АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«РУССКИЙ УНИВЕРСИТЕТ МЕТАТЕХНОЛОГИЙ»

Кафедра ПС

Отчёт
по лабораторной работе №9
«Разработка устройств обработки сигналов на программных логических интегральных схемах»
по дисциплине «Основы микропроцессорных систем»

Выполнил ст. гр. ПС-11
Поскряков Н.В.
Проверил: Электроник
кафедры ПС
Костицына К.О.

Йошкар-Ола
2025

Теоретические сведения

ПЛИС (программируемая логическая интегральная схема) - это электронное устройство, состоящие из большого количества логических элементов, которое можно программируемым образом настроить на выполнение нужной функции. Использование ПЛИС позволяет существенно ускорить разработку и производство электронной аппаратуры, поскольку значительная часть необходимой логики может быть реализована на одной микросхеме.

Практическая часть

Задание 1

Цель: собрать RS-Триггер с помощью компонентов NAND и NOR

Ход работы

Таблица 1 - Таблица истинности RS-Триггера

R	S	Q	~Q
0	0	0	1
0	1	1	0
1	0	0	1
1	1	?	?

1. RS-Триггер на компонентах NAND

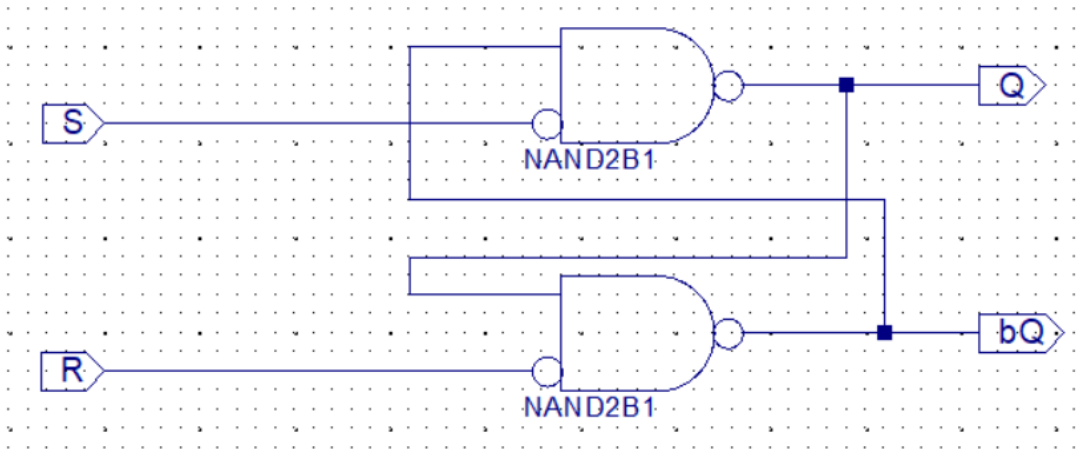

Рисунок 1 - Схема RS-Триггера на NAND

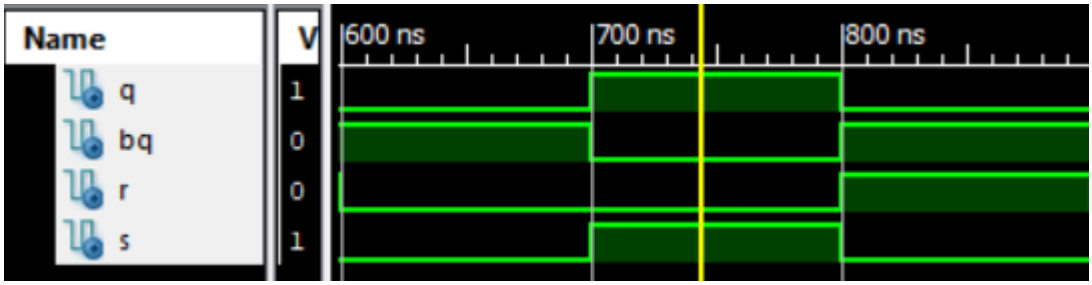

Рисунок 2 - Тестирование RS-Триггера на NAND

2. RS-Триггер на компонентах NOR

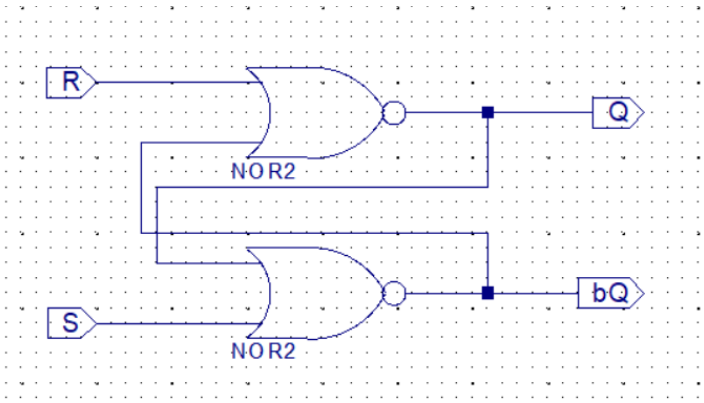

Рисунок 3 - Схема RS-Триггера на NOR

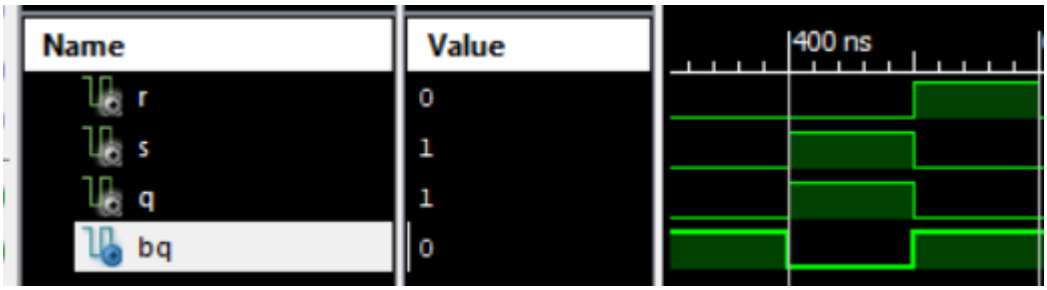

Рисунок 4 - Тестирование RS-Триггера на NOR

3. Тестирование RS-Триггера на плате

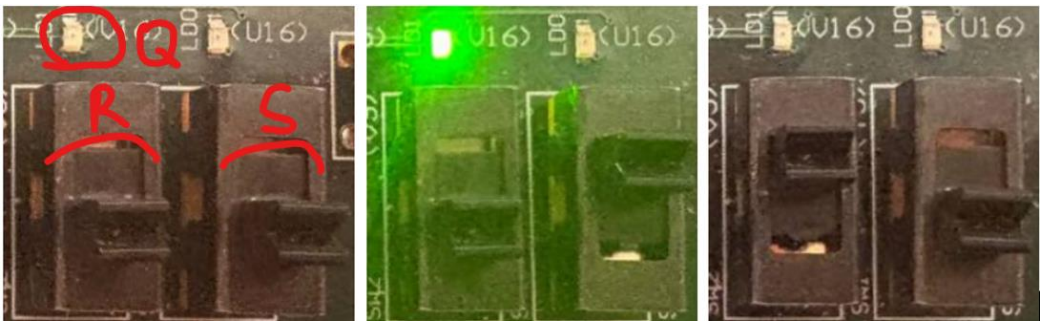

Рисунок 5 - Тестирование RS-Триггера на плате Spartan-6

Вывод: В результате проделанной работы были собраны RS-Триггеры на основе NOR и NAND, тестирование показало что результат выполнения этих вариаций никак не отличается друг от друга. При тестировании не было рассмотрено значение R = 1 и S = 1, т.к. возникает неопределённость.

Задание 2

Цель: Изучить работу D-, T- и JK-Триггеров.

