

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 15

Виконав: ст. гр. КІ-201

Коростенська С.В

Прийняв:

ст. викладач

Козак Н.Б

Львів 2024

Мета роботи:

На базі стенда реалізувати цифровий автомат світлових ефектів

Варіант виконання роботи:

Варіант 3

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

- 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 (SPEED):
 - Якщо $SPEED=0$ то автомат працює зі швидкістю, визначеною за замовчуванням.
 - Якщо $SPEED=1$ то автомат працює зі швидкістю, **В 4 РАЗИ ВИЩОЮ** ніж в режимі ($SPEED=0$).
- Для керування сигналом MODE використати будь який з 8 DIP перемикачів.
- Для керування сигналами RESET/SPEED використати будь які з PUSH BUTTON кнопок.

Виконання роботи:

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;
```

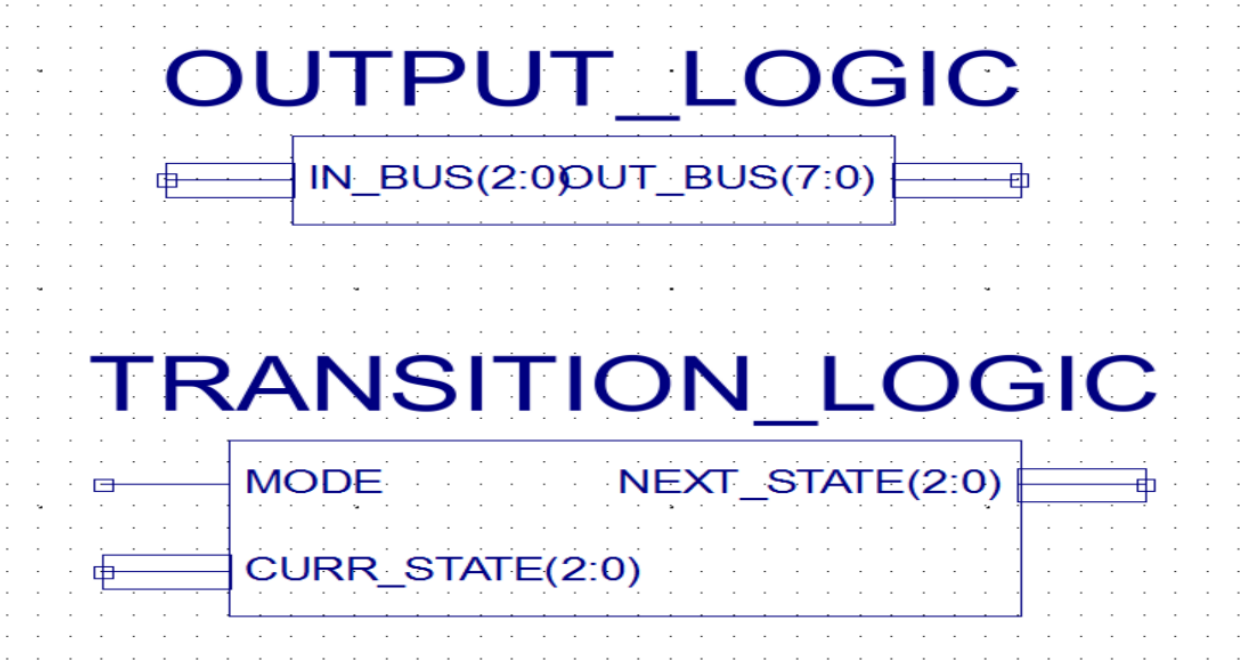
VHDL опис вихідних сигналів

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

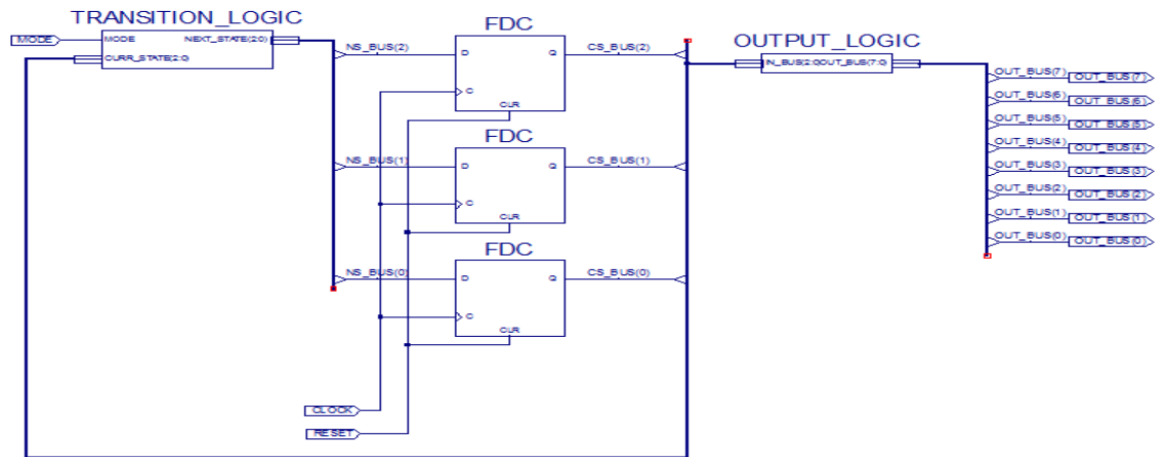
entity OUTPUT_LOGIC is
  Port ( IN_BUS : in std_logic_vector(2 downto 0);
        OUT_BUS : out std_logic_vector(7 downto 0)
  );
end OUTPUT_LOGIC;

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))) after 1ns;
  OUT_BUS(1) <= (not(IN_BUS(2)) and not(IN_BUS(1))) after 1ns;
  OUT_BUS(2) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and IN_BUS(0)) or (not(IN_BUS(2)) and IN_BUS(1) and not(IN_BUS(0)))) after 1ns;
  OUT_BUS(3) <= (not(IN_BUS(2)) and IN_BUS(1)) after 1ns;
  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)))) after 1ns;
  OUT_BUS(5) <= (IN_BUS(2) and not(IN_BUS(1))) after 1ns;
  OUT_BUS(6) <= ((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)))) after 1ns;
  OUT_BUS(7) <= (IN_BUS(2) and IN_BUS(1)) after 1ns;
end OUTPUT_LOGIC_ARCH;
```

