

Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

до лабораторної роботи № 3

з дисципліни «Моделювання комп'ютерних систем» на
тему:

«Поведінковий опис цифрового автомата Перевірка роботи автомата за
допомогою стенда Elbert V2 – Spartan 3A FPGA»

Варіант №16

Виконав:
ст. гр. КІ-201
Вітик С. А.
Прийняв:
ст. викладач
каф. ЕОМ
Козак Н. Б.

Львів 2024

Мета роботи: На базі стенда реалізувати цифровий автомат для обчислення значення виразів.

Завдання:

ВАРІАНТ(KI-201)

ВИРАЗ

6

$((OP1 + OP2 + 10) - 2) \ll OP2$

Виконання завдання:

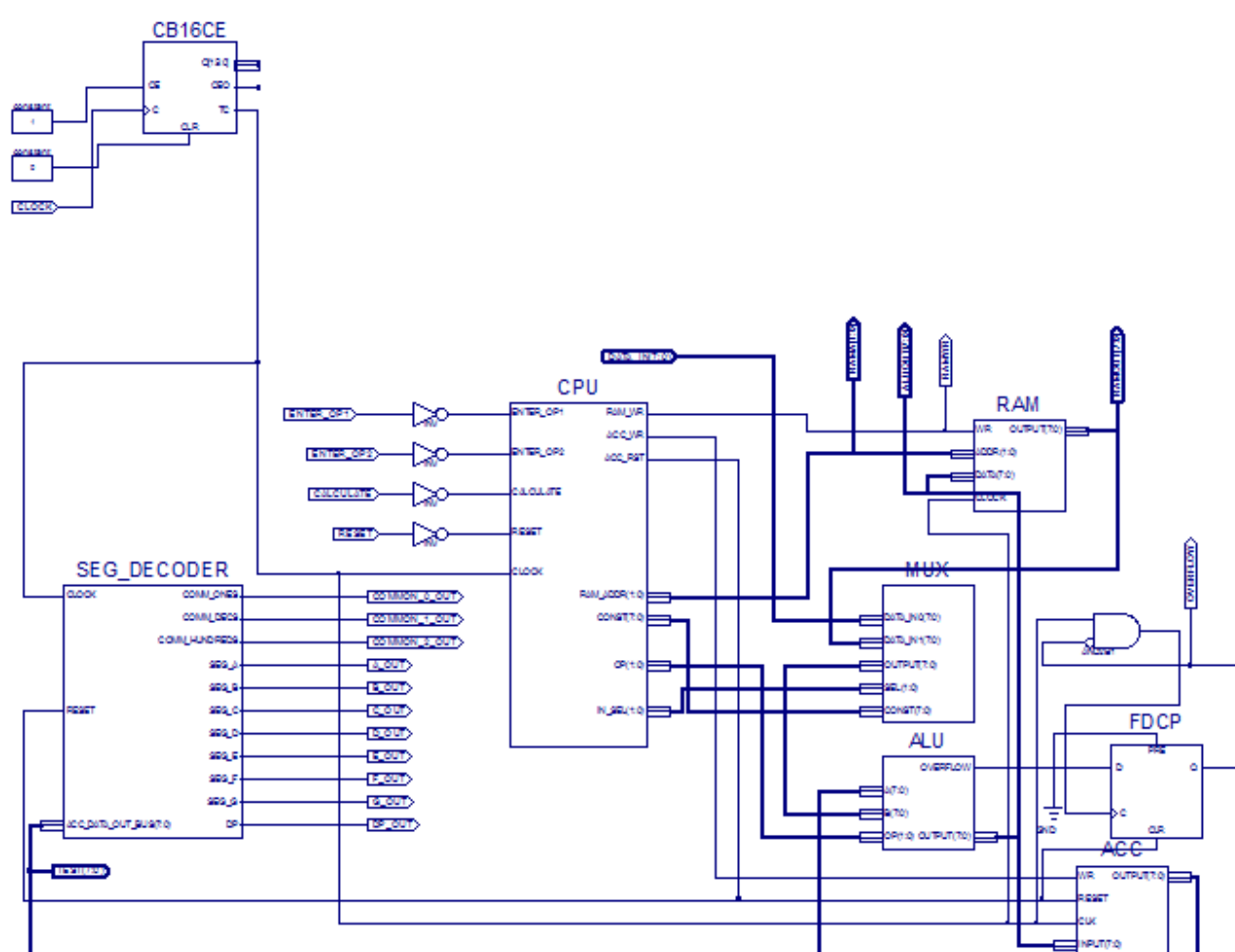


Рис.1 – Top Level

Файл ACC.vhd

-- Company:

