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

Виконав: ст. гр. КІ-201

Гришканич А. М.

Прийняв:

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:

Табл.1.1 Вихідні сигнали для кожного стану.

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	1
1	0	1	0	0	0	0	1	0
2	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	0
4	0	0	1	1	1	1	0	0
5	0	1	1	1	1	1	1	0
6	1	1	1	1	1	1	1	1
7	0	0	0	0	0	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 (SPEED):
  - Якщо *SPEED=0* то автомат працює зі швидкістю, визначеною за замовчуванням.
  - Якщо *SPEED=1* то автомат працює зі швидкістю, В 4 РАЗИ НИЖЧОЮ 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:

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

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
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;
```

Рис.2.1. VHDL опис логіки переходів.

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 (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0)))) after 1 ns;

OUT_BUS(1) <= ((not(IN_BUS(1)) and IN_BUS(0)) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0)))) after 1 ns;

OUT_BUS(2) <= ((IN_BUS(1) and not(IN_BUS(0))) or (IN_BUS(2) and not(IN_BUS(1)))) after 1 ns;

OUT_BUS(3) <= ((IN_BUS(2) and not(IN_BUS(1))) or (IN_BUS(2) and not(IN_BUS(0))) or (not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0))) after 1 ns;

OUT_BUS(4) <= ((IN_BUS(2) and not(IN_BUS(1))) or (IN_BUS(2) and not(IN_BUS(0))) or (not(IN_BUS(2)) and IN_BUS(1) and IN_BUS(0))) after 1 ns;

OUT_BUS(5) <= ((IN_BUS(1) and not(IN_BUS(0))) or (IN_BUS(2) and not(IN_BUS(1)))) after 1 ns;

OUT_BUS(6) <= ((not(IN_BUS(1)) and IN_BUS(0)) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0)))) after 1 ns;

OUT_BUS(7) <= ((not(IN_BUS(2)) and not(IN_BUS(1)) and not(IN_BUS(0))) or (IN_BUS(2) and IN_BUS(1) and not(IN_BUS(0)))) after 1 ns;

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

end OUTPUT_LOGIC_ARCH;

```

Рис.2.4. VHDL опис вихідних сигналів.

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

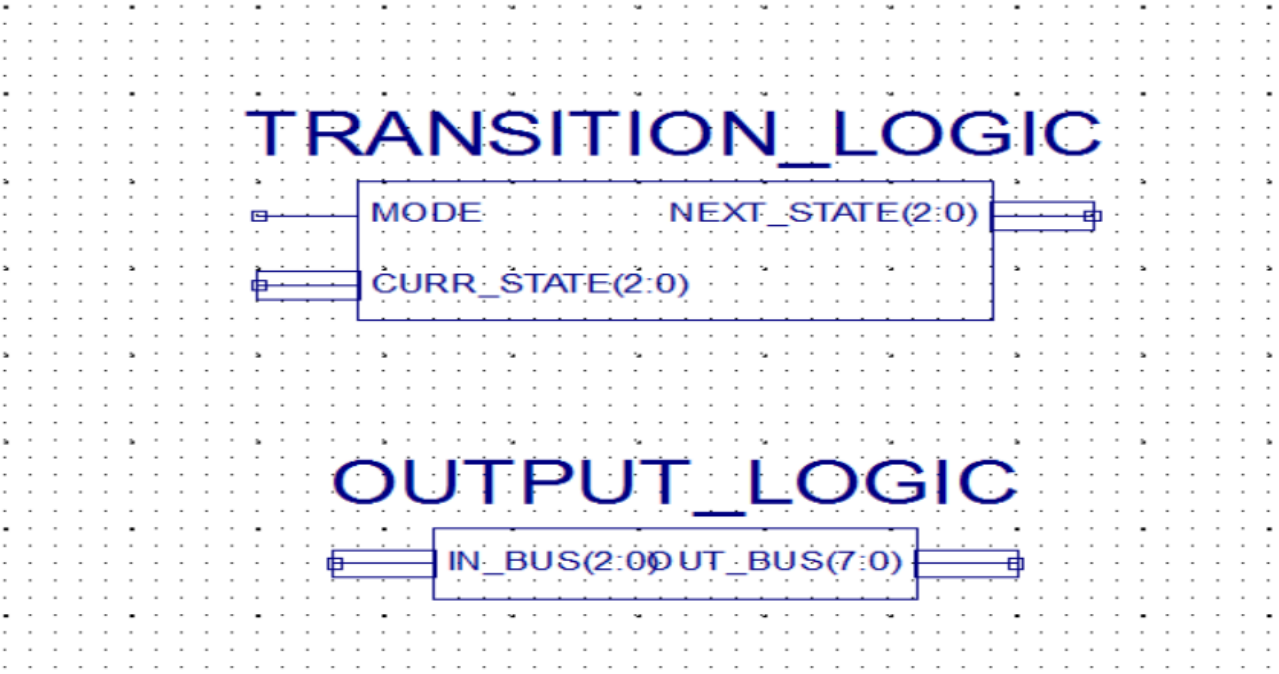

Рис.2.5. Згенеровані схематичні символи.

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.

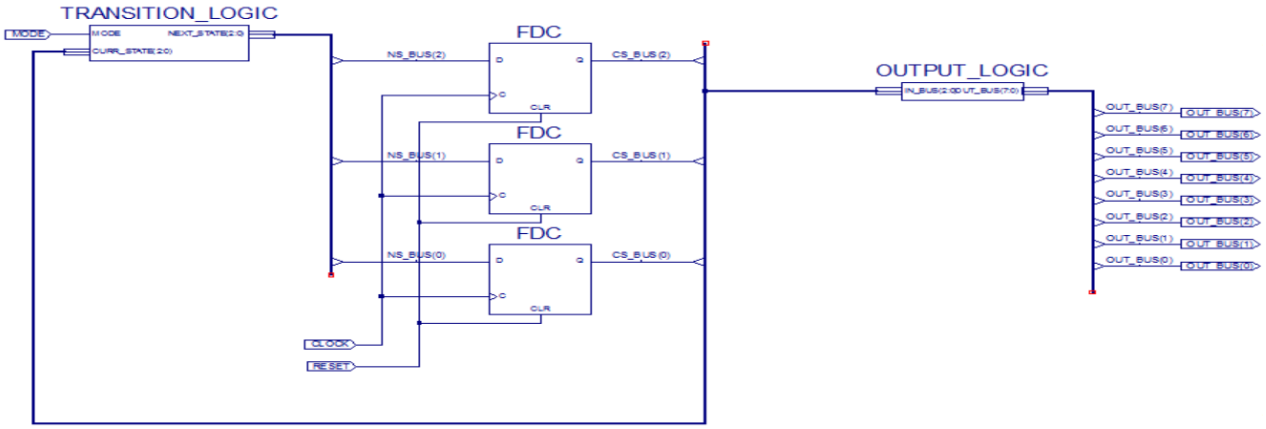

Рис.2.6. Інтеграція всіх створених компонентів разом з пам'ятю стану автомата.