Ход работы

1. D-Триггер

Таблица 2 - Таблица истинности D-Триггера

D	C	Q	~Q
0	0	Q	~Q
0	1	0	1
1	0	Q	~Q
1	1	1	0

```
41  begin
42      process(C, D)
43      begin
44          if (C'event AND C = '1') then
45              Q <= D;
46              bQ <= not(D);
47          end if;
48      end process;
49
50  end Behavioral;
```

Рисунок 6 - Реализация D-Триггера в VHDL

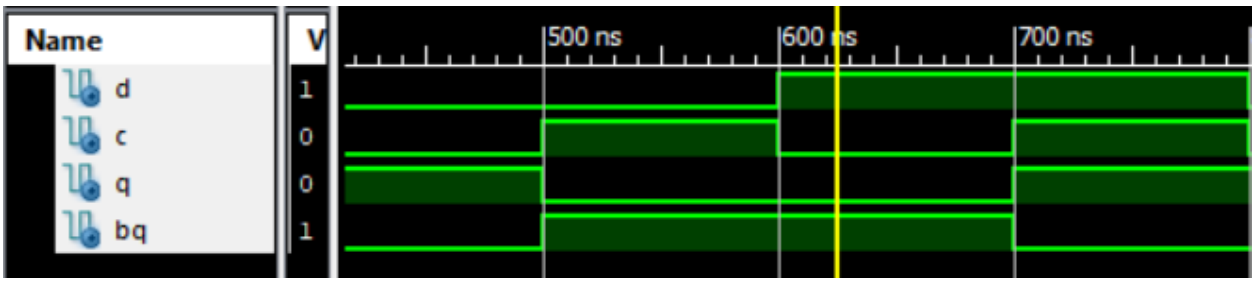

Рисунок 7 - Тестирование D-Триггера

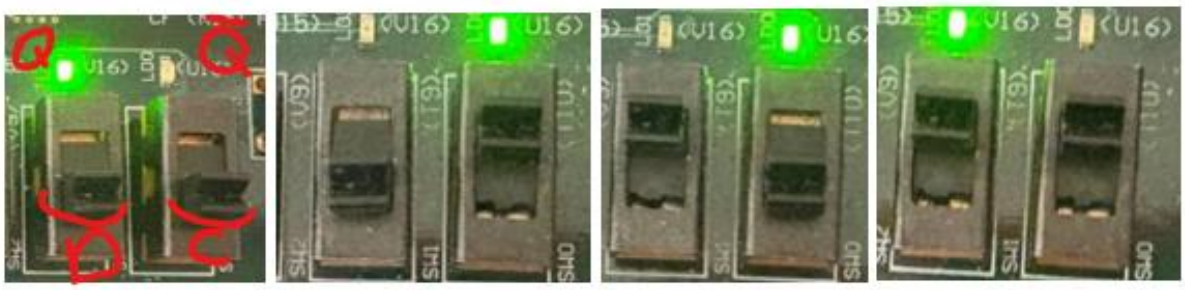

Рисунок 8 - Результаты тестирования D-Триггера на плате Spartan-6

Данная таблица истинности описывает принцип работы D-Триггера с динамическим управлением синхронизации, значение D записывается в триггер при восходящем фронте синхроимпульса.