-- Engineer:

--

-- Create Date: 17:16:45 04/29/2024

-- Design Name:

-- Module Name: ACC - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ACC is

Port (WR : in STD_LOGIC;

RESET : in STD_LOGIC;

CLK : in STD_LOGIC;

INPUT : in STD_LOGIC_VECTOR (7 downto 0);

OUTPUT : out STD_LOGIC_VECTOR (7 downto 0));

end ACC;

architecture ACC_arch of ACC is

signal DATA : STD_LOGIC_VECTOR (7 downto 0);

begin

```

process (CLK)
begin
    if rising_edge(CLK) then
        if RESET = '1' then
            DATA <= (others => '0');
        elsif WR = '1' then
            DATA <= INPUT;
        end if;
    end if;
end process;

```

```

OUTPUT <= DATA;

```

```

end ACC_arch;

```

Файл MUX.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

```

```

entity MUX is

```

```

    PORT(
        SEL: in STD_LOGIC_VECTOR(1 downto 0);
        CONST: in STD_LOGIC_VECTOR(7 downto 0);
        --CONST1: in STD_LOGIC_VECTOR()
        DATA_IN0: in STD_LOGIC_VECTOR(7 downto 0);
        DATA_IN1: in STD_LOGIC_VECTOR(7 downto 0);
        OUTPUT: out STD_LOGIC_VECTOR(7 downto 0)
    );

```

```

end MUX;

```

```

architecture Behavioral of MUX is

```

```

begin
    process (SEL, DATA_IN0, DATA_IN1, CONST)

```

```

begin
  if (SEL = "00") then
    OUTPUT <= DATA_IN0;
  elsif (SEL = "01") then
    OUTPUT <= DATA_IN1;
  else
    OUTPUT <= CONST;
  end if;
end process;
end Behavioral;

```

Файл RAM.vhd

```

-----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity RAM is
  port(
    WR : IN STD_LOGIC;
    ADDR : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    DATA : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    CLOCK: IN STD_LOGIC;
    OUTPUT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
end RAM;

architecture RAM_arch of RAM is
  type ram_type is array (3 downto 0) of STD_LOGIC_VECTOR(7 downto 0);
  signal UNIT : ram_type;

begin

```

```
process(ADDR, CLOCK, UNIT)
begin
  if(rising_edge(CLOCK)) then
    if (WR = '1') then
      UNIT(conv_integer(ADDR)) <= DATA;
    end if;
  end if;
  OUTPUT <= UNIT(conv_integer(ADDR));
end process;
end RAM_arch;
```

Файл SEG_DECODER.vhd

vetlittz, [24.05.2024 22:02]

```
-----
-- Company:
-- Engineer:
--
-- Create Date: 02:34:19 04/21/2024
-- Design Name:
-- Module Name: BIN_TO_BCD - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----
```

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

entity SEG_DECODER is

```
port( CLOCK : IN STD_LOGIC;
      RESET : IN STD_LOGIC;
      ACC_DATA_OUT_BUS : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      COMM_ONES      : OUT STD_LOGIC;
      COMM_DECS      : OUT STD_LOGIC;
      COMM_HUNDREDS  : OUT STD_LOGIC;
      SEG_A  : OUT STD_LOGIC;
      SEG_B  : OUT STD_LOGIC;
      SEG_C  : OUT STD_LOGIC;
      SEG_D  : OUT STD_LOGIC;
      SEG_E  : OUT STD_LOGIC;
      SEG_F  : OUT STD_LOGIC;
      SEG_G  : OUT STD_LOGIC;
      DP    : OUT STD_LOGIC);
```

end SEG_DECODER;

architecture Behavioral of SEG_DECODER is

```
signal ONES_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
signal DECS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0001";
signal HONDREDS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
```

begin