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

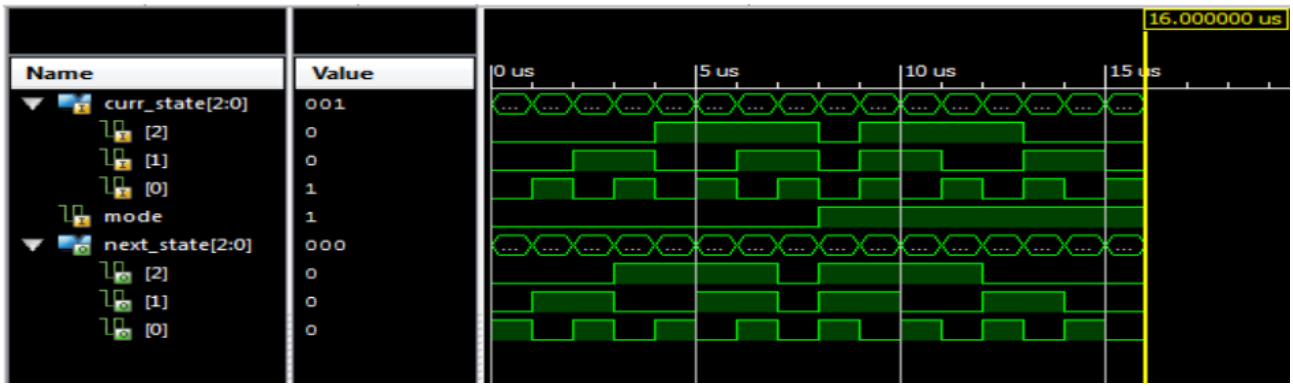

Рис.2.7. Результати симуляції логіки переходів в ISim.

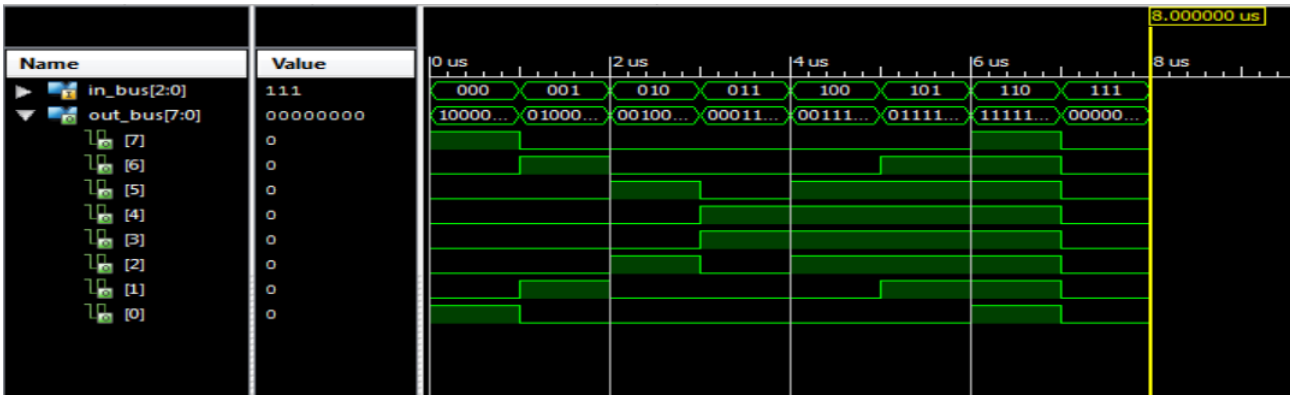

Рис.2.8. Результати симуляції логіки вихідних сигналів в ISim.

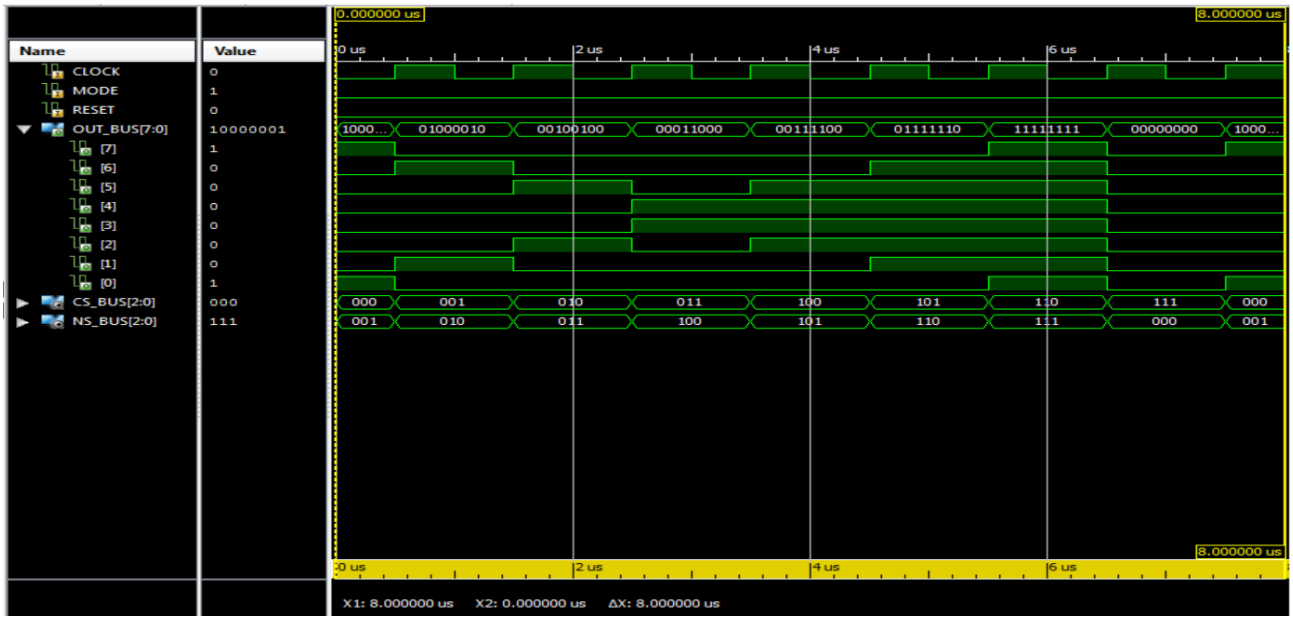

Рис.2.9. Результати симуляції автомата (MODE = 0, RESET = 0).

