Міністерство освіти і науки України
Національний університет „Львівська політехніка”
Кафедра ЕОМ

Звіт
з лабораторної роботи №2
з дисципліни: “Моделювання комп’ютерних систем”
на тему: “Структурний опис цифрового автомата”

Виконав:
ст. гр. КІ-201
Нескромний Д.П.

Прийняв:
Козак Н.Б.

Львів 2023

Мета: **“**На базі стенда реалізувати цифровий автомат світлових ефектів**”.**
Варіант 4
Завдання:

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:

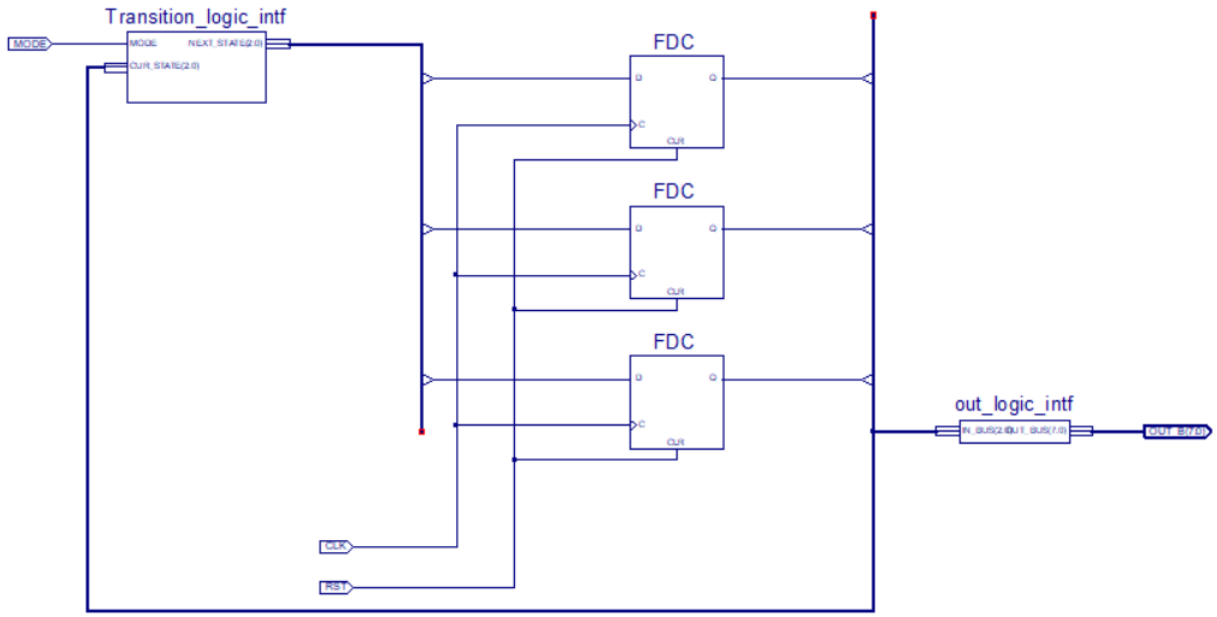
1. Створюю OutputLogic.vhd

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

entity out_logic_intf is
    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;
```

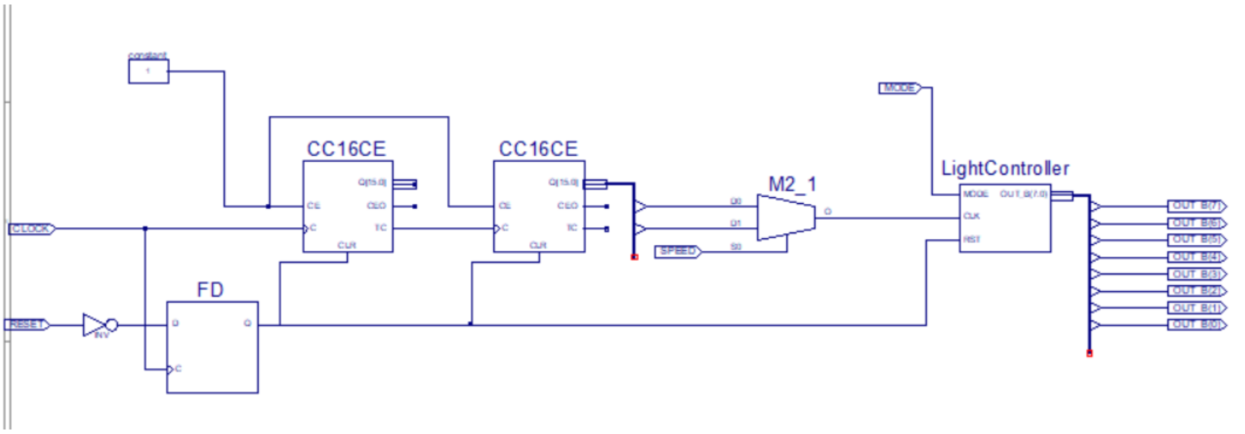
2. Створюю TransitionLogic.vhd

