Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Звіт

про виконання лабораторної роботи №2

з дисципліни: «Моделювання комп'ютерних систем»

На тему: «Структурний опис цифрового автомата. Перевірка роботи автомата за допомогою стенда Elbert V2 – Spartan 3A FPGA.»

Варіант - 4

Виконала:

ст.гр. КІ-202

Максимчук К.С.

Прийняв:

Козак Н.Б.

Львів 2023

Мета роботи: на базі стенда Elbert V2 – Spartan 3A FPGA реалізувати цифровий автомат світлових ефектів згідно вимог.

Завдання згідно з варіантом:

Варіант – 4:

- Пристрій повинен реалізувати 8 комбінацій вихідних сигналів згідно таблиці:

Стан#	LED_0	LED_1	LED_2	LED_3	LED_4	LED_5	LED_6	LED_7
0	1	0	0	0	0	0	0	1
1	0	1	0	0	0	0	1	0
2	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	0
4	0	0	1	1	1	1	0	0
5	0	1	1	1	1	1	1	0
6	1	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0

- Пристрій повинен використовувати *12MHz* тактовий сигнал від мікроконтролера *IC1* і знижувати частоту за допомогою внутрішнього подільника. Мікроконтролер *IC1* є частиною стенда *Elbert V2 – Spartan 3A FPGA*. Тактовий сигнал заведено нв вхід *LOC = P129 FPGA* (див. ***Додаток – 1***).
- Інтерфейс пристрою повинен мати вхід синхронного скидання (*RESET*).
- Інтерфейс пристрою повинен мати вхід керування режимом роботи (*MODE*):
  - Якщо *MODE=0* то стан пристрою інкрементується по зростаючому фронту тактового сигналу пам'яті станів (0->1->2->3->4->5->6->7->0…).
  - Якщо *MODE=1* то стан пристрою декрементується по зростаючому фронту тактового сигналу пам'яті станів (0->7->6->5->4->3->2->1->0…).
- Інтерфейс пристрою повинен мати однорозрядний вхід керування швидкістю роботи(*SPEED*):
  - Якщо *SPEED=0* то автомат працює зі швидкістю, визначеною за замовчуванням.
  - Якщо *SPEED=1* то автомат працює зі швидкістю, В 4 РАЗИ НИЖЧОЮ ніж в режимі (*SPEED= 0*).
- Для керування сигналом *MODE* використати будь який з 8 *DIP* перемикачів (див. **Додаток – 1**).
- Для керування сигналами *RESET/SPEED* використати будь якІ з *PUSH BUTTON* кнопок (див. **Додаток – 1**).

Виконання роботи:

1. Створив проект.

2. Додав VHDL Module для імплементації логіки формування вихідних сигналів.

Код OutputLogic:

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

entity out_logic_intf is2
   Port ( IN_BUS : in  std_logic_vector(2 downto 0);
        OUT_BUS : out  std_logic_vector(7 downto 0)
                    );
end out_logic_intf;