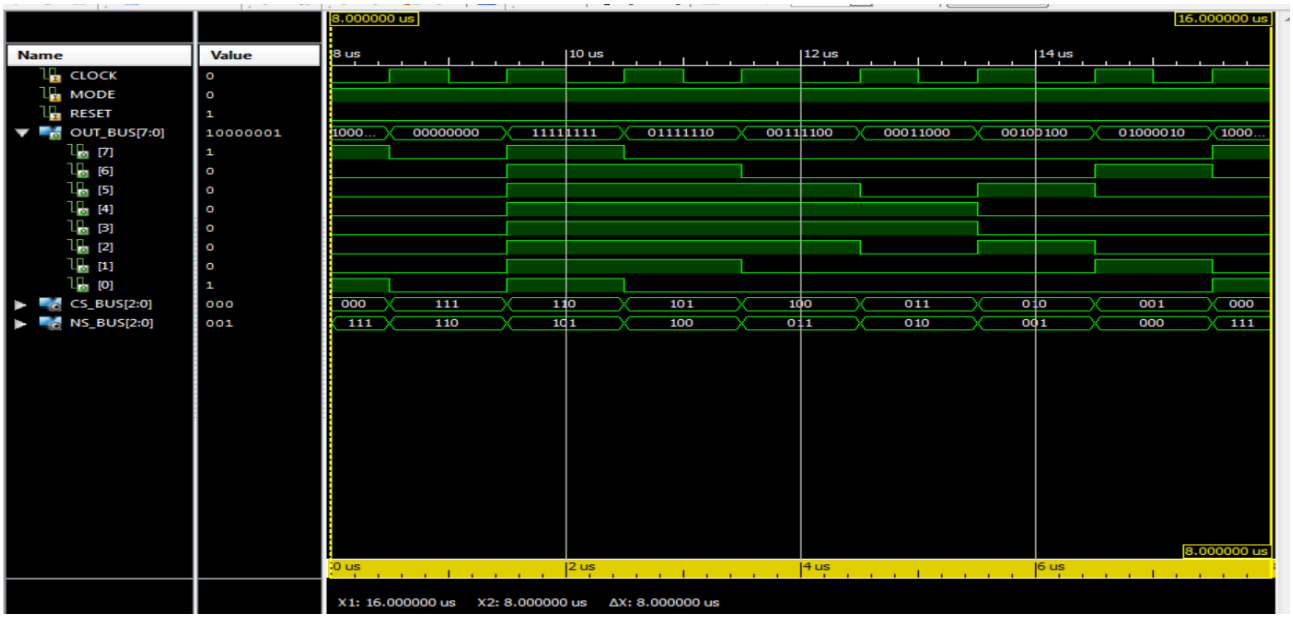

Рис.2.10. Результати симуляції автомата (MODE = 1, RESET = 0).

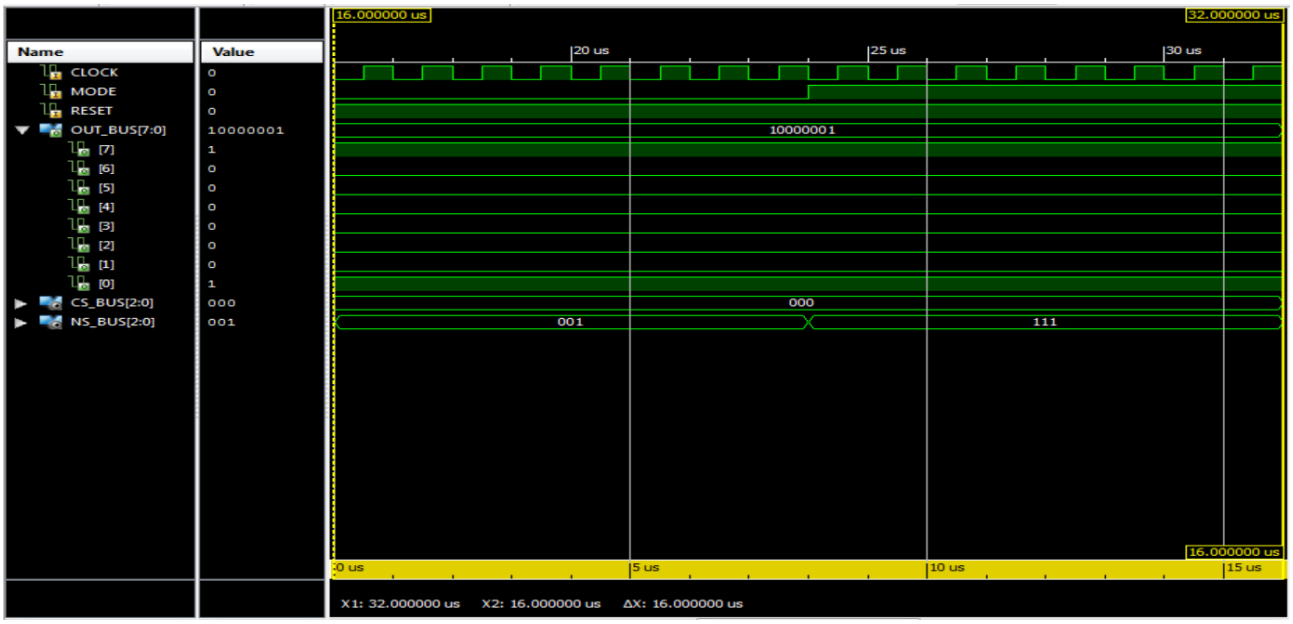

Рис.2.11. Результати симуляції автомата (MODE = 0, RESET = 1).

