2Міністерство 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 25***

Виконав: ст. гр. КІ-21

Фундальська Д. І.

Прийняв:

Козак Н. Б.

**Львів 2024**

**Мета роботи:**

На базі стенда реалізувати цифровий автомат світлових ефектів

згідно заданих вимог.

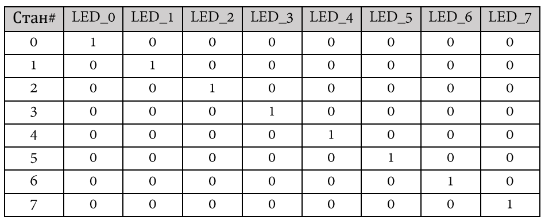
**Етапи роботи:**

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:

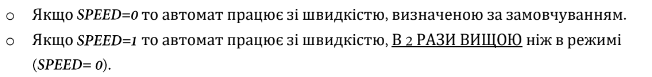
Табл.1.1 Вихідні сигнали для кожного стану..



* 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):



* Інтерфейс пристрою повинен мати однорозрядний вхід (SPEED):



* 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.

Табл.2.2.1-2.2.3. Логіка переходів для всіх станів автомата.

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(1)** |
| 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(2)** |
| 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))));

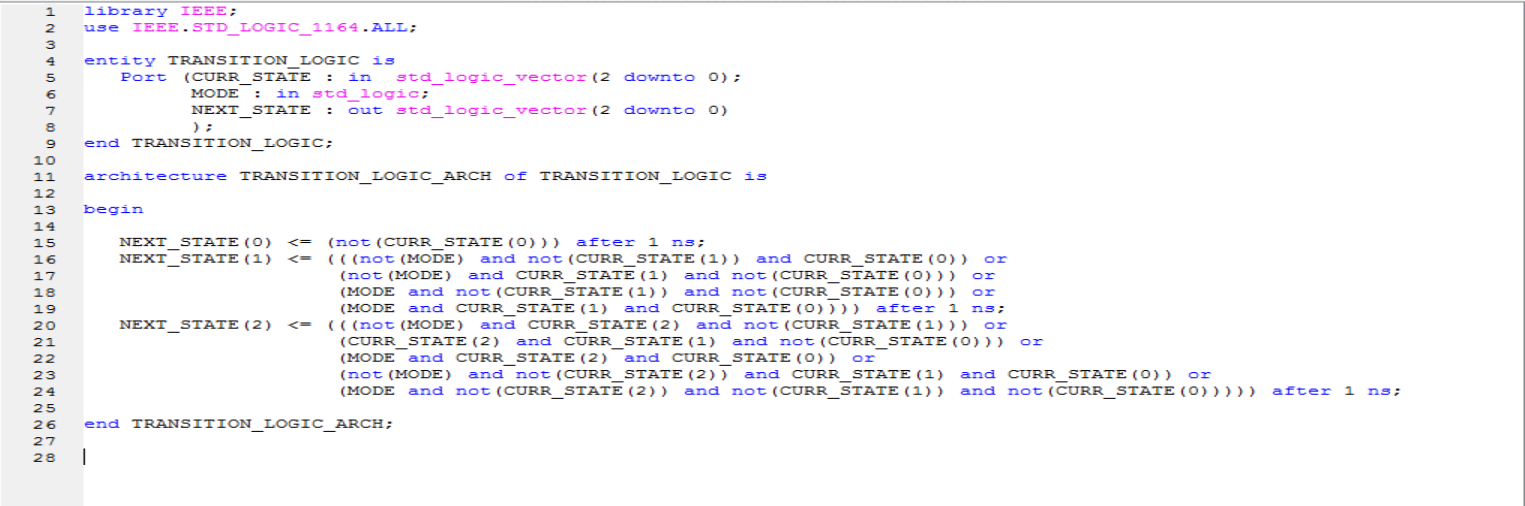


Рис.2.1. VHDL опис логіки переходів.

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)));

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));

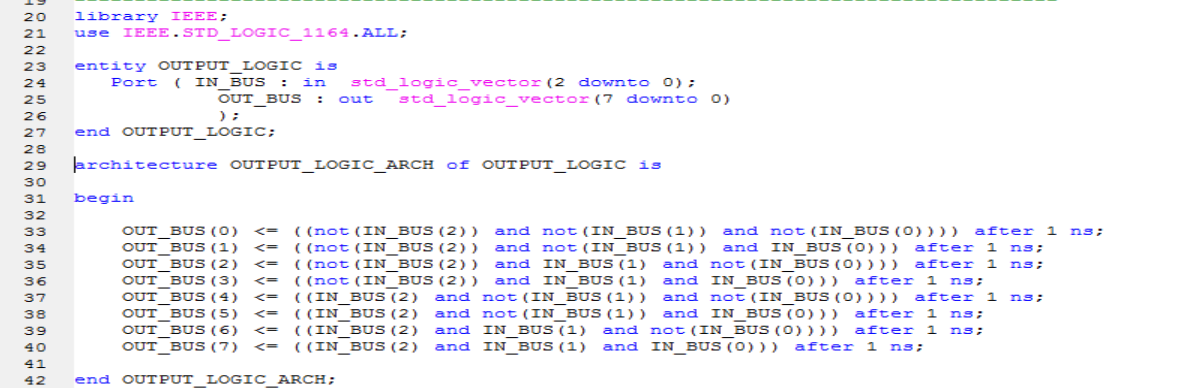


Рис.2.4. VHDL опис вихідних сигналів.

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.

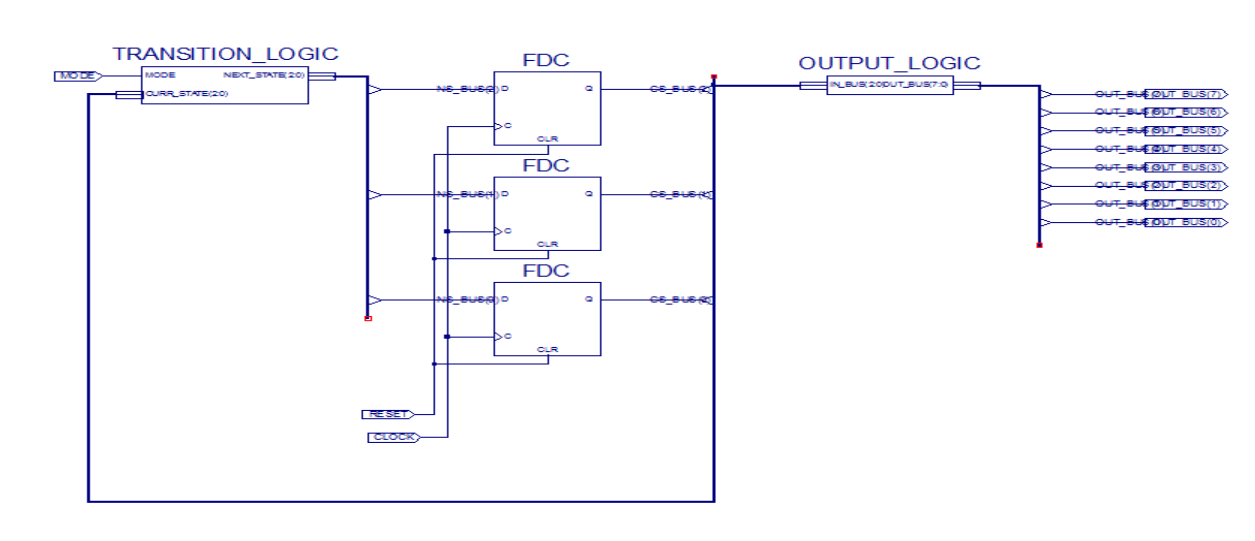


Рис.2.6. Інтеграція всіх створених компонентів разом з пам’ятю стану автомата.