architecture out_logic_arch of out_logic_intf is

begin

    OUT_BUS(0) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and not(IN_BUS(0))) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
    OUT_BUS(1) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0)) or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0))  or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
    OUT_BUS(2) <= ((not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0))) or (IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0)))  or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0))  or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
    OUT_BUS(3) <= ((not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0))  or (IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0)))  or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0))  or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
    OUT_BUS(4) <= ((not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0))  or (IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0)))  or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0))  or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
    OUT_BUS(5) <= ((not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0))) or (IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0)))  or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0))  or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));
```

OUT_BUS(6) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0)) or (IN_BUS(2) and not (IN_BUS(1)) and IN_BUS(0)) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));

OUT_BUS(7) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and not(IN_BUS(0))) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))));

end out_logic_arch;

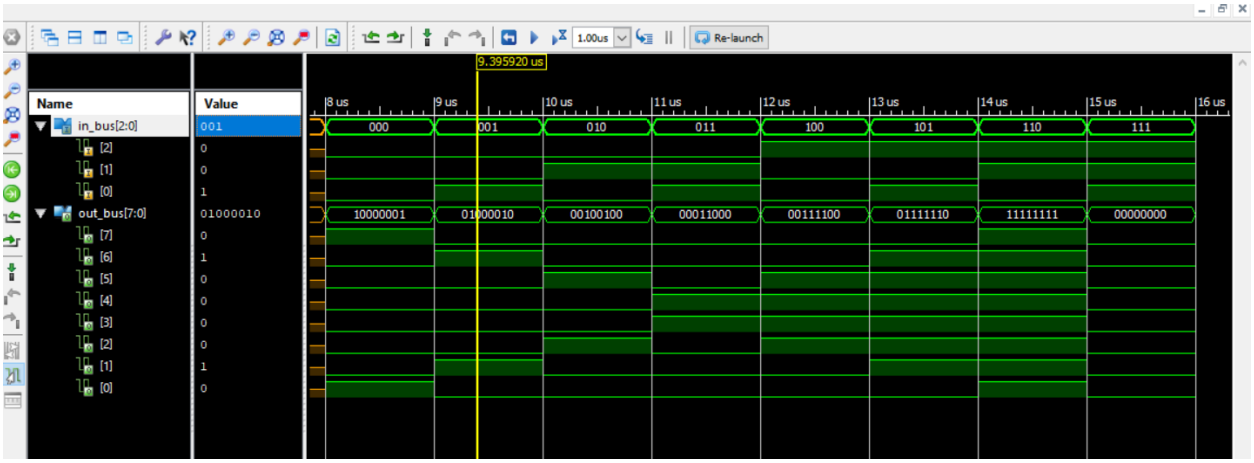


Рис. 1 Діаграма проведеної симуляції для OutputLogic

3. Додав VHDL Module для імплементації логіки формування переходів.

Код TransitionLogic:

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

entity transition_logic_intf is
   Port ( CUR_STATE : in  std_logic_vector(2 downto 0);
          MODE : in  std_logic;
          RES : in  std_logic;
          NEXT_STATE : out  std_logic_vector(2 downto 0)
 );
end transition_logic_intf;

architecture transition_logic_arch of transition_logic_intf is
```

```
begin

NEXT_STATE(0) <= (not(RES) and not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or  -- 000 -> 001
  (not(RES) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and not(CUR_STATE(0))) or  -- 010 -> 011
  (not(RES) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or  -- 100 -> 101
  (not(RES) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or   -- 110 -> 111
  (not(RES) and MODE and not(CUR_STATE(2)) and (CUR_STATE(1)) and not(CUR_STATE(0))) or    -- 001 <- 010
  (not(RES) and MODE and (CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or    -- 011 <- 100
  (not(RES) and MODE and CUR_STATE(2) and (CUR_STATE(1)) and not(CUR_STATE(0))) or   -- 101 <- 110
  (not(RES) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));  -- 111 <- 000


NEXT_STATE(1) <=  (not(RES) and not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and CUR_STATE(0)) or  -- 001 -> 010
  (not(RES) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and not(CUR_STATE(0))) or  -- 010 -> 011
  (not(RES) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or   -- 101 -> 110
  (not(RES) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or   -- 110 -> 111
  (not(RES) and MODE and not(CUR_STATE(2)) and (CUR_STATE(1)) and (CUR_STATE(0))) or   -- 010 <- 011
  (not(RES) and MODE and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or   -- 011 <- 100
  (not(RES) and MODE and CUR_STATE(2) and CUR_STATE(1) and CUR_STATE(0)) or    -- 110 <- 111
  (not(RES) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));  -- 111 <- 000


NEXT_STATE(2) <=  (not(RES) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and CUR_STATE(0)) or   -- 011 -> 100
  (not(RES) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or  -- 100 -> 101
  (not(RES) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or   -- 101 -> 110
  (not(RES) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or   -- 110 -> 111
  (not(RES) and MODE and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or   -- 100 <- 101
```

```
(not(RES) and MODE and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or   -- 101 <- 110
(not(RES) and MODE and CUR_STATE(2) and CUR_STATE(1) and CUR_STATE(0)) or    -- 110 <- 111
(not(RES) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));   -- 111 <- 000

end transition_logic_arch;
```