```
32  entity Transition_logic_intf is
33      Port ( CUR_STATE : in  std_logic_vector(2 downto 0);
34             MODE : in  std_logic;
35             NEXT_STATE : out  std_logic_vector(2 downto 0)
36           );
37  end transition_logic_intf;
38
39  architecture transition_logic_arch of transition_logic_intf is
40
41  begin
42
43     NEXT_STATE(0) <= (not(MODE) and not(CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
44                 (not(MODE) and not(CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
45                 (not(MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
46                 (not(MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
47                 (    (MODE) and not(CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
48                 (    (MODE) and not(CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
49                 (    (MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
50                 (    (MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0)));
51
52     NEXT_STATE(1) <= (not(MODE) and not(CUR_STATE(2)) and not (CUR_STATE(1)) and     (CUR_STATE(0))) or
53                 (not(MODE) and not(CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
54                 (not(MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and     (CUR_STATE(0))) or
55                 (not(MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
56                 (    (MODE) and not(CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
57                 (    (MODE) and not(CUR_STATE(2)) and     (CUR_STATE(1)) and     (CUR_STATE(0))) or
58                 (    (MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
59                 (    (MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and     (CUR_STATE(0)));
60
61     NEXT_STATE(2) <= (not(MODE) and not(CUR_STATE(2)) and     (CUR_STATE(1)) and     (CUR_STATE(0))) or
62                 (not(MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
63                 (not(MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and     (CUR_STATE(0))) or
64                 (not(MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
65                 (    (MODE) and not(CUR_STATE(2)) and not (CUR_STATE(1)) and not (CUR_STATE(0))) or
66                 (    (MODE) and     (CUR_STATE(2)) and not (CUR_STATE(1)) and     (CUR_STATE(0))) or
67                 (    (MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and not (CUR_STATE(0))) or
68                 (    (MODE) and     (CUR_STATE(2)) and     (CUR_STATE(1)) and     (CUR_STATE(0)));
69
70  end transition_logic_arch;
```

3. Створюю схему LightController.sch



4. Створюю схему TopLevel.sch

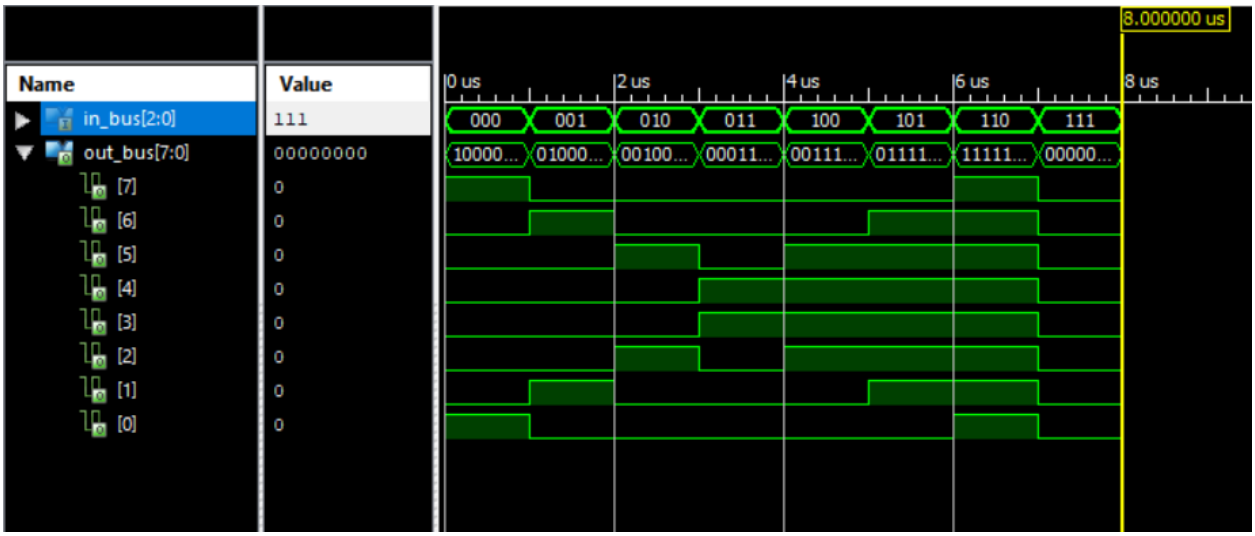


Щоб забезпечити можливість зменшення вхідної частоти в чотири рази, додаю мультиплексор.

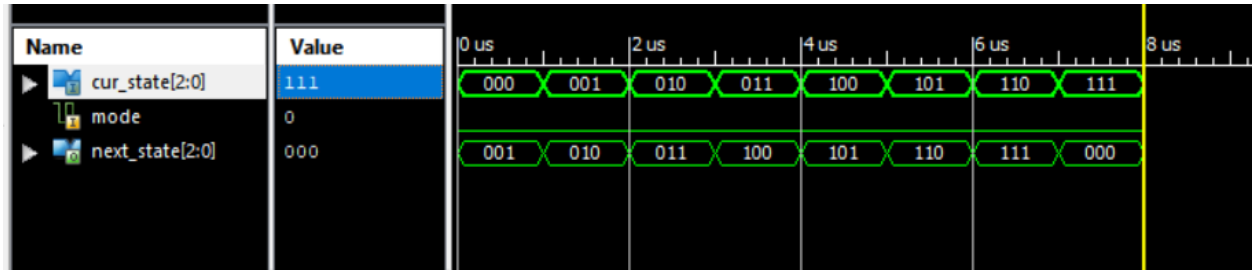
5. Додаю файл Constraints.ucf

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