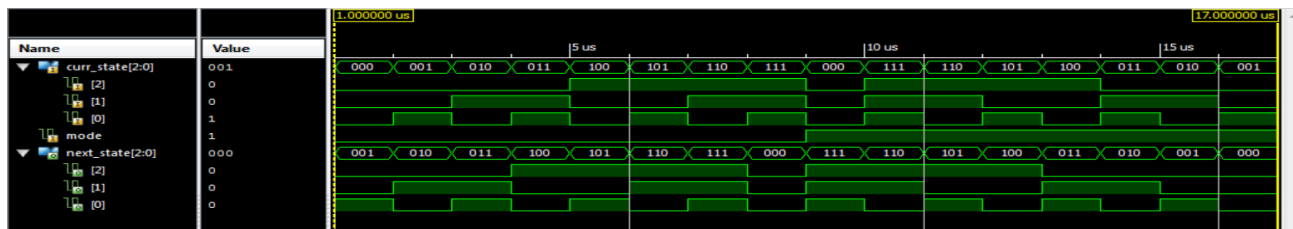
Згенеровані схематичні символи



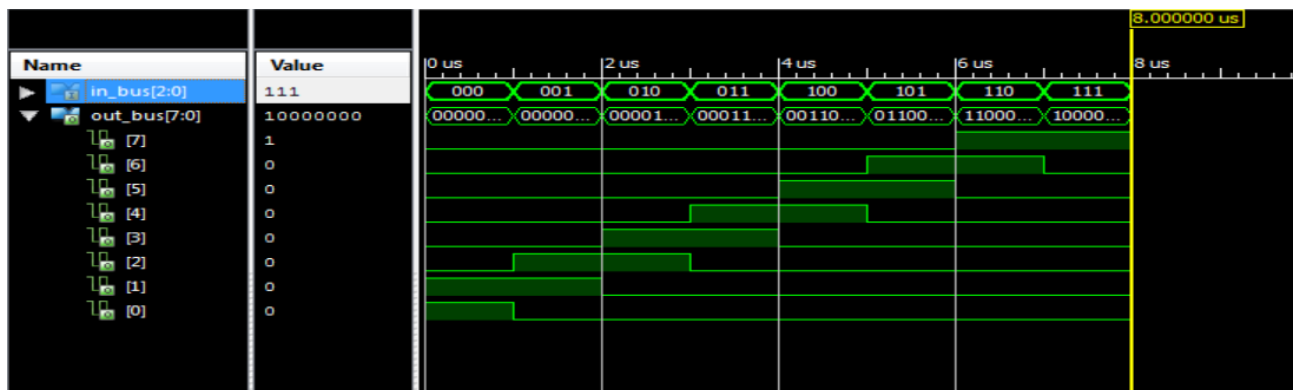
Інтеграція всіх створених компонентів разом з пам'яттю стану автомата



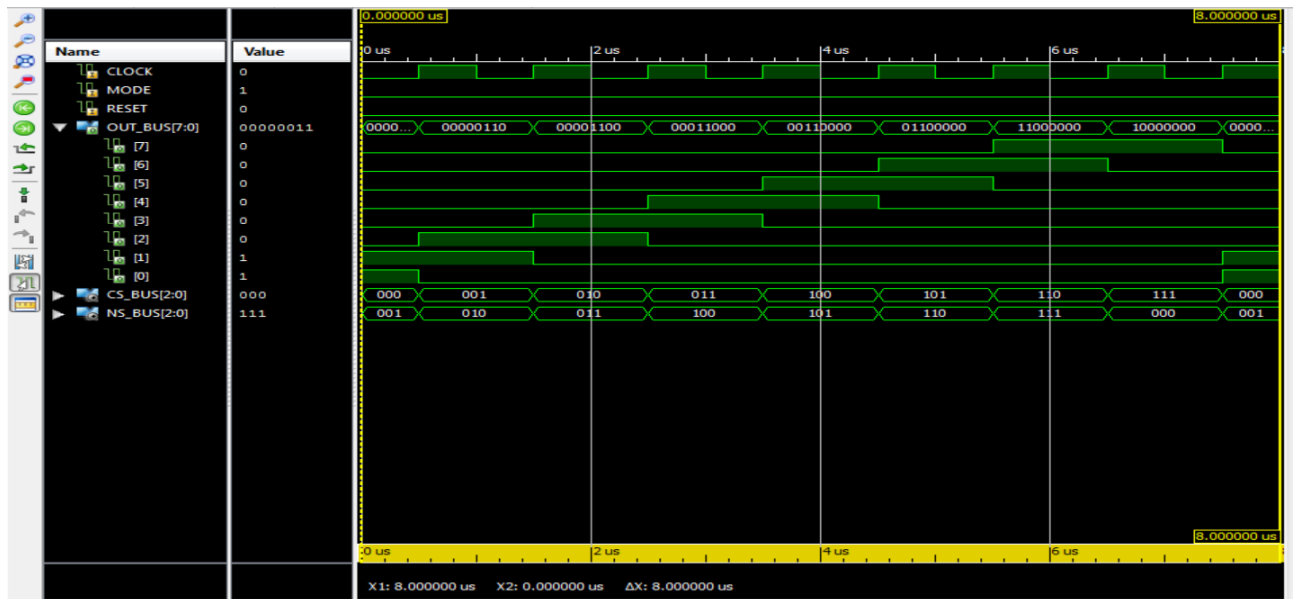
Результати симуляції логіки переходів в ISim