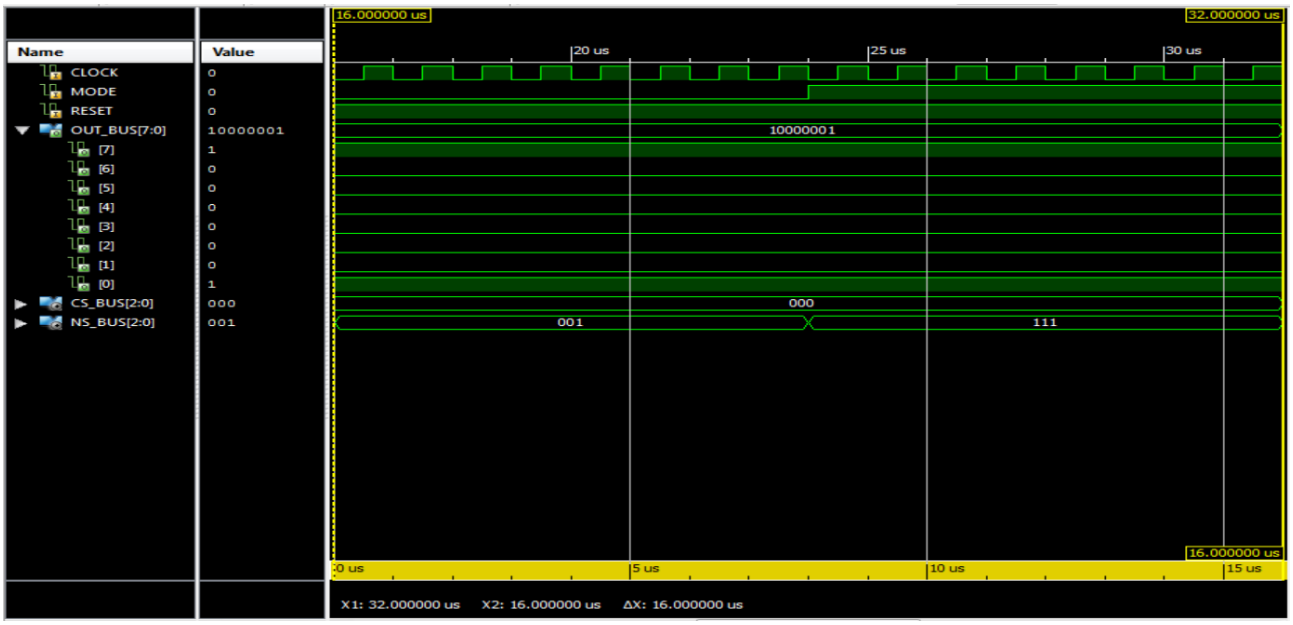

Рис.2.12. Результати симуляції автомата (MODE = 1, RESET = 1).

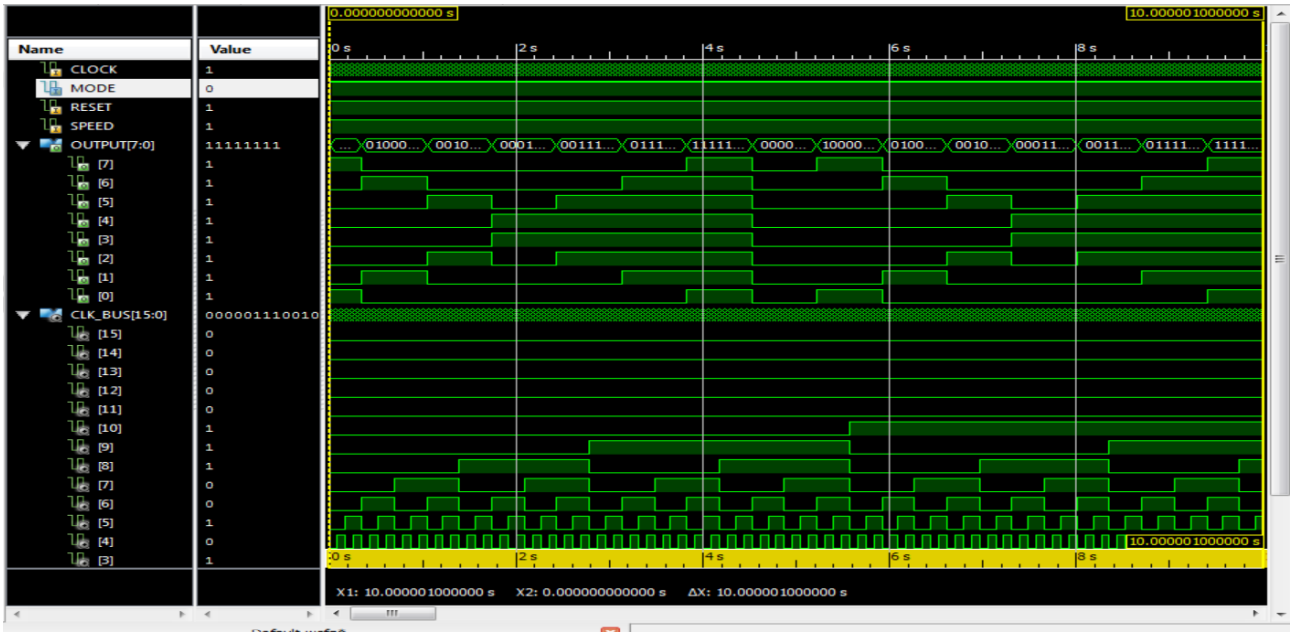

Рис.2.13. Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 0).

