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

Звіт з лабораторної роботи № 2

Виконав:
ст.гр. КІ-202
Куйбіда Н.В.
Прийняв
Ст.викладач
Козак Н.Б.

Львів – 2023

Тема: Структурний опис цифрового автомата. Перевірка роботи автомата за допомогою стенда Elbert V2-Spartan 3A FPGA

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

1. Інтерфейс пристрою та функціонал реалізувати згідно отриманого варіанту завдання
2. Логіку переходів реалізувати з використанням мови опису апаратних засобів VHDL. Заборонено використовувати оператори if, switch, for, when
3. Логіку формування вихідних сигналів реалізувати з використанням мови опису апаратних засобів VHDL. Заборонено використовувати оператори if, switch, for, when
4. Згенерувати Schematic символи для VHDL описів логіки переходів та логіки формування вихідних сигналів
5. Зінтегрувати всі компоненти (логіку переходів логіку формування вихідних сигналів та пам'ять станів) в єдину систему за допомогою ISE WebPACK Schematic Capture. Пам'ять станів реалізувати за допомогою графічних компонентів з бібліотеки
6. Промоделювати роботу окремих частин автомата та автомата в цілому за допомогою симулятора ISim
7. Інтегрувати створений автомат зі стендом Elbert V2 – Spartan 3A FPGA (додати подільник частоти для вхідного тактового сигналу призначити фізичні виводи на FPGA)
8. Згенерувати BIT файл та перевірити роботу за допомогою стенда Elbert V2 – Spartan 3A FPGA
9. Підготувати і захистити звіт

ЗАВДАННЯ:

Варіант 7

Варіант – 1:

- 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	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	1

- 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 2 РАЗИ ВИЩОЮ 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:

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


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

architecture out_logic_arc of out_logic_int 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 not (IN_BUS(1)) and IN_BUS(0));
   OUT_BUS(2) <= (not (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));
   OUT_BUS(4) <= ( IN_BUS(2) and not (IN_BUS(1)) and not (IN_BUS(0)));
   OUT_BUS(5) <= ( IN_BUS(2) and not (IN_BUS(1)) and  IN_BUS(0));
   OUT_BUS(6) <= ( IN_BUS(2) and  IN_BUS(1) and not (IN_BUS(0)));
   OUT_BUS(7) <= ( IN_BUS(2) and  IN_BUS(1) and  IN_BUS(0));
end out_logic_arc;
```

Рис.1 VHDL файл, який реалізує логіку формування сигналів (OutputLogic.vhd).

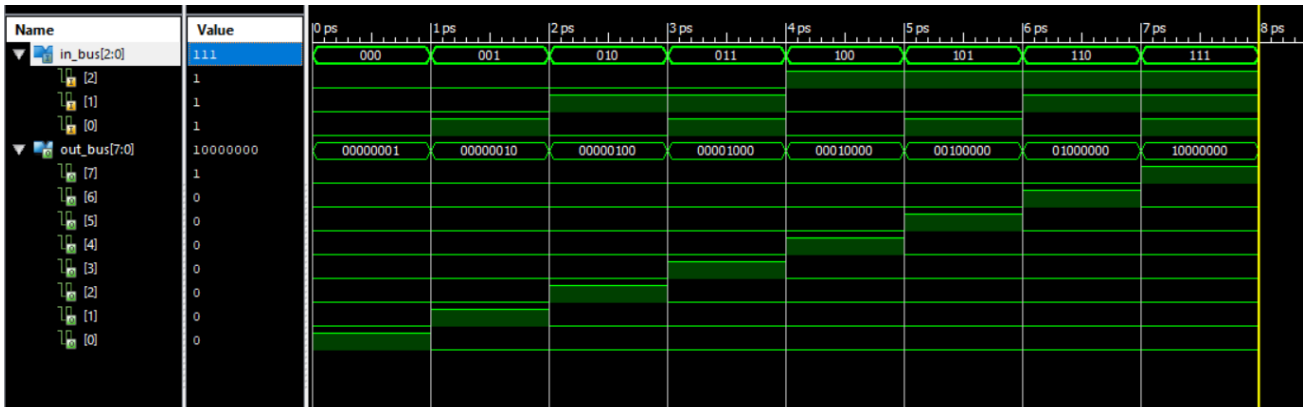

Рис.2 Промодельована робота схеми формування вихідних сигналів з усіма можливими наборами сигналів.