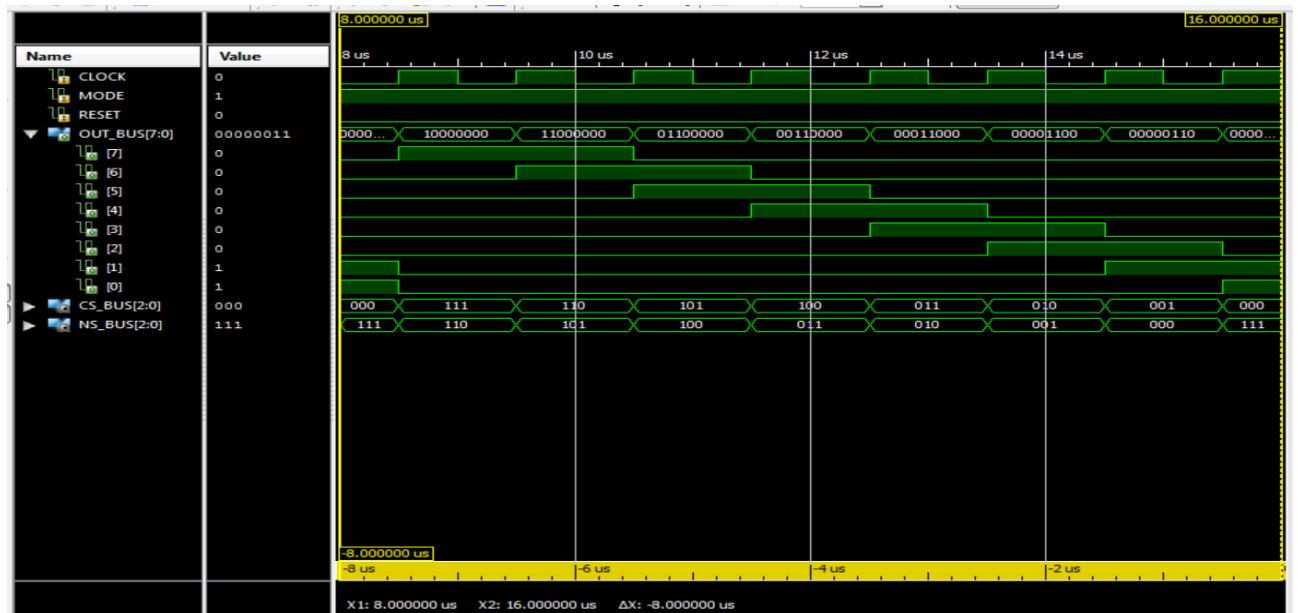
Результати симуляції логіки вихідних сигналів в ISim



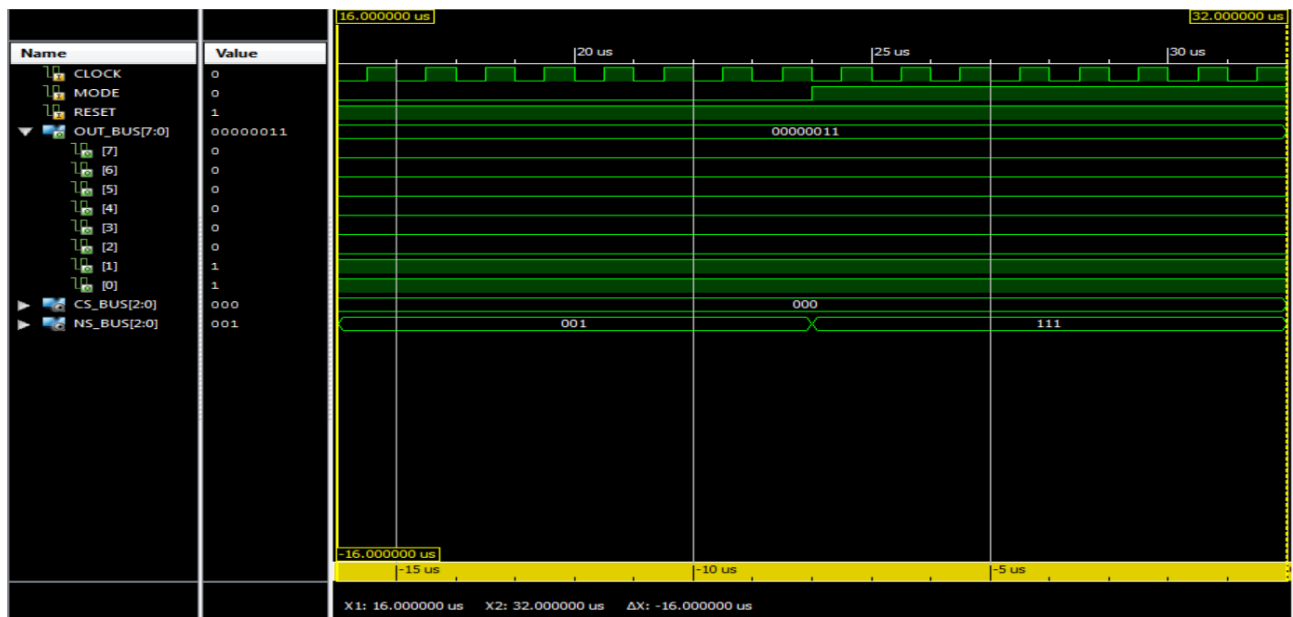
Результати симуляції автомата (MODE = 0, RESET = 0)



