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 Перевірка роботи автомата за допомогою стенда Elbert V2 – Spartan3A FPGA»

Варіант 23

Виконав

Студент групи КІ-201

Равчак В.А.

Перевірив:

Козак Н.Б.

Львів 2024

Мета роботи: на базі стенда реалізувати цифровий автомат світлових ефектів згідно заданих вимог.

Етапи роботи:

1. Інтерфейс пристрою та функціонал реалізувати згідно отриманого варіанту завдання.
2. Логіку переходів реалізувати з використанням мови опису апаратних засобів.
3. Логіку формування вихідних сигналів реалізувати з використанням мови опису апаратних засобів.
4. Згенерувати символи для 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. Пам'ять станів реалізувати за допомогою графічних компонентів з бібліотеки.
6. Промоделювати роботу окремих частин автомата та автомата вцілому за допомогою симулятора ISim.
7. Інтегрувати створений автомат зі стендом додати подільник частоти для вхідного тактовового сигналу призначити фізичні виводи на FPGA.
8. Згенерувати файал та перевірити роботу за допомогою стенда Elbert V2 – Spartan3A FPGA.
9. Підготувати і захистити звіт.

Варіант виконання роботи:

Пристрій повинен реалізувати 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 тактовий сигнал 12MHz від мікроконтролера і знижувати частоту за допомогою внутрішнього подільника Мікроконтролер є частиною стенда Elbert V2 – Spartan3A FPGA. Тактовий сигнал заведено на вхід LOC = P129 FPGA.

- 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:
  - 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.
- Для керування сигналами RESET/TEST використати будь які з PUSH BUTTON кнопок.

Виконання роботи:

1) Логіку переходів реалізувати з використанням мови опису апаратних засобів.

1)

MODE	CUR_STATE(2)	CUR_STATE(1)	CUR_STATE(0)	NEXT_STATE(0)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

2)

MODE	CUR_STATE(2)	CUR_STATE(1)	CUR_STATE(0)	NEXT_STATE(0)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

3)

MODE	CUR_STATE(2)	CUR_STATE(1)	CUR_STATE(0)	NEXT_STATE(0)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Мінімізовані функції наступних станів автомата:

NEXT_STATE(0) = not(CURR_STATE(0));

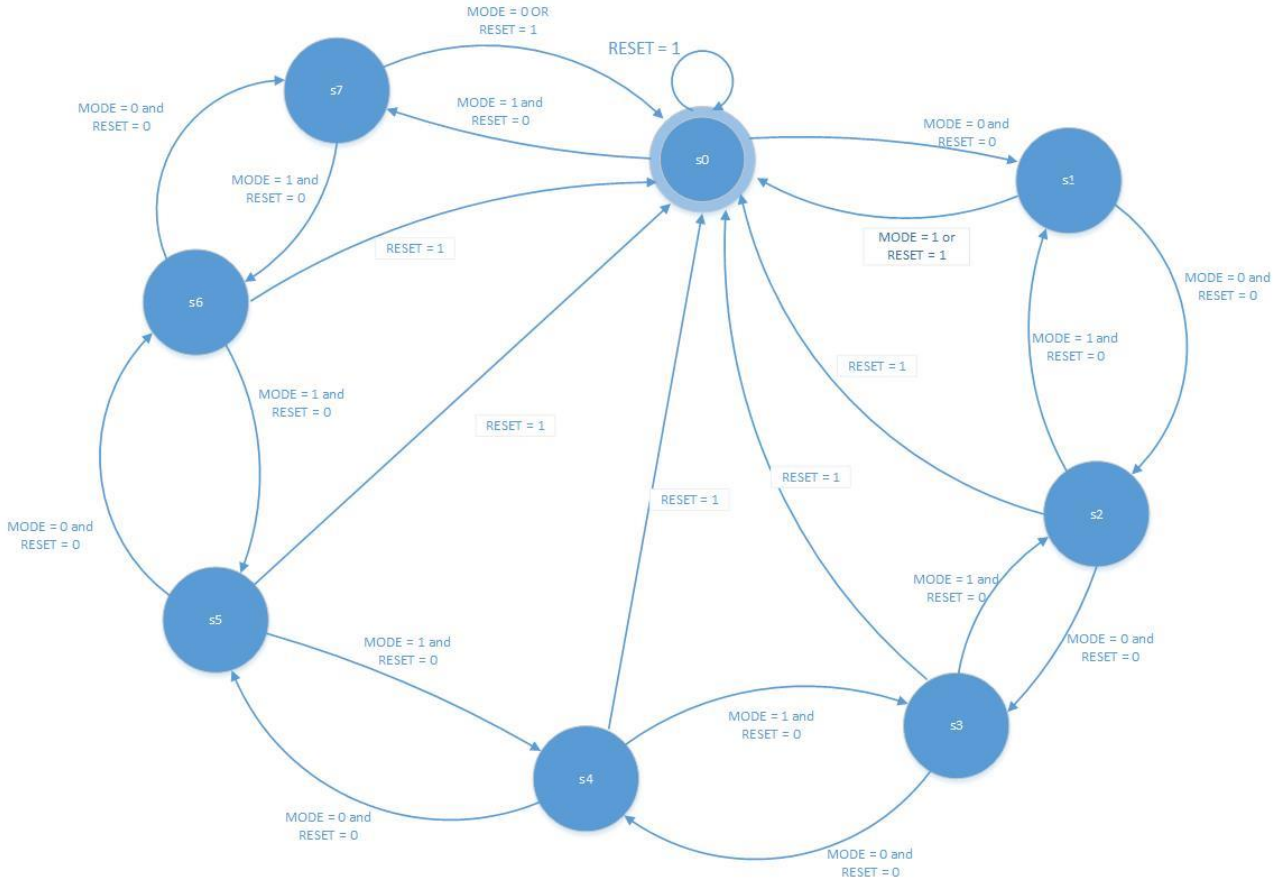
NEXT_STATE(1) = ((not(MODE) and not(CURR_STATE(1)) and CURR_STATE(0)) or (not(MODE) and CURR_STATE(1) and not(CURR_STATE(0))) or (MODE and not(CURR_STATE(1)) and not(CURR_STATE(0))) or (MODE and CURR_STATE(1) and CURR_STATE(0)));