```
20  NEXT_STATE(0) <=
21
22      (not(RESET) and not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or
23      (not(RESET) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and not(CUR_STATE(0))) or
24      (not(RESET) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or
25      (not(RESET) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or
26      (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or
27      (not(RESET) and MODE and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or
28      (not(RESET) and MODE and not(CUR_STATE(2)) and CUR_STATE(1) and not(CUR_STATE(0))) or
29      (not(RESET) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));
30
31
32  NEXT_STATE(1)<=
33
34      (not(RESET) and not(MODE) and not(CUR_STATE(2)) and not(CUR_STATE(1)) and CUR_STATE(0)) or
35      (not(RESET) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and not(CUR_STATE(0))) or
36      (not(RESET) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or
37      (not(RESET) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or
38      (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1) and CUR_STATE(0)) or
39      (not(RESET) and MODE and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or
40      (not(RESET) and MODE and not(CUR_STATE(2)) and CUR_STATE(1) and CUR_STATE(0)) or
41      (not(RESET) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));
42
43  NEXT_STATE(2)<=
44
45      (not(RESET) and not(MODE) and not(CUR_STATE(2)) and CUR_STATE(1) and CUR_STATE(0)) or
46      (not(RESET) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and not(CUR_STATE(0))) or
47      (not(RESET) and not(MODE) and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or
48      (not(RESET) and not(MODE) and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or
49      (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1) and CUR_STATE(0)) or
50      (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1) and not(CUR_STATE(0))) or
51      (not(RESET) and MODE and CUR_STATE(2) and not(CUR_STATE(1)) and CUR_STATE(0)) or
52      (not(RESET) and MODE and not(CUR_STATE(2)) and not(CUR_STATE(1)) and not(CUR_STATE(0)));
```

Рис.3 VHDL файл, який реалізує логіку формування переходів стану автомата(TransitionLogic.vhd).