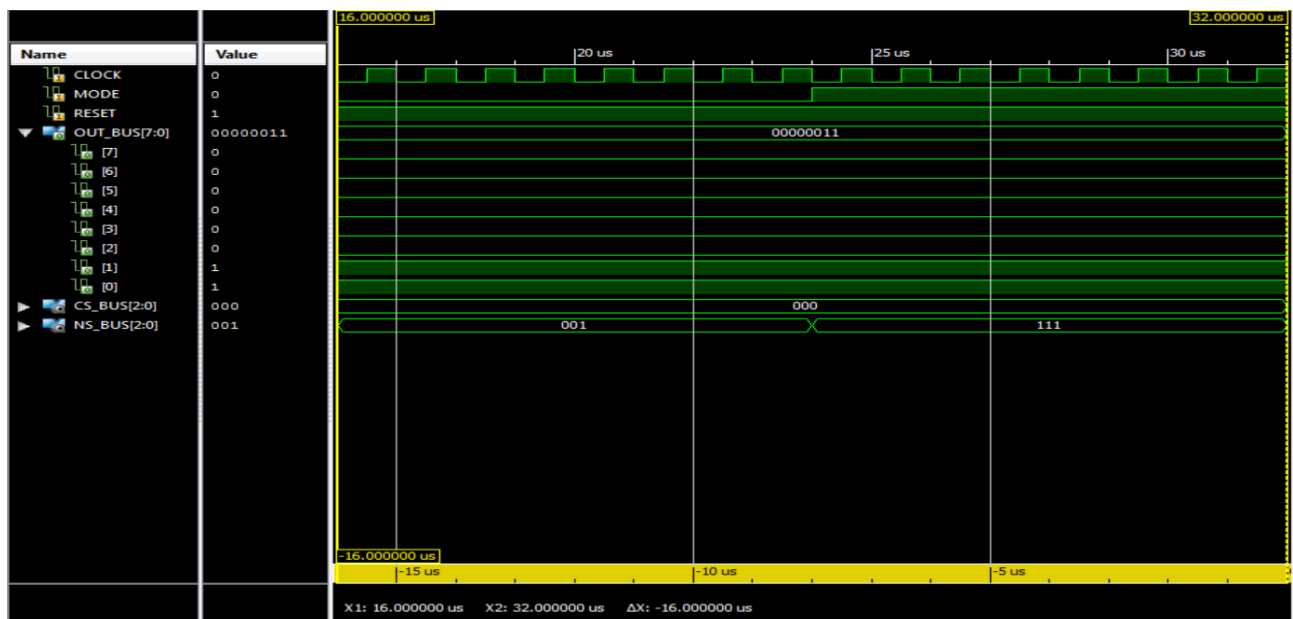
Результати симуляції автомата (MODE = 1, RESET = 0)



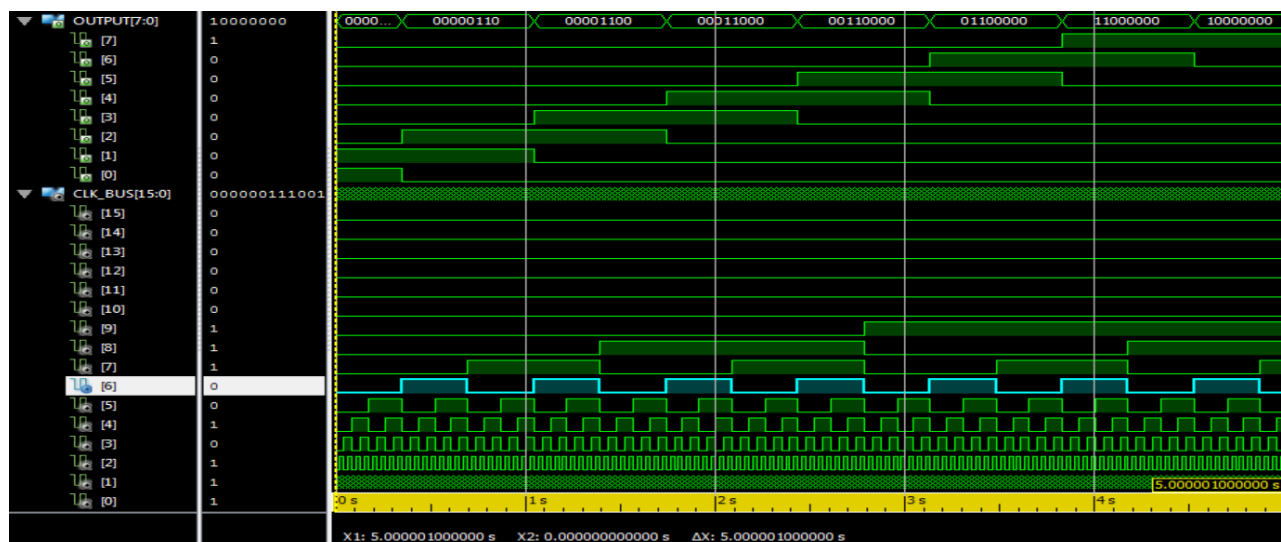
Результати симуляції автомата (MODE = 0, RESET = 1)