2. T-Триггер

Таблица 3 - Таблица истинности T-Триггера

T	Q	~Q
0	Q	~Q
1	~Q	Q

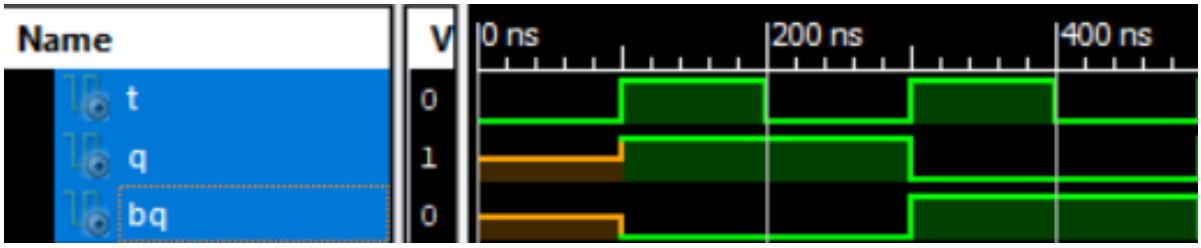

Рисунок 9 - Тестирование T-Триггера

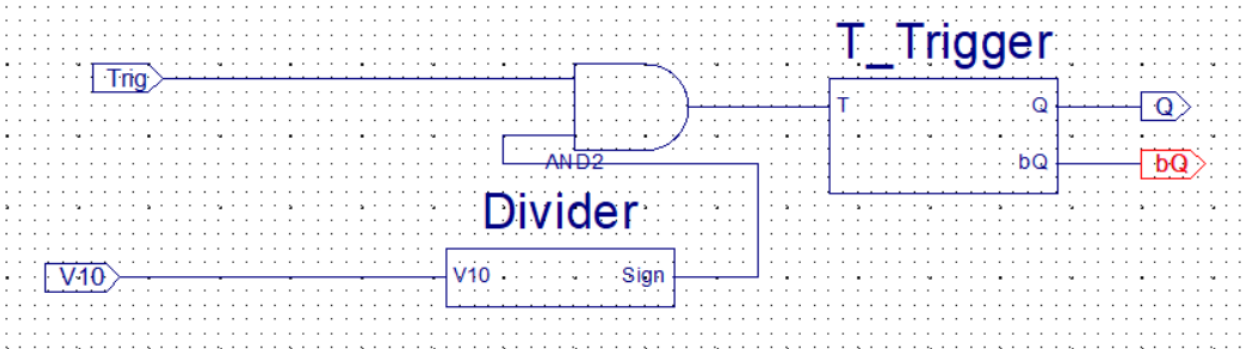

Рисунок 10 - Схема T-Триггера

```
10  architecture Behavioral of T_Trigger is
11  signal clk : std_logic := '1';
12  begin
13     process(T)
14     begin
15        if (T'event AND T = '1') then
16           clk <= not(clk);
17        end if;
18     end process;
19     Q <= clk;
20     bQ <= not(clk);
21
22  end Behavioral;
```

Рисунок 11 - Реализация Т-Триггера на VHDL

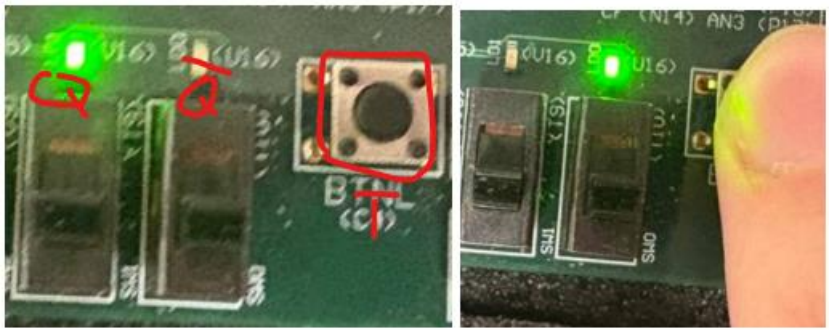

Рисунок 12 - Результаты тестирования Т-Триггера на плате Spartan-6

Данная таблица истинности описывает Т-Триггер с динамическим управлением синхронизации. Так как Т-Триггер не имеет входа синхроимпульса, мы не можем использовать простой триггер на свитчах. Из-за этого перенесём Т, на кнопку. Чтобы не возникал дребезг контактов – интегрируем схему гашения. Тогда импульсы подаваемые на Триггер должны быть короткими.