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



Рис.2.7. Результати симуляції логіки переходів в ISim.

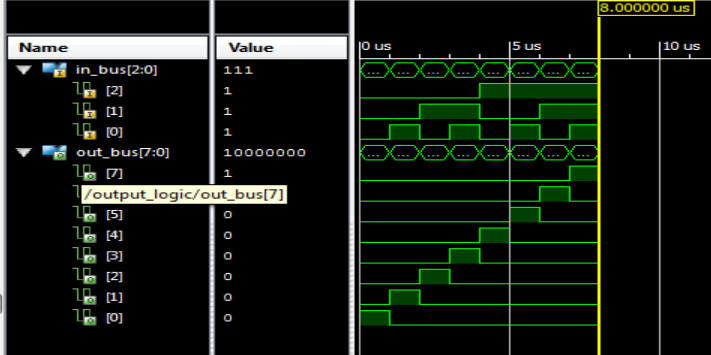


Рис.2.8. Результати симуляції логіки вихідних сигналів в ISim.

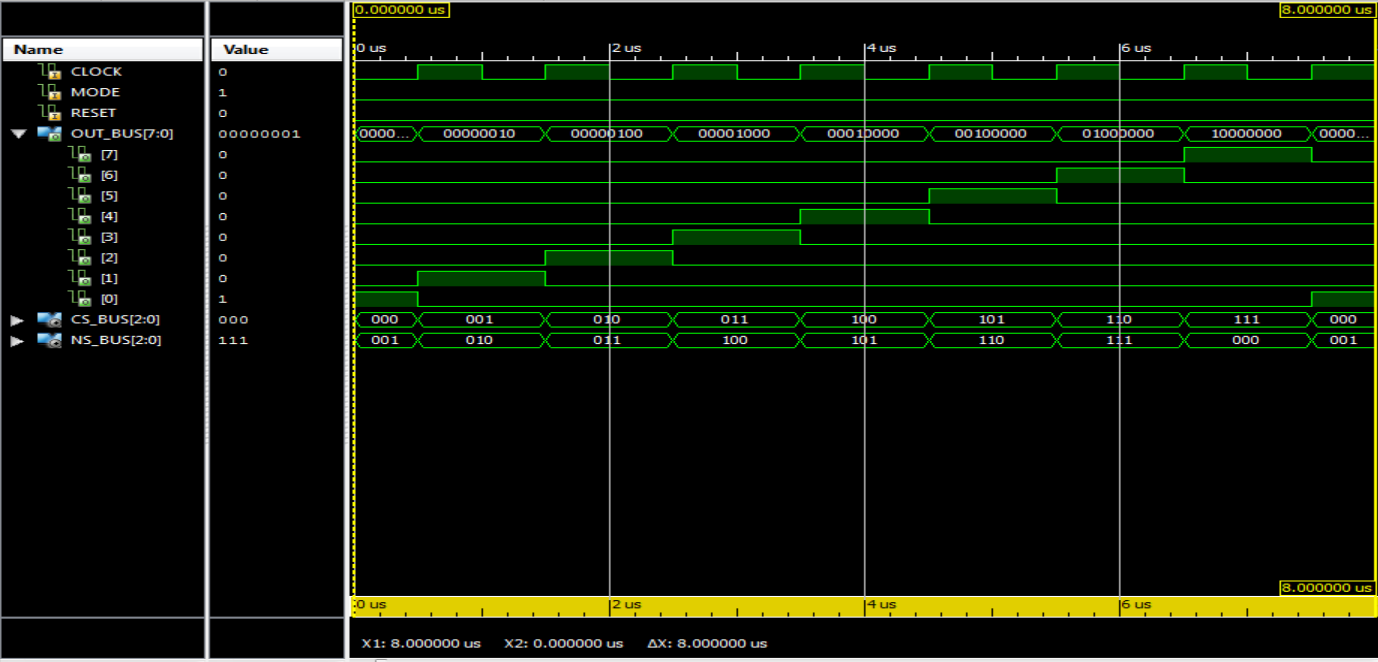


Рис.2.9. Результати симуляції автомата (MODE = 0, RESET = 0).

