**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение   
высшего образования

«Национальный исследовательский Томский политехнический университет»

|  |  |
| --- | --- |
| Школа / филиал | Инженерная школа неразрушающего контроля и безопасности |
| Обеспечивающее подразделение | Отделение электронной инженерии |
| Направление подготовки | 11.03.04 «Электроника и наноэлектроника» |
| Образовательная программа | Прикладная электронная инженерия |

**ОТЧЕТ**

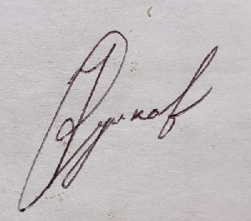
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

по дисциплине «Основы ПЛИС»



Выполнил:

студент гр. 1А22 27.02.2025 \_\_\_\_\_\_\_\_\_\_\_\_\_ Кравцов О.К.



Проверил:

ассистент ОЭИ 27.02.2025 \_\_\_\_\_\_\_\_\_\_\_\_\_ Сизиков Ф.А.

# ЦЕЛЬ РАБОТЫ

Освоить принцип работы светодиодов и кнопок на плате BASYS 3.

# ПОСТАНОВКА ЗАДАЧИ

1. Составить алгоритм горения светодиодов согласно своему варианту.
2. Рассчитать разрядность регистра и его значение для задержки смены комбинаций горения светодиодов согласно варианту и частоте входного сигнала 100 МГц.
3. Показать преподавателю расчёт задержки и рассказать алгоритм горения светодиодов
4. Написать код на VHDL.
5. Написать тестовый файл на VHDL. Промоделировать работу написанного кода в режиме «Run Post-Implementation Timing Simulation».
6. Сравнить результаты моделирования с алгоритмом горения светодиодов и реакцией на кнопки, при несоответствии исправить допущенные ошибки.
7. Назначить ножки ПЛИС для входов и выходов.
8. Показать преподавателю результаты моделирования и назначенные ножки ПЛИС.
9. Продемонстрировать результаты моделирования и назначенные ножки преподавателю.
10. Проверить работу написанного кода на плате *Basys 3*.
11. Продемонстрировать работу преподавателю.
12. Сделать несколько фотографий работы кода на плате.

# ОПИСАНИЕ АЛГОРИТМА

По заданию 11 варианта требуется, чтобы комбинации свечения светодиодов, показанные на рисунке 1, последовательно сменяли друг друга с задержкой между комбинациями 0,43 секунды, при нажатии кнопка «Пауза» (*BTNU*) изменение комбинаций прекращалось, а при нажатии кнопки «Сброс» (*BTNL*) происходил возврат к самой первой комбинации.

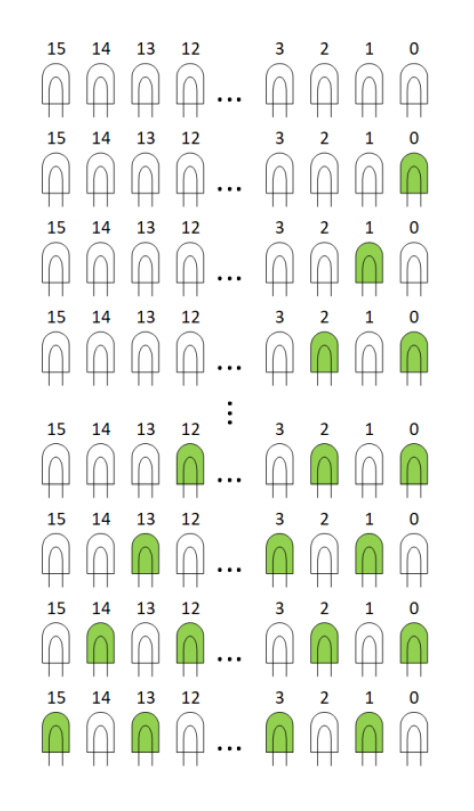


Рисунок 1 – Последовательность комбинаций свечения светодиодов

Блок схема разработанного алгоритма представлена на рисунке 2.

