



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

\_\_\_\_\_  
(Подпись, дата)

М.А.Гейне

(И.О.Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

А.Ю.Попов

(И.О.Фамилия)

Москва, 2021

**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.

## **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 полученными функциями DLY\_EN, CNT, SN(0), SN(1) на основе шаблона;
3. В интегрированном редакторе тестов САПР Xilinx ISE разработать тест для 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 шине  $D[0..3]$  на декодер 7-сегментных индикаторов;

5. Разработать поведенческое VHDL описание схемы преобразования четырехразрядного информационного кода  $D[0..3]$  в код активизации 7-сегментного индикатора  $LED[0..7]$ ;
6. В редакторе схем САПР ISE добавить исходное описание, заменить пропущенные сигналы;
7. В программе XilinxPACЕ создать файл ограничений \*.ucf или добавьте в проект 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 отладочного набора 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 подавлениядребезга кнопки. При нажатии и отжатиикнопка на платедребезжит, из-за чего можно наблюдать лишниесрабатывания устройства. Для подавлениядребезга 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 подавлениядребезга можно реализовать в виде конечного автомата, диаграмма состояний которого приведена на рисунке 1.

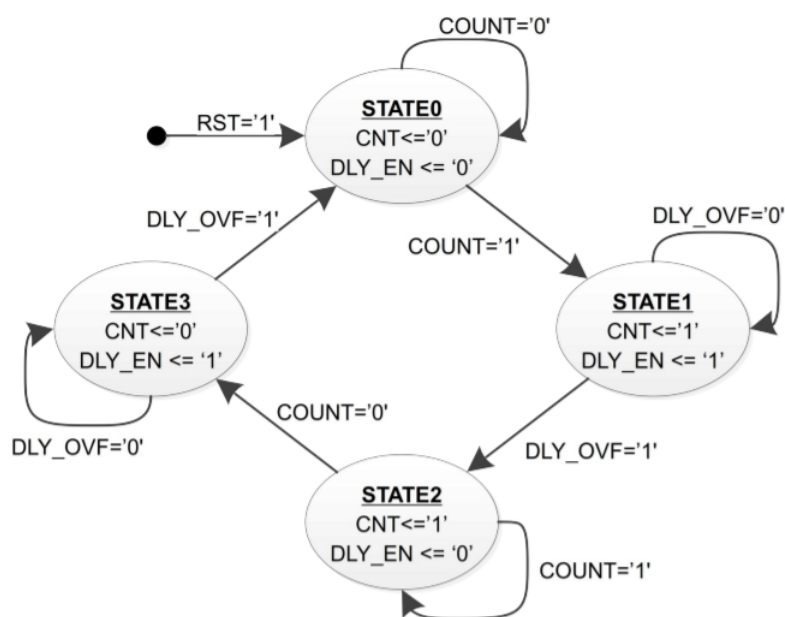


Рисунок 1 – Диаграмма состояния автомата подавлениядребезга

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

Табл. 2

state	S1	S0	CNT	DLY_EN
State0	0	0	0	0
State1	1	0	1	1
State2	1	1	1	0
State3	0	1	0	1

$$CNT = S1\bar{S0} \vee S1S0 = S1$$

$$DLY\_EN = S1\bar{S0} \vee S1S0 = S1 \oplus S0$$

Рисунок 2 – Таблица выходов

На основе данной таблицы были составлены функции следующих сигналов:  $CNT = S1$  и  $DLY\_EN = S1 \oplus S0$ .

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

Табл. 3.

Count	CS	S1(1)	S0(1)	S1(0)	S0(0)	SN(1)	SN(0)	Desk
0	X	0	0	0	0	0	0	Откл. лампы
1	X	0	0	1	0	1	0	Лампа
X	0	1	0	1	0	1	0	Откл. эквип. сл.
X	1	1	0	1	1	1	1	Конец смены
1	X	1	1	1	1	1	1	Откл. отключ.
0	X	1	1	0	1	0	1	Отключ. кнопки
X	0	0	1	0	1	0	1	Откл. эквип. смены
X	1	0	1	0	0	0	0	Конец смены

SN1

CS	S0	S1	SN1
00	0	0	0
01	0	1	1
11	1	1	1
10	1	0	1

SN0

CS	S0	S1	SN0
00	0	0	0
01	0	1	1
11	1	1	1
10	1	0	0

$$SN1 = S1\bar{S0} \vee CS1 \vee CS0$$

$$SN0 = S1S0 \vee DS1 \vee \bar{DS0}$$

Рисунок 3 – Таблица переходов состояний автомата

На основе данной таблицы были составлены функции следующих управляющих сигналов:  $SN(1) = S(1)\bar{S}(0) \vee CS(1) \vee \bar{S}(0)$  и  $SN(0) = S(1)S(0) \vee DS(1) \vee \bar{DS}(0)$ .