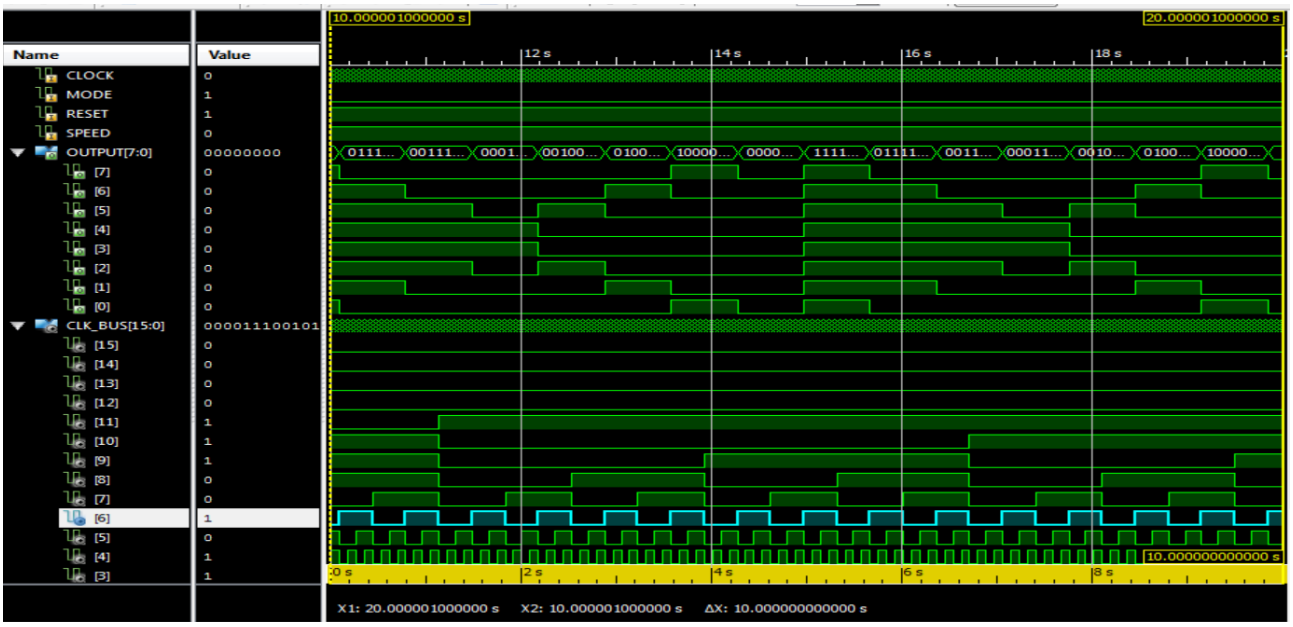

Рис.2.14. Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 0).

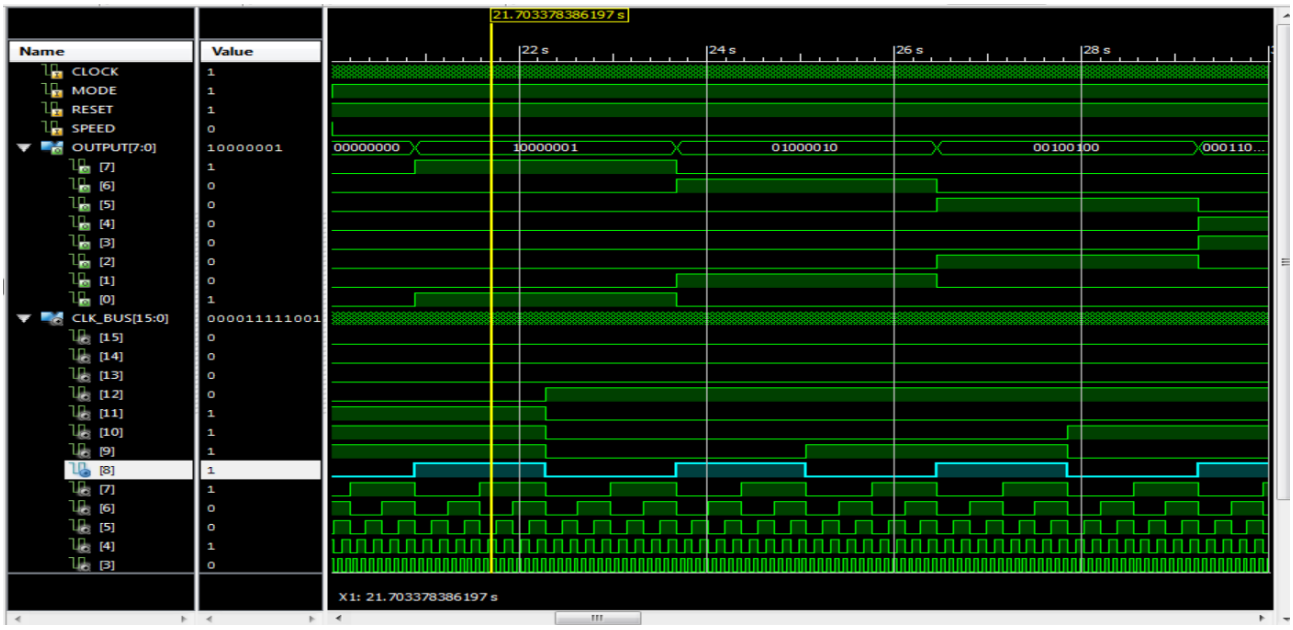

Рис.2.15. Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 0).