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

VHDL опис логіки переходів:

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

entity TRANSITION_LOGIC is
    Port (CURR_STATE : in  std_logic_vector(2 downto 0);
          MODE : in std_logic;
          NEXT_STATE : out std_logic_vector(2 downto 0)
          );
end TRANSITION_LOGIC;

architecture TRANSITION_LOGIC_ARCH of TRANSITION_LOGIC is

begin
    NEXT_STATE(0) <= (not(CURR_STATE(0))) after 1 ns;
    NEXT_STATE(1) <= ((not(MODE) and not(CURR_STATE(1)) and CURR_STATE(0)) or
                      (not(MODE) and CURR_STATE(1) and not(CURR_STATE(0))) or
                      (MODE and not(CURR_STATE(1)) and not(CURR_STATE(0))) or
                      (MODE and CURR_STATE(1) and CURR_STATE(0))) after 1 ns;
    NEXT_STATE(2) <= ((not(MODE) and CURR_STATE(2) and not(CURR_STATE(1))) or
                      (CURR_STATE(2) and CURR_STATE(1) and not(CURR_STATE(0))) or
                      (MODE and CURR_STATE(2) and CURR_STATE(0)) or
                      (not(MODE) and not(CURR_STATE(2)) and CURR_STATE(1) and CURR_STATE(0)) or
                      (MODE and not(CURR_STATE(2)) and not(CURR_STATE(1)) and not(CURR_STATE(0)))) after 1 ns;


end TRANSITION_LOGIC_ARCH;

```

Граф переходів між станами:



2) Логіку формування вихідних сигналів реалізувати з використанням мови опису апаратних засобів VHDL.

Логічні вирази для вихідних сигналів:

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

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

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

OUT_BUS(3) = ((IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))) or TEST);

OUT_BUS(4) = ((IN_BUS(2) and IN_BUS(1) and IN_BUS(0)) or TEST);

OUT_BUS(5) = ((IN_BUS(2) and not(IN_BUS(1)) and IN_BUS(0)) or TEST);

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

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

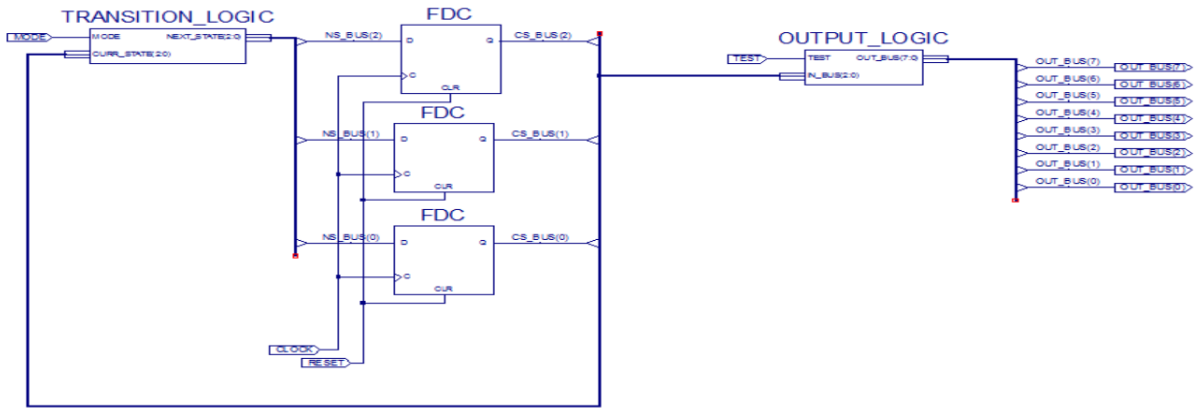
VHDL опис вихідних сигналів:

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

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

architecture OUTPUT_LOGIC_ARCH of OUTPUT_LOGIC is

begin

    OUT_BUS(0) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and not(IN_BUS(0))) or TEST) after 1 ns;
    OUT_BUS(1) <= ((not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0))) or TEST) after 1 ns;
    OUT_BUS(2) <= ((IN_BUS(2) and not(IN_BUS(1)) and not(IN_BUS(0))) or TEST) after 1 ns;
    OUT_BUS(3) <= ((IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0))) or TEST) after 1 ns;
    OUT_BUS(4) <= ((IN_BUS(2) and IN_BUS(1) and IN_BUS(0)) or TEST) after 1 ns;
    OUT_BUS(5) <= ((IN_BUS(2) and not(IN_BUS(1)) and IN_BUS(0)) or TEST) after 1 ns;
    OUT_BUS(6) <= ((not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0)) or TEST) after 1 ns;
    OUT_BUS(7) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0)) or TEST) after 1 ns;

end OUTPUT_LOGIC_ARCH;

|
```

