**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**

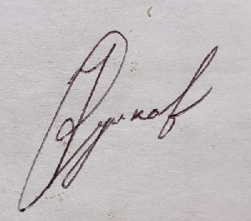
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

по дисциплине «Основы ПЛИС»



Выполнил:

студент гр. 1А22 21.10.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_ Кравцов О.К.



Проверил:

ассистент ОЭИ 21.10.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_ 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.

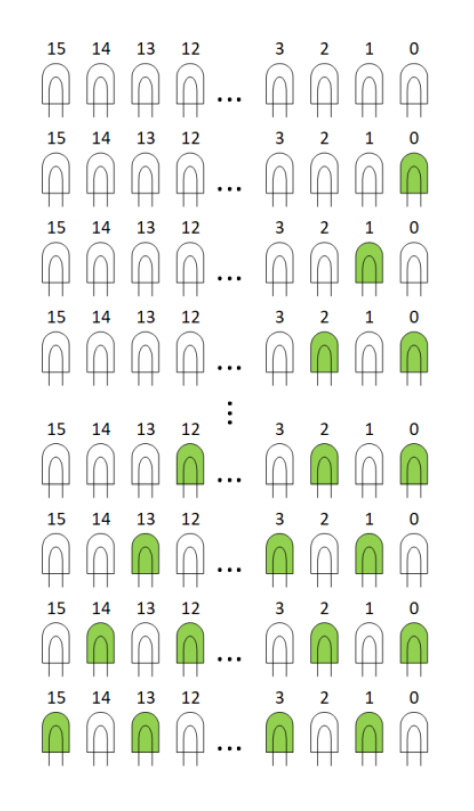


Рисунок 1 – Последовательность комбинаций свечения светодиодов

Блок схема разработанного алгоритма представлена на рисунке 2.