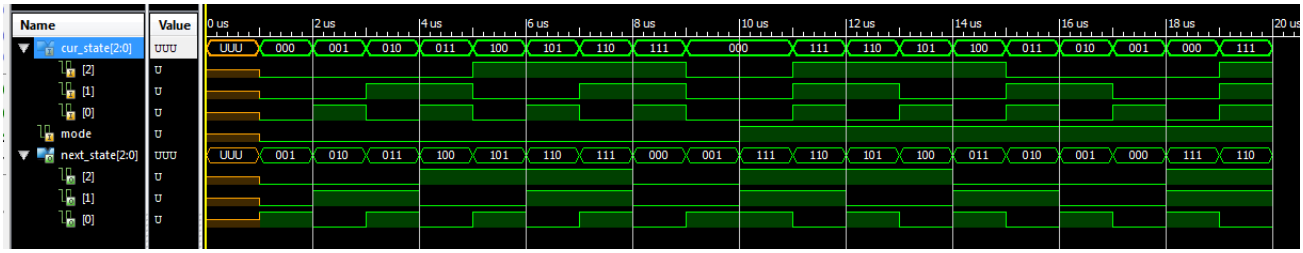


Рис. 2 Діаграма проведеної симуляції для TransitionLogic

4. Згенерував Schematic файли для OutputLogic та TransitionLogic.

5. Створив Schematic файл LightController, реалізував в ньому пам’ять стану автомата та зв’язав між собою всі його частини.