3) Згенерувати символи для описів логіки переходів та логіки формування вихідних сигналів.

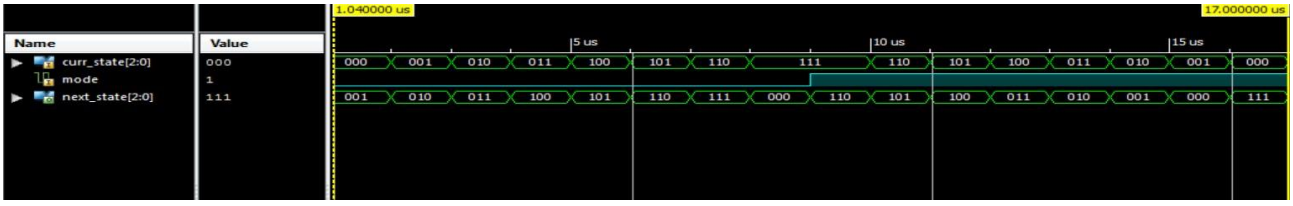
4) Зінтегрувати всі компоненти логіку переходів логіку формування вихідних сигналів та пам ять станів в єдину систему за допомогою ISE WebPACK Schematic Capture. Пам ять станів реалізувати за допомогою графічних компонентів з бібліотеки.

Інтеграція всіх створених компонентів разом з пам'ятю стану автомата:

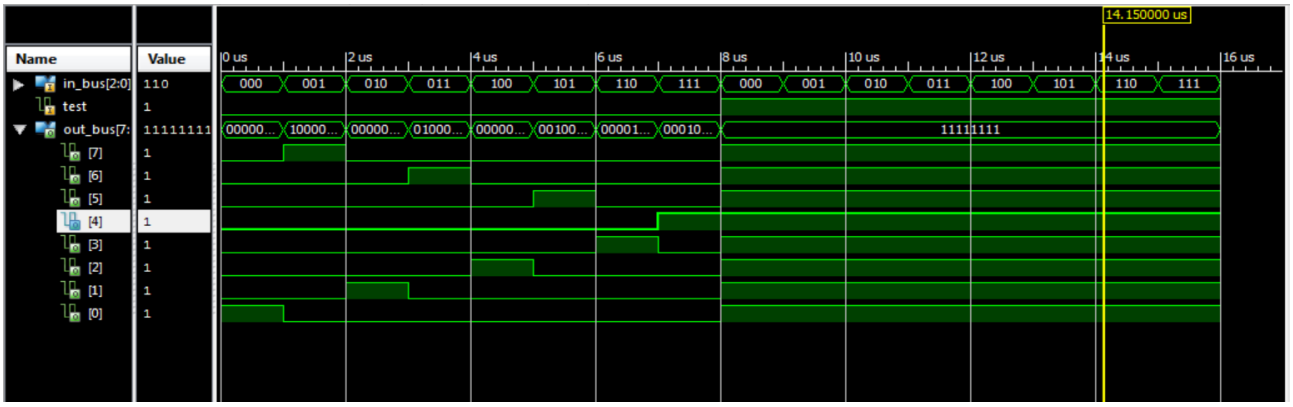


5) Промоделював роботу окремих частин автомата та автомата вцілому за допомогою симулятора ISim.