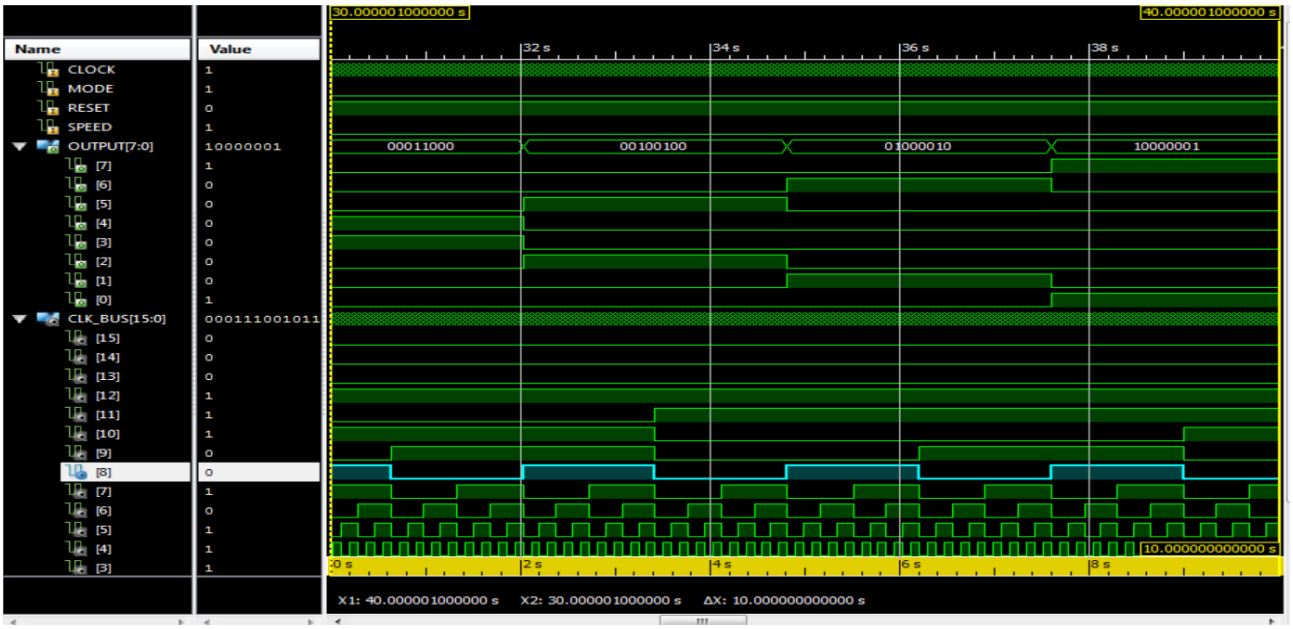

Рис.2.16. Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 0).

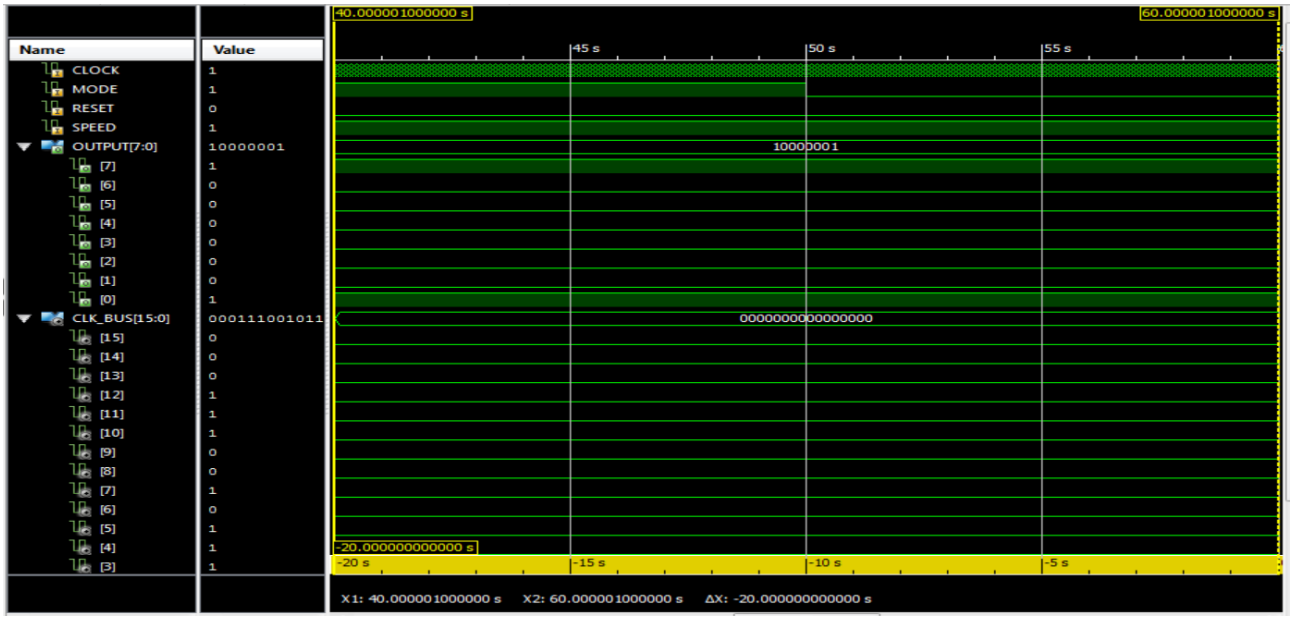

Рис.2.17. Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 1).

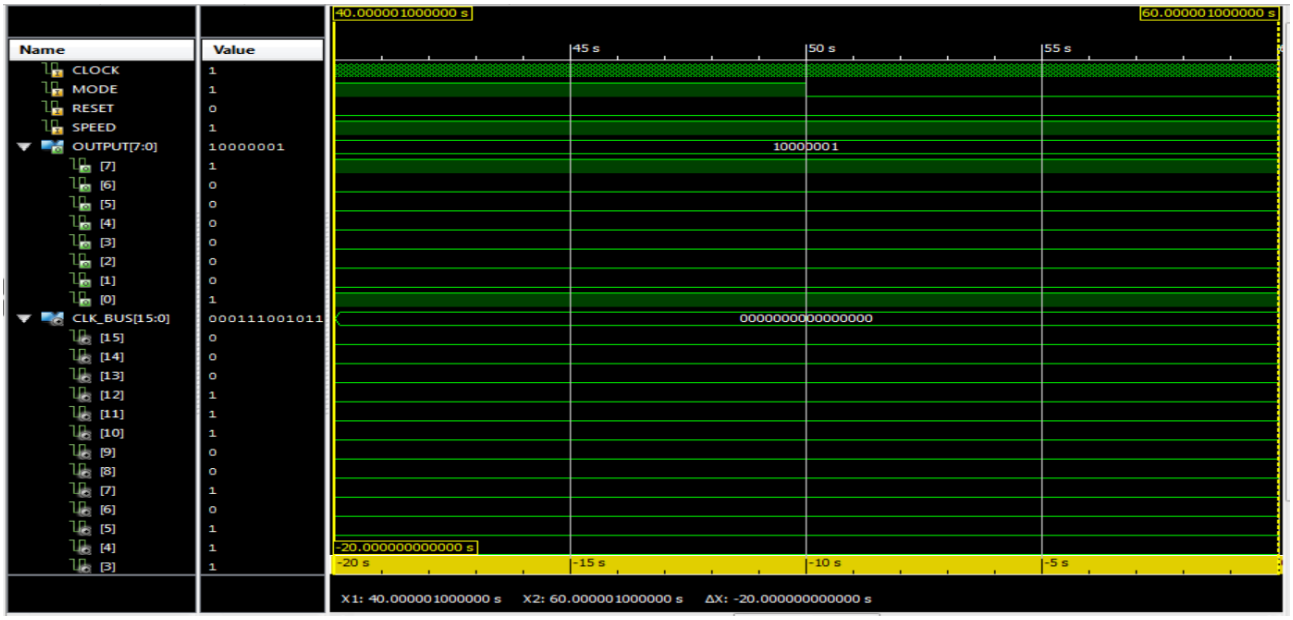

Рис.2.18. Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 1).