```
BIN_TO_BCD : process (ACC_DATA_OUT_BUS)
    variable hex_src : STD_LOGIC_VECTOR(7 downto 0) ;
    variable bcd     : STD_LOGIC_VECTOR(11 downto 0) ;
begin
```

```

bcd      := (others => '0') ;
hex_src  := ACC_DATA_OUT_BUS;

for i in hex_src'range loop
    if bcd(3 downto 0) > "0100" then
        bcd(3 downto 0) := bcd(3 downto 0) + "0011" ;
    end if ;
    if bcd(7 downto 4) > "0100" then
        bcd(7 downto 4) := bcd(7 downto 4) + "0011" ;
    end if ;
    if bcd(11 downto 8) > "0100" then
        bcd(11 downto 8) := bcd(11 downto 8) + "0011" ;
    end if ;

    bcd := bcd(10 downto 0) & hex_src(hex_src'left) ; -- shift bcd + 1 new entry
    hex_src := hex_src(hex_src'left - 1 downto hex_src'right) & '0' ; -- shift src + pad with 0
end loop ;

```

vetlittz, [24.05.2024 22:02]

```

HONDREDS_BUS    <= bcd (11 downto 8);
DECS_BUS        <= bcd (7 downto 4);
ONES_BUS        <= bcd (3 downto 0);

```

end process BIN_TO_BCD;

INDICATE : process(CLOCK)

type DIGIT_TYPE is (ONES, DECS, HUNDREDS);

variable CUR_DIGIT : DIGIT_TYPE := ONES;

variable DIGIT_VAL : STD_LOGIC_VECTOR(3 downto 0) := "0000";

variable DIGIT_CTRL : STD_LOGIC_VECTOR(6 downto 0) := "0000000";

variable COMMONS_CTRL : STD_LOGIC_VECTOR(2 downto 0) := "000";


```

begin
  if (rising_edge(CLOCK)) then
    if(RESET = '0') then
      case CUR_DIGIT is
        when ONES =>
          DIGIT_VAL := ONES_BUS;
          CUR_DIGIT := DECS;
          COMMONS_CTRL := "001";
        when DECS =>
          DIGIT_VAL := DECS_BUS;
          CUR_DIGIT := HUNDREDS;
          COMMONS_CTRL := "010";
        when HUNDREDS =>
          DIGIT_VAL := HONDREDS_BUS;
          CUR_DIGIT := ONES;
          COMMONS_CTRL := "100";
        when others =>
          DIGIT_VAL := ONES_BUS;
          CUR_DIGIT := ONES;
          COMMONS_CTRL := "000";
      end case;

      case DIGIT_VAL is          --abcdefg
        when "0000" => DIGIT_CTRL := "1111110";
        when "0001" => DIGIT_CTRL := "0110000";
        when "0010" => DIGIT_CTRL := "1101101";
        when "0011" => DIGIT_CTRL := "1111001";
        when "0100" => DIGIT_CTRL := "0110011";
        when "0101" => DIGIT_CTRL := "1011011";
        when "0110" => DIGIT_CTRL := "1011111";
        when "0111" => DIGIT_CTRL := "1110000";
        when "1000" => DIGIT_CTRL := "1111111";
        when "1001" => DIGIT_CTRL := "1111011";
      end case;
    end if;
  end if;
end;

```

```

        when others => DIGIT_CTRL := "0000000";
    end case;
else
    DIGIT_VAL := ONES_BUS;
    CUR_DIGIT := ONES;
    COMMONS_CTRL := "000";
end if;

COMM_ONES    <= not COMMONS_CTRL(0);
COMM_DECS    <= not COMMONS_CTRL(1);
COMM_HUNDREDS <= not COMMONS_CTRL(2);

SEG_A <= not DIGIT_CTRL(6);
SEG_B <= not DIGIT_CTRL(5);
SEG_C <= not DIGIT_CTRL(4);
SEG_D <= not DIGIT_CTRL(3);
SEG_E <= not DIGIT_CTRL(2);
SEG_F <= not DIGIT_CTRL(1);
SEG_G <= not DIGIT_CTRL(0);
DP    <= '1';

end if;
end process INDICATE;

```

end Behavioral;

Файл CPU.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

```

entity CPU is

```

    port( ENTER_OP1 : IN STD_LOGIC;
          ENTER_OP2 : IN STD_LOGIC;