```

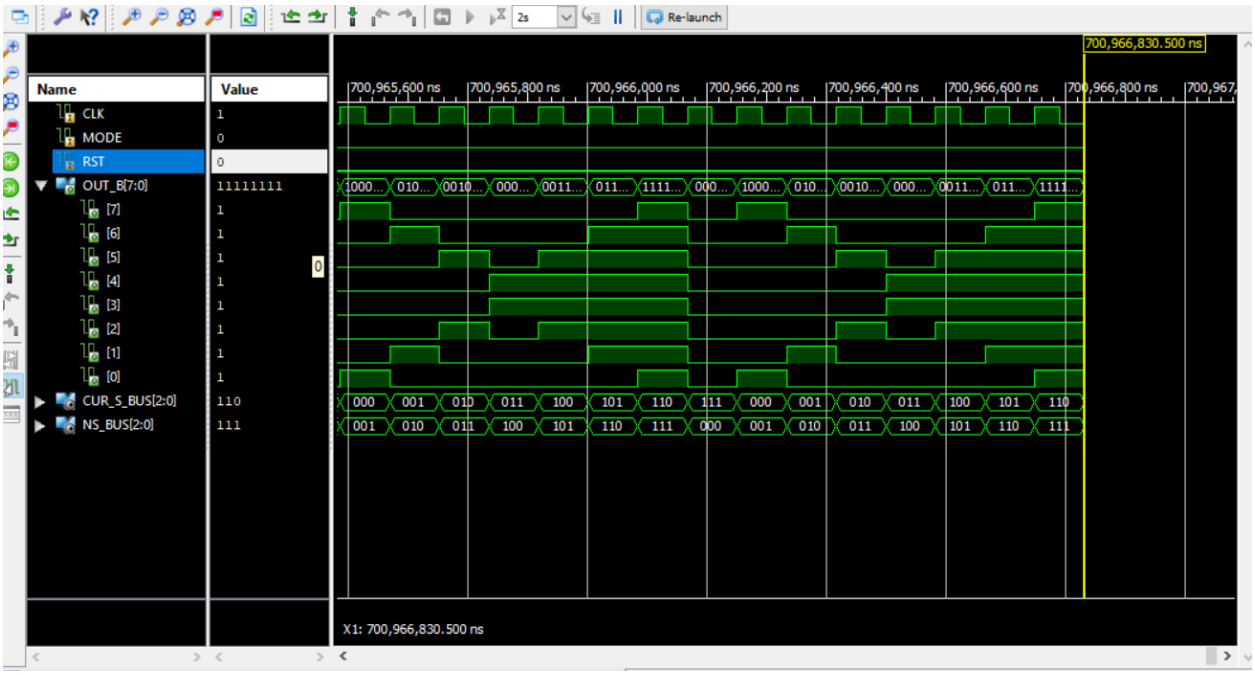
6. Симулюю роботу OutputLogic :



