

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
Кісіль В. І.

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 на 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_Logics_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_Logics_intf;

architecture Transition_Logics_arch of Transition_Logics_intf is

begin
    NEXT_STATE(0) <= (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))) or
                    ((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)));

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

    NEXT_STATE(2) <= (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

```

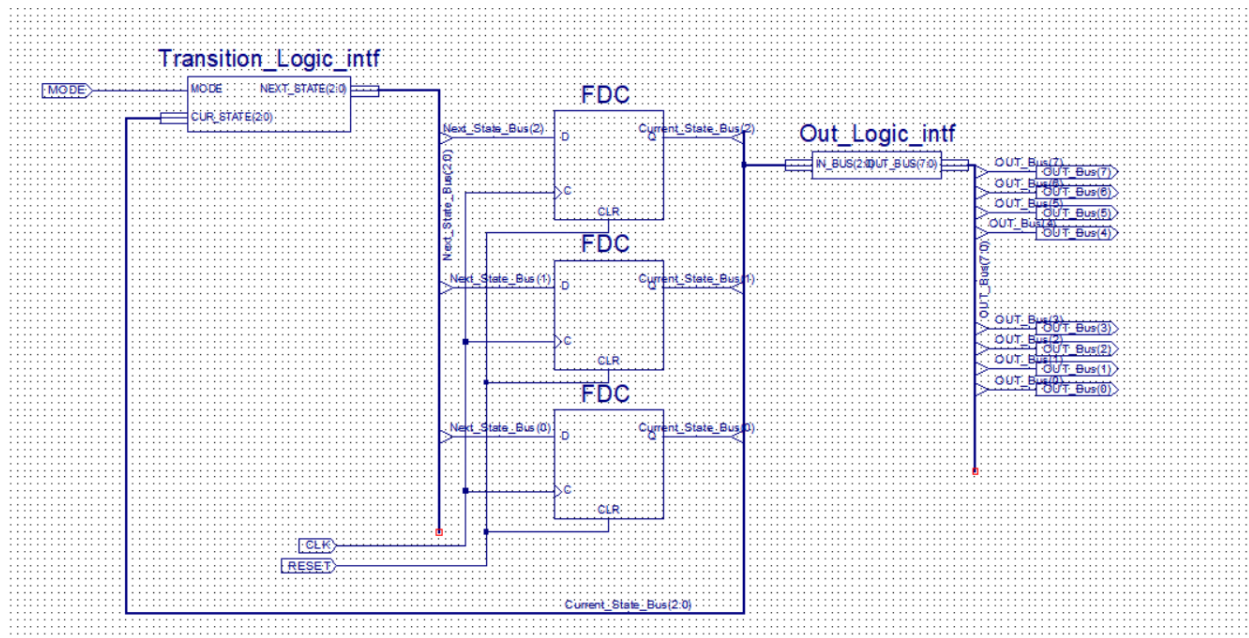
2) Створюю OutputLogic.vhd