## 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 4.



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 симулируетдребезг кнопки. На выходе автомата cntдребезг 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 счётчике четыре тетрады. Схема управления последовательно выделяет тетрады и подаёт их на свой выход. Кроме того, необходимо обеспечить одновременное 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 диаграмма с которого приведена на рисунке 5.

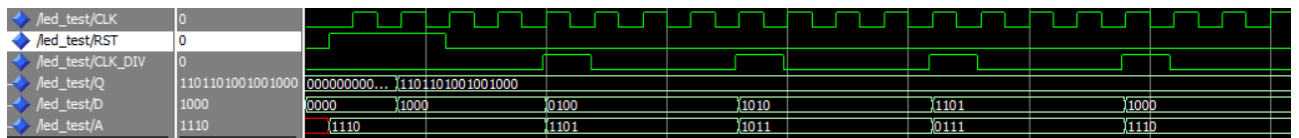


Рисунок 5 – Временная диаграмма теста устройства управления

## 5 Схема преобразования кода цифры в код индикатора

На семисегментный индикатор должен приходить не четырёхразрядный код некоторой цифры, а восьмиразрядный код, определяющий включенные сегменты индикатора. Для этого необходимо разработать преобразователь, который получив на входе 4 разряда цифры выдаст 8 разрядов кода активации индикатора. Такой преобразователь представляет из себя, по сути, Look-Up Table. Текст описания устройства приведён ниже.

---

```
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;

```

---

## 6 Разработка основного модуля проекта

Основной модуль проекта является устройством верхнего уровня, который и будет считать нажатия на кнопку. Основной модуль включает в себя разработанные ранее модули, а также 16-разрядный счётчик нажатий и делитель частоты для корректной работы схемы управления индикаторами. Код разработанного основного модуля приведён ниже.

---

```

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 delay_module
    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_CODE: 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_CODE,
                  A(3 DOWNTO 0)>=A
                  );

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

    lab2_example_inst : delay_module
        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 счётчик
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;

```

Для проверки корректности работы устройства был разработан testbench, временная диаграмма работы которого приведена на рисунке 6.

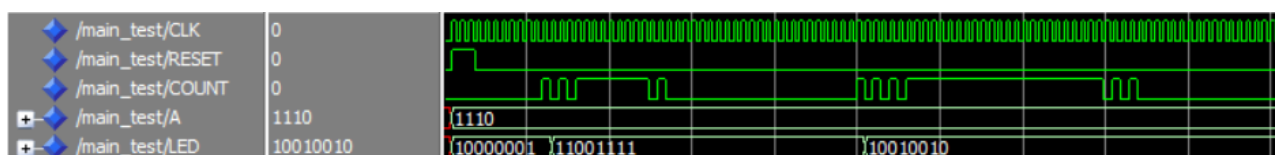


Рисунок 6 – Временная диаграмма теста основного модуля

## 7 Файл ограничений

В проект был добавлен файл ограничений, в котором были назначены внешние выводы сигналам разрабатываемого устройства в соответствии с заданными в методических указаниях для платы XC3S200. Код файла ограничений приведён ниже.

---

```
#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
```

---

## 8 Синтез схемы

В САПР ISE был проведён синтез полученного устройства. Сведения о результатах проектирования устройства отображены на рисунке 7.

lab2 Project Status (05/05/2021 - 15:31:57)					
Project File:	lab2.isc	Current State:	Programming File Generated		
Module Name:	main	• Errors:	No Errors		
Target Device:	xc3s200-5ft256	• Warnings:	No Warnings		
Product Version:	ISE 10.1 - WebPACK	• Routing Results:	<a href="#">All Signals Completely Routed</a>		
Design Goal:	Balanced	• Timing Constraints:	<a href="#">All Constraints Met</a>		
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	0 ( <a href="#">Timing Report</a> )		

lab2 Partition Summary		<a href="#">H</a>
No partition information was found.		