3. JK-Триггер

Таблица 4 - Таблица истинности JK-Триггера

J	C	K	Q
0	0	0	Q
0	0	1	Q
1	0	0	Q
1	0	1	Q
0	1	0	Q
0	1	1	0
1	1	0	1
1	1	1	Q

```
12  architecture Behavioral of JK_Trigger is
13
14  begin
15     process(J, C, K)
16     begin
17        if (C'event AND C = '1') then
18           if (J = '1') AND (K = '1') then
19              null;
20           else
21              if (J = '1') then
22                 Q <= '1';
23              elsif (K = '1') then
24                 Q <= '0';
25              end if;
26           end if;
27        end if;
28     end process;
29
30  end Behavioral;
```

Рисунок 13 - Реализация JK-Триггера с помощью VHDL

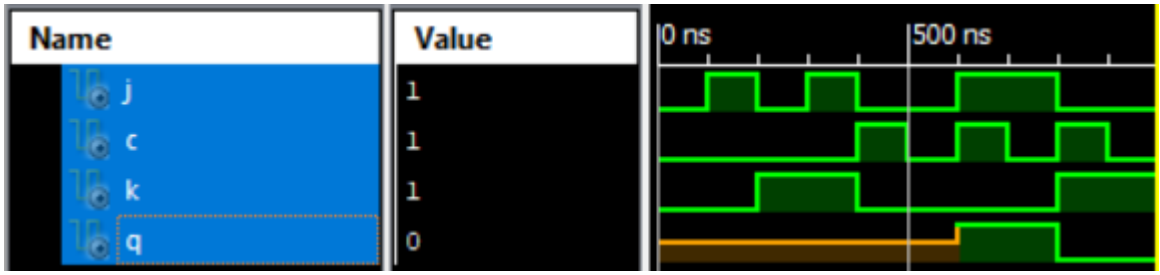

Рисунок 14 - Тестирование JK-Триггера

Так как при C = 0 состояние триггера никак не будет изменятся – проверять все эти варианты не имеет смысла.

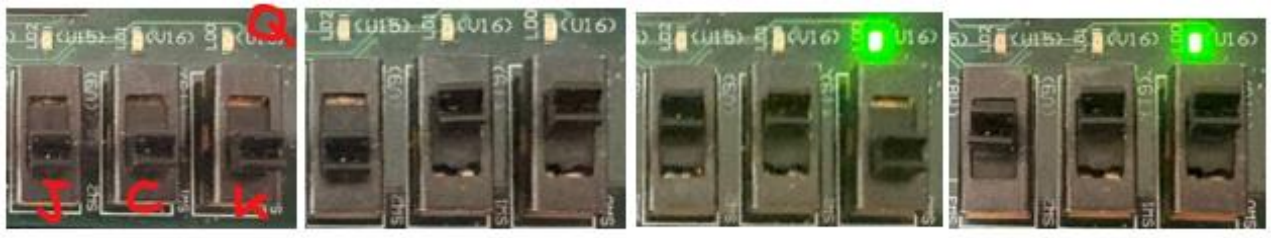

Рисунок 15 - Тестирование JK-Триггера на плате Spartan-6

Вывод: В результате проделанной работы была изучена работа D-, T- и JK-Триггеров. Составлены таблицы истинности, временные диаграммы и протестированы на плате Spartan-6. Собранные триггеры с динамическим управлением синхронизацией на восходящем фронте. Для T-Триггера использовалась конструкция гашения дребезга контактов, рассчитанная на кратчайшее нажатие кнопки. Все экспериментальные значения совпадают с таблицей истинности.

Задание 3

Цель: Создать схему динамической индикацией семисегментного модуля, с внедрением регистров.

Ход работы

1. Для удобства и лаконичности схемы – объединим 4 регистра в блок и назовём его демультиплексором чисел. В таком случае у пользователя будет возможность выбирать регистры с помощью двух свитчей и записывать в него число.
2. Будет использоваться параллельный 4-х битный регистр. В дальнейшем число будет дешифроваться на сегменты.