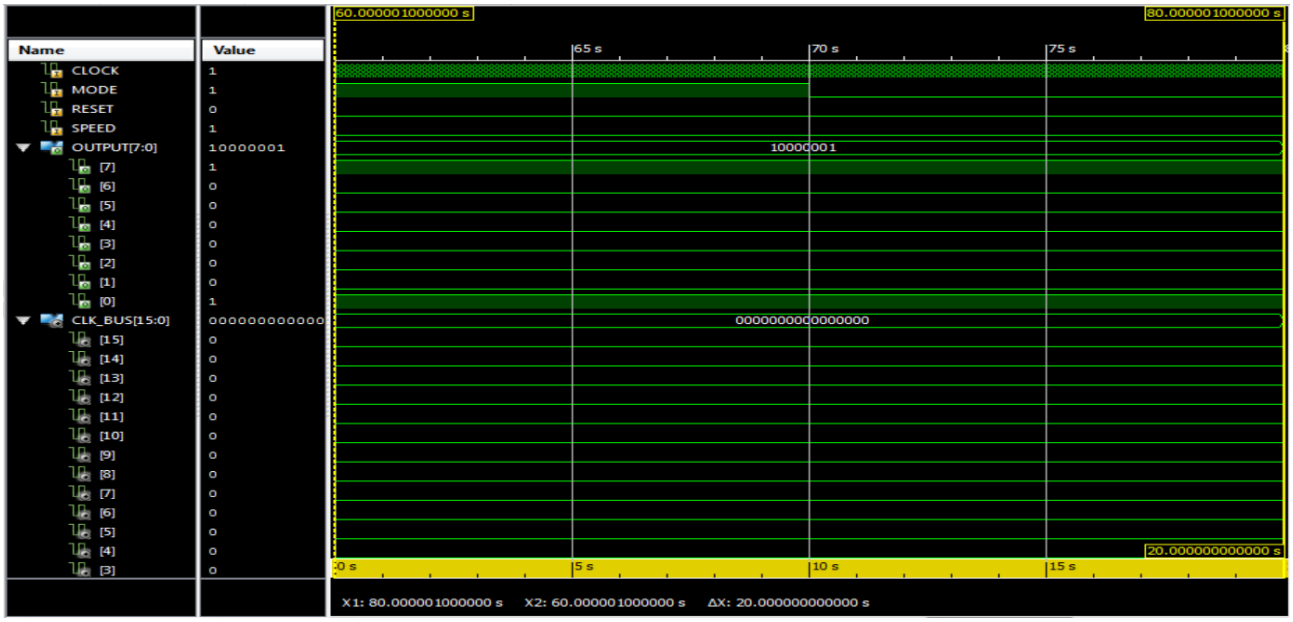

Рис.2.19. Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 1).

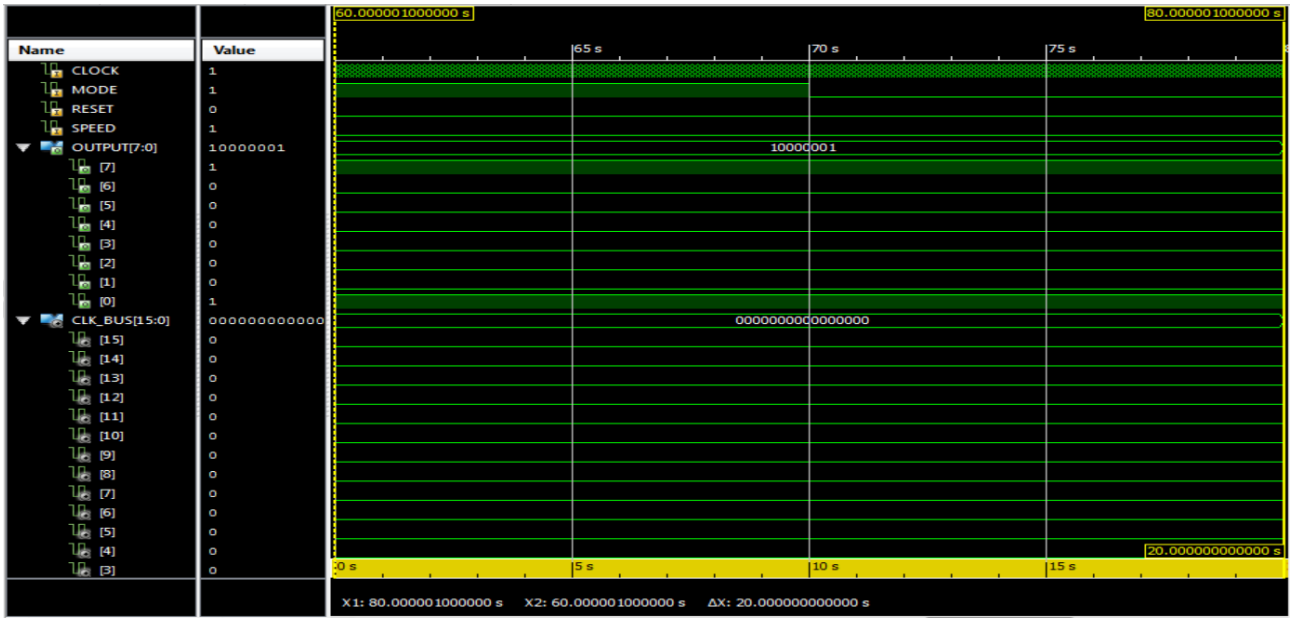

Рис.2.20. Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 1).

