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
Грицай В.О.

Прийняв:
Козак Н.Б.

Львів 2023

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

Завдання до варіанту № 5:

Варіант – 5:

- 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	0
1	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0
4	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	1	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 (*TEST*) для подачі логічної «1» на всі виходи одночасно:
  - Якщо *TEST=0* то автомат перемикає сигнали на виходах згідно заданого алгоритму.
  - Якщо *TEST=1* то на всіх виходах повинна бути логічна «1» (всі *LED* увімкнені).
- Для керування сигналом *MODE* використати будь який з 8 *DIP* перемикачів (див. **Додаток** – 1).
- Для керування сигналами *RESET/TEST* використати будь якІ з *PUSH BUTTON* кнопок (див. **Додаток** – 1).

Хід виконання:

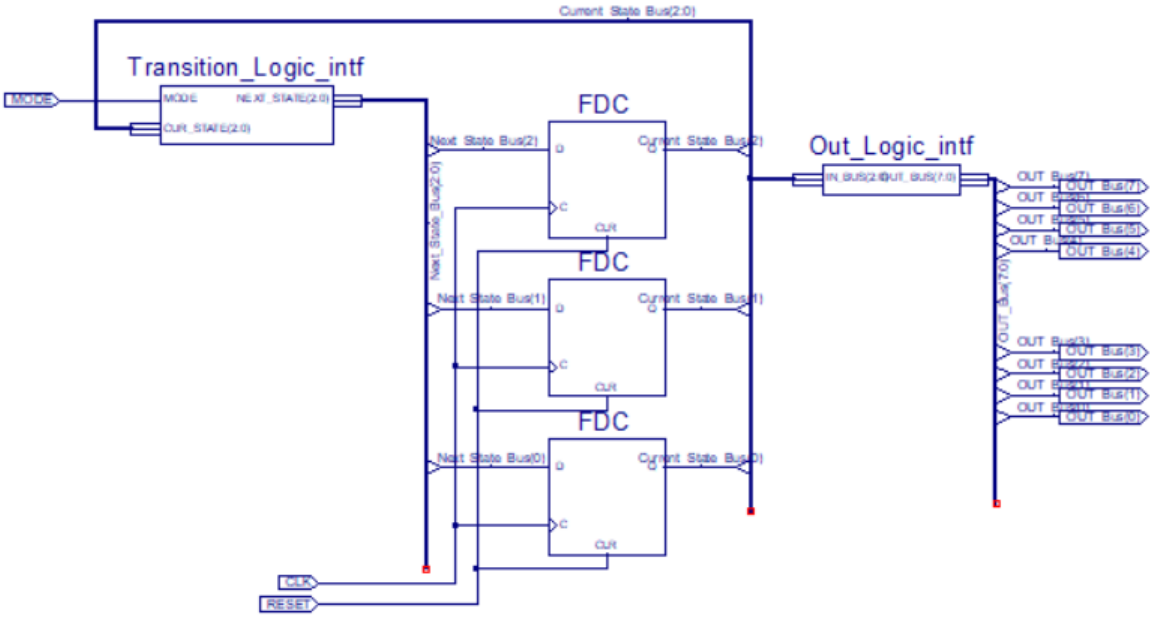
1) Створюю TransitionLogic.vhd

```
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;
        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) <= (MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and not(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and not(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0)));

    NEXT_STATE(1) <= (MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and (CUR_STATE(1)) and(CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and not (CUR_STATE(2)) and (CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and not(CUR_STATE(2)) and (CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and not(CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0)));

    NEXT_STATE(2) <= (MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and (CUR_STATE(1)) and(CUR_STATE(0))) or
                     (MODE and(CUR_STATE(2)) and (CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (MODE and (CUR_STATE(2)) and not(CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and not(CUR_STATE(2)) and(CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and not(CUR_STATE(1)) and not (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and not(CUR_STATE(1)) and (CUR_STATE(0))) or
                     (not(MODE) and(CUR_STATE(2)) and(CUR_STATE(1)) and not (CUR_STATE(0)));

end Transition_Logic_arch;
```