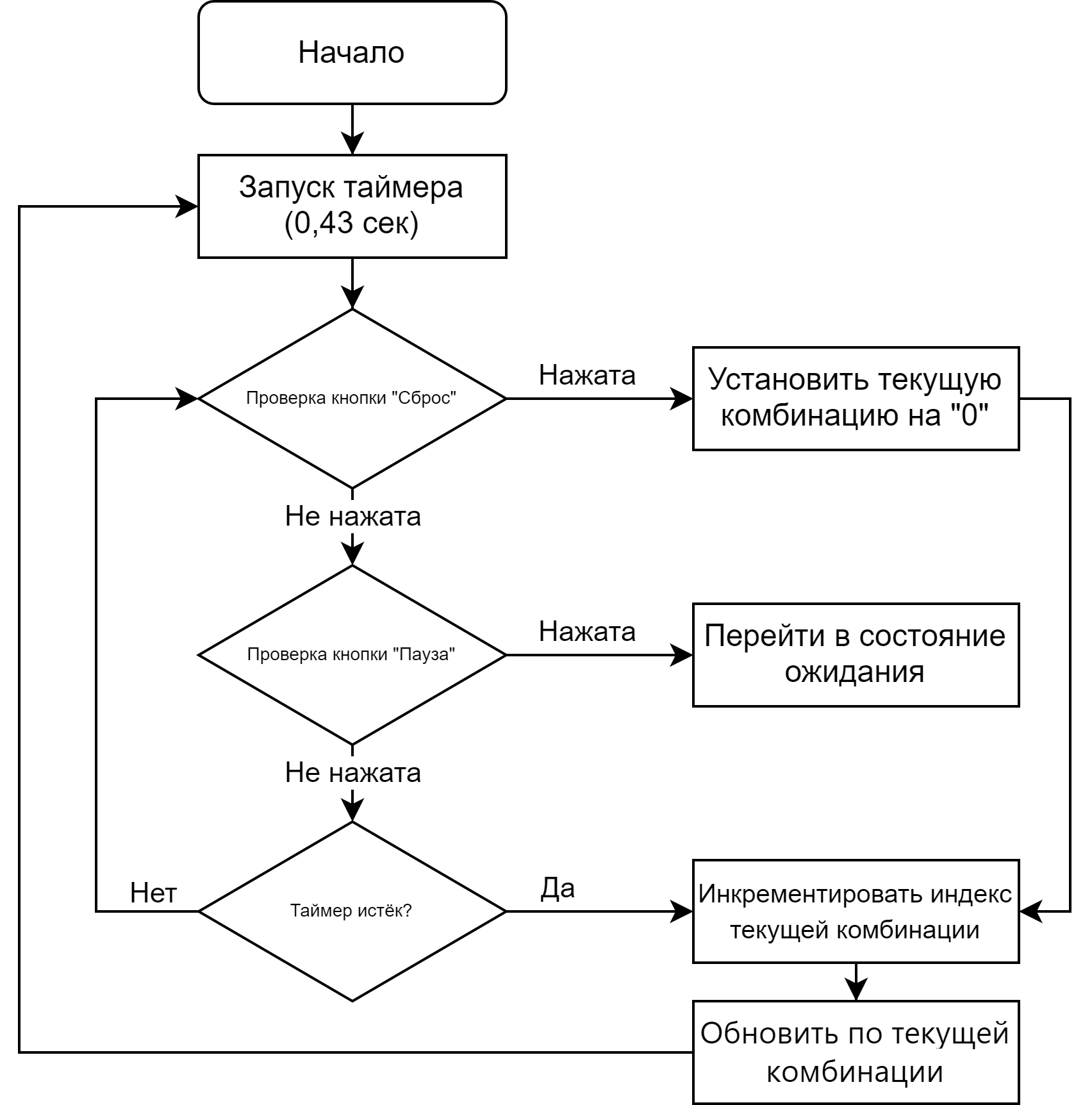


Рисунок 2 – Блок-схема алгоритма работы

# КОД НА HDL

**2.1 Модуль led\_controller**

Разработанный модуль led\_controller на языке VHDL предназначен для управления последовательностью свечения светодиодов на плате BASYS 3 в соответствии с заданным алгоритмом. Основной задачей модуля является вывод индекса текущей комбинации с заданной задержкой, а также обработка нажатий кнопок «Пауза» и «Сброс».

**Порты:**

* clk (вход): тактовый сигнал с частотой 100 МГц.
* reset (вход): сигнал сброса, активный при нажатии «Сброс».
* pause (вход): сигнал паузы, активный при нажатии «Пауза».
* combination\_index (выход): индекс текущей комбинации (5-разрядный вектор).