TEST BENCH:

```
 -- Vhdl test bench created from schematic
C:\mks\projectsXILINX\LAB_2\LAB_2_V4\POHREBNYAK_LAB2\TOP_SCHEM
E.sch - Mon Mar 25 22:31:01 2024
--
-- Notes:
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
```

```
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--

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 DOWNTO 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 600ms;
    wait until RESET = '0';
```

```
assert OUTPUT = "10000001";
wait for 174064us;
assert OUTPUT = "01000010";
wait for 348128us;
assert OUTPUT = "00100100";
wait for 348128us;
assert OUTPUT = "00011000";
wait for 348128us;
assert OUTPUT = "00111100";
wait for 348128us;
assert OUTPUT = "01111110";
wait for 348128us;
assert OUTPUT = "11111111";
wait for 348128us;
assert OUTPUT = "00000000";
wait for 348128us;

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

assert OUTPUT = "10000001";
wait for 696507us;
assert OUTPUT = "01000010";
wait for 1392509us;
assert OUTPUT = "00100100";
wait for 1392509us;
assert OUTPUT = "00011000";
wait for 1392509us;
assert OUTPUT = "00111100";
wait for 1392509us;
assert OUTPUT = "01111110";
wait for 1392509us;
assert OUTPUT = "11111111";
wait for 1392509us;
assert OUTPUT = "00000000";
wait for 1392509us;

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

assert OUTPUT = "10000001";
wait for 696507us;
assert OUTPUT = "00000000";
wait for 1392509us;
```

```
assert OUTPUT = "11111111";
wait for 1392509us;
assert OUTPUT = "01111110";
wait for 1392509us;
assert OUTPUT = "00111100";
wait for 1392509us;
assert OUTPUT = "00011000";
wait for 1392509us;
assert OUTPUT = "00100100";
wait for 1392509us;
assert OUTPUT = "01000010";
wait for 1392509us;

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

assert OUTPUT = "10000001";
wait for 174064us;
assert OUTPUT = "01000010";
wait for 348128us;
assert OUTPUT = "00100100";
wait for 348128us;
assert OUTPUT = "00011000";
wait for 348128us;
assert OUTPUT = "00111100";
wait for 348128us;
assert OUTPUT = "01111110";
wait for 348128us;
assert OUTPUT = "11111111";
wait for 348128us;
assert OUTPUT = "00000000";
wait for 348128us;

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

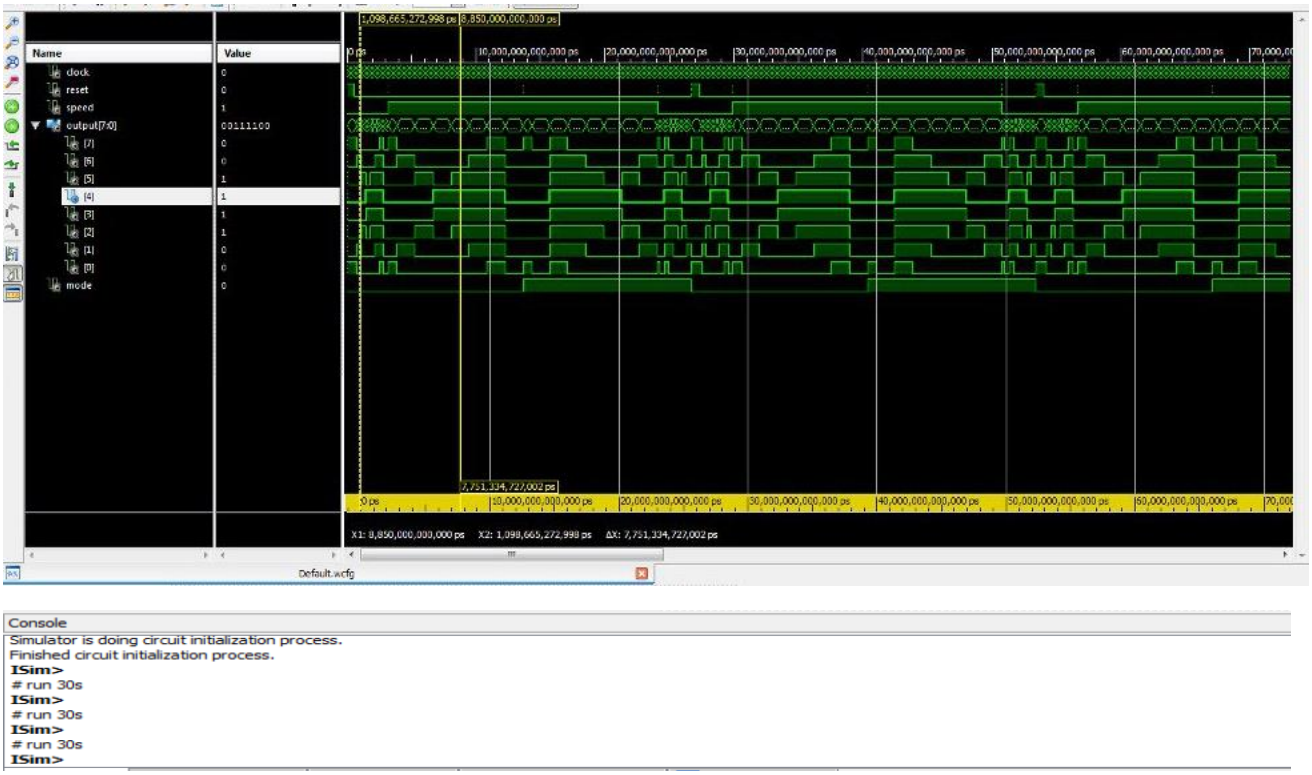

Рис.2.21. Часова діаграма.

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

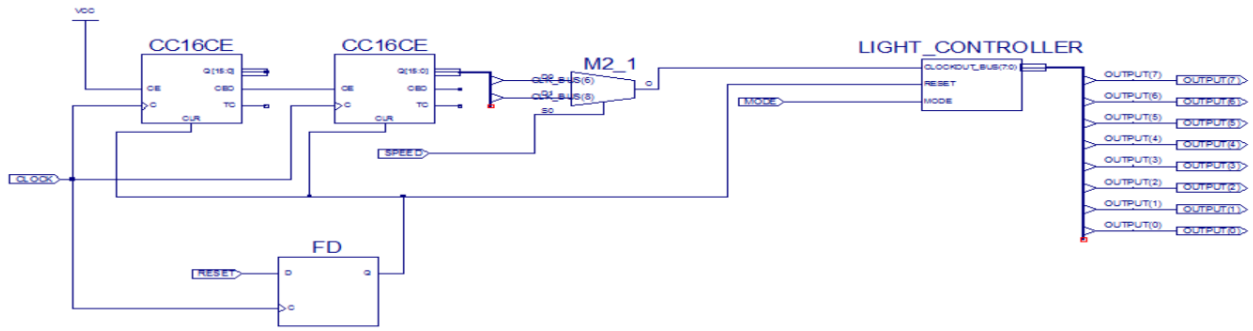

Рис.2.22. Автомат світлових сигналів та подільник тактового сигналу.

Висновок:

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