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 №2

Виконав:
ст. гр. КІ-201
Бовтач О.М
Прийняв:
Козак Н. Б.

Львів 2024

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

Виконання роботи:

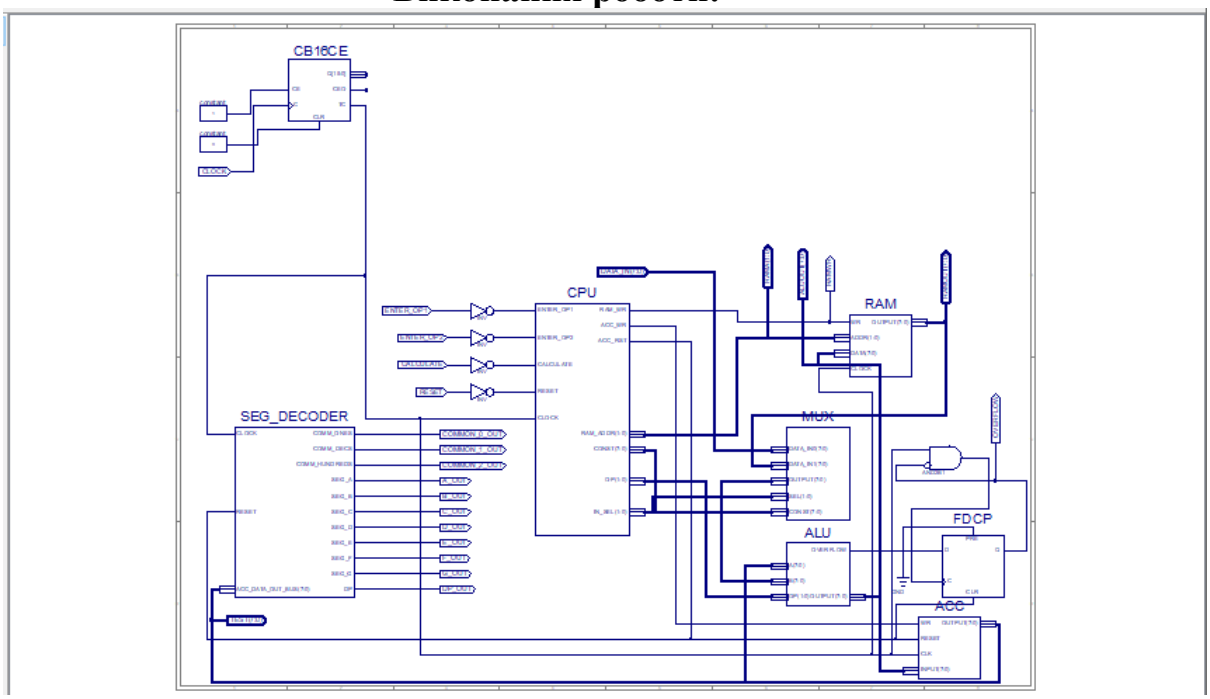

Рис. 1 – Top Level

Файл ACC.vhd

```
1.  ----------------------------------------------------------------------------------
2.  -- Company:
3.  -- Engineer:
4.  --
5.  -- Create Date:    17:16:45 04/29/2024
6.  -- Design Name:
7.  -- Module Name:    ACC - Behavioral
8.  -- Project Name:
9.  -- Target Devices:
10. -- Tool versions:
11. -- Description:
12. --
13. -- Dependencies:
14. --
15. -- Revision:
16. -- Revision 0.01 - File Created
17. -- Additional Comments:
18. --
19. ----------------------------------------------------------------------------------
    -
20. library IEEE;
21. use IEEE.STD_LOGIC_1164.ALL;
22.
23. -- Uncomment the following library declaration if using
24. -- arithmetic functions with Signed or Unsigned values
25. --use IEEE.NUMERIC_STD.ALL;
26.
27. -- Uncomment the following library declaration if instantiating
28. -- any Xilinx primitives in this code.
29. --library UNISIM;
```

```
30. --use UNISIM.VComponents.all;
31.
32. entity ACC is
33.     Port ( WR   : in  STD_LOGIC;
34.            RESET  : in  STD_LOGIC;
35.            CLK  : in  STD_LOGIC;
36.            INPUT : in  STD_LOGIC_VECTOR (7 downto 0);
37.            OUTPUT : out  STD_LOGIC_VECTOR (7 downto 0));
38. end ACC;
39.
40. architecture ACC_arch of ACC is
41.     signal DATA : STD_LOGIC_VECTOR (7 downto 0);
42. begin
43.     process (CLK)
44.     begin
45.         if rising_edge(CLK) then
46.             if RESET = '1' then
47.                 DATA <= (others => '0');
48.             elsif WR = '1' then
49.                 DATA <= INPUT;
50.             end if;
51.         end if;
52.     end process;
53.
54.
55.     OUTPUT <= DATA;
56.
57. end ACC_arch;
```