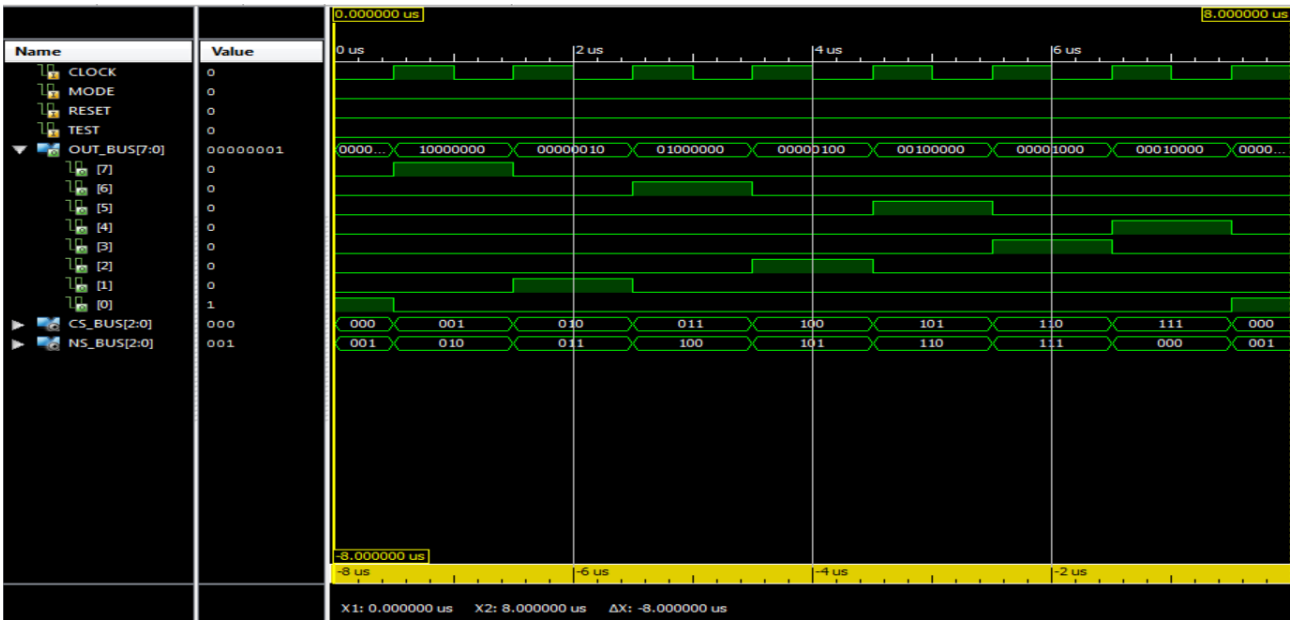
Симуляція логіки переходів:



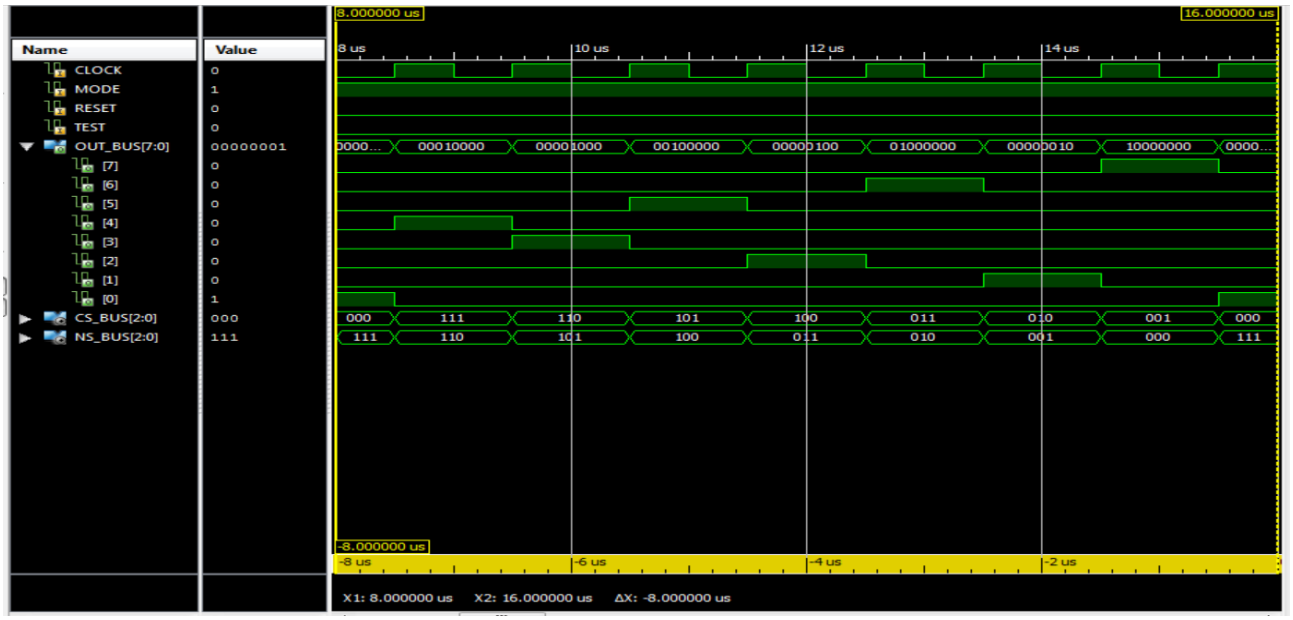
Симуляція логіки вихідних сигналів(TEST = 0):