```

```

    CALCULATE : IN STD_LOGIC;
    RESET : IN STD_LOGIC;
    CLOCK : IN STD_LOGIC;
    RAM_WR : OUT STD_LOGIC;
    RAM_ADDR : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
    CONST : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    ACC_WR : OUT STD_LOGIC;
    ACC_RST : OUT STD_LOGIC;
    IN_SEL : OUT STD_LOGIC_VECTOR(1 downto 0);
    OP : OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
end CPU;

```

architecture CPU_arch of CPU is

```

type STATE_TYPE is (RST, IDLE, LOAD_OP1, LOAD_OP2, RUN_CALC0, RUN_CALC1,
RUN_CALC2, RUN_CALC3, RUN_CALC4, FINISH);

```

```

signal CUR_STATE : STATE_TYPE;

```

```

signal NEXT_STATE : STATE_TYPE;

```

```

begin

```

```

    SYNC_PROC: process (CLOCK)

```

```

    begin

```

```

        if (rising_edge(CLOCK)) then

```

```

            if (RESET = '1') then

```

```

                CUR_STATE <= RST;

```

```

            else

```

```

                CUR_STATE <= NEXT_STATE;

```

```

            end if;

```

```

        end if;

```

```

    end process;

```

```

NEXT_STATE_DECODE: process (CLOCK, ENTER_OP1, ENTER_OP2, CALCULATE)

```

begin

 NEXT_STATE <= CUR_STATE;

case(CUR_STATE) is

 when RST =>

 NEXT_STATE <= IDLE;

 when IDLE =>

 if (ENTER_OP1 = '1') then

 NEXT_STATE <= LOAD_OP1;

 elsif (ENTER_OP2 = '1') then

 NEXT_STATE <= LOAD_OP2;

 elsif (CALCULATE = '1') then

 NEXT_STATE <= RUN_CALC0;

 else

 NEXT_STATE <= IDLE;

 end if;

 when LOAD_OP1 =>

 NEXT_STATE <= IDLE;

 when LOAD_OP2 =>

 NEXT_STATE <= IDLE;

 when RUN_CALC0 =>

 NEXT_STATE <= RUN_CALC1;

 when RUN_CALC1 =>

 NEXT_STATE <= RUN_CALC2;

 when RUN_CALC2 =>

 NEXT_STATE <= RUN_CALC3;

 when RUN_CALC3 =>

 NEXT_STATE <= RUN_CALC4;

 when RUN_CALC4 =>

 NEXT_STATE <= FINISH;

 when FINISH =>

 NEXT_STATE <= FINISH;

 when others =>

```

    NEXT_STATE <= IDLE;

end case;

end process;

OUTPUT_DECODE: process (CUR_STATE)
begin
    case (CUR_STATE) is
        when RST =>
            RAM_WR <= '0';
            RAM_ADDR <= "00";
            CONST <= "000000000";
            ACC_WR <= '0';
            ACC_RST <= '1';
            IN_SEL <= "00";
            OP <= "00";

        when LOAD_OP1 =>
            RAM_WR <= '1';
            RAM_ADDR <= "00";
            CONST <= "000000000";
            ACC_WR <= '0';
            ACC_RST <= '1';
            IN_SEL <= "00";
            OP <= "00";

        when LOAD_OP2 =>
            RAM_WR <= '1';
            RAM_ADDR <= "01";
            CONST <= "000000000";
            ACC_WR <= '0';
            ACC_RST <= '1';
            IN_SEL <= "00";
            OP <= "00";

        when RUN_CALC0 =>
            RAM_WR <= '0';