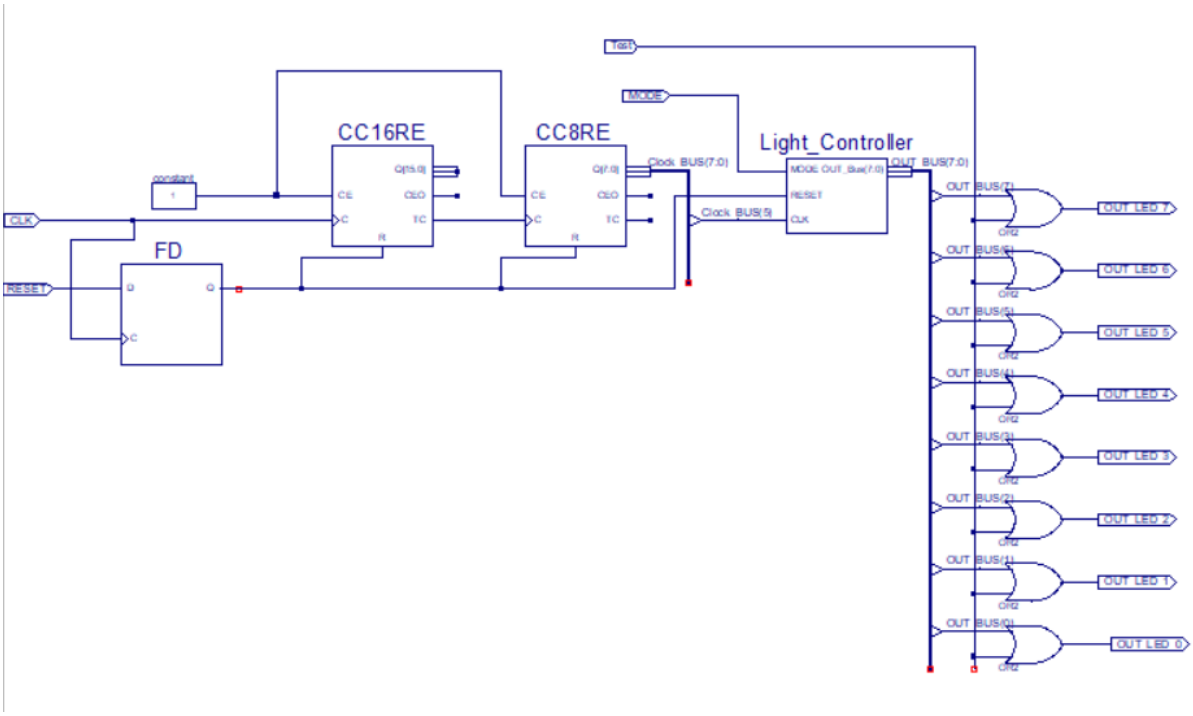
2) Створюю OutputLogic.vhd

```
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)));
    OUT_BUS(1) <= (not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0)));
    OUT_BUS(2) <= (IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0)));
    OUT_BUS(3) <= (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0)));
    OUT_BUS(4) <= (IN_BUS(2) and IN_BUS(1) and IN_BUS(0));
    OUT_BUS(5) <= (IN_BUS(2) and not(IN_BUS(1)) and IN_BUS(0));
    OUT_BUS(6) <= (not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0));
    OUT_BUS(7) <= (not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0));

end Out_Logic_arch;
```