Device Utilization Summary					<a href="#">H</a>
Logic Utilization	Used	Available	Utilization	Note(s)	
Number of Slice Flip Flops	87	3,840	2%		
Number of 4 input LUTs	42	3,840	1%		
<b>Logic Distribution</b>					
Number of occupied Slices	65	1,920	3%		
Number of Slices containing only related logic	65	65	100%		
Number of Slices containing unrelated logic	0	65	0%		
<b>Total Number of 4 input LUTs</b>	<b>119</b>	<b>3,840</b>	<b>3%</b>		
Number used as logic	42				
Number used as a route-thru	77				
Number of bonded <a href="#">IOBs</a>	15	173	8%		
Number of BUFGMUXs	1	8	12%		

Performance Summary				<a href="#">H</a>
Final Timing Score:	0	Pinout Data:	<a href="#">Pinout Report</a>	
Routing Results:	<a href="#">All Signals Completely Routed</a>	Clock Data:	<a href="#">Clock Report</a>	
Timing Constraints:	<a href="#">All Constraints Met</a>			

Detailed Reports						<a href="#">H</a>
Report Name	Status	Generated	Errors	Warnings	Infos	
<a href="#">Synthesis Report</a>	Current	Wed May 5 15:31:49 2021	0	0	0	
<a href="#">Translation Report</a>	Current	Wed May 5 15:31:51 2021	0	0	0	
<a href="#">Map Report</a>	Current	Wed May 5 15:31:52 2021	0	0	2 Infos	
<a href="#">Place and Route Report</a>	Current	Wed May 5 15:31:55 2021	0	0	2 Infos	

Рисунок 7 – Результаты проектирования

В этом отчёте можно получить информацию о количестве элементов, используемых при синтезе: задействовано 42 LUT, 87 Slice Flip Flops.

Также ISE генерирует отчёт о таймингах, который приведён на рисунке 8.

Data Sheet report:					
-----					
All values displayed in nanoseconds (ns)					
Setup/Hold to clock CLK					
-----+-----+-----+-----+-----+					
Source	Setup to clk (edge)	Hold to clk (edge)	Internal Clock(s)	Clock Phase	
-----+-----+-----+-----+-----+					
COUNT	1.182(R)	0.330(R)	CLK_BUF GP	0.000	
RESET	4.124(R)	0.942(R)	CLK_BUF GP	0.000	
-----+-----+-----+-----+-----+					
Clock CLK to Pad					
-----+-----+-----+-----+-----+					
Destination	clk (edge) to PAD	Internal Clock(s)	Clock Phase		
-----+-----+-----+-----+-----+					
A<0>	9.117(R)	CLK_BUF GP	0.000		
A<1>	8.904(R)	CLK_BUF GP	0.000		
A<2>	9.046(R)	CLK_BUF GP	0.000		
A<3>	9.331(R)	CLK_BUF GP	0.000		
LED<0>	13.231(R)	CLK_BUF GP	0.000		
LED<1>	12.964(R)	CLK_BUF GP	0.000		
LED<2>	11.986(R)	CLK_BUF GP	0.000		
LED<3>	13.008(R)	CLK_BUF GP	0.000		
LED<4>	12.379(R)	CLK_BUF GP	0.000		
LED<5>	12.690(R)	CLK_BUF GP	0.000		
LED<6>	13.455(R)	CLK_BUF GP	0.000		
-----+-----+-----+-----+-----+					
Clock to Setup on destination clock CLK					
-----+-----+-----+-----+-----+					
Source Clock	Src:Rise	Src:Fall	Src:Rise	Src:Fall	
	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall	
-----+-----+-----+-----+-----+					
CLK	5.672				
-----+-----+-----+-----+-----+					

Рисунок 8 – Отчёт о быстродействии

Из отчёта становится известно, что на распространение CLK до портов устройства требуется не больше 14ns. От нажатия на клавишу подсчёта до clk пройдёт не более 1.2ns, а в случае сброса - не более 4.2ns. Сигнал нужно при этом задерживать менее 1ns.



## Выводы

Изучено явление дребезга кнопки и методы фильтрации дребезга.

Изучены основы проектирования на языке VHDL.

Спроектировано устройство подавления дребезга кнопки.

Спроектирована схема управления 4 семисегментными индикаторами.

Спроектировано устройство преобразования четырёхразрядного кода цифры в восьмиразрядный код активации индикатора.

Спроектировано основное устройство, включающее в себя разработанные ранее устройства, а также счётчик и делитель частоты.

Составлен файл ограничений с назначением внутренних сигналов внешним выводам платы.

Произведено тестирование разработанных устройств с использованием Testbench Waveform.

Произведён синтез разработанного устройства. Проведены оценки быстродействия.

Произведена прошивка платы, проведено ручное тестирование устройства.