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


entity Num_demultiplexer is
    Port ( ENABLE : in  STD_LOGIC;
           DIGIT : in  STD_LOGIC_VECTOR (1 downto 0);
           NUMBER : in  STD_LOGIC_VECTOR (3 downto 0);
           NUM_0 : out  STD_LOGIC_VECTOR (3 downto 0);
           NUM_1 : out  STD_LOGIC_VECTOR (3 downto 0);
           NUM_2 : out  STD_LOGIC_VECTOR (3 downto 0);
           NUM_3 : out  STD_LOGIC_VECTOR (3 downto 0));
end Num_demultiplexer;

architecture Behavioral of Num_demultiplexer is

begin
   process(ENABLE, DIGIT, NUMBER)
   begin
      if (ENABLE = '1') then
         case(DIGIT) is
            when "00" => NUM_0 <= NUMBER;
            when "01" => NUM_1 <= NUMBER;
            when "10" => NUM_2 <= NUMBER;
            when "11" => NUM_3 <= NUMBER;
            when others => null;
         end case;
      end if;
   end process;

end Behavioral;
```

Рисунок 16 - Демультиплексор для переключения параллельных регистров и записи в них значений

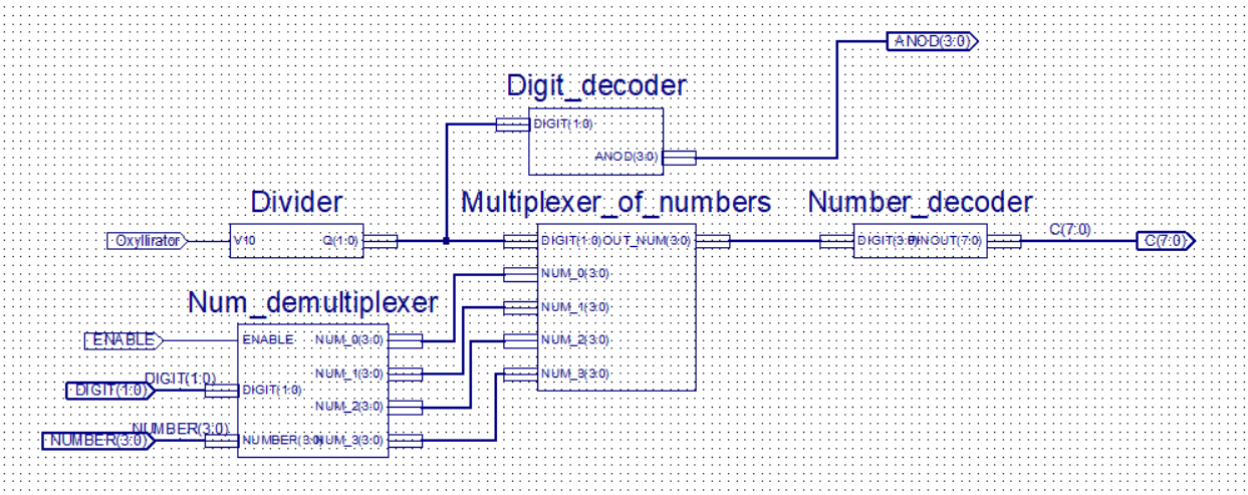

Рисунок 17 - Динамическая индикация с регистрами на схеме

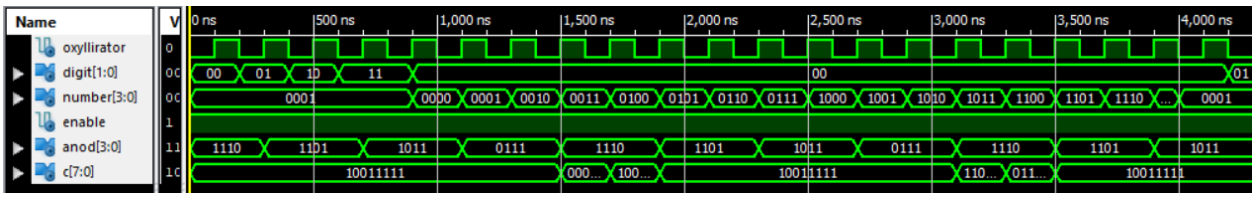

Рисунок 18 - Тестирование динамической индикации

Рисунок 19 - Результаты тестирования на плате Spartan-6

Вывод: В результате проделанной работы функционал динамической индикации содержит возможность выбора регистра для записи числа от 0 до F. Добавлен модуль, который содержит в себе 4 параллельных регистра и демультиплексор, который направляет входное число в один из 4-х регистров. В дальнейшем регистры записываются на свой разряд в семисегментном модуле.

Задание 4

Цель: Реализовать двухступенчатый JK-Триггер

Ход работы

Таблица 5 - Таблица истинности JK-Триггера

C	J	K	Q
0	0	0	Q
1	1	0	Q
0	1	0	1
1	0	1	Q
0	0	1	0