**Работа модуля:**

При нажатии кнопки «Сброс» текущий индекс комбинации сбрасывается в ноль, и счётчик задержки обнуляется.

При нажатии кнопки «Пауза» смена комбинаций приостанавливается.

При отсутствии паузы счётчик увеличивается на каждом такте, и при достижении заданной задержки (0,43 секунды) индекс комбинации увеличивается.

**Расчёт задержки:**

Для реализации задержки смены комбинаций светодиодов в 0,43 секунды при частоте тактового сигнала 100 МГц (*W5* на *Basys 3*) необходимо определить количество тактовых импульсов, соответствующих этой задержке:

Значение 43,000,000 представлено в шестнадцатеричной системе как X"29020C0" и используется в качестве константы *DELAY* для сравнения с текущим значением счётчика *counter*.

Листинг модуля *led\_controller* на языке *VHDL:*

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity led\_controller is

    Port (

        clk     : in  STD\_LOGIC;

        reset   : in  STD\_LOGIC;

        pause   : in  STD\_LOGIC;

        combination\_index   : out STD\_LOGIC\_VECTOR (4 downto 0)

    );

end led\_controller;

architecture Behavioral of led\_controller is

    signal current\_index : UNSIGNED(4 downto 0) := (others => '0');

    signal counter       : UNSIGNED(31 downto 0) := (others => '0');

    constant MAX\_INDEX   : UNSIGNED(4 downto 0) := "10000"; -- 17 combinations

    constant DELAY       : UNSIGNED(31 downto 0) := X"029020C0"; -- 43,000,000 for 100 MHz

    signal running       : STD\_LOGIC := '1';

begin

    process(clk, reset)

    begin

        if reset = '1' then

            current\_index <= (others => '0');

            counter <= (others => '0');

            running <= '1';

        elsif rising\_edge(clk) then

            running <= not pause;

            if running = '1' then

                if counter < DELAY then

                    counter <= counter + 1;

                else

                    counter <= (others => '0');

                    current\_index <= current\_index + 1;

                    if current\_index > MAX\_INDEX then

                        current\_index <= (others => '0');

                    end if;

                end if;

            end if;

        end if;

    end process;

    combination\_index <= std\_logic\_vector(current\_index);

end Behavioral;

**2.2 Модуль led\_pattern**

Модуль led\_pattern реализует выбор последовательности свечения светодиодов на основе заданного индекса.

**Порты:**

* *index* (вход): 5-разрядный вектор, определяющий текущую комбинацию светодиодов.
* *leds* (выход): 16-разрядный вектор, управляющий состоянием светодиодов на плате.

**Работа модуля:**

На основе входного индекса формируется комбинация светодиодов.

Если индекс превышает допустимое значение, он ограничивается.

Листинг модуля *led\_pattern* на языке *VHDL:*

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity led\_pattern is

    Port (

        index   : in  STD\_LOGIC\_VECTOR(4 downto 0);

        leds    : out STD\_LOGIC\_VECTOR(15 downto 0)

    );

end led\_pattern;

architecture Behavioral of led\_pattern is

begin

    process(index)

        variable pattern : STD\_LOGIC\_VECTOR(15 downto 0);

        variable idx\_int : integer;

    begin

        idx\_int := to\_integer(unsigned(index));

        pattern := (others => '0');

        -- Limit the range of idx\_int

        if idx\_int > 16 then

            idx\_int := 16;  -- Set index to 16 if it exceeds 16

        end if;

        if idx\_int = 0 then

            pattern := (others => '0');  -- Keep all bits '0' if index is 0

        else

            for i in 0 to 15 loop  -- Fixed range loop

                if i < idx\_int then

                    if ((idx\_int mod 2 = 0) and (i mod 2 = 1)) or

                       ((idx\_int mod 2 = 1) and (i mod 2 = 0)) then

                        pattern(i) := '1';  -- Set bit to '1' according to condition

                    end if;

                end if;

            end loop;

        end if;

        leds <= pattern;  -- Assign the result to the output signal leds

    end process;

end Behavioral;

**2.1 Модуль top\_module**

Модуль верхнего уровня объединяет в себе два подмодуля:

1. *led\_controller*.