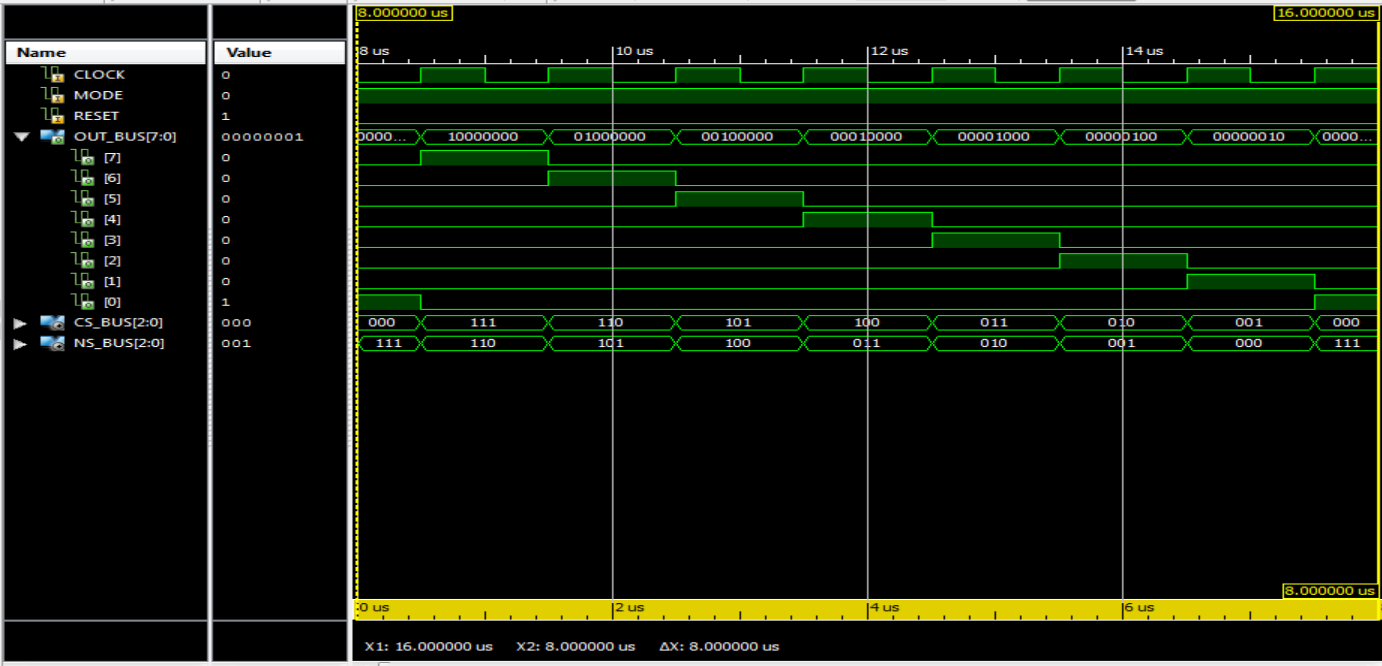


Рис.2.10. Результати симуляції автомата (MODE = 1, RESET = 0).

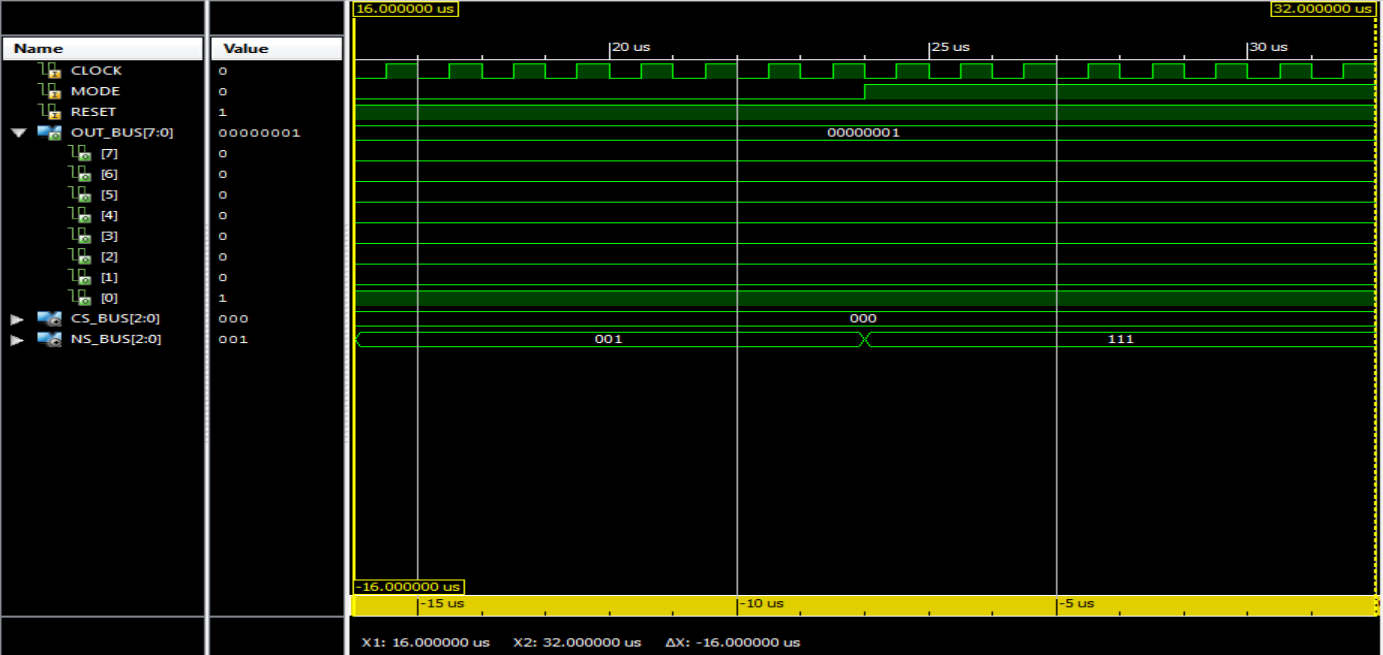


Рис.2.11. Результати симуляції автомата (MODE = 0, RESET = 1).

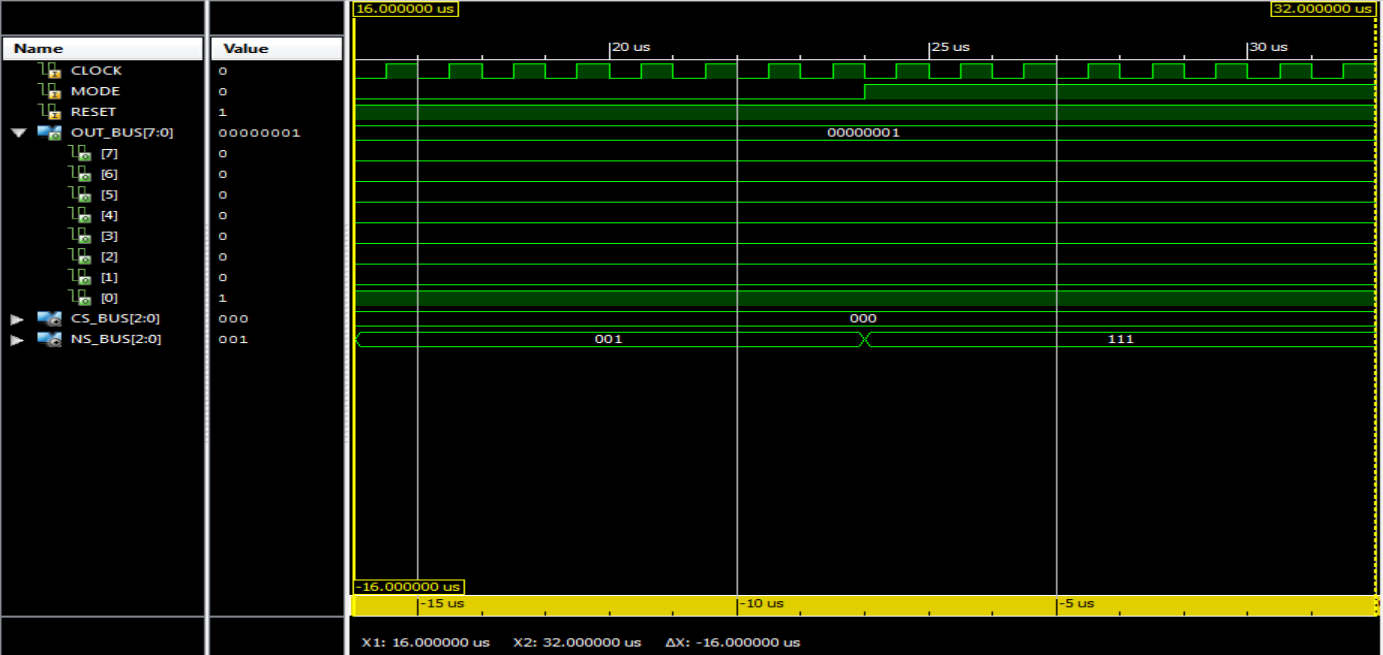


Рис.2.12. Результати симуляції автомата (MODE = 1, RESET = 1).