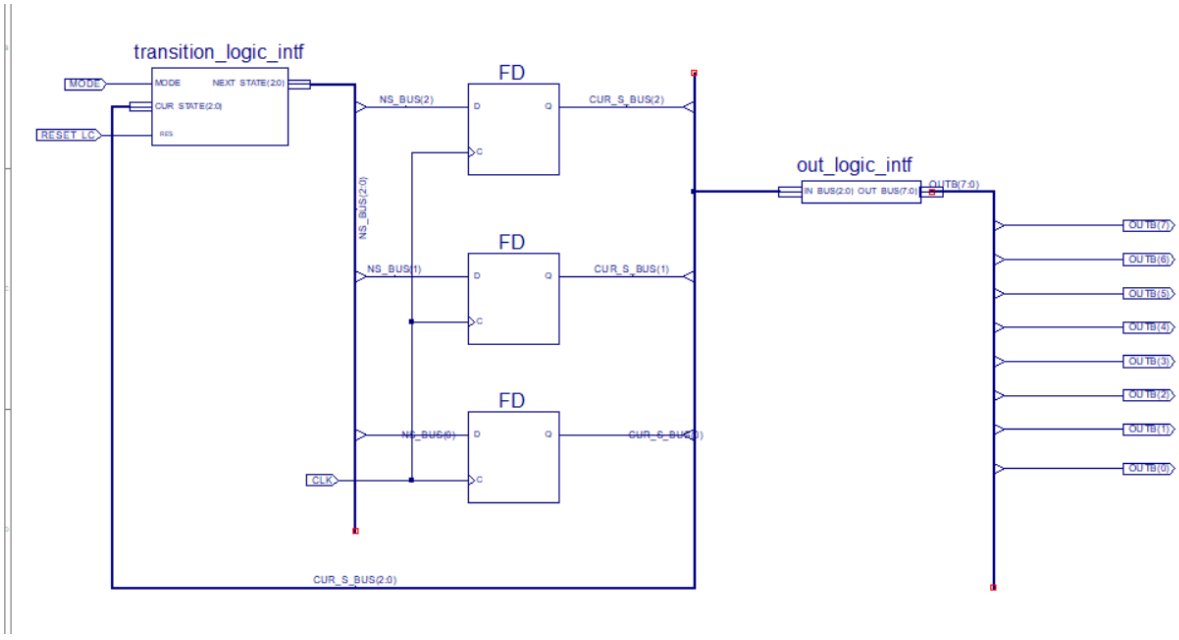


Рис. 3 Схема LightController

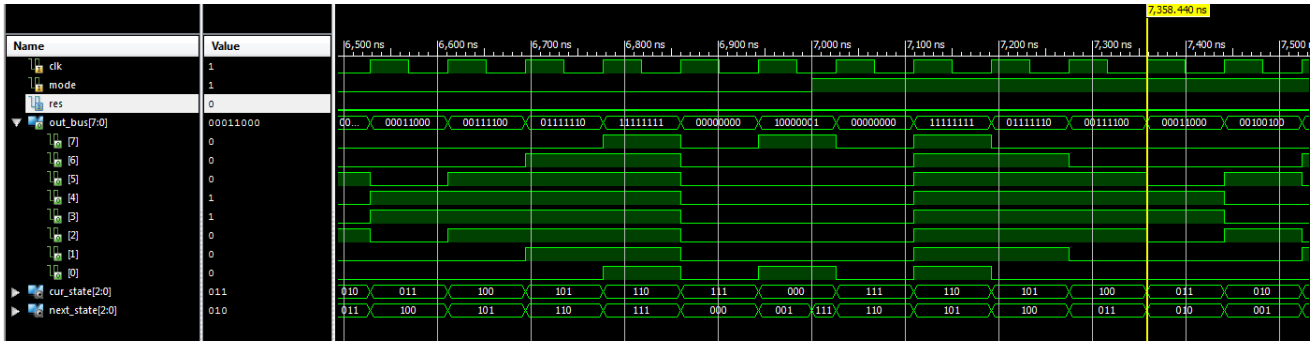


Рис. 4 Діаграма проведеної симуляції для LightController

6. Створив Schematic файл TopLevel, в якому реалізував подільник вхідної частоти та інтегрував його зі створеним автоматом, попередньо згенерувавши для нього Schematic файл.