3) Створюю схему LightController.sch



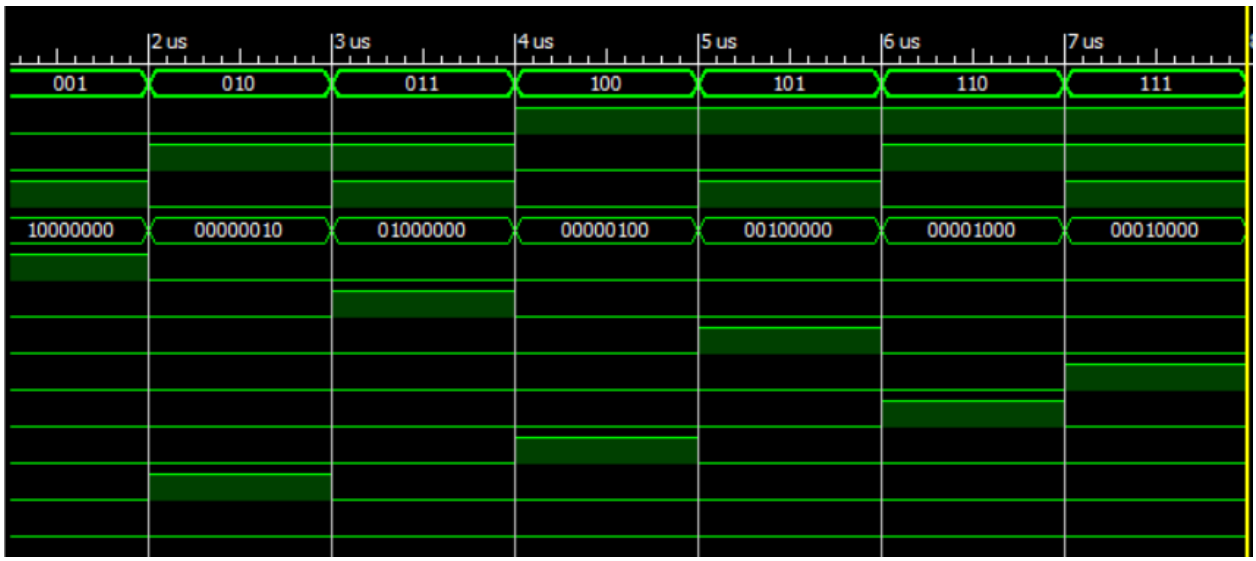
4) Створюю файл TopLevel.sch



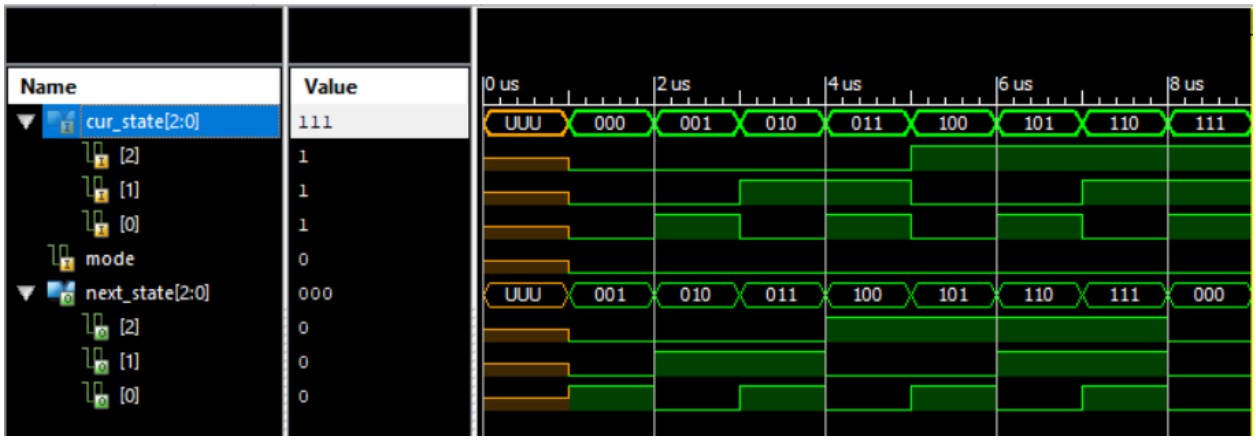
5) Додаю Constraints.ucf файл

```
CONFIG VCCAUX = "3.3" ;

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

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

    NET "OUT_LED_0"             LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_1"             LOC = P47   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_2"             LOC = P48   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_3"             LOC = P49   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_4"             LOC = P50   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_5"             LOC = P51   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_6"             LOC = P54   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "OUT_LED_7"             LOC = P55   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

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

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

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

    NET "Test"          LOC = P80   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "RESET"         LOC = P79   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##########################################################################################################

```

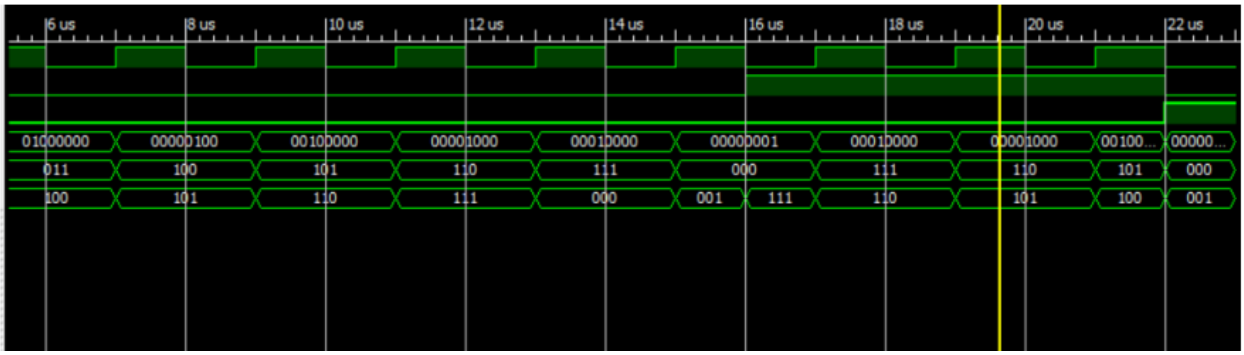
6) Симулюю роботу OutputLogic :