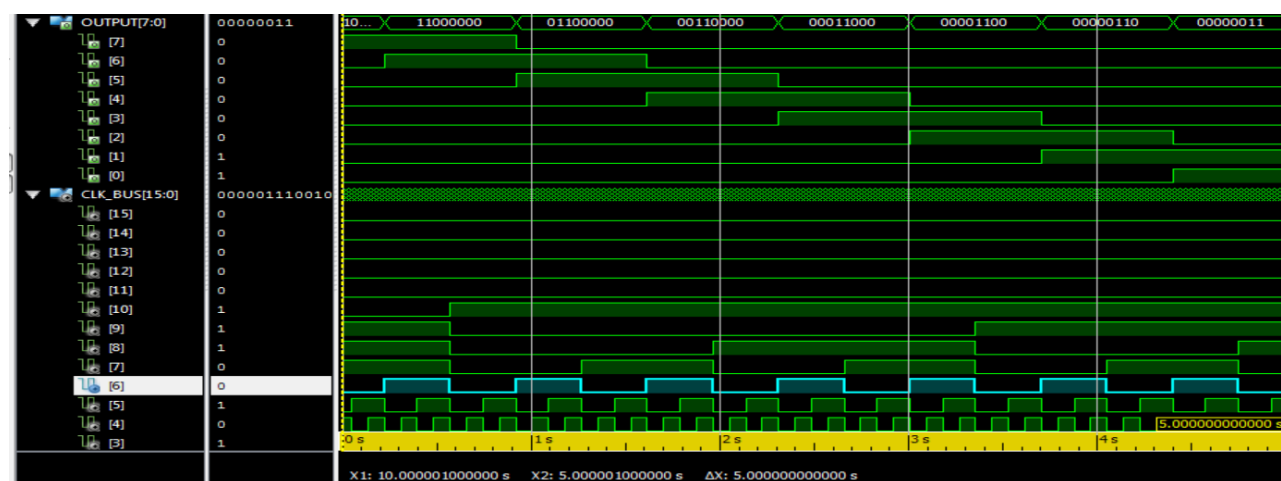
Результати симуляції автомата (MODE = 1, RESET = 1)



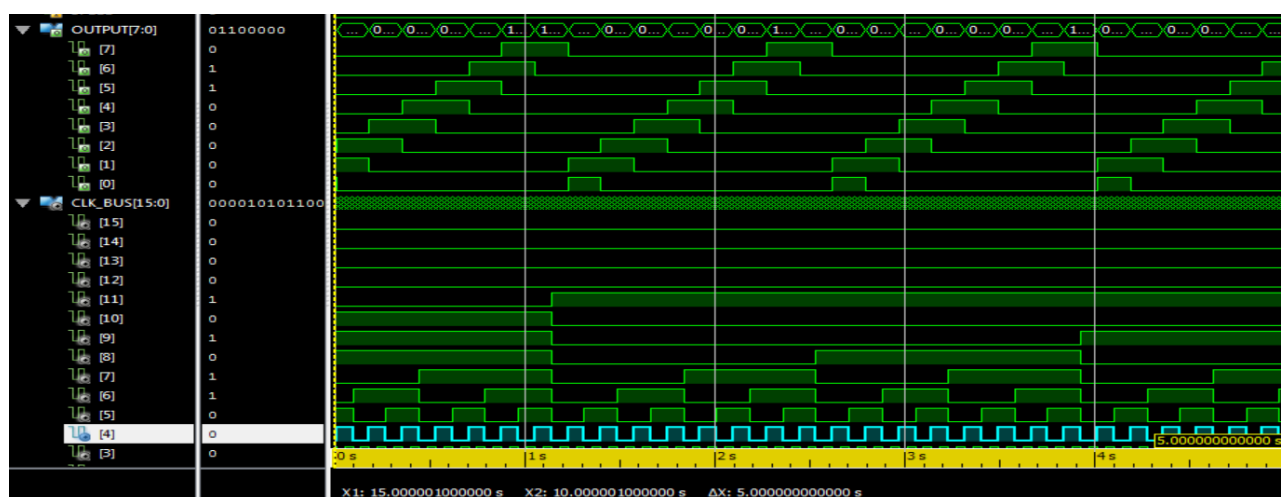
Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 0)