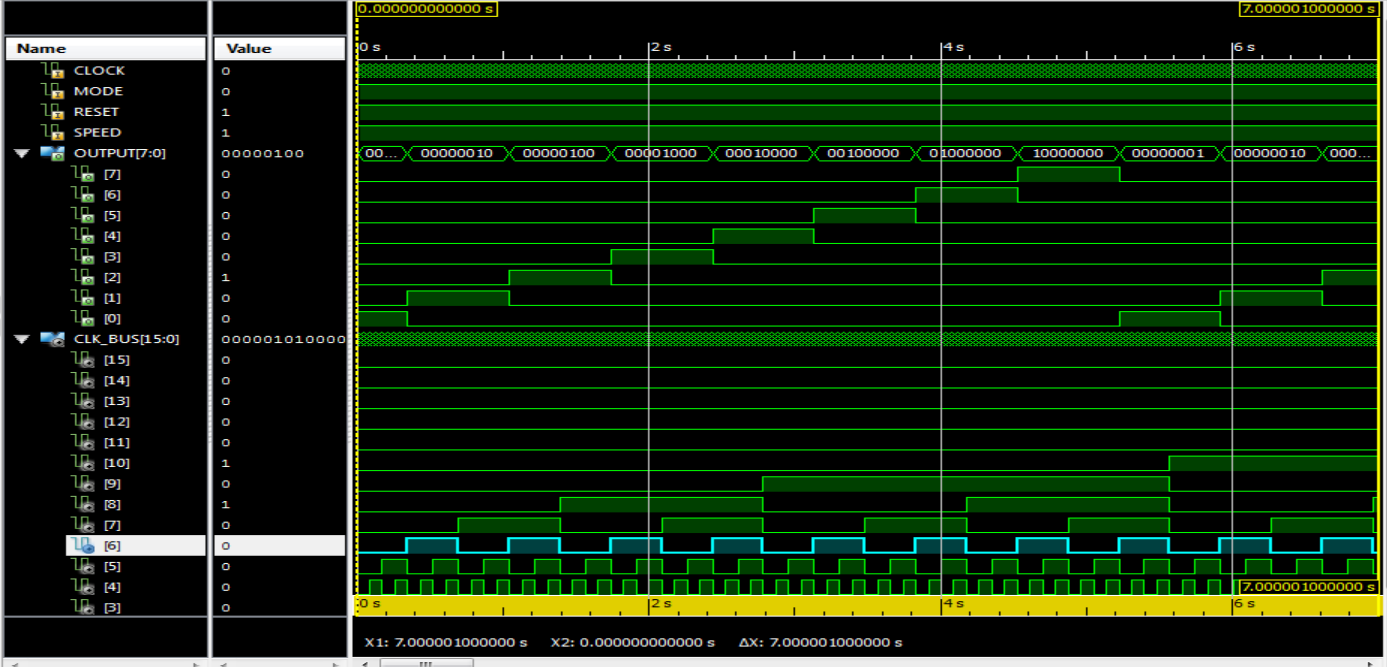


Рис.2.13. Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 0).

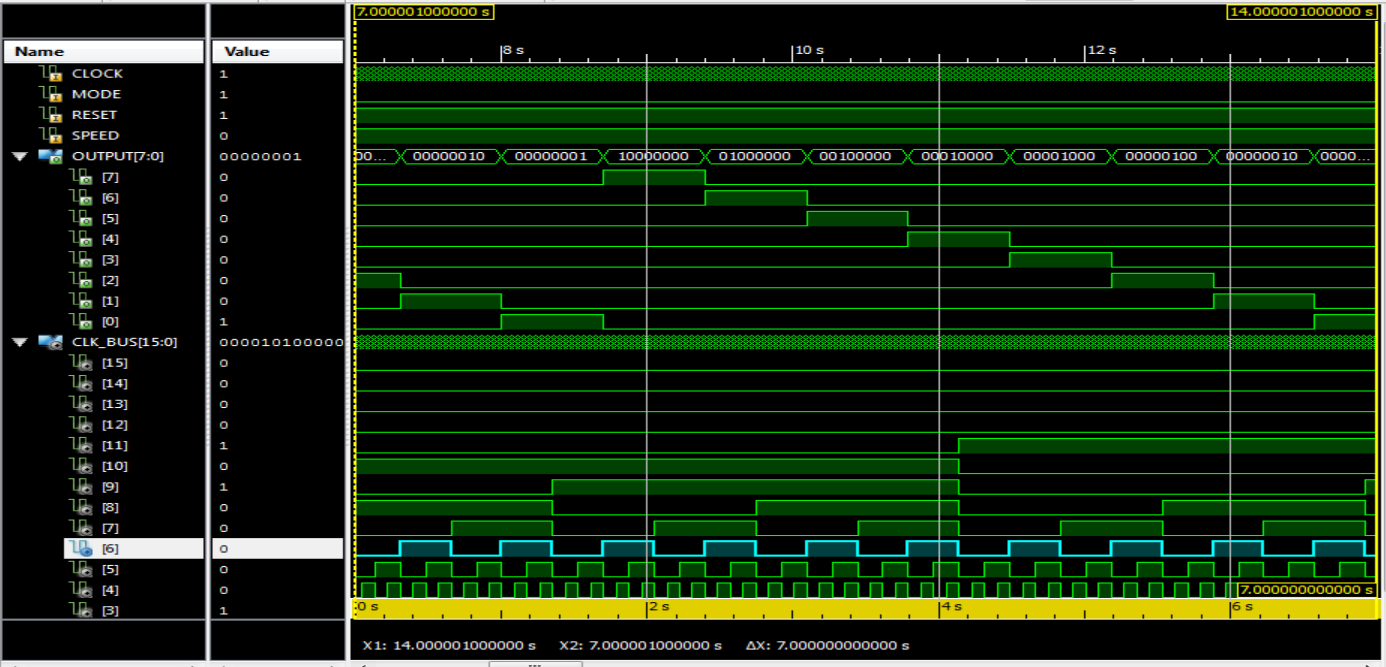


Рис.2.14. Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 0).