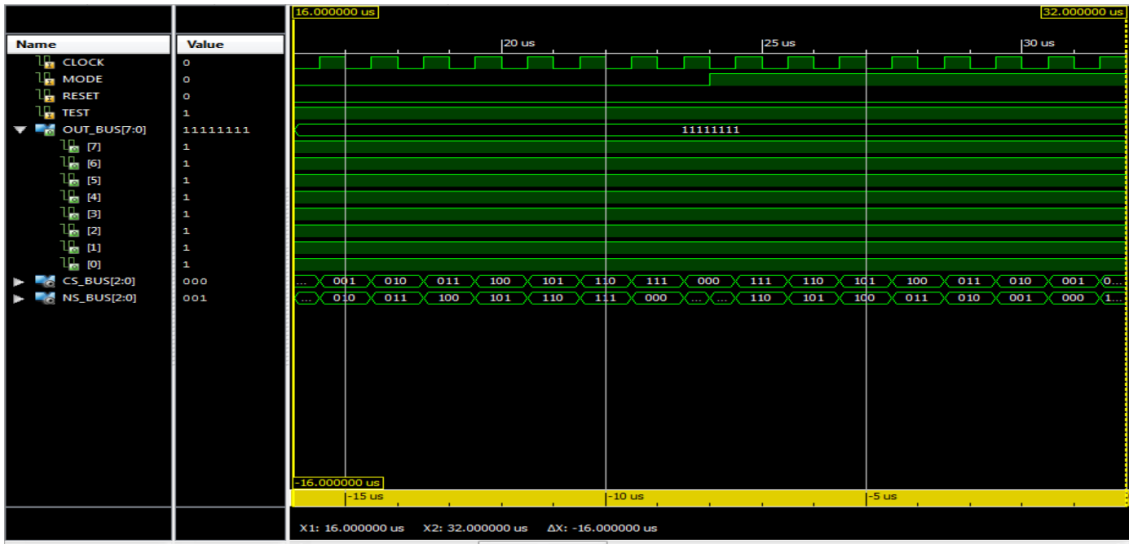
Результати симуляції автомата(MODE = 0, TEST = 0, RESET = 0)



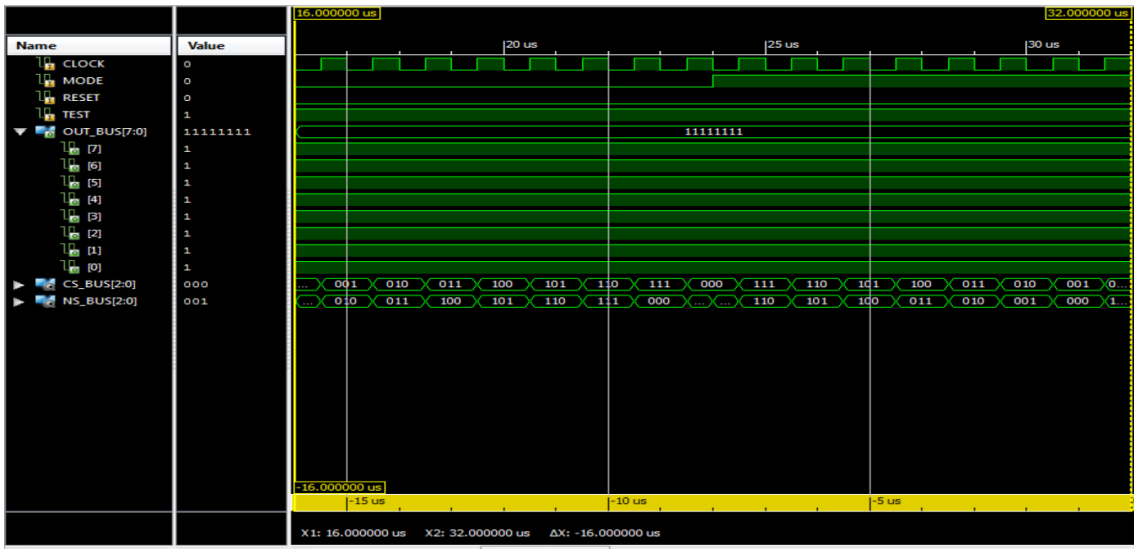
Результати симуляції автомата(MODE = 1, TEST = 0, RESET = 0)