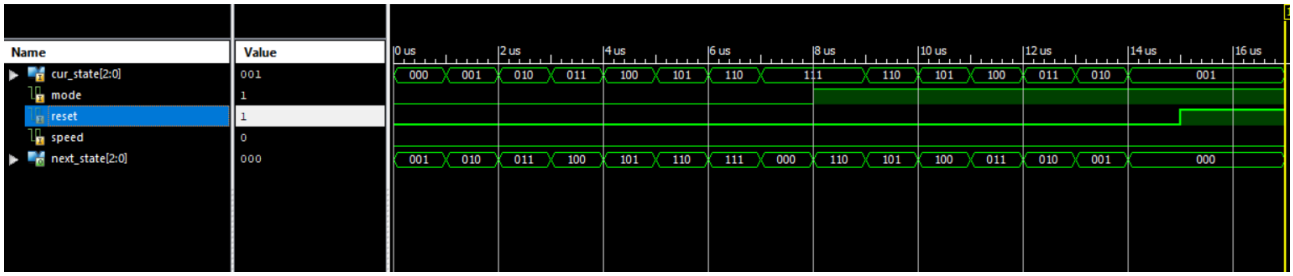

Рис.4 Промодельована роботу схеми формування вихідних сигналів з усіма можливими наборами сигналів.

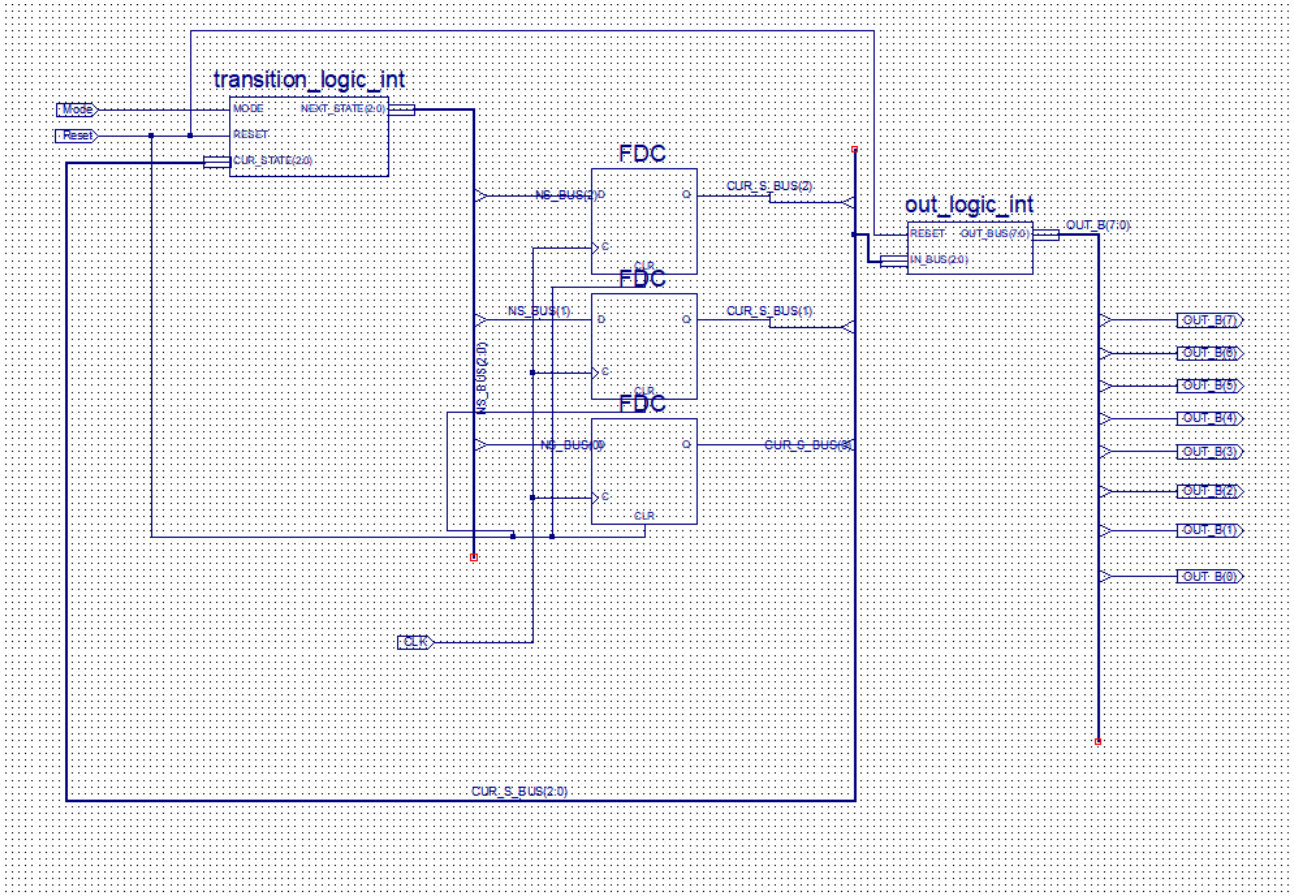

Рис.5 Schematic файл (LightController.sch), в якому реалізована пам'ять стану автомата.

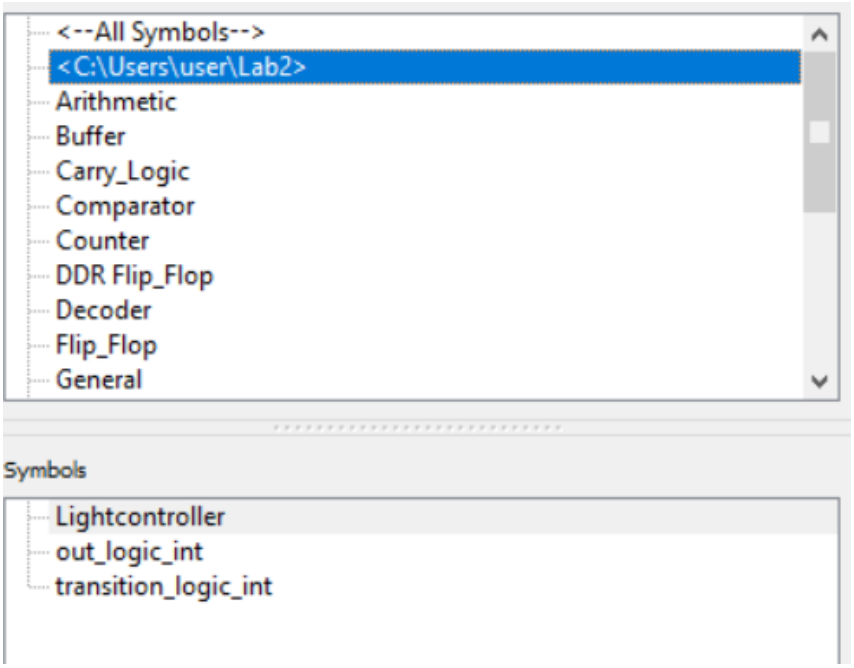

Рис.6 Згенеровані Schematic символи.

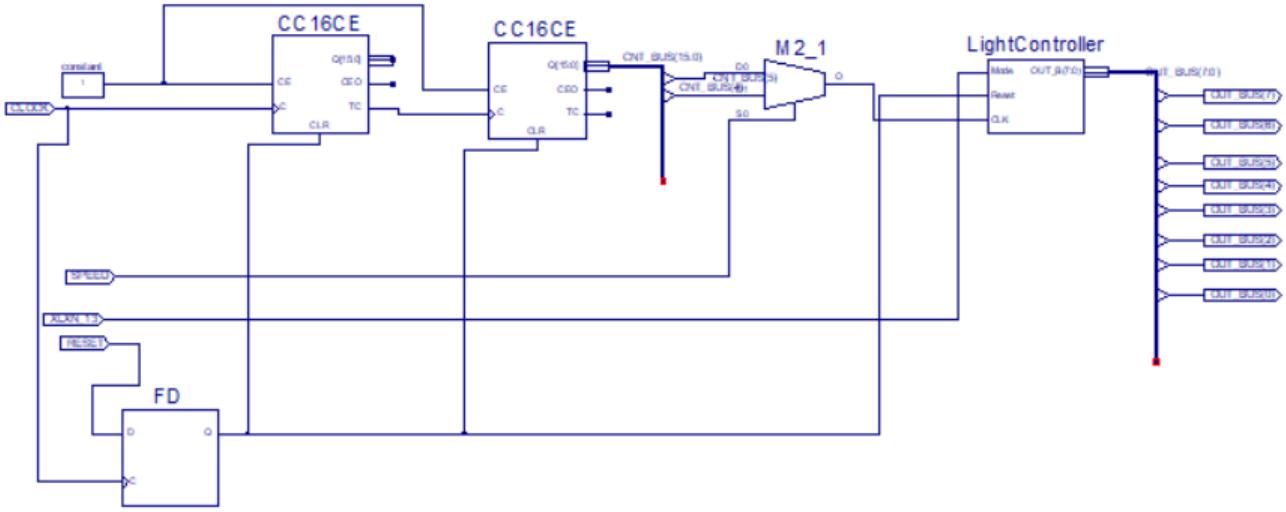

Рис.7 Schematic файл для кінцевої схеми.(TopLevel.sch). Реалізований подільник вхідної частоти.

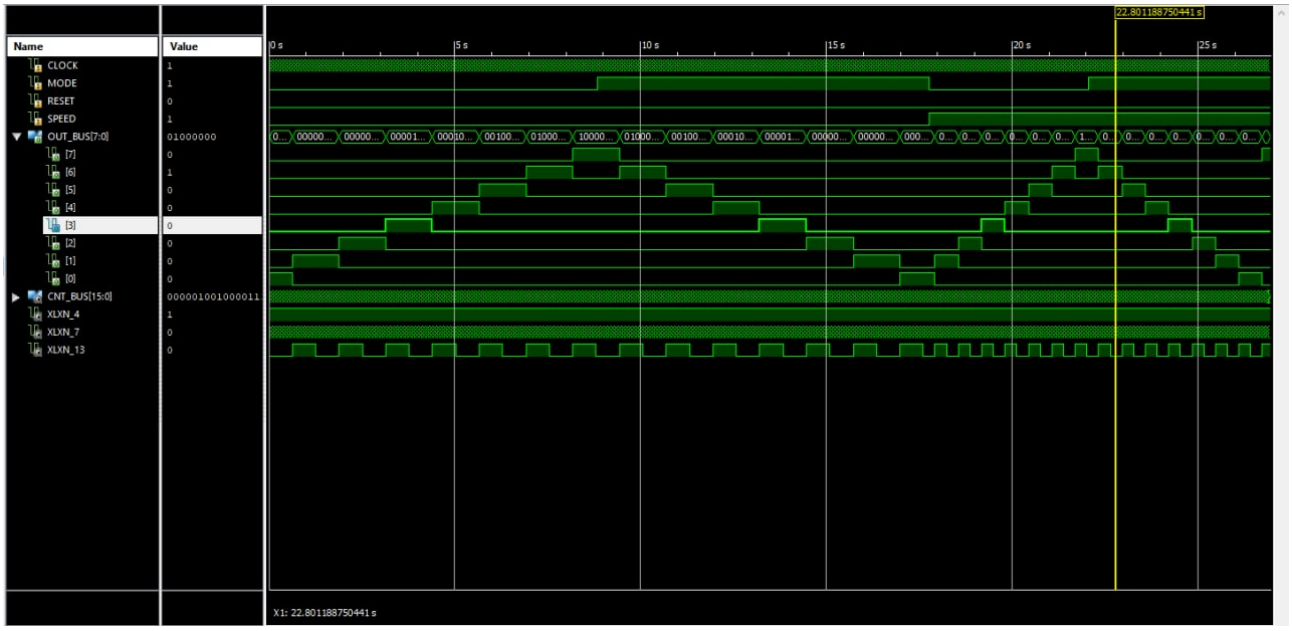

Рис.8 Симуляція системи різними значеннями сигналів MODE/RESET/SPEED.

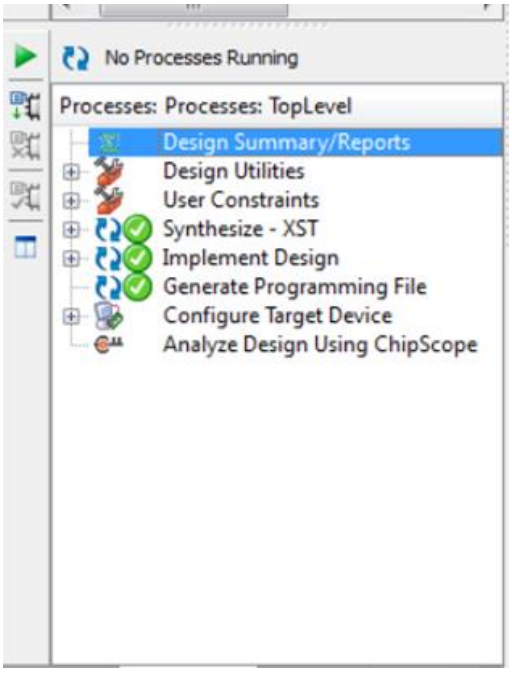

Рис.9 Згенерований бінарний файл.

Лістинг1 (VHDL OutputLogic)

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

entity out_logic_int is
Port ( IN_BUS : in  std_logic_vector(2 downto 0);
     OUT_BUS : out  std_logic_vector(7 downto 0);
RESET:in std_logic
      );
end out_logic_int;

architecture out_logic_arc of out_logic_int is

begin

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

end out_logic_arc;
```