```
11  architecture Behavioral of TwoStep_JK_Trigger is
12  signal QM : STD_LOGIC := '0';
13  begin
14     process(J, C, K)
15     begin
16        if (C = '1') then
17           if (J = '1' AND K = '1') then
18              null;
19           elsif (J = '1') then
20              QM <= '1';
21           elsif (K = '1') then
22              QM <= '0';
23           end if;
24        elsif (C = '0') then
25           Q <= QM;
26        end if;
27     end process;
28
29  end Behavioral;
```

Рисунок 20 - Реализация Двуступенчатого JK-Триггера на VHDL

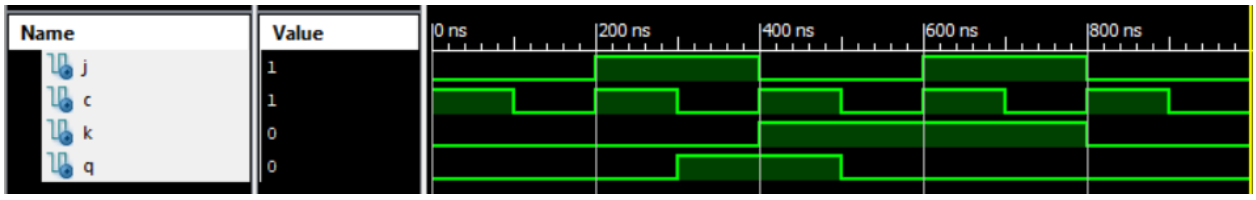

Рисунок 21 - Тестирование двуступенчатого JK-Триггера

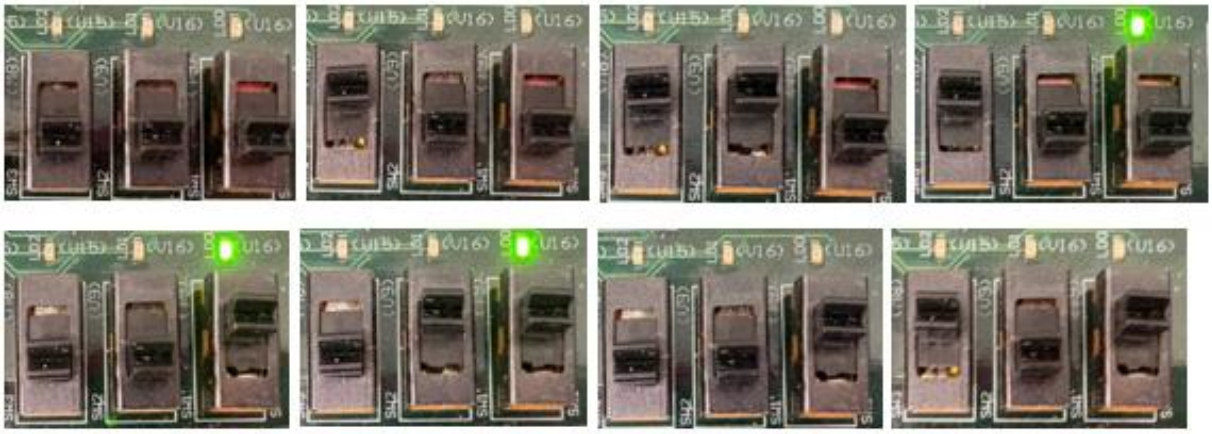

Рисунок 22 - Тестирование Двустпенчатого JK-Триггера на плате Spartan-6

Вывод: В результате проделанной работы был реализован Двуступенчатый JK-Триггер. Такой триггер является самым примитивным триггером с динамическим управлением синхронных сигналов. Первый свитч отвечает за J, второй на C, третий за K. Правый светодиод отображает значение выходного сигнала триггера.

Задание 5

Цель: Разработать игру 13 спичек.

Ход работы

1. Первоначально требуется реализовать главный игровой модуль, это будет модуль вычитания двух чисел, числа спичек и выбранного игроком числа.