```

```
RAM_ADDR <= "00";
CONST <= "000000000";
ACC_WR <= '1';
ACC_RST <= '0';
IN_SEL <= "01";
OP <= "01";
when RUN_CALC1 =>
    RAM_WR <= '0';
    RAM_ADDR <= "01";
    CONST <= "000000000";
    ACC_WR <= '1';
    ACC_RST <= '0';
    IN_SEL <= "01";
    OP <= "01";
when RUN_CALC2 =>
    RAM_WR <= '0';
    RAM_ADDR <= "01";
    CONST <= "00001010";
    ACC_WR <= '1';
    ACC_RST <= '0';
    IN_SEL <= "10";
    OP <= "01";
when RUN_CALC3 =>
    RAM_WR <= '0';
    RAM_ADDR <= "01";
    CONST <= "00000010";
    ACC_WR <= '1';
    ACC_RST <= '0';
    IN_SEL <= "10";
    OP <= "10";
when RUN_CALC4 =>
    RAM_WR <= '0';
    RAM_ADDR <= "01";
```

```

CONST <= "000000000";

ACC_WR <= '1';

ACC_RST <= '0';

IN_SEL <= "01";

OP <= "11";

when IDLE =>

    RAM_WR <= '0';

    RAM_ADDR <= "00";

    CONST <= "000000000";

    ACC_WR <= '0';

    ACC_RST <= '0';

    IN_SEL <= "00";

    OP <= "00";

when others =>

    RAM_WR <= '0';

    RAM_ADDR <= "00";

    CONST <= "000000000";

    ACC_WR <= '0';

    ACC_RST <= '0';

    IN_SEL <= "00";

    OP <= "00";

end case;

end process;

end CPU_arch;

```

Часові діаграми:

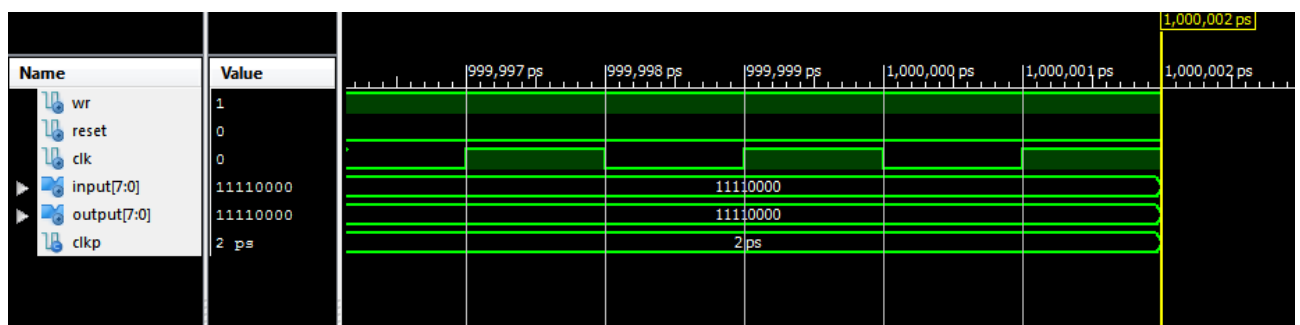


Рис. 2 – Часова діаграма ACC

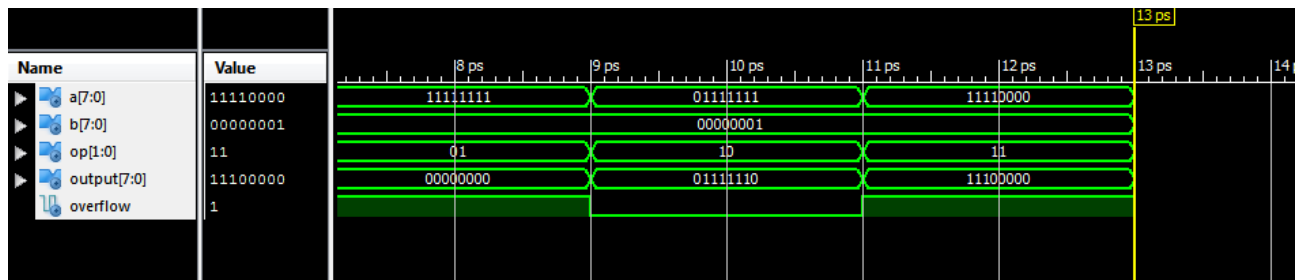


Рис. 3 – Часова діаграма ALU

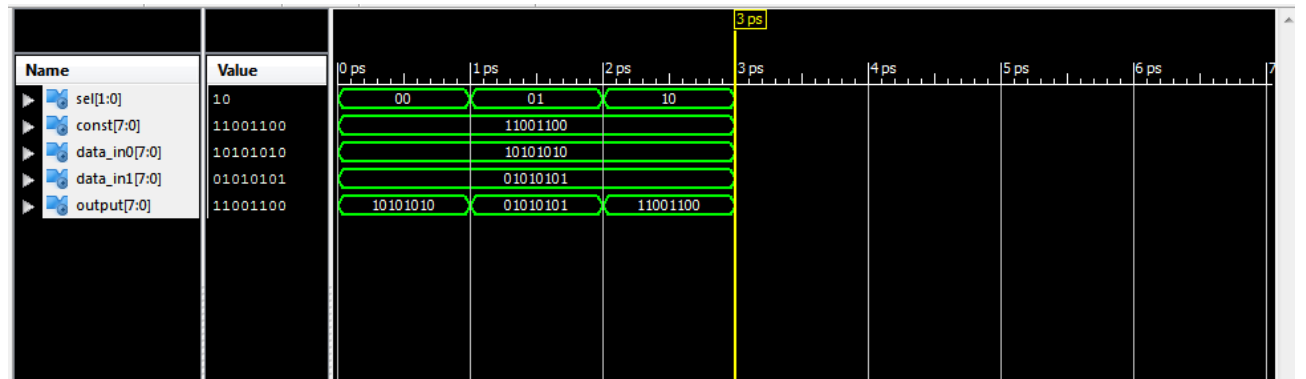


Рис. 4 – Часова діаграма MUX

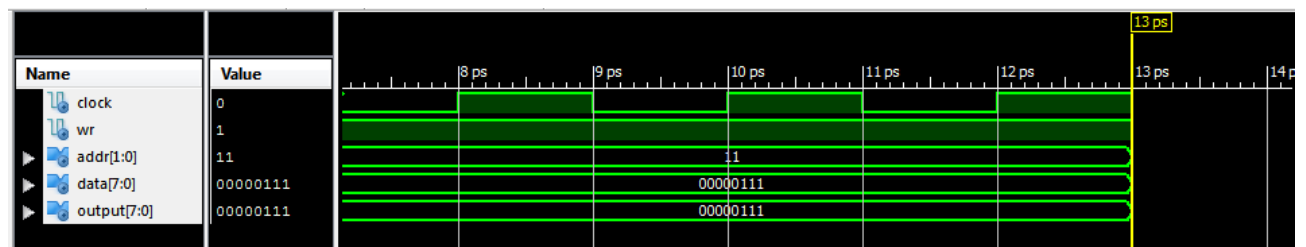


Рис. 5 – Часова діаграма RAM

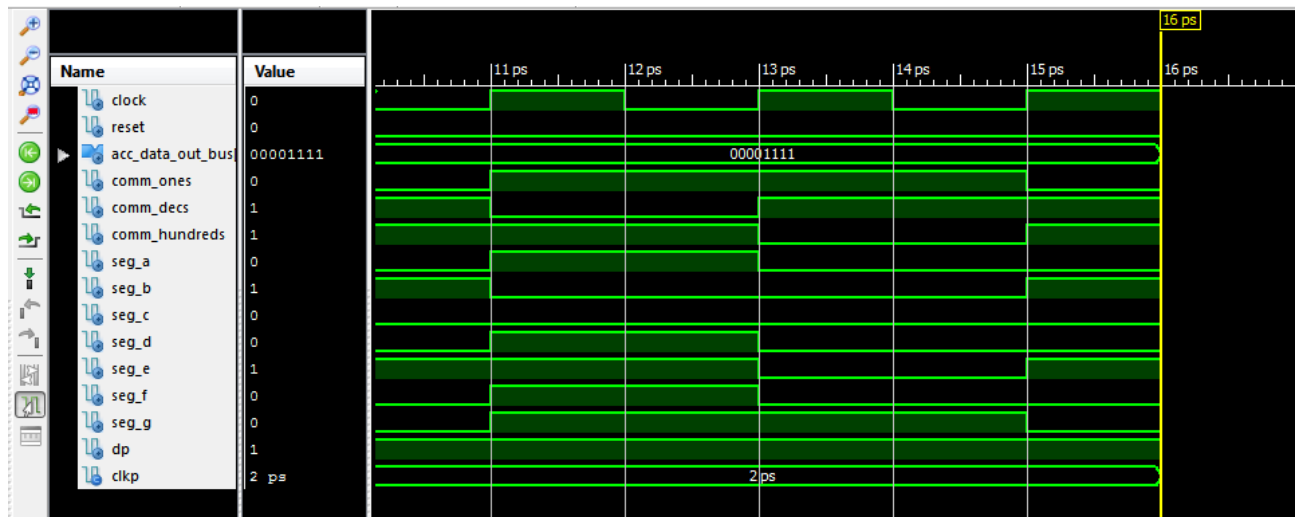