Получает на вход тактовый сигнал *clk*, сигнал сброса *reset* и сигнал паузы *pause*. Генерирует 5-битный индекс *combination\_index*, который определяет текущий паттерн светодиодов. Индекс передается через сигнал index\_signal.

2. *led\_pattern*:

Принимает на вход сигнал *index\_signal*. На основе полученного индекса формирует соответствующий 16-битный паттерн для светодиодов. Выводит сформированный паттерн на *led\_output*.

Листинг модуля *top\_module* на языке *VHDL:*

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity top\_module is

    Port (

        clk         : in  STD\_LOGIC;

        reset       : in  STD\_LOGIC;

        pause       : in  STD\_LOGIC;

        led\_output  : out STD\_LOGIC\_VECTOR(15 downto 0)

    );

end top\_module;

architecture Behavioral of top\_module is

    signal index\_signal : STD\_LOGIC\_VECTOR(4 downto 0);

begin

    U1: entity work.led\_controller

        Port map (

            clk => clk,

            reset => reset,

            pause => pause,

            combination\_index => index\_signal

        );

    U2: entity work.led\_pattern

        Port map (

            index => index\_signal,

            leds => led\_output

        );

end Behavioral;

# НАзначение ВЫВОДОВ плис

Файл *basys3.xdc* определяет привязку логических портов проекта к физическим выводам ПЛИС на плате *Basys 3*:

* Тактовый сигнал *clk* подключен к выводу *W5*.
* Сигнал сброса *reset* подключен к кнопке "*Button Left*" на выводе *W19*.
* Сигнал паузы *pause* подключен к кнопке "*Button Up*" на выводе *T18*.
* Выходной вектор светодиодов *led\_output[0..15]* подключен к выводам U16, E19, U19, V19, ..., L1, управляющим 16 светодиодами на плате.
* Для всех сигналов установлен стандарт логических уровней *LVCMOS33* (3.3 В).

Листинг *basys3.xdc*:

## Clock signal

set\_property PACKAGE\_PIN W5 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

## Reset button (Button Left)

set\_property PACKAGE\_PIN W19 [get\_ports reset]

set\_property IOSTANDARD LVCMOS33 [get\_ports reset]

## Pause button (Button Up)

set\_property PACKAGE\_PIN T18 [get\_ports pause]

set\_property IOSTANDARD LVCMOS33 [get\_ports pause]

# Output signal

set\_property PACKAGE\_PIN U16 [get\_ports {led\_output[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[0]}]

set\_property PACKAGE\_PIN E19 [get\_ports {led\_output[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[1]}]

set\_property PACKAGE\_PIN U19 [get\_ports {led\_output[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[2]}]

set\_property PACKAGE\_PIN V19 [get\_ports {led\_output[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[3]}]

set\_property PACKAGE\_PIN W18 [get\_ports {led\_output[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[4]}]

set\_property PACKAGE\_PIN U15 [get\_ports {led\_output[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[5]}]

set\_property PACKAGE\_PIN U14 [get\_ports {led\_output[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[6]}]

set\_property PACKAGE\_PIN V14 [get\_ports {led\_output[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[7]}]

set\_property PACKAGE\_PIN V13 [get\_ports {led\_output[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[8]}]

set\_property PACKAGE\_PIN V3 [get\_ports {led\_output[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[9]}]

set\_property PACKAGE\_PIN W3 [get\_ports {led\_output[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[10]}]

set\_property PACKAGE\_PIN U3 [get\_ports {led\_output[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[11]}]

set\_property PACKAGE\_PIN P3 [get\_ports {led\_output[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[12]}]

set\_property PACKAGE\_PIN N3 [get\_ports {led\_output[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[13]}]

set\_property PACKAGE\_PIN P1 [get\_ports {led\_output[14]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[14]}]

set\_property PACKAGE\_PIN L1 [get\_ports {led\_output[15]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led\_output[15]}]

# Фотографии макета

На рисунках 4-6 представлены фотографии работы макета.