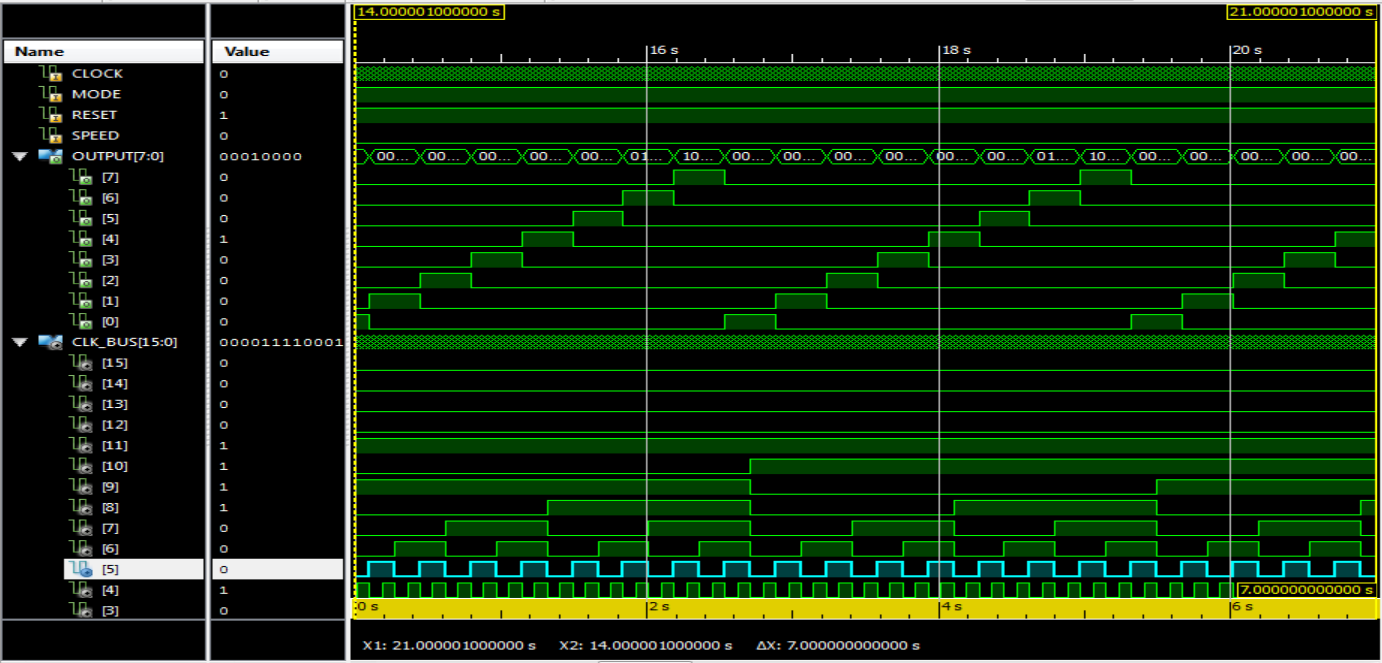


Рис.2.15. Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 0).

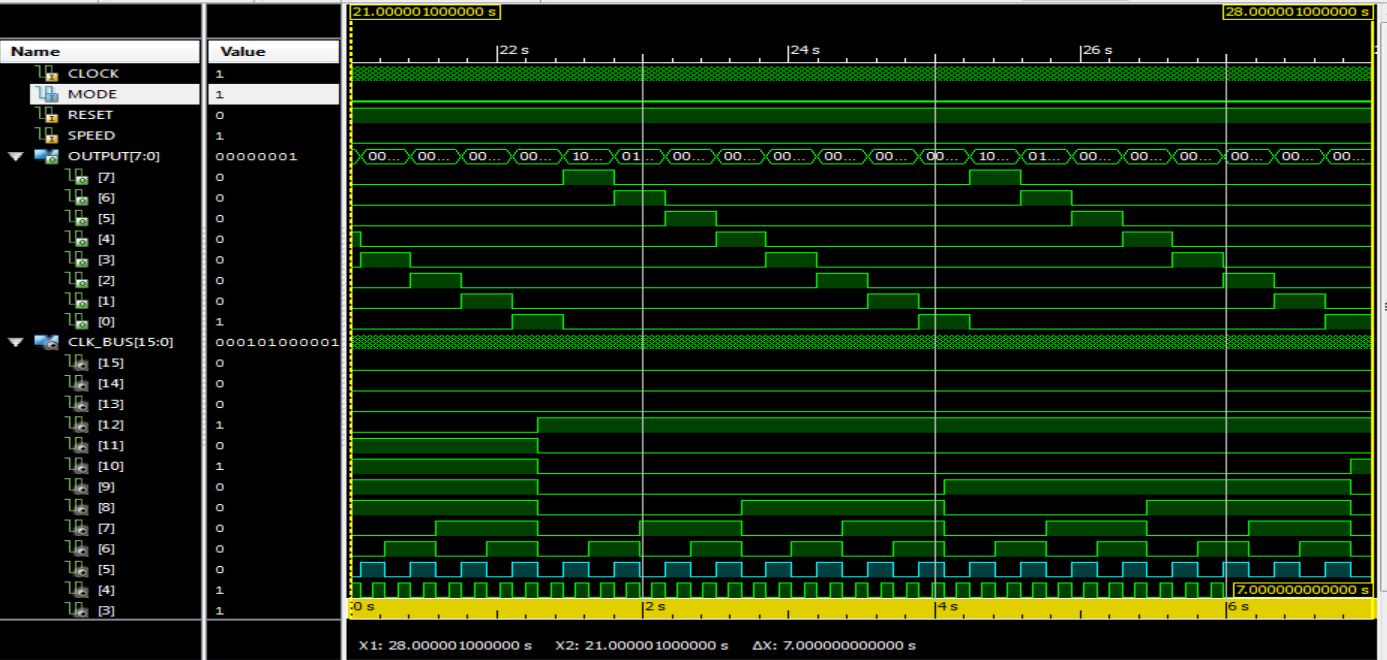


Рис.2.16. Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 0).