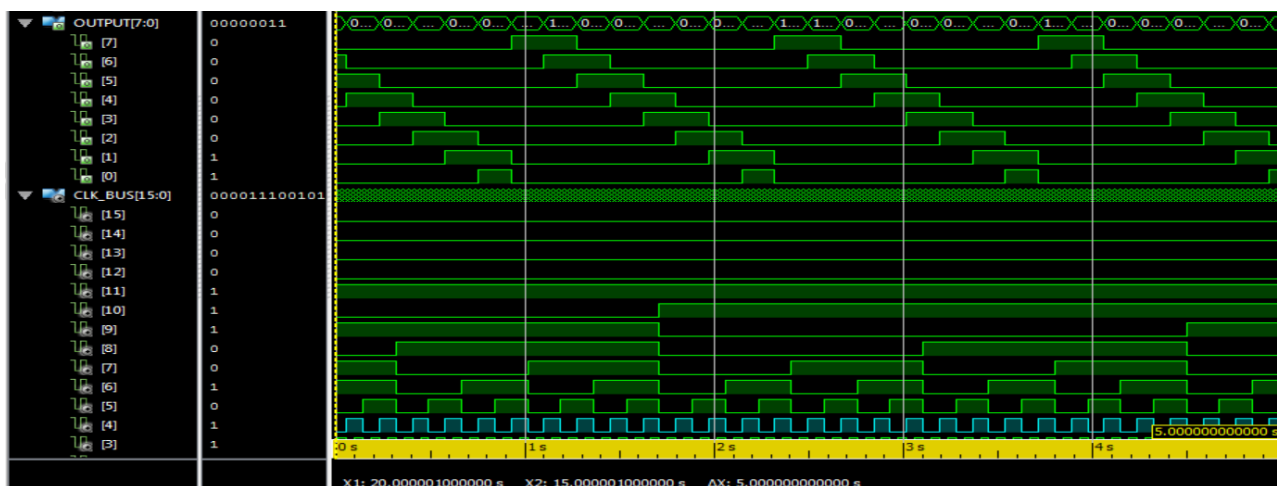
Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 0)



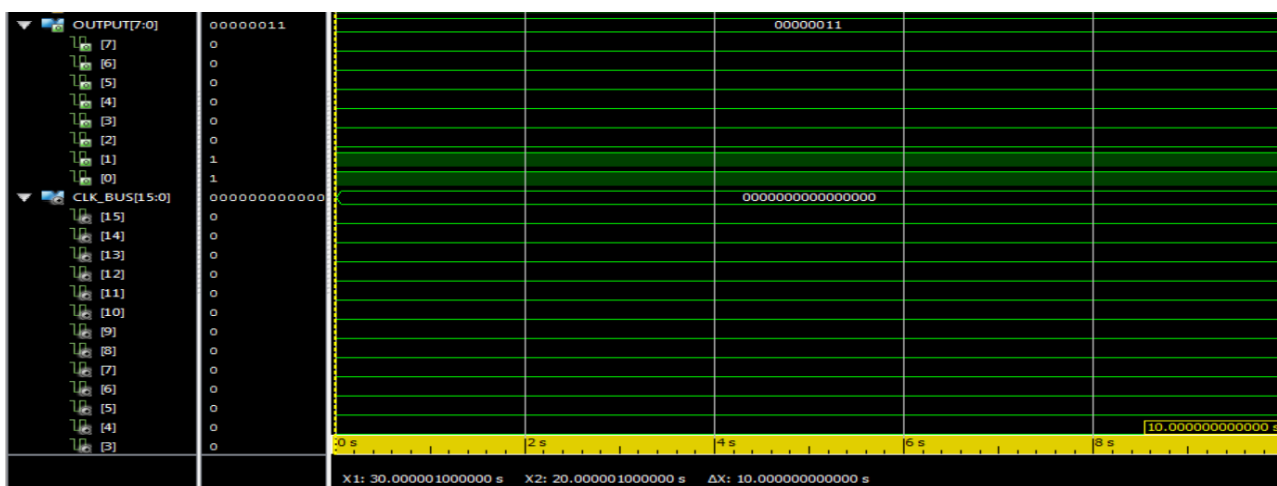
Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 0)



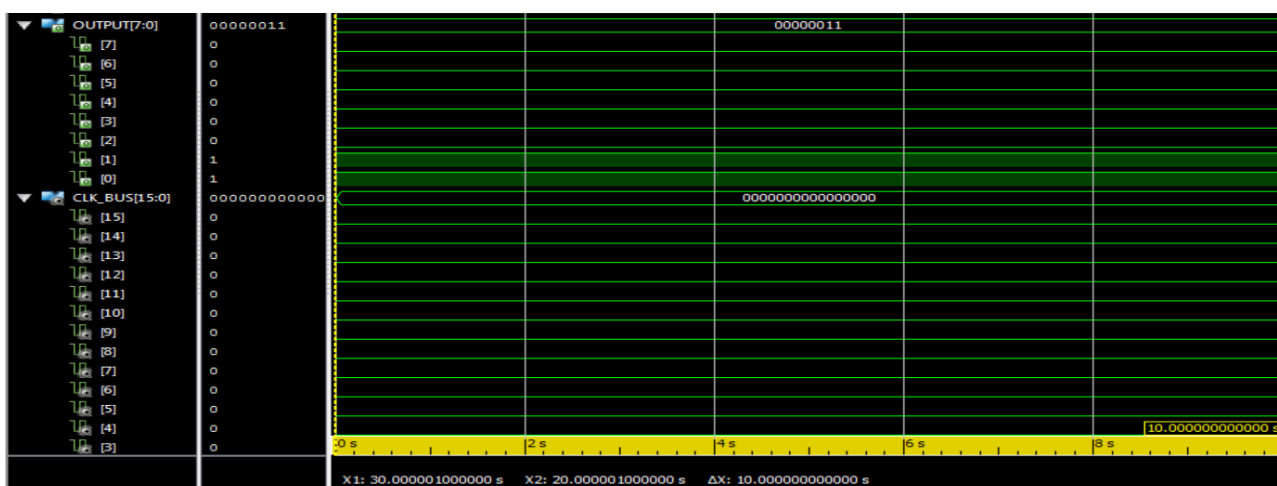
Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 0)