Файл ALU.vhd

```
1.  - Company:
2.  -- Engineer:
3.  --
4.  -- Create Date:   11:09:43 05/07/2024
5.  -- Design Name:
6.  -- Module Name:   C:/Users/User/Documents/Lab_3_v6/ACCTest.vhd
7.  -- Project Name:  Lab_3_Example
8.  -- Target Device:
9.  -- Tool versions:
10. -- Description:
11. --
12. -- VHDL Test Bench Created by ISE for module: ACC
13. --
14. -- Dependencies:
15. --
16. -- Revision:
17. -- Revision 0.01 - File Created
18. -- Additional Comments:
19. --
20. -- Notes:
21. -- This testbench has been automatically generated using types std_logic and
22. -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
23. -- that these types always be used for the top-level I/O of a design in order
24. -- to guarantee that the testbench will bind correctly to the post-implementation
25. -- simulation model.
26. --------------------------------------------------------------------------------
27. LIBRARY ieee;
28. USE ieee.std_logic_1164.ALL;
29.
30. -- Uncomment the following library declaration if using
31. -- arithmetic functions with Signed or Unsigned values
32. --USE ieee.numeric_std.ALL;
33.
34. ENTITY ACCTest IS
```

```
35. END ACCTest;
36.
37. ARCHITECTURE behavior OF ACCTest IS
38.
39.     -- Component Declaration for the Unit Under Test (UUT)
40.
41.     COMPONENT ACC
42.     PORT(
43.          WR : IN  std_logic;
44.          RESET : IN  std_logic;
45.          CLK : IN  std_logic;
46.          INPUT : IN  std_logic_vector(7 downto 0);
47.          OUTPUT : OUT  std_logic_vector(7 downto 0)
48.         );
49.     END COMPONENT;
50.
51.
52.    --Inputs
53.    signal WR : std_logic := '0';
54.    signal RESET : std_logic := '0';
55.    signal CLK : std_logic := '0';
56.    signal INPUT : std_logic_vector(7 downto 0) := (others => '0');
57.
58.        --Outputs
59.    signal OUTPUT : std_logic_vector(7 downto 0);
60.
61.    -- Clock period definitions
62.    constant CLKP: time := 2 ps;
63.
64. BEGIN
65.
66.        -- Instantiate the Unit Under Test (UUT)
67.    uut: ACC PORT MAP (
68.           WR => WR,
69.           RESET => RESET,
70.           CLK => CLK,
71.           INPUT => INPUT,
72.           OUTPUT => OUTPUT
73.         );
74.
75.    -- Clock process definitions
76.    CLK_process :process
77.    begin
78.                CLK <= '0';
79.                wait for CLKP/2;
80.                CLK <= '1';
81.                wait for CLKP/2;
82.    end process;
83.
84.
85.    -- Stimulus process
86.    stim_proc: process
87.    begin
88.                RESET <= '1';
89.       wait for 4 * CLKP;
90.                RESET <= '0';
91.                WR <= '0';
92.                INPUT <= "00001111";
93.                wait for CLKP;
94.                assert OUTPUT = "00000000" severity failure;
95.                WR <= '1';
96.                wait for CLKP;
97.                assert OUTPUT = "00001111" severity failure;
98.                INPUT <= "11110000";
99.                wait for CLKP;
100.                       assert OUTPUT = "11110000" severity failure;
```

```
101.
102.             wait;
103.          end process;
104.
105. END;
```