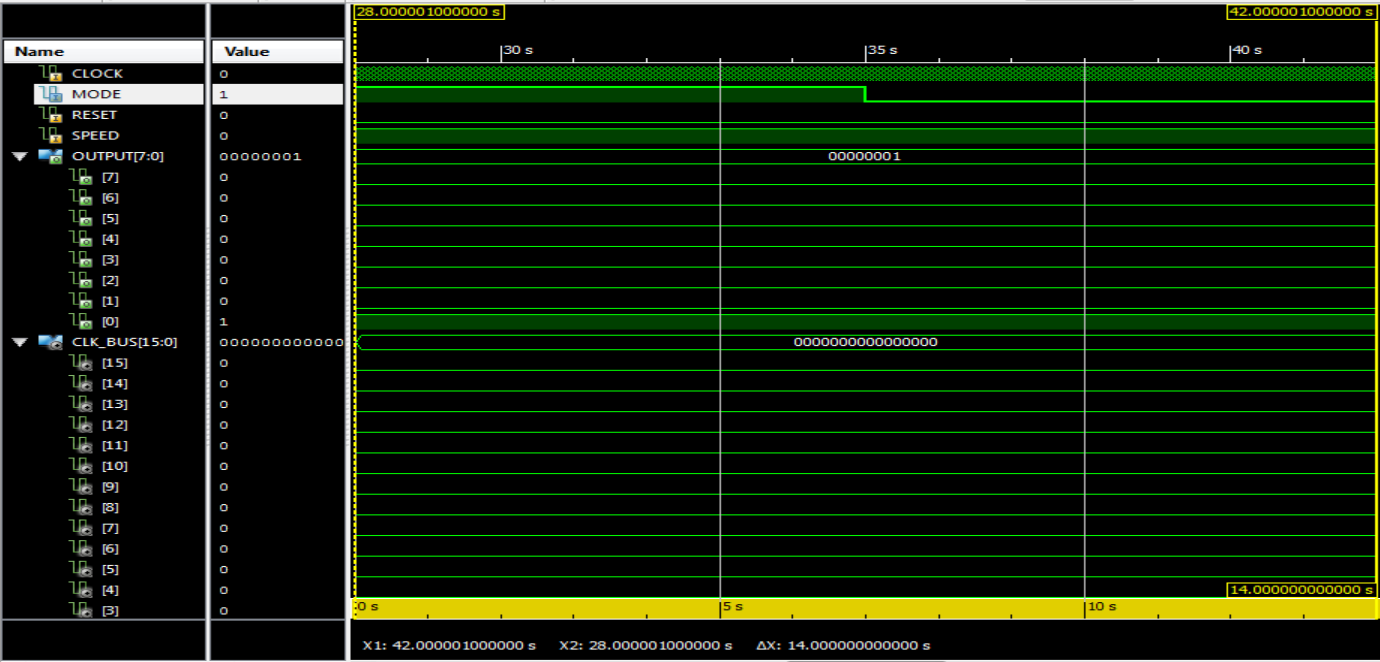


Рис.2.17. Результати симуляції фінальної схеми (MODE = 0, SPEED = 0, RESET = 1).

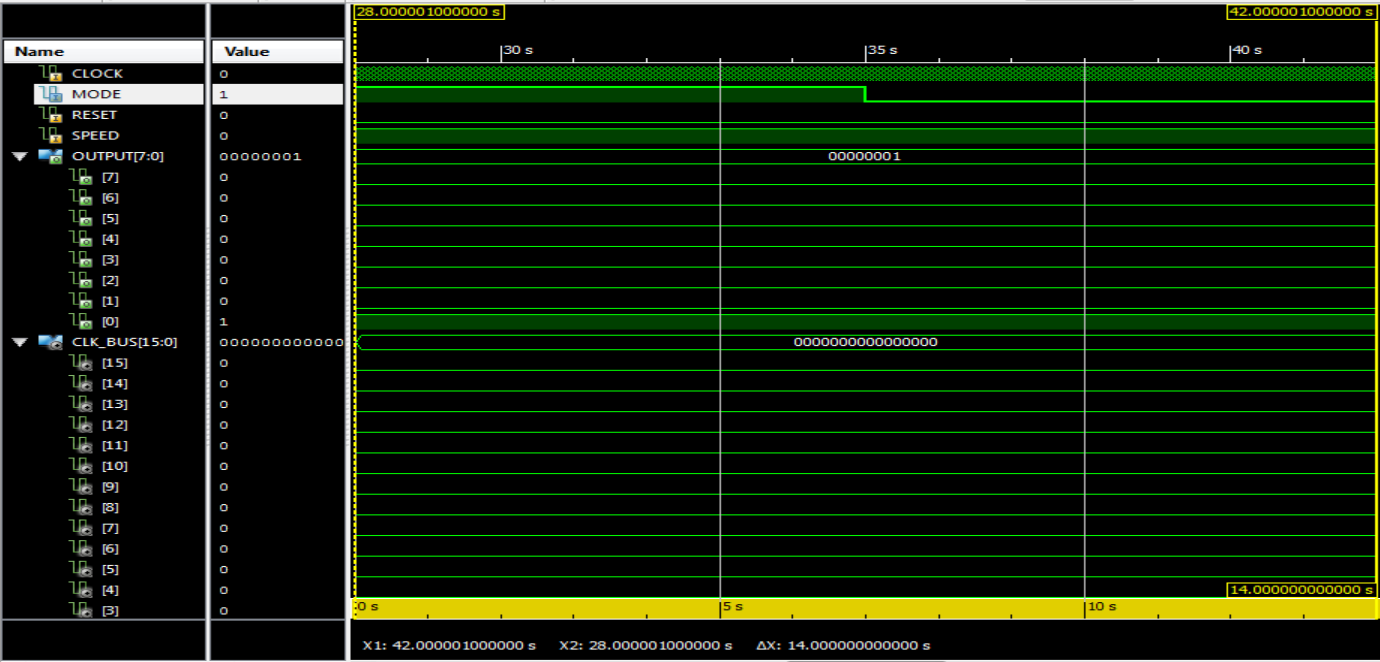


Рис.2.18. Результати симуляції фінальної схеми (MODE = 1, SPEED = 0, RESET = 1).