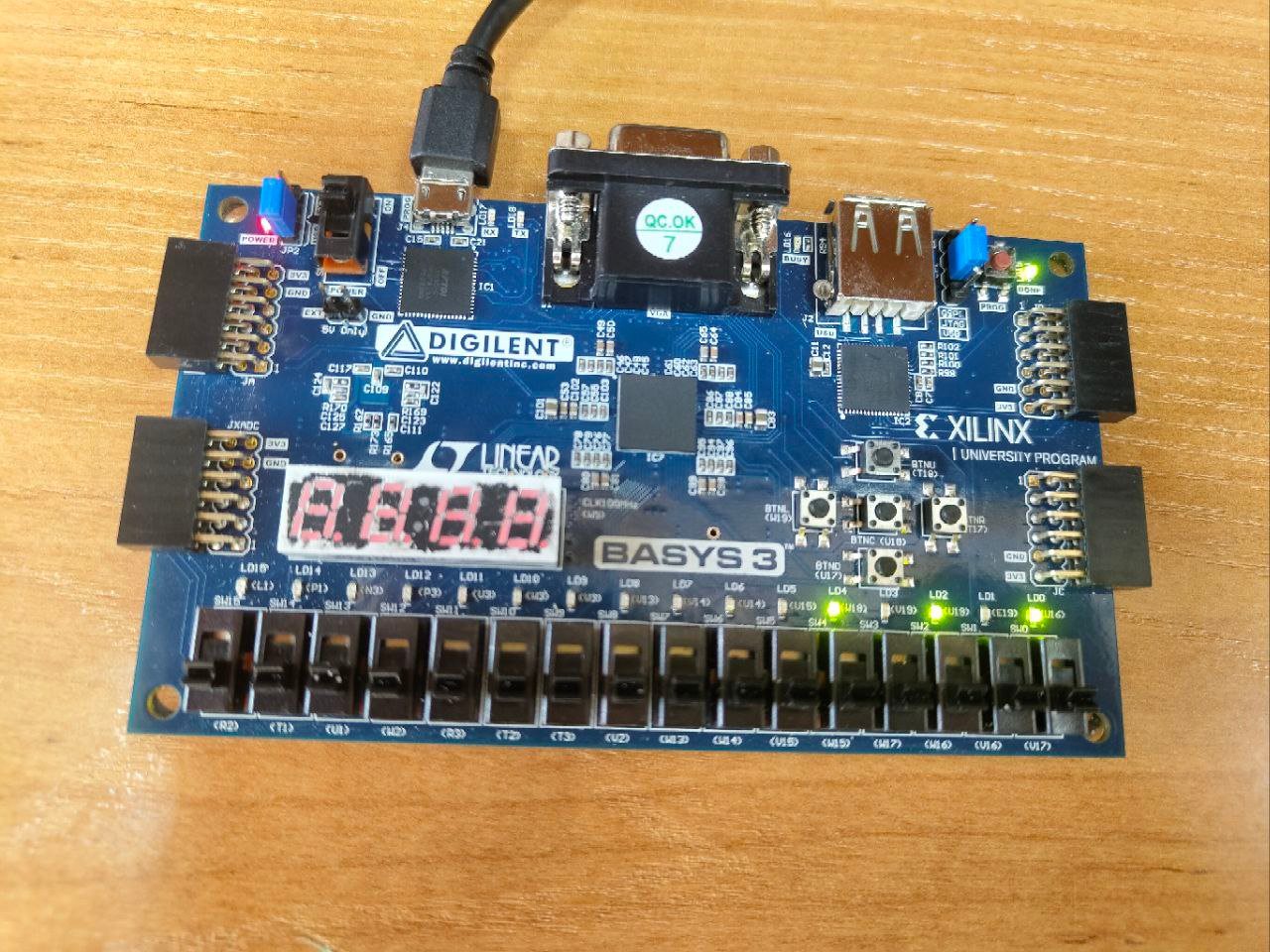


Рисунок 4 – Фотография макета



Рисунок 5 – Фотография макета

# Выводы

В ходе лабораторной работы №2 по дисциплине «Основы ПЛИС» была реализована последовательность свечения светодиодов на плате *Basys 3* согласно заданному алгоритму. Были рассчитаны необходимые задержки для смены комбинаций с учётом частоты тактового сигнала 100 МГц. Разработан код на *VHDL*, который успешно прошёл программную симуляцию и был проверен на реальной плате. Реализация корректно реагирует на нажатия кнопок «Пауза» и «Сброс», что подтверждает правильность работы алгоритма. Цель работы достигнута: освоен принцип работы светодиодов и кнопок на плате *Basys 3*, а также приобретены практические навыки разработки цифровых схем на языке *VHDL*.