```
5  entity Game_Calc is
6      Port ( E : in  STD_LOGIC;
7             RST : in  STD_LOGIC;
8             DB_IN : in  STD_LOGIC_VECTOR (1 downto 0);
9             DBM_IN : in  STD_LOGIC_VECTOR (3 downto 0) := "1011";
10            DB_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
11 end Game_Calc;
12
13 architecture Behavioral of Game_Calc is
14 signal A : STD_LOGIC_VECTOR (3 downto 0) := DBM_IN;
15 signal B : STD_LOGIC_VECTOR (1 downto 0) := DB_IN;
16 begin
17    process(E)
18    begin
19       if (E'event and E = '1') then
20          DB_OUT <= std_logic_vector(unsigned(A) - unsigned(B));
21       end if;
22    end process;
23
24 end Behavioral;
25
26
```

Рисунок 23 - Реализация игрового счёта на VHDL

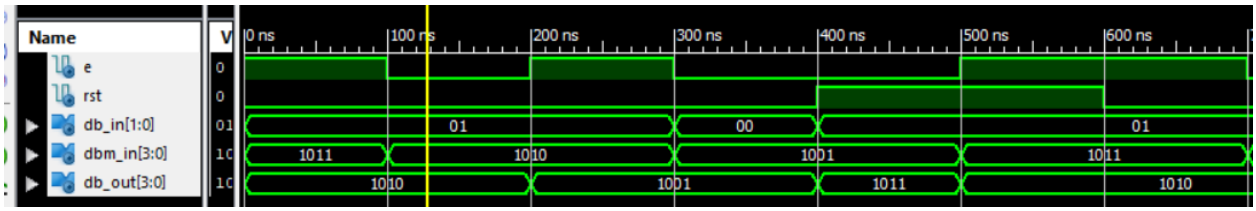

Рисунок 24 - Тестирование игрового счёта

Следует обратить внимание, что при тесте рассматривались различные комбинации сигнала E (Разрешение вычитания) и RST (Сброс), в том числе случай когда E и RST одинаково принимают лог. 1.

2. Теперь вычтенное число нам необходимо направить на индикатор и обратно в вычитающий компонент. Таким образом при каждом такте стробирующего сигнала у нас будет срабатывать вычитание.

Геймплей отправлен Ксении Олеговне на Лёрн.

Вывод: В результате выполнения задания была разработана игра “13 спичек” на плате Spartan-6, игровыми элементами являются свитчи T9, T10, которые обозначают число спичек которое берёт игров (в двоичной СС). Также присутствуют кнопки управления, C4 и B8, они соответствуют кнопке сбора спичек и кнопки “сброса” игры. По правилам игры выигрывает игрок, который совершив последний ход – забрал все спички. В таком случае загорается светодиод U16.

Общий вывод: В результате проделанной работы были исследованы Триггеры, регистры, конструкции гашения дребезга с помощью тактового генератора. Также был расширен функционал схемы динамической индикации и разработана игра “13 спичек”.