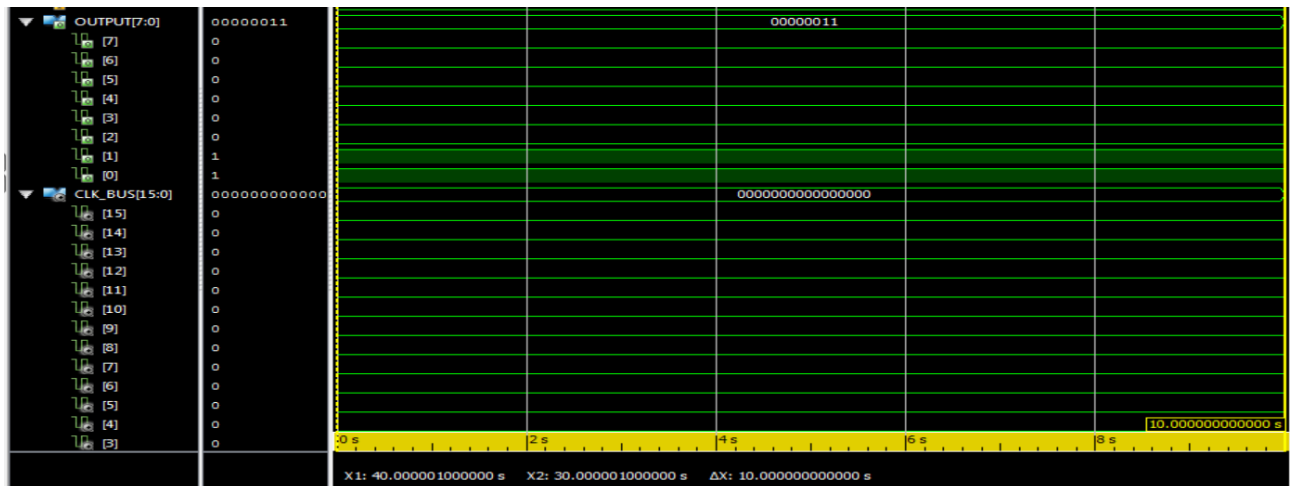
Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 1)



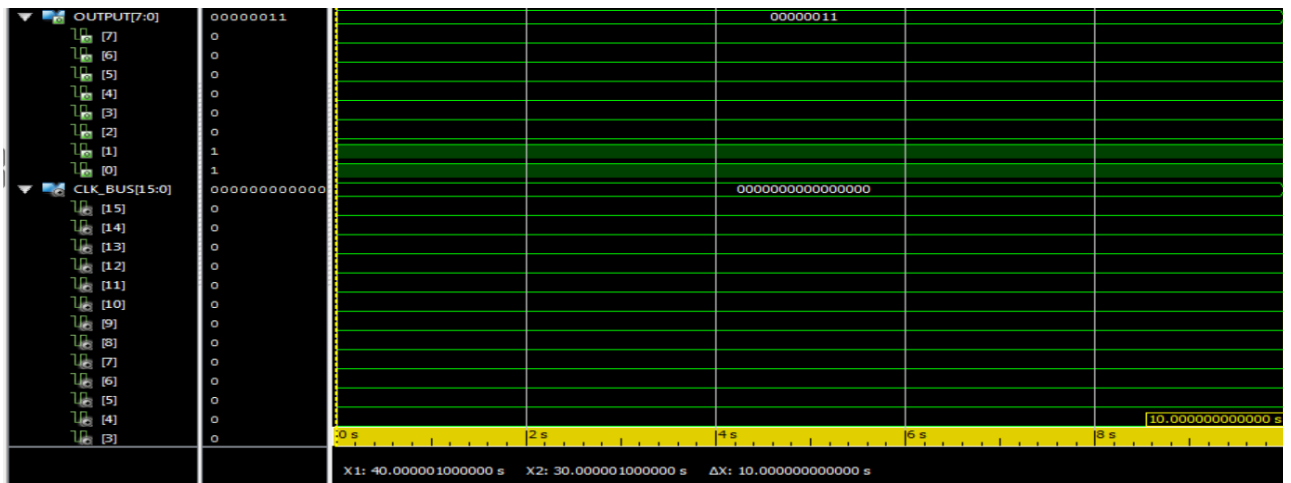
Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 1)



Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 1)



Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 1)



TEST BENCH:

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY TOP_SCHEME_TOP_SCHEME_sch_tb IS
END TOP_SCHEME_TOP_SCHEME_sch_tb;
ARCHITECTURE behavioral OF TOP_SCHEME_TOP_SCHEME_sch_tb IS

    COMPONENT TOP_SCHEME
    PORT( CLOCK : IN STD_LOGIC;
          RESET : IN STD_LOGIC;
          SPEED : IN STD_LOGIC;
          OUTPUT : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
          MODE : IN STD_LOGIC);
    END COMPONENT;