Лістинг2 (VHDL TransitionLogic)

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



entity transition_logic_int is

 Port ( CUR_STATE : in  std_logic_vector(2 downto 0);
        MODE : in  std_logic;
        RESET : in std_logic;

        NEXT_STATE : out  std_logic_vector(2 downto 0)
      );

end transition_logic_int;

architecture transition_logic_arc of transition_logic_int is

begin

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


        (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1)
and not(CUR_STATE(0))) or
        (not(RESET) and MODE and CUR_STATE(2) and
not(CUR_STATE(1)) and not(CUR_STATE(0))) or
        (not(RESET) and MODE and not(CUR_STATE(2)) and
CUR_STATE(1) and not(CUR_STATE(0))) or
        (not(RESET) and MODE and not(CUR_STATE(2)) and
```

```
not(CUR_STATE(1)) and not(CUR_STATE(0)));



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


        (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1)
and CUR_STATE(0)) or
        (not(RESET) and MODE and CUR_STATE(2) and
not(CUR_STATE(1)) and not(CUR_STATE(0))) or
        (not(RESET) and MODE and not(CUR_STATE(2)) and
CUR_STATE(1) and CUR_STATE(0)) or
        (not(RESET) and MODE and not(CUR_STATE(2)) and
not(CUR_STATE(1)) and not(CUR_STATE(0)));

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


        (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1)
and CUR_STATE(0)) or
        (not(RESET) and MODE and CUR_STATE(2) and CUR_STATE(1)
and not(CUR_STATE(0))) or
```

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


    end transition_logic_arc;
```

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