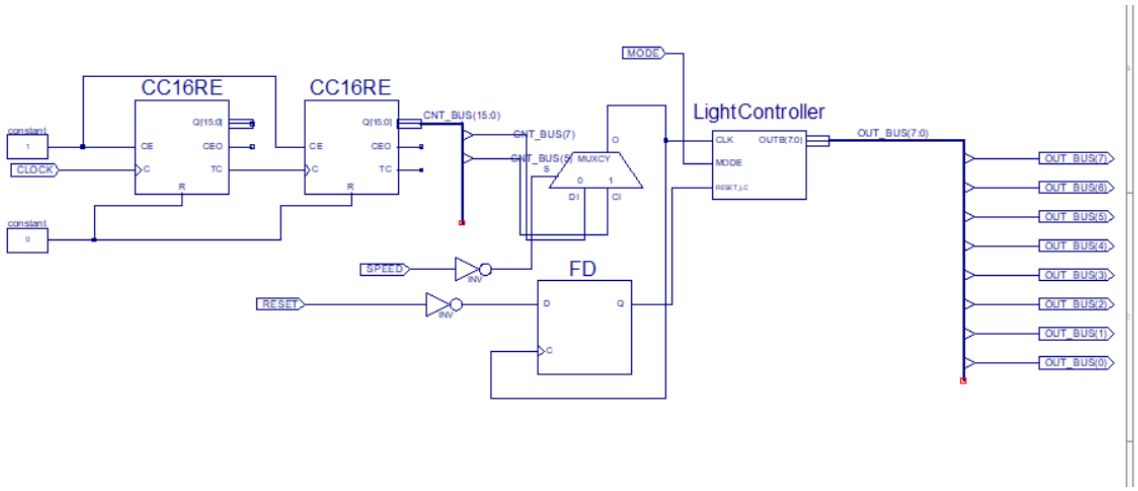


Рис. 5 Схема TopLevel

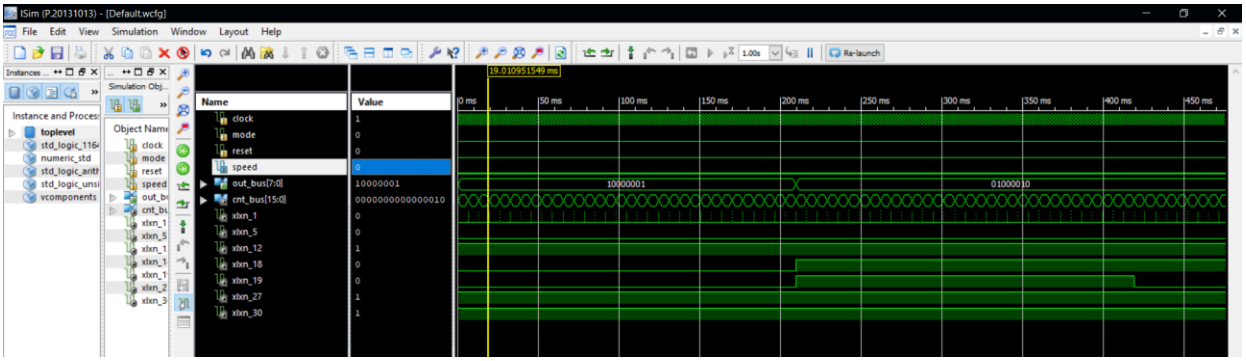


Рис. 6 Діаграма проведеної симуляції для TopLevel

7. Додав файл Constraints.ucf та призначив виводам схеми фізичної виводи FPGA.

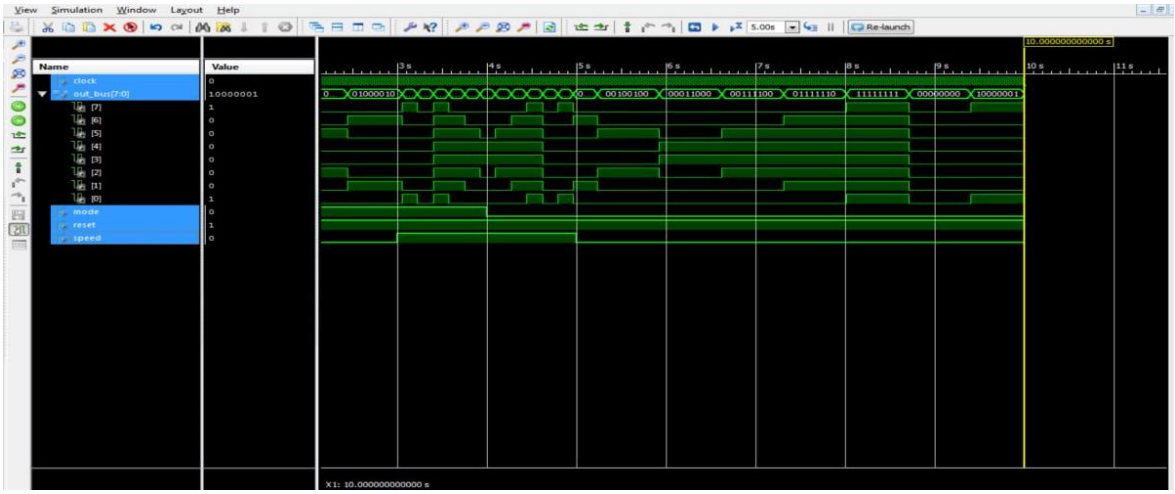
```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
# This file is a .ucf for ElbertV2 Development Board                                                            #
# To use it in your project :                                                                                   #
# * Remove or comment the lines corresponding to unused pins in the project                                     #
# * Rename the used signals according to the your project                                                       #
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#

#**************************************************************************************************************#
#                                        UCF for ElbertV2 Development Board                                     #
#**************************************************************************************************************#
CONFIG VCCAUX = "3.3" ;

 # Clock 12 MHz
  NET "CLOCK"               LOC = P129  | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;

##############################################################################################################
#######
#                                LED
##############################################################################################################
#######

   NET "OUT_BUS(0)"          LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "OUT_BUS(1)"          LOC = P47   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "OUT_BUS(2)"          LOC = P48   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "OUT_BUS(3)"          LOC = P49   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
          NET "OUT_BUS(4)"           LOC = P50   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE =
12;
   NET "OUT_BUS(5)"          LOC = P51   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "OUT_BUS(6)"          LOC = P54   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "OUT_BUS(7)"          LOC = P55   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##############################################################################################################
#######
#                             DP Switches
##############################################################################################################
#######

   NET "MODE"       LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##############################################################################################################
#######
#                              Switches
##############################################################################################################
#######

          NET "RESET"       LOC = P80   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE
= 12;
```

```
NET "SPEED"        LOC = P79   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

TestBench:

```
-- Vhdl test bench created from schematic C:\rubbish\MKS\2\Telegram Desktop\Lab_2\Lab_2\TopLevel.sch - Fri Apr 28 09:57:47 2023
--
-- Notes:
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY TopLevel_TopLevel_sch_tb IS
END TopLevel_TopLevel_sch_tb;
ARCHITECTURE behavioral OF TopLevel_TopLevel_sch_tb IS

  COMPONENT TopLevel
  PORT( CLOCK  :  IN  STD_LOGIC;
        OUT_BUS  :  OUT  STD_LOGIC_VECTOR (7 DOWNTO 0);
        MODE  :  IN  STD_LOGIC;
        RESET  :  IN  STD_LOGIC;
        SPEED  :  IN  STD_LOGIC);
  END COMPONENT;

  SIGNAL CLOCK  :  STD_LOGIC := '0';
  SIGNAL OUT_BUS  :  STD_LOGIC_VECTOR (7 DOWNTO 0);
  SIGNAL MODE  :  STD_LOGIC;
  SIGNAL RESET  :  STD_LOGIC;
  SIGNAL SPEED  :  STD_LOGIC;

BEGIN

  UUT: TopLevel PORT MAP(
   CLOCK => CLOCK,
   OUT_BUS => OUT_BUS,
   MODE => MODE,
   RESET => RESET,
   SPEED => SPEED
  );

-- *** Test Bench - User Defined Section ***
```

```
   tb : PROCESS
BEGIN
   -- Apply input stimuli to the design
   MODE <= '0';
   RESET <= '0';
   SPEED <= '0';
   wait for 1000 ms;

   MODE <= '1';
   RESET <= '1';
   SPEED <= '1';

   wait for 1000 ms;

   MODE <= '1';
   SPEED <= '0';

   wait for 1000 ms;

  MODE <= '1';
   SPEED <= '1';

   wait for 1000 ms;

   MODE <= '0';
   SPEED <= '1';

   wait for 1000 ms;

  MODE <= '0';
   SPEED <= '0';

   wait for 1000 ms;



   -- Add output check statements to verify the expected output
   --assert OUT_BUS = "00000000" report "Test failed: Output does not match expected value" severity error;

   WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***


  tb_clk : PROCESS
BEGIN

  CLOCK <= not CLOCK;
  wait for 83 ns;

END PROCESS;

END;
```



8. Згенерував бінарний файл для цифрового автомата світлових ефектів.

Висновок: виконавши лабораторну роботу, здобуто навики реалізації цифрових автоматів світлових ефектів.