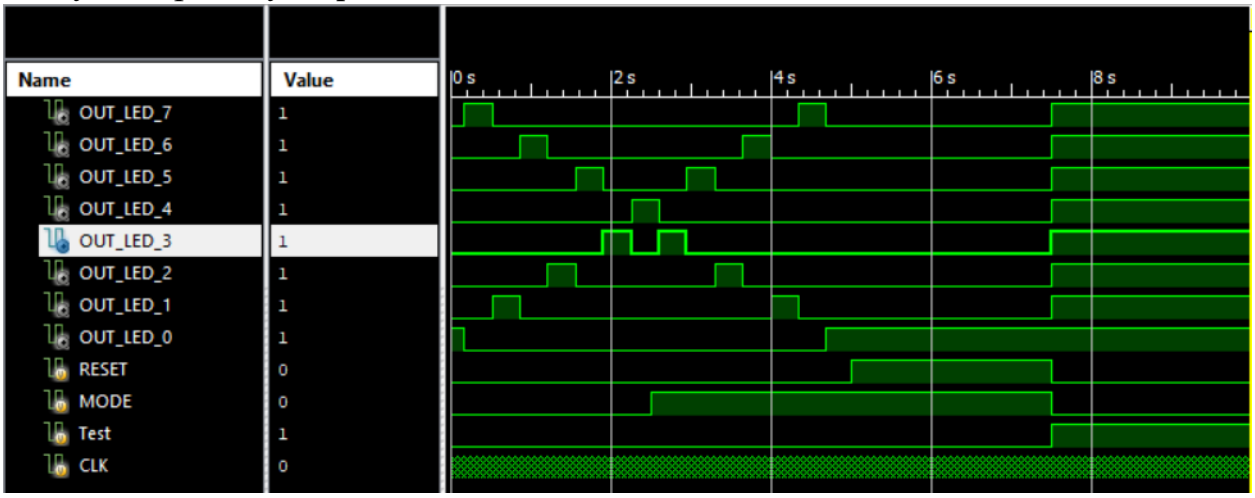
7) Симулюю роботу TransitionLogic :



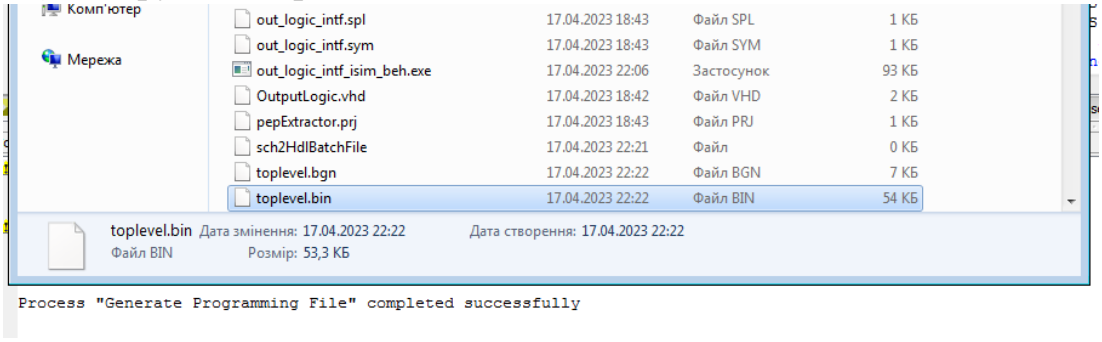
8) Симулюю роботу LightController.sch :



9) Симулюю роботу TopLevel.sch :



10) Генерую BIN файл :




```
Process "Generate Programming File" completed successfully
```

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.