    SIGNAL CLOCK : STD_LOGIC := '0';
    SIGNAL RESET : STD_LOGIC;
    SIGNAL SPEED : STD_LOGIC;
    
```

```
SIGNAL OUTPUT : STD_LOGIC_VECTOR (7 DOWNT0 0);  
SIGNAL MODE : STD_LOGIC;
```

```
BEGIN
```

```
CLOCK <= not CLOCK after 83ns;  
UUT: TOP_SCHEME PORT MAP(  
  CLOCK => CLOCK,  
  RESET => RESET,  
  SPEED => SPEED,  
  OUTPUT => OUTPUT,  
  MODE => MODE  
);
```

```
-- *** Test Bench - User Defined Section ***
```

```
tb : PROCESS
```

```
BEGIN
```

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

```
assert OUTPUT = "00000011";  
wait for 696255us;  
assert OUTPUT = "00000110";  
wait for 1392509us;  
assert OUTPUT = "00001100";  
wait for 1392509us;  
assert OUTPUT = "00011000";  
wait for 1392509us;  
assert OUTPUT = "00110000";  
wait for 1392509us;  
assert OUTPUT = "01100000";  
wait for 1392509us;  
assert OUTPUT = "11000000";  
wait for 1392509us;  
assert OUTPUT = "10000000";  
wait for 1392509us;
```

```
SPEED <= '1';  
MODE <= '1';  
RESET <= '1', '0' after 1ms;  
wait until RESET = '0';
```

```
assert OUTPUT = "00000011";  
wait for 175065us;  
assert OUTPUT = "10000000";  
wait for 348149us;  
assert OUTPUT = "11000000";  
wait for 348149us;  
assert OUTPUT = "01100000";  
wait for 348149us;
```

```
assert OUTPUT = "00110000";
wait for 348149us;
assert OUTPUT = "00011000";
wait for 348149us;
assert OUTPUT = "00001100";
wait for 348149us;
assert OUTPUT = "00000110";
wait for 348149us;
```

```
MODE <= '0';
SPEED <= '1';
RESET <= '1', '0' after 1ms;
```

```
wait until RESET = '0';
assert OUTPUT = "00000011";
wait for 175065us;
assert OUTPUT = "10000000";
wait for 348149us;
assert OUTPUT = "11000000";
wait for 348149us;
assert OUTPUT = "01100000";
wait for 348149us;
assert OUTPUT = "00110000";
wait for 348149us;
assert OUTPUT = "00011000";
wait for 348149us;
assert OUTPUT = "00001100";
wait for 348149us;
assert OUTPUT = "00000110";
wait for 348149us;
```

```
MODE <= '1';
SPEED <= '0';
RESET <= '1', '0' after 1ms;
wait until RESET = '0';
```

```
assert OUTPUT = "00000011";
wait for 696255us;
assert OUTPUT = "00000110";
wait for 1392509us;
assert OUTPUT = "00001100";
wait for 1392509us;
assert OUTPUT = "00011000";
wait for 1392509us;
assert OUTPUT = "00110000";
wait for 1392509us;
assert OUTPUT = "01100000";
wait for 1392509us;
assert OUTPUT = "11000000";
wait for 1392509us;
assert OUTPUT = "10000000";
```

```
wait for 1392509us;

MODE <= '0';
SPEED <= '0';
RESET <= '1', '0' after 1ms;
wait until RESET = '0';
```

```
END PROCESS;
```

```
...-- *** End Test Bench - User Defined Section ***
```

```
END;
```



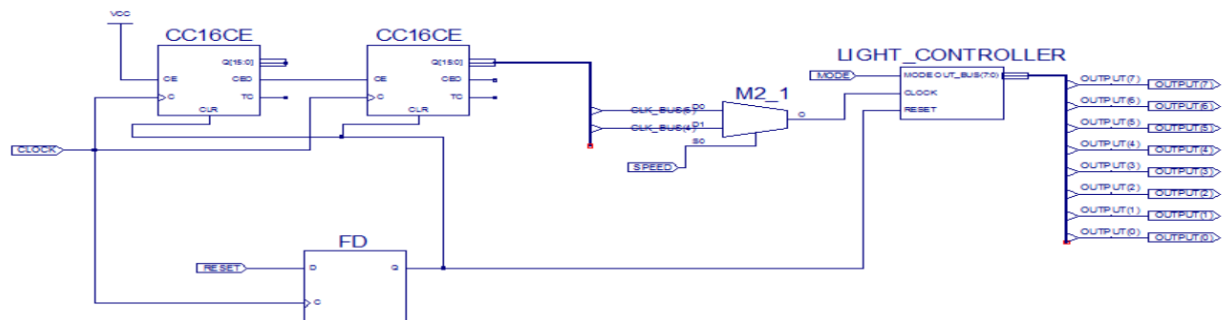
Console

```
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# run 10s
ISim>
# run 10s
ISim>
```

Below the console window are tabs for: Console, Compilation Log, Breakpoints, Find in Files Results, and Search Results.

Результати TEST BENCH

Автомат світлових сигналів та подільник тактового сигналу



Висновок:

В ході виконання цієї лабораторної роботи я реалізувала на базі стенда Elbert V2 – Spartan3A FPGA цифровий автомат світлових ефектів згідно заданих вимог