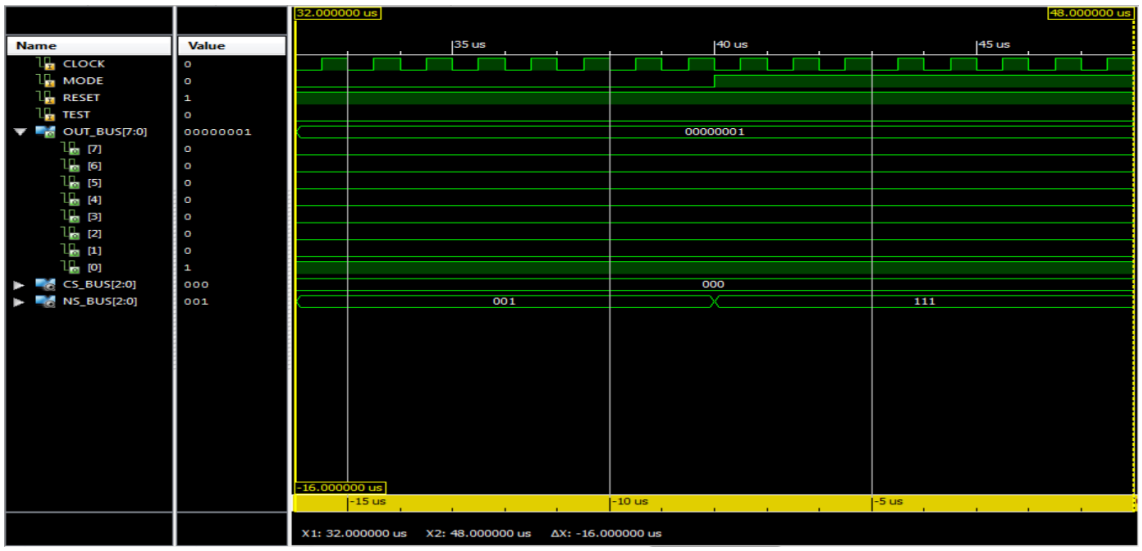
Результати симуляції автомата(MODE = 0, TEST = 1, RESET = 0)



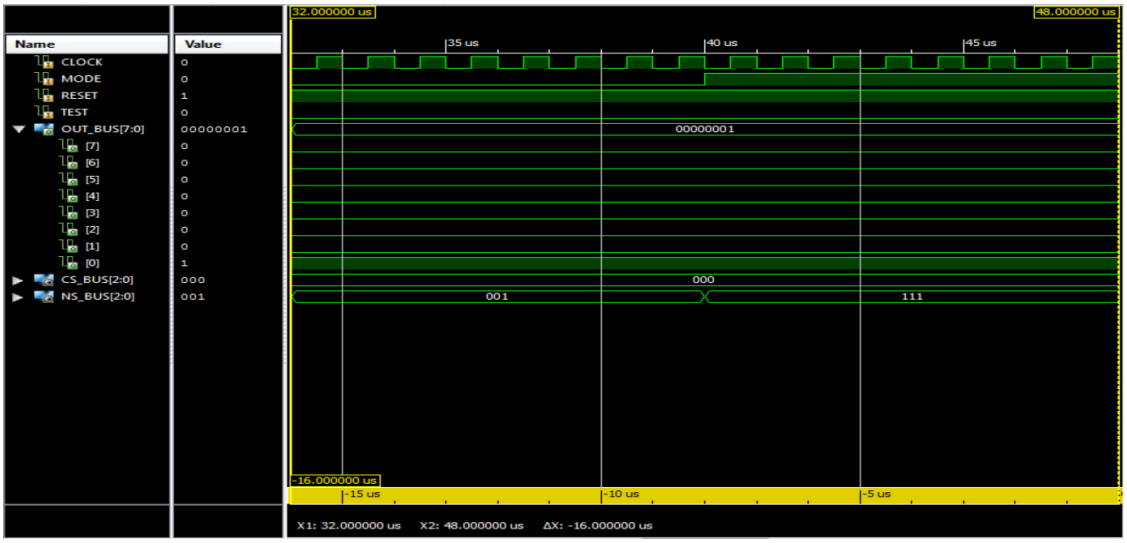
Результати симуляції автомата(MODE = 1, TEST = 1, RESET = 0)