Файл CPU.vhd

```
1.  library IEEE;
2.  use IEEE.STD_LOGIC_1164.ALL;
3.
4.
5.  entity CPU is
6.          port( ENTER_OP1 : IN STD_LOGIC;
7.                          ENTER_OP2 : IN STD_LOGIC;
8.                          CALCULATE : IN STD_LOGIC;
9.                          RESET : IN STD_LOGIC;
10.                         CLOCK : IN STD_LOGIC;
11.                         RAM_WR : OUT STD_LOGIC;
12.                         RAM_ADDR : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
13.                         CONST : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
14.                         ACC_WR : OUT STD_LOGIC;
15.                         ACC_RST : OUT STD_LOGIC;
16.                         IN_SEL : OUT STD_LOGIC_VECTOR(1 downto 0);
17.                         OP : OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
18. end CPU;
19.
20. architecture CPU_arch of CPU is
21.
22. type   STATE_TYPE is (RST, IDLE, LOAD_OP1, LOAD_OP2, RUN_CALC0, RUN_CALC1,
    RUN_CALC2, RUN_CALC3, RUN_CALC4, FINISH);
23. signal CUR_STATE  : STATE_TYPE;
24. signal NEXT_STATE : STATE_TYPE;
25.
26. begin
27.         SYNC_PROC: process (CLOCK)
28.    begin
29.       if (rising_edge(CLOCK)) then
30.          if (RESET = '1') then
31.             CUR_STATE <= RST;
32.          else
33.             CUR_STATE <= NEXT_STATE;
34.          end if;
35.       end if;
36.    end process;
37.
38.
39.         NEXT_STATE_DECODE: process (CLOCK, ENTER_OP1, ENTER_OP2, CALCULATE)
40.    begin
41.       NEXT_STATE <= CUR_STATE;
42.
43.                case(CUR_STATE) is
44.                        when RST =>
45.                                NEXT_STATE <= IDLE;
46.                        when IDLE             =>
47.                                if (ENTER_OP1 = '1') then
48.                                        NEXT_STATE <= LOAD_OP1;
49.                                elsif (ENTER_OP2 = '1') then
50.                                        NEXT_STATE <= LOAD_OP2;
51.                                elsif (CALCULATE = '1') then
52.                                        NEXT_STATE <= RUN_CALC0;
53.                                else
54.                                        NEXT_STATE <= IDLE;
55.                                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 <= "00000000";
                                ACC_WR <= '0';
                                ACC_RST <= '1';
                                IN_SEL <= "00";
                                OP <= "00";
                        when LOAD_OP1 =>
                                RAM_WR <= '1';
                                RAM_ADDR <= "00";
                                CONST <= "00000000";
                                ACC_WR <= '0';
                                ACC_RST <= '1';
                                IN_SEL <= "00";
                                OP <= "00";
                        when LOAD_OP2 =>
                                RAM_WR <= '1';
                                RAM_ADDR <= "01";
                                CONST <= "00000000";
                                        ACC_WR <= '0';
                                        ACC_RST <= '1';
                                        IN_SEL <= "00";
                                        OP <= "00";
                                when RUN_CALC0 =>
                                        RAM_WR <= '0';
                                        RAM_ADDR <= "00";
                                        CONST <= "00000000";
                                        ACC_WR <= '1';
                                        ACC_RST <= '0';
                                        IN_SEL <= "01";
                                        OP <= "00";
                                when RUN_CALC1 =>
                                        RAM_WR <= '0';
                                        RAM_ADDR <= "01";
                                        CONST <= "00000000";
                                        ACC_WR <= '1';
                                        ACC_RST <= '0';
                                        IN_SEL <= "01";
                                        OP <= "11";
                                when RUN_CALC2 =>
                                        RAM_WR <= '0';
```

```
122.                                    RAM_ADDR <= "01";
123.                                    CONST <= "00000000";
124.                                    ACC_WR <= '1';
125.                                    ACC_RST <= '0';
126.                                    IN_SEL <= "01";
127.                                    OP <= "01";
128.                            when RUN_CALC3 =>
129.                                    RAM_WR <= '0';
130.                                    RAM_ADDR <= "01";
131.                                    CONST <= "00001010";
132.                                    ACC_WR <= '1';
133.                                    ACC_RST <= '0';
134.                                    IN_SEL <= "10";
135.                                    OP <= "01";
136.                            when RUN_CALC4 =>
137.                                    RAM_WR <= '0';
138.                                    RAM_ADDR <= "00";
139.                                    CONST <= "00000011";
140.                                    ACC_WR <= '1';
141.                                    ACC_RST <= '0';
142.                                    IN_SEL <= "10";
143.                                    OP <= "10";
144.                            when IDLE =>
145.                                    RAM_WR <= '0';
146.                                    RAM_ADDR <= "00";
147.                                    CONST <= "00000000";
148.                                    ACC_WR <= '0';
149.                                    ACC_RST <= '0';
150.                                    IN_SEL <= "00";
151.                                    OP <= "00";
152.                            when others =>
153.                                    RAM_WR <= '0';
154.                                    RAM_ADDR <= "00";
155.                                    CONST <= "00000000";
156.                                    ACC_WR <= '0';
157.                                    ACC_RST <= '0';
158.                                    IN_SEL <= "00";
159.                                    OP <= "00";
160.                    end case;
161.          end process;
162.       end CPU_arch;
```