```

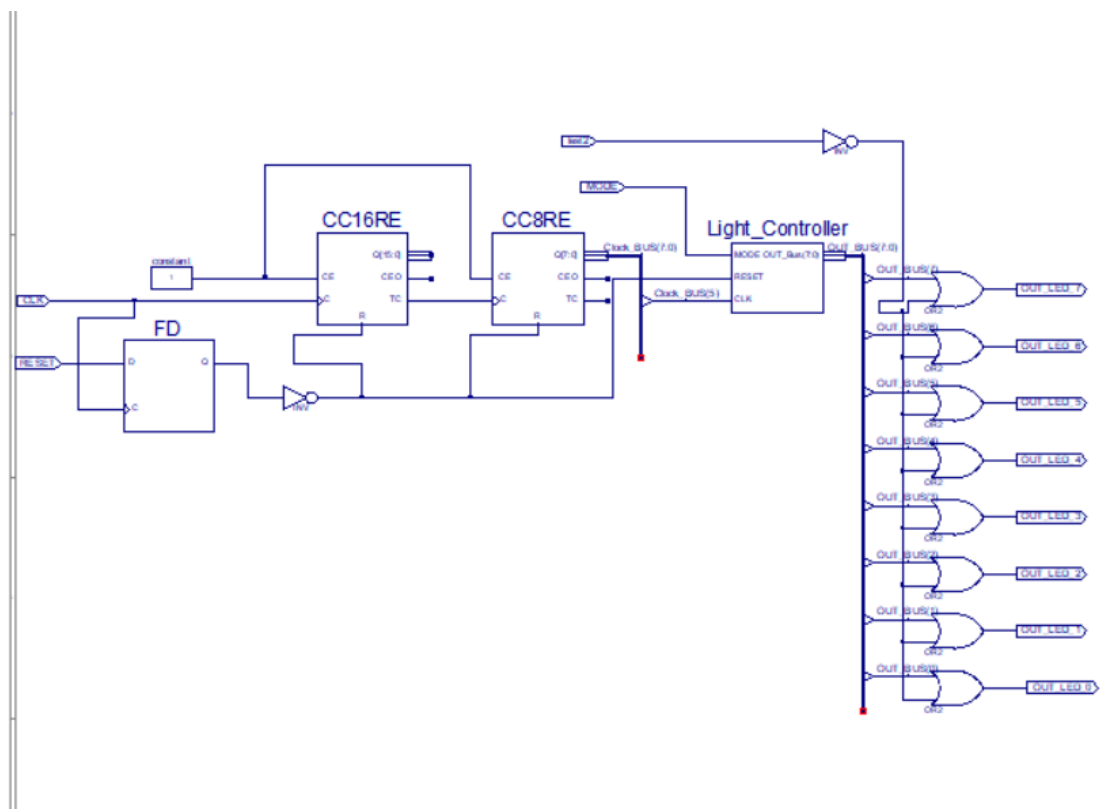
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23
24 entity Out_Logic_intf is
25 Port ( IN_BUS : in  std_logic_vector(2 downto 0);
26        OUT_BUS : out std_logic_vector(7 downto 0)
27        );
28 end Out_Logic_intf;
29
30 architecture Out_Logic_arch of Out_Logic_intf is
31
32 begin
33     OUT_BUS(0) <= (not(IN_BUS(2)) and not(IN_BUS(1)) and not(IN_BUS(0)));
34     OUT_BUS(2) <= (not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0));
35     OUT_BUS(4) <= (not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0)));
36     OUT_BUS(6) <= (not(IN_BUS(2)) and IN_BUS(1) and(IN_BUS(0)));
37     OUT_BUS(7) <= ((IN_BUS(2)) and not IN_BUS(1) and not(IN_BUS(0)));
38     OUT_BUS(5) <= ((IN_BUS(2)) and not IN_BUS(1) and(IN_BUS(0)));
39     OUT_BUS(3) <= ((IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0)));
40     OUT_BUS(1) <= ((IN_BUS(2)) and IN_BUS(1) and(IN_BUS(0)));
41
42 end Out_Logic_arch;

```

3) Створюю схему LightController.sch



4) Створюю файл TopLevel.sch



Також, щоб забезпечити можливість зменшення вхідної частоти в чотири рази – умова завдання, додаю мультіплексор.

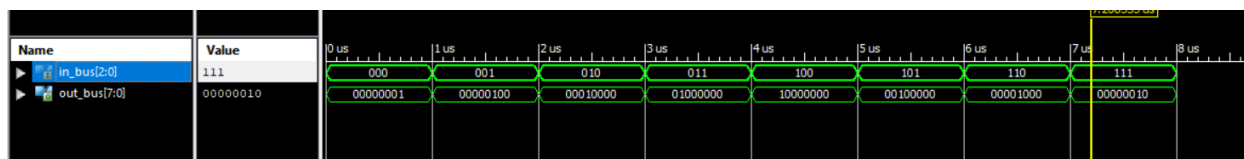
5) Добавлю Constraints.ucf файл