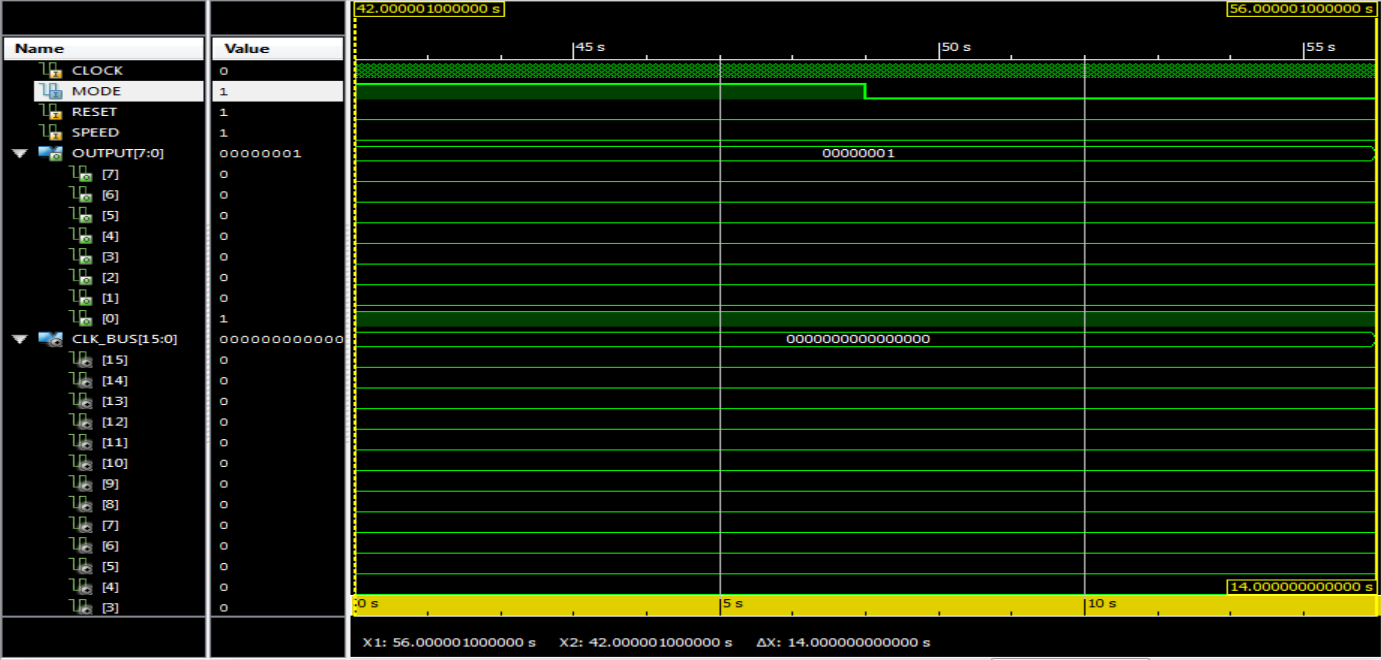


Рис.2.19. Результати симуляції фінальної схеми (MODE = 0, SPEED = 1, RESET = 1).

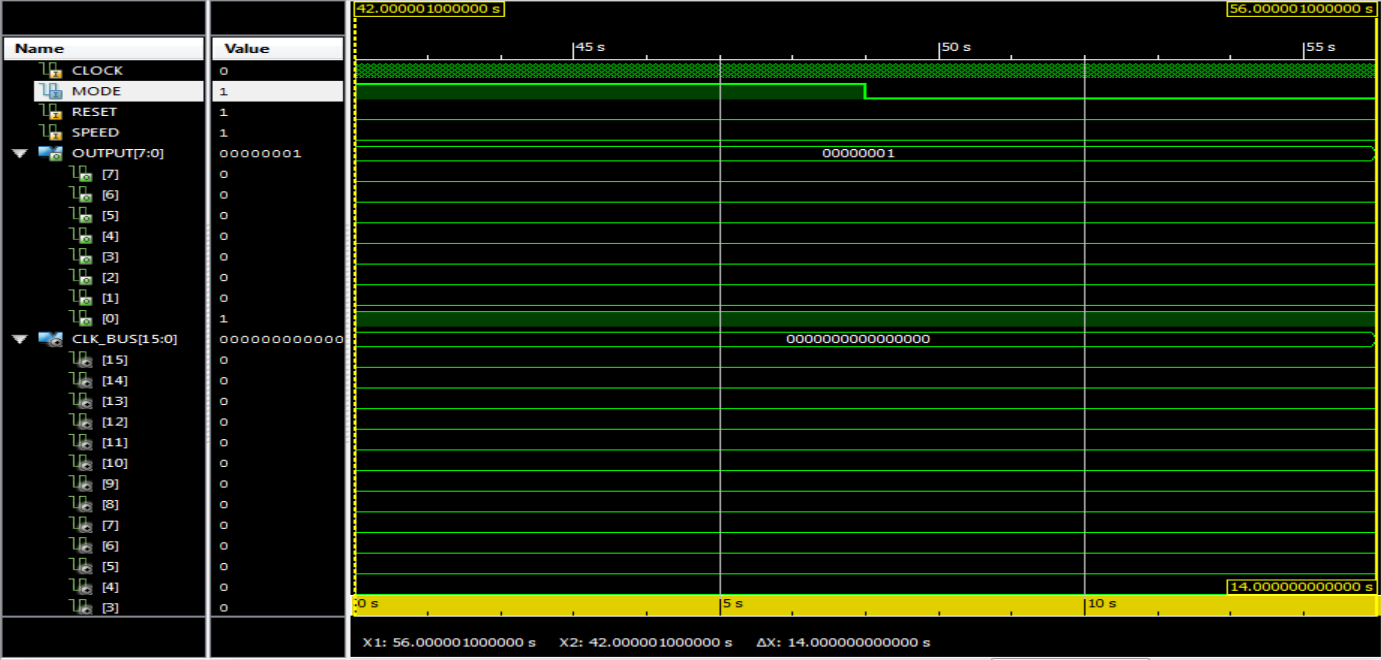


Рис.2.20. Результати симуляції фінальної схеми (MODE = 1, SPEED = 1, RESET = 1).

TEST BENCH:

-- Vhdl test bench created from schematic C:\mks\projectsXILINX\LAB\_2\LAB\_2\_V1\FIX\MIAKISHEV\_LAB2\TOP\_SCHEME.sch - Mon Mar 25 20:37:03 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 1ms;

wait until RESET = '0';

assert OUTPUT = "00000001";

wait for 696255us;

assert OUTPUT = "00000010";

wait for 1392509us;

assert OUTPUT = "00000100";

wait for 1392509us;

assert OUTPUT = "00001000";

wait for 1392509us;

assert OUTPUT = "00010000";

wait for 1392509us;

assert OUTPUT = "00100000";

wait for 1392509us;

assert OUTPUT = "01000000";

wait for 1392509us;

assert OUTPUT = "10000000";

wait for 1392509us;

MODE <= '1';

RESET <= '1', '0' after 1ms;

wait until RESET = '0';

assert OUTPUT = "00000001";

wait for 696255us;

assert OUTPUT = "10000000";

wait for 1392509us;

assert OUTPUT = "01000000";

wait for 1392509us;

assert OUTPUT = "00100000";

wait for 1392509us;

assert OUTPUT = "00010000";

wait for 1392509us;

assert OUTPUT = "00001000";

wait for 1392509us;

assert OUTPUT = "00000100";

wait for 1392509us;

assert OUTPUT = "00000010";

wait for 1392509us;

MODE <= '0';

SPEED <= '1';

RESET <= '1', '0' after 167ns;

wait until RESET = '0';

assert OUTPUT = "00000001";

wait for 348128us;

assert OUTPUT = "00000010";

wait for 696255us;

assert OUTPUT = "00000100";

wait for 696255us;

assert OUTPUT = "00001000";

wait for 696255us;

assert OUTPUT = "00010000";

wait for 696255us;

assert OUTPUT = "00100000";

wait for 696255us;

assert OUTPUT = "01000000";

wait for 696255us;

assert OUTPUT = "10000000";

wait for 696255us;

MODE <= '1';

SPEED <= '1';

RESET <= '1', '0' after 167ns;

wait until RESET = '0';

assert OUTPUT = "00000001";

wait for 348128us;

assert OUTPUT = "00000010";

wait for 696255us;

assert OUTPUT = "00000100";

wait for 696255us;

assert OUTPUT = "00001000";

wait for 696255us;

assert OUTPUT = "00010000";

wait for 696255us;

assert OUTPUT = "00100000";

wait for 696255us;

assert OUTPUT = "01000000";

wait for 696255us;

assert OUTPUT = "10000000";

wait for 696255us;

SPEED <= '0';

RESET <= '1', '0' after 167ns;

wait until RESET = '0';

END PROCESS;

-- \*\*\* End Test Bench - User Defined Section \*\*\*

END;

Рис.2.21. Файл TEST BENCH.