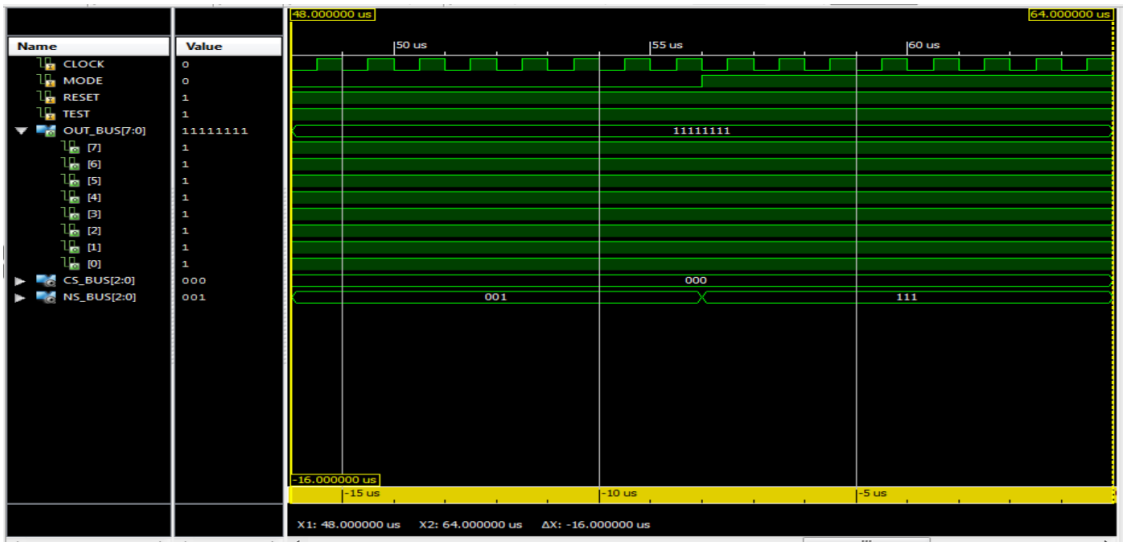
Результати симуляції автомата(MODE = 0, TEST = 0, RESET = 1)



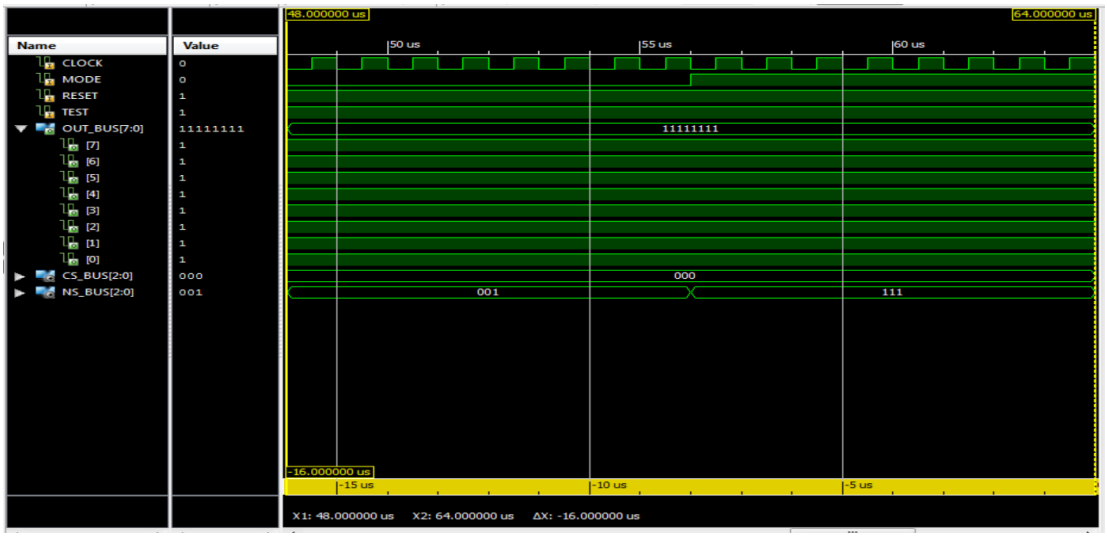
Результати симуляції автомата(MODE = 1, TEST = 0, RESET = 1)



Результати симуляції автомата(MODE = 0, TEST = 1, RESET = 1)



Результати симуляції автомата(MODE = 1, TEST = 1, RESET = 1)