Рис. 6 – Часова діаграма SEG_DECODER

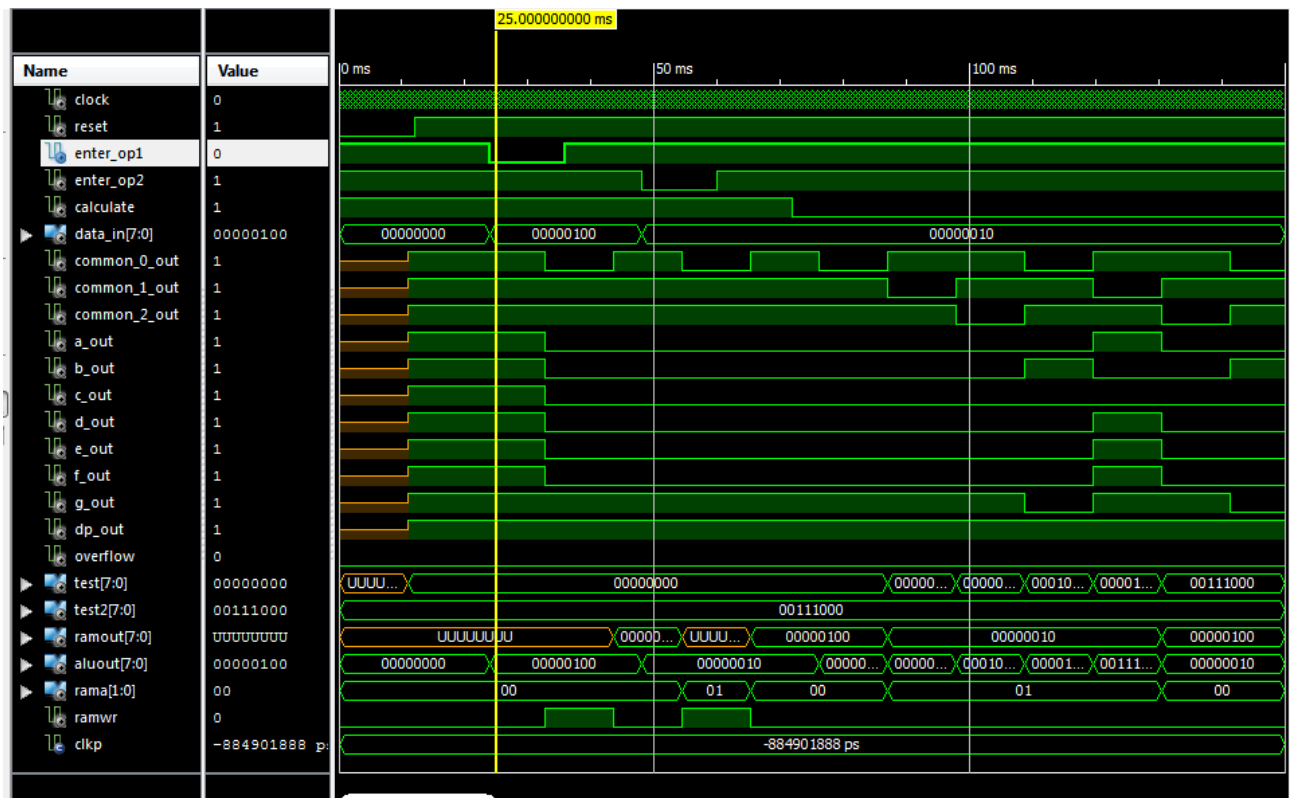


Рис. 7 – Часова діаграма TopLevel

Файл TopLevelTest.vhd

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

tb : PROCESS

BEGIN

lp1: for i in 4 to 6 loop

lp2: for j in 2 to 2 loop

TEST2 <= std_logic_vector(to_unsigned(i + j + 10 - 2, 8) sll j);

ENTER_OP1 <= '1';

ENTER_OP2 <= '1';