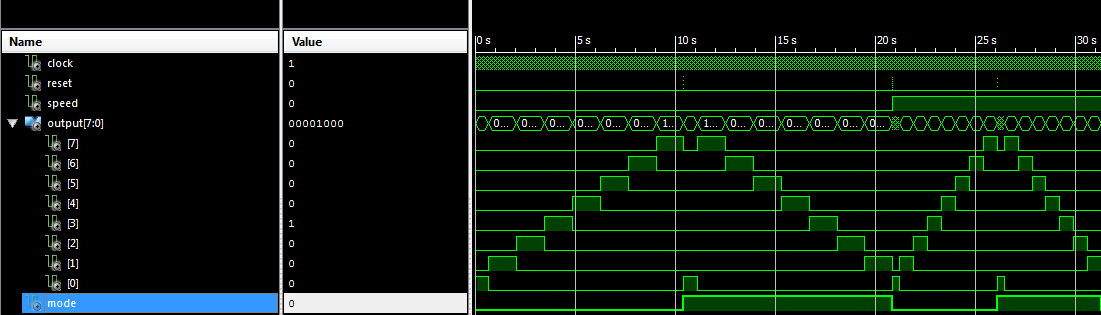


Рис.2.22. Результати виконання.

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

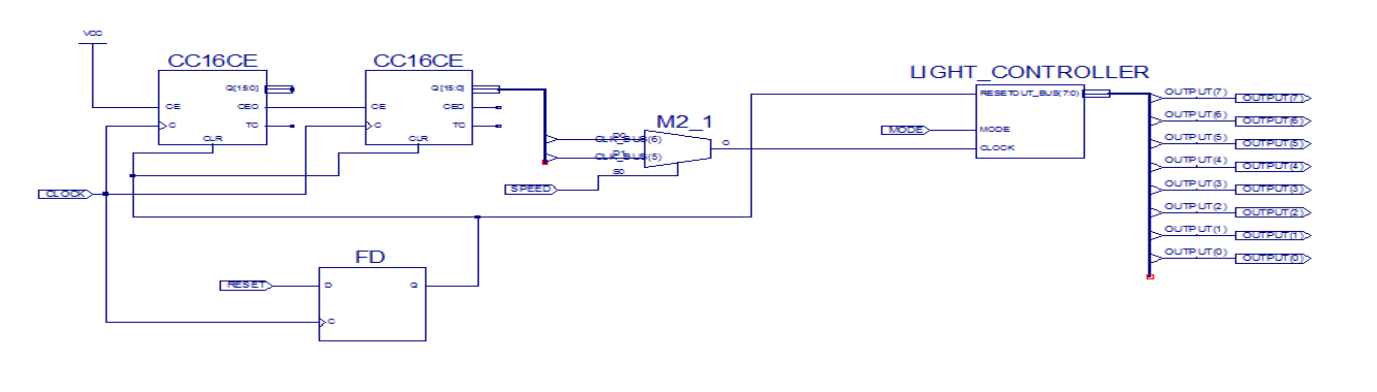


Рис.2.23. Автомат світлових сигналів та подільник тактового сигналу.

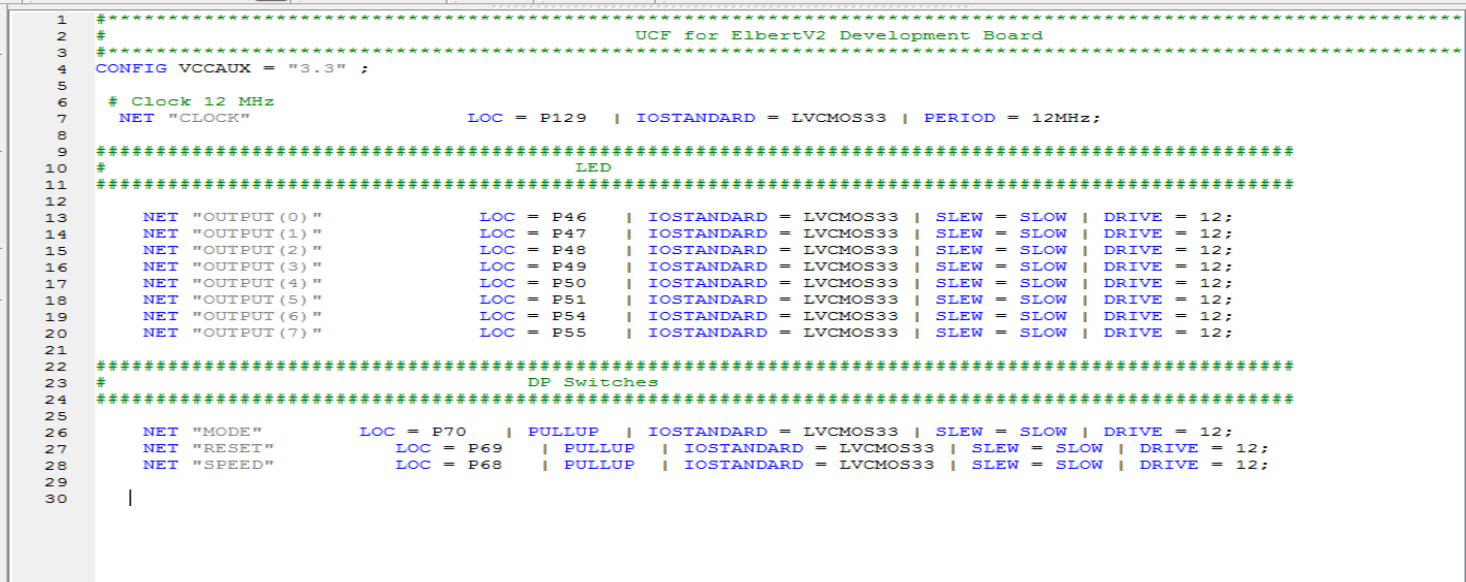


Рис.2.24. Призначення фізичних входів та виходів.

**Висновок:**

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