```

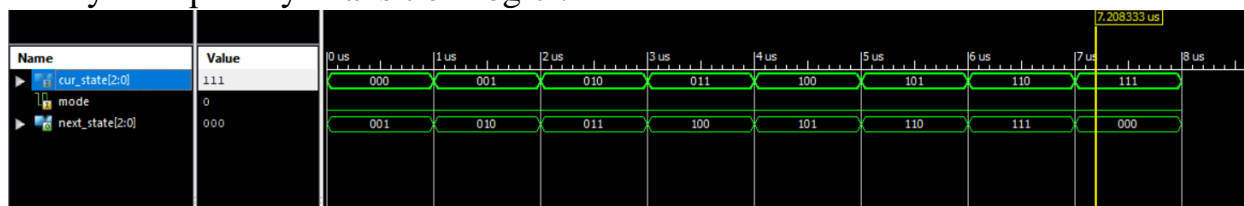
1  CONFIG VCCAUX = "3.3" ;
2
3  # Clock 12 MHz
4  NET "CLK" LOC = P129 | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;
5
6  #####
7  # LED
8  #####
9
10 NET "OUT_LED_0" LOC = P46 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
11 NET "OUT_LED_1" LOC = P47 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
12 NET "OUT_LED_2" LOC = P48 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
13 NET "OUT_LED_3" LOC = P49 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
14 NET "OUT_LED_4" LOC = P50 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
15 NET "OUT_LED_5" LOC = P51 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
16 NET "OUT_LED_6" LOC = P54 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
17 NET "OUT_LED_7" LOC = P55 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
18
19 #####
20 # DP Switches
21 #####
22
23 NET "MODE" LOC = P70 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
24
25 #####
26 # Switches
27 #####
28
29 NET "test2" LOC = P78 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
30 NET "RESET" LOC = P79 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
31
32 #####
33

```

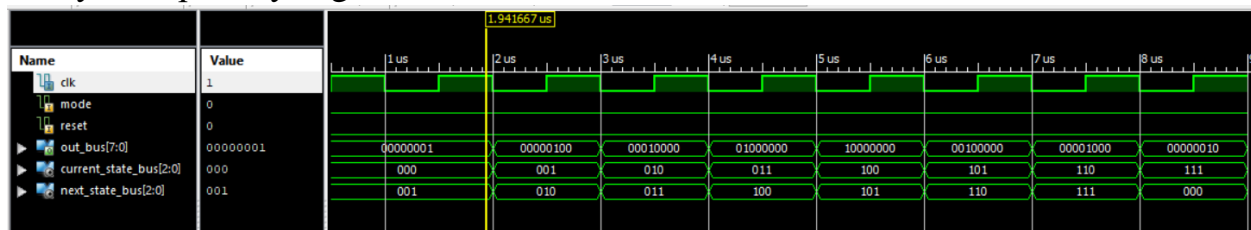
6) Симулюю работу OutputLogic :



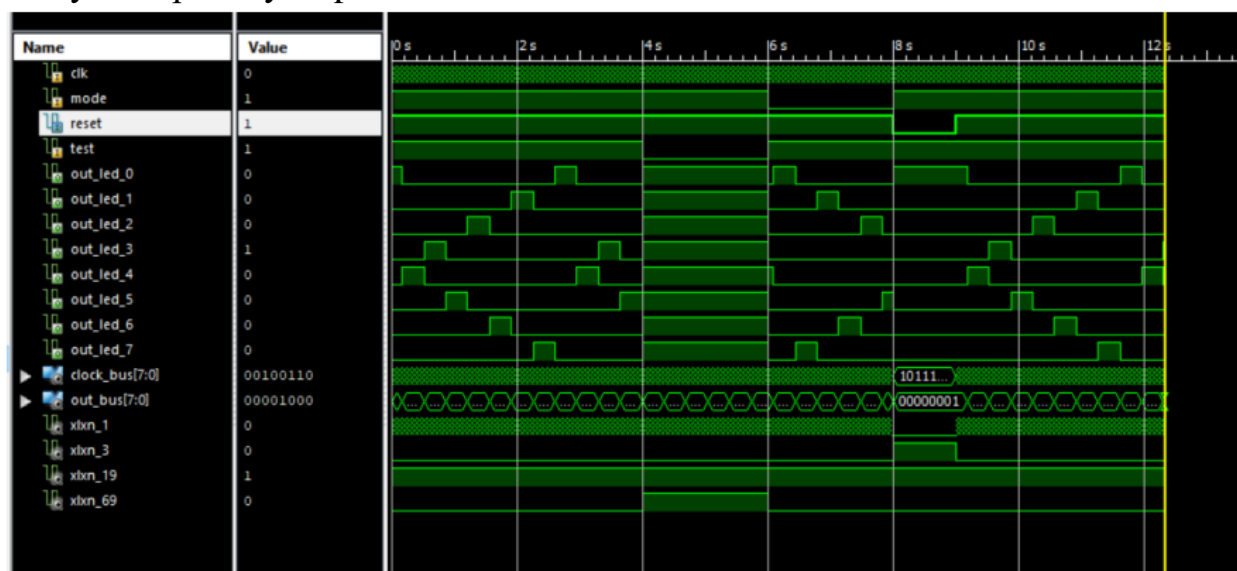
7) Симулюю работу TransitionLogic :



8) Симулюю роботу LightController:



9) Симулюю роботу TopLevel.sch :



10) Генерую BIN файл :

Локальний диск (D:) > xilinx > LAB2				
Ім'я	Дата змінення	Тип	Розмір	
sch2HdlBatchFile	31.05.2023 1:18	Файл	0 КБ	
testbench.v	02.05.2023 15:16	Файл V	2 КБ	
top_level.bgn	02.05.2023 16:17	Файл BGN	7 КБ	
top_level.bin	02.05.2023 16:17	Файл BIN	54 КБ	
top_level.bit	02.05.2023 16:17	Файл BIT	54 КБ	
Top_Level.bld	02.05.2023 16:17	Файл BLD	2 КБ	
Top_Level.cmd_log	02.05.2023 16:17	Файл CMD_LOG	1 КБ	
top_level.drc	02.05.2023 16:17	DRC Video File (VL...	1 КБ	
Top_Level.jhd	02.05.2023 16:16	Файл JHD	1 КБ	
Top_Level.iso	02.05.2023 16:17	Файл LSO	1 КБ	
Top_Level.ncd	02.05.2023 16:17	Файл NCD	15 КБ	
Top_Level.ngc	02.05.2023 16:17	Файл NGC	18 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.