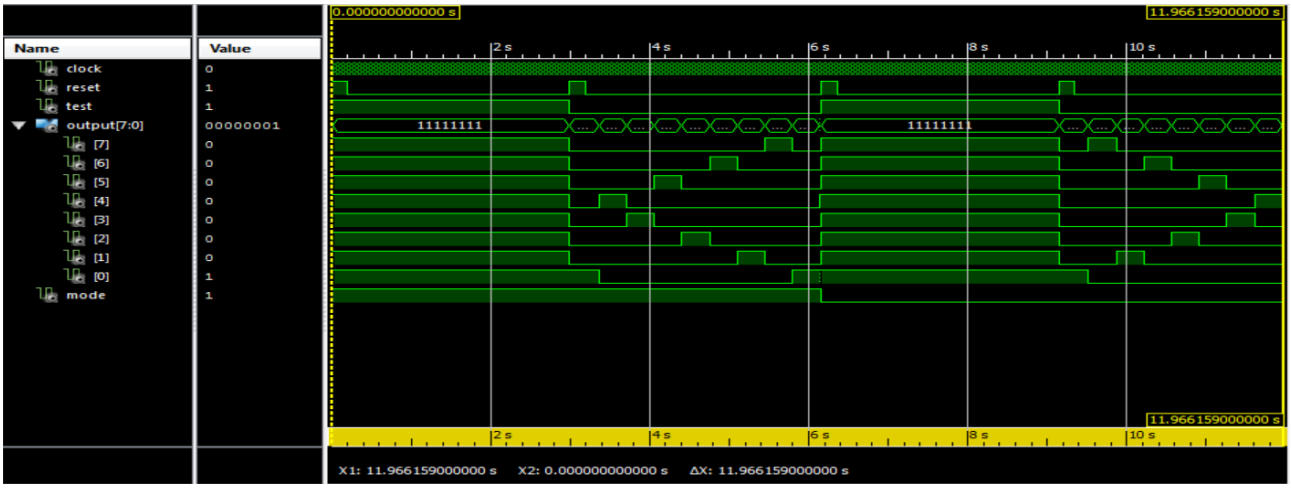
TEST BENCH:

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
                MODE <= '1';
                TEST <= '1';
                RESET <= '1', '0' after 200 ms;
                wait until RESET = '0';
                assert OUTPUT = "11111111";
                wait for 2786028us;

                TEST <= '0';
                RESET <= '1', '0' after 200 ms;
                wait until RESET = '0';

                assert OUTPUT = "00000001";
                wait for 174864us;
                assert OUTPUT = "00010000";
                wait for 349625us;
                assert OUTPUT = "00001000";
                wait for 349625us;
                assert OUTPUT = "00100000";
                wait for 349625us;
                assert OUTPUT = "00000100";
                wait for 349625us;
                assert OUTPUT = "01000000";
                wait for 349625us;
                assert OUTPUT = "00000010";
                wait for 349625us;
                assert OUTPUT = "10000000";
```

```
wait for 349625us;
assert OUTPUT = "00000001";
wait for 349625us;



TEST <= '1';
MODE <= '0';
RESET <= '1', '0' after 200 ms;
wait until RESET = '0';

assert OUTPUT = "11111111";
wait for 2786028us;



TEST <= '0';
RESET <= '1', '0' after 200 ms;
wait until RESET = '0';

assert OUTPUT = "00000001";
wait for 174864us;
assert OUTPUT = "10000000";
wait for 349625us;
assert OUTPUT = "00000010";
wait for 349625us;
assert OUTPUT = "01000000";
wait for 349625us;
assert OUTPUT = "00000100";
wait for 349625us;
assert OUTPUT = "00100000";
```

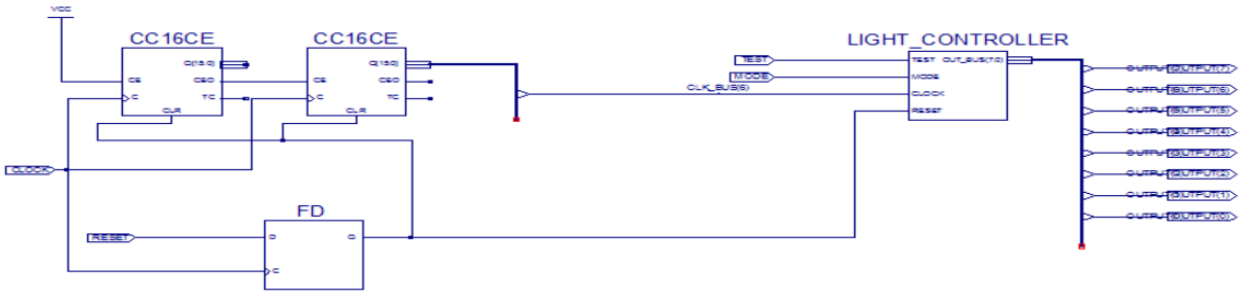
```
        wait for 349625us;
        assert OUTPUT = "00001000";
        wait for 349625us;
        assert OUTPUT = "00010000";
        wait for 349625us;
    END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

Результати TEST BENCH:



6) Інтегрувати створений автомат зі стендом Elbert V2 – Spartan3A FPGA. Додати подільник частоти для вхідного тактовового сигналу призначити фізичні виводи на FPGA.

Схема автомата світлових сигналів та подільник тактового сигналу:



Призначення фізичних входів та виходів:

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

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

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

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

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

    NET "MODE"          LOC = P70    | PULLUP   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "RESET"           LOC = P69    | PULLUP   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
    NET "TEST"          LOC = P68    | PULLUP   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

Висновок: виконуючи дану лабораторну роботу я реалізував цифровий автомат світлових ефектів на базі стенда Elbert V2 – Spartan3A FPGA згідно заданих вимог.