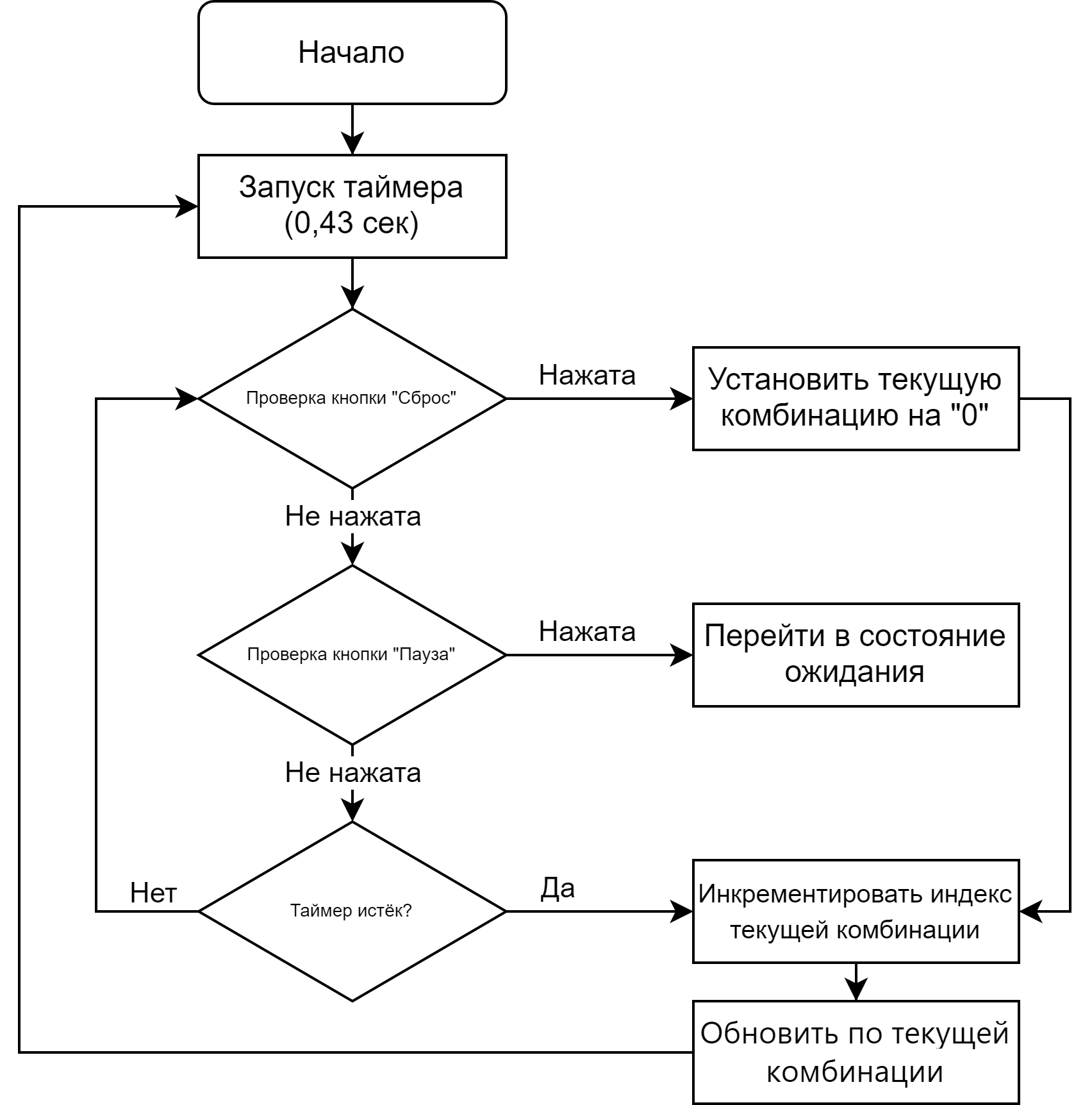


Рисунок 2 – Блок-схема алгоритма работы

# 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

(4-разрядный вектор).

**Работа модуля:**

1. При активации сброса (*reset = '1'*) все внутренние сигналы (*current\_index*, *counter*) и состояние *running* устанавливаются в начальные значения.

2. Основной процесс:

При каждом переднем фронте тактового сигнала (*rising\_edge(clk)*) происходит проверка состояния кнопки «Пауза».

Если модуль не на паузе (*running = '1'*), то счётчик *counter* инкрементируется на каждом такте. Когда значение счётчика достигает *DELAY* счётчик сбрасывается, и происходит переход к следующей комбинации светодиодов (инкрементируется текущий индекс комбинации).

Если *current\_index* превышает *MAX\_INDEX*, то он возвращается к нулю, обеспечивая циклическое повторение последовательности.

Таким образом, данный модуль обеспечивает последовательную смену комбинаций светодиодов с точной задержкой 0,43 секунды, а также позволяет управлять процессом смены с помощью кнопок «Пауза» и «Сброс».

**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 (3 downto 0)

    );

end led\_controller;

architecture Behavioral of led\_controller is

    signal current\_index : UNSIGNED(3 downto 0) := (others => '0');

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

    constant MAX\_INDEX   : UNSIGNED(3 downto 0) := "1111"; -- 16 combinations

    constant DELAY       : UNSIGNED(31 downto 0) := X"29020C0"; -- 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): 4-разрядный вектор, определяющий текущую комбинацию светодиодов.
* *leds* (выход): 16-разрядный вектор, управляющий состоянием светодиодов на плате.

**Работа модуля:**

В архитектуре модуля определён массив *patterns*, содержащий 16 предустановленных комбинаций свечения светодиодов. Каждая комбинация представлена 16-разрядным вектором, соответствующим определённому паттерну.

На основе входного сигнала *index* модуль выбирает соответствующий паттерн из массива *patterns* и выводит его на выход *leds*. Преобразование индекса из логического вектора в целочисленное значение осуществляется с помощью функции *to\_integer* и *unsigned*.

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

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity led\_pattern is

    Port (

        index   : in  STD\_LOGIC\_VECTOR(3 downto 0);

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

    );

end led\_pattern;

architecture Behavioral of led\_pattern is

    type led\_array is array(0 to 15) of STD\_LOGIC\_VECTOR(15 downto 0);

    constant patterns : led\_array := (

        "0000000000000000",  -- 0

        "0000000000000001",  -- 1

        "0000000000000010",  -- 2

        "0000000000000101",  -- 3

        "0000000000001010",  -- 4

        "0000000000101010",  -- 5

        "0000000001010101",  -- 6

        "0000000010101010",  -- 7

        "0000000101010101",  -- 8

        "0000001010101010",  -- 9

        "0000010101010101",  -- 10

        "0000101010101010",  -- 11

        "0001010101010101",  -- 12

        "0010101010101010",  -- 13

        "0101010101010101",  -- 14

        "1010101010101010"   -- 15

    );

begin

    leds <= patterns(to\_integer(unsigned(index)));

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 4-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(3 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;

# МОДЕЛИРОВАНИЕ

Моделирование кода даёт возможность проверить работу модулей не используя плату. Для проведения симуляции был написан тестовый файл.