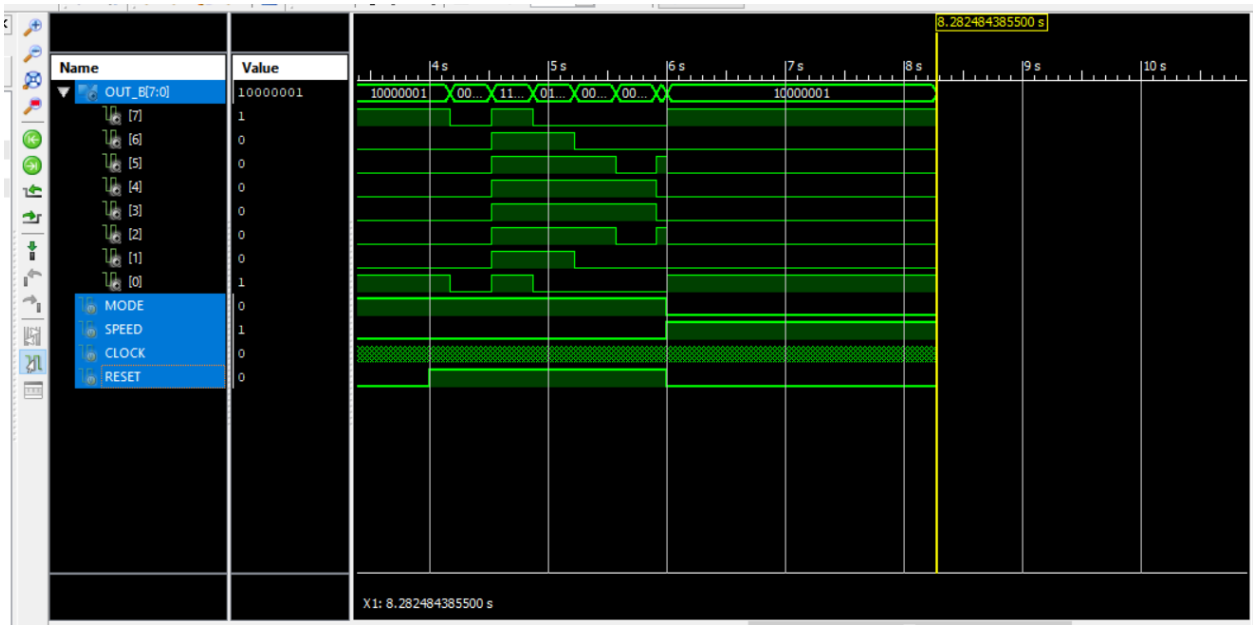
7. Симулюю роботу TransitionLogic :



8. Симулюю роботу LightController.sch:



9. Симулюю роботу TopLevel.sch:



10. Генерую bin файл

sch2HdlBatchFile	14.05.2023 21:43	Файл	0 КБ
TestBench.v	29.04.2023 23:36	Файл V	2 КБ
toplevel.bgn	14.05.2023 21:43	Файл BGN	7 КБ
toplevel.bit	14.05.2023 21:43	Файл BIT	54 КБ
TopLevel.bld	14.05.2023 21:43	Файл BLD	2 КБ
TopLevel.cmd_log	14.05.2023 21:43	Файл CMD_LOG	1 КБ
toplevel.drc	14.05.2023 21:43	Файл DRC	1 КБ

Висновок:На даній лабораторній роботі я на базі стенда Elbert V2- Spartan 3A FPGA реалізував цифровий автомат світлових ефектів. Навчився створювати нові елементи і описувати логіку їх роботи засобами VHDL.