектирования цифровых устройств на основе программируемых логических интегральных схем (ПЛИС)**.** Были получены знания и навыки разработки устройства для подавления дребезга и работы с 7-сегментным индикатором (с применением динамической индикации).