CALCULATE <= '1';

DATA_IN <= (others => '0');

RESET <= '0';

wait for CLKP;

RESET <= '1';

wait for CLKP;

DATA_IN <= std_logic_vector(to_unsigned(i, 8)); -- A

ENTER_OP1 <= '0';

```

wait for CLKP;
ENTER_OP1 <= '1';
wait for CLKP;
DATA_IN <= std_logic_vector(to_unsigned(j, 8)); -- B
ENTER_OP2 <= '0';
wait for CLKP;
ENTER_OP2 <= '1';
wait for CLKP;
CALCULATE <= '0'; -- START CALCULATION
wait for CLKP* 7;
assert TEST = TEST2 severity FAILURE;
wait for CLKP;
end loop;
end loop;

WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;

```

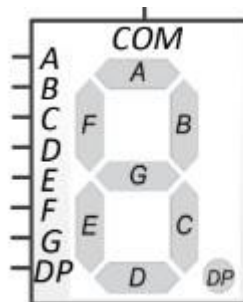


Рис.8 – 7-сегментний індикатор

Переглянемо часову діаграму 7seg decoder.

Бачимо, що для вхідного числа $0110\ 1000\ (2) = 104(10)$

Ми отримуємо такі значення(якщо значення не вказане, вважаємо його за одиницю):

COMM_ONES = 0: F, G, B, C = 0, що відповідає числу 4

COMM_DECS = 0: A, B, C, D, E, F = 0, що відповідає числу 0

COMM_HUNDREDS = 0: B, C = 0, що відповідає числу 1

Отримуємо значення 104, що збігається з даним йому.

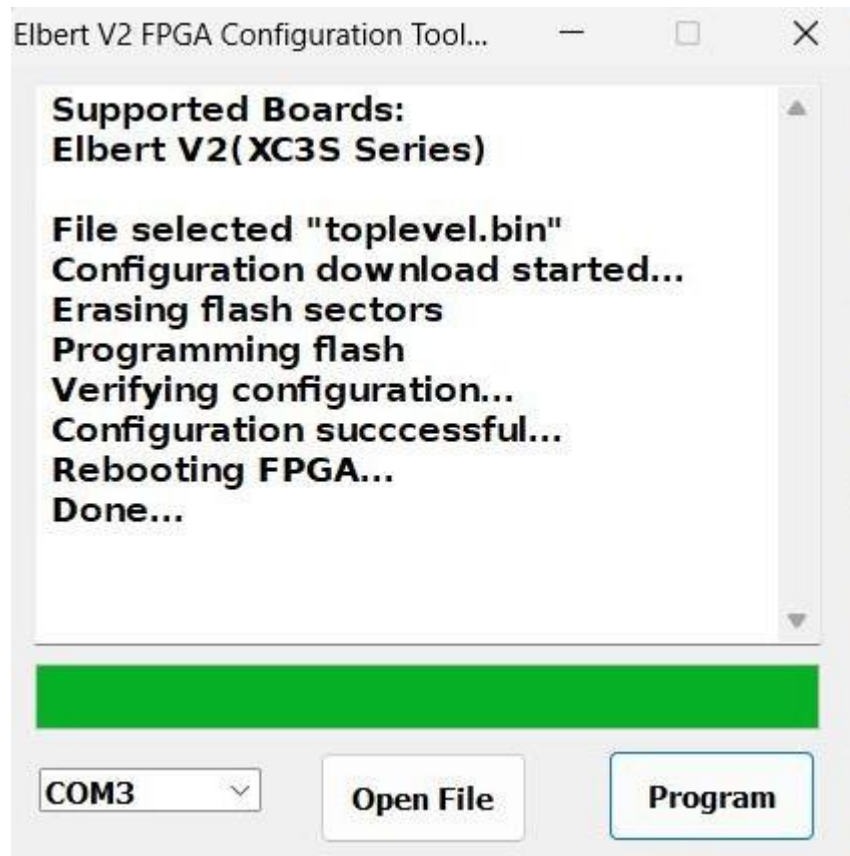


Рис.9 – Успішна прошивка

Висновок: Виконуючи дану лабораторну роботу я навчився реалізовувати цифровий автомат для обчислення значення виразів використовуючи засоби VHDL.