Листинг тестового файла на языке *VHDL:*

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY top\_module\_tb IS

END top\_module\_tb;

ARCHITECTURE Behavioral OF top\_module\_tb IS

    -- Сигналы для подключения к top\_module

    SIGNAL clk        : STD\_LOGIC := '0';

    SIGNAL reset      : STD\_LOGIC := '0';

    SIGNAL pause      : STD\_LOGIC := '0';

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

    -- Константа для периода тактового сигнала (например, 10ns для 100MHz)

    CONSTANT clk\_period : TIME := 10 ns;

BEGIN

    -- Генерация тактового сигнала

    clk\_process : PROCESS

    BEGIN

        LOOP

            clk <= '0';

            WAIT FOR clk\_period / 2;

            clk <= '1';

            WAIT FOR clk\_period / 2;

        END LOOP;

    END PROCESS;

    -- Инстанциация тестируемого модуля

    uut: entity work.top\_module

        PORT MAP (

            clk        => clk,

            reset      => reset,

            pause      => pause,

            led\_output => led\_output

        );

-- Процесс для генерации сигналов reset и pause

    stim\_proc: PROCESS

    BEGIN

        -- Инициализация: reset и pause неактивны

        reset <= '0';

        pause <= '0';

        WAIT FOR 2 \* clk\_period;

        -- 1. Прогон всех 16 комбинаций (минимум 16 периодов)

        WAIT FOR 20 \* clk\_period;

        -- 2. Тестирование сигнала reset

        -- Активируем reset на один тактовый период

        reset <= '1';

        WAIT FOR clk\_period;

        reset <= '0';

        WAIT FOR 4 \* clk\_period;

        -- 3. Тестирование сигнала pause

        -- Активируем pause, после чего модуль должен перестать обновлять led\_output

        pause <= '1';

        WAIT FOR 10 \* clk\_period;

        pause <= '0';

        WAIT FOR 40 ns;

        -- Завершаем симуляцию

        WAIT;

    END PROCESS;

    -- Процесс для мониторинга выходов

    monitor\_proc: PROCESS(clk)

    BEGIN

        IF rising\_edge(clk) THEN

            REPORT "Time: " & TIME'image(now) &

                   " | Reset: " & STD\_LOGIC'image(reset) &

                   " | Pause: " & STD\_LOGIC'image(pause) &

                   " | LED Output: " & to\_hstring(led\_output);

        END IF;

    END PROCESS;

END Behavioral;

На рисунке 3 представлены скриншоты симуляции. Видно, что полученные сфазированные временные диаграммы состояний входов и выходов полностью соответствуют требуемой логике работы устройства.

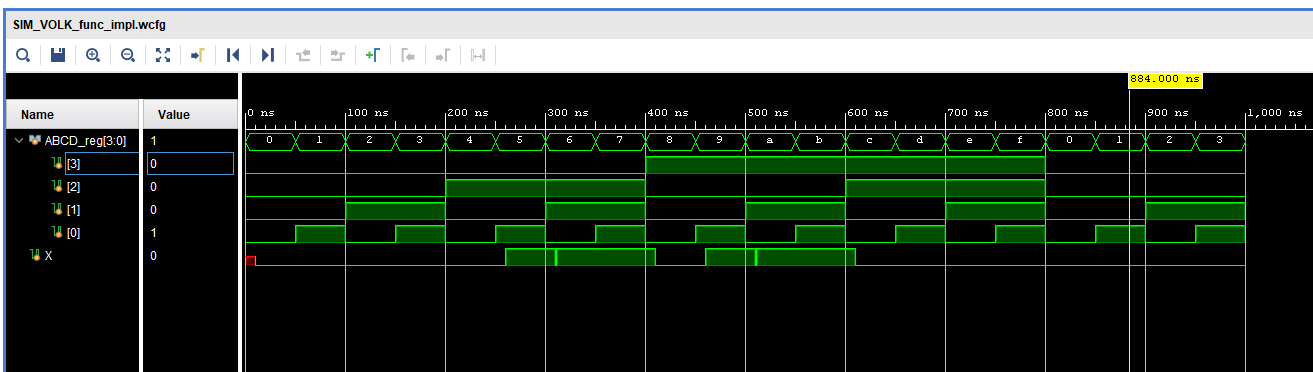


Рисунок 3 – Моделирование поведения модуля с учётом реальных задержек в кристалле

# НАзначение ножек 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 *T18*.
* Сигнал паузы *pause* подключен к кнопке "*Button Up*" на выводе *W19*.
* 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 T18 [get\_ports reset]

set\_property IOSTANDARD LVCMOS33 [get\_ports reset]

## Pause button (Button Up)

set\_property PACKAGE\_PIN W19 [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

Рисунок 6 – 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*.