Файл MUX.vhd

```
1.  library IEEE;
2.  use IEEE.STD_LOGIC_1164.ALL;
3.
4.  entity MUX is
5.          PORT(
6.                  SEL: in STD_LOGIC_VECTOR(1 downto 0);
7.                  CONST: in STD_LOGIC_VECTOR(7 downto 0);
8.                  --CONST1: in STD_LOGIC_VECTOR()
9.                  DATA_IN0: in STD_LOGIC_VECTOR(7 downto 0);
10.                 DATA_IN1: in STD_LOGIC_VECTOR(7 downto 0);
11.                 OUTPUT: out STD_LOGIC_VECTOR(7 downto 0)
12.         );
13. end MUX;
14.
15. architecture Behavioral of MUX is
16. begin
17.         process (SEL, DATA_IN0, DATA_IN1, CONST)
18.         begin
19.                 if (SEL = "00") then
20.                         OUTPUT <= DATA_IN0;
21.                 elsif (SEL = "01") then
```

```
                        OUTPUT <= DATA_IN1;
                else
                        OUTPUT <= CONST;
                end if;
        end process;
end Behavioral;
```

Файл RAM.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    02:03:24 04/21/2024
-- Design Name:
-- Module Name:    RAM - RAM_arch
-- 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.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

```
1.  ----------------------------------------------------------------------------------
    -
2.  -- Company:
3.  -- Engineer:
4.  --
5.  -- Create Date:    02:34:19 04/21/2024
6.  -- Design Name:
7.  -- Module Name:    BIN_TO_BCD - Behavioral
8.  -- Project Name:
9.  -- Target Devices:
10. -- Tool versions:
11. -- Description:
12. --
13. -- Dependencies:
14. --
15. -- Revision:
16. -- Revision 0.01 - File Created
17. -- Additional Comments:
18. --
19. ----------------------------------------------------------------------------------
    -
20. library IEEE;
21. use IEEE.STD_LOGIC_1164.ALL;
22. use IEEE.STD_LOGIC_ARITH.ALL;
23. use IEEE.STD_LOGIC_UNSIGNED.ALL;
24.
25.
26. entity SEG_DECODER is
27.         port( CLOCK : IN STD_LOGIC;
28.                       RESET : IN STD_LOGIC;
29.                       ACC_DATA_OUT_BUS : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
30.                       COMM_ONES                    : OUT STD_LOGIC;
31.                       COMM_DECS             : OUT STD_LOGIC;
32.                       COMM_HUNDREDS   : OUT STD_LOGIC;
33.                       SEG_A    : OUT STD_LOGIC;
34.                       SEG_B    : OUT STD_LOGIC;
35.                       SEG_C    : OUT STD_LOGIC;
36.                       SEG_D    : OUT STD_LOGIC;
37.                       SEG_E    : OUT STD_LOGIC;
38.                       SEG_F    : OUT STD_LOGIC;
39.                       SEG_G    : OUT STD_LOGIC;
40.                       DP              : OUT STD_LOGIC);
41. end SEG_DECODER;
42.
43. architecture Behavioral of SEG_DECODER is
44.
45.         signal ONES_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
46.         signal DECS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0001";
47.         signal HONDREDS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
48.
49. begin
50.         BIN_TO_BCD : process (ACC_DATA_OUT_BUS)
51.          variable hex_src : STD_LOGIC_VECTOR(7 downto 0) ;
52.          variable bcd     : STD_LOGIC_VECTOR(11 downto 0) ;
53.      begin
54.          bcd             := (others => '0') ;
55.          hex_src         := ACC_DATA_OUT_BUS;
56.
57.          for i in hex_src'range loop
58.              if bcd(3 downto 0) > "0100" then
59.                  bcd(3 downto 0) := bcd(3 downto 0) + "0011" ;
60.              end if ;
61.              if bcd(7 downto 4) > "0100" then
```

```
62.                 bcd(7 downto 4) := bcd(7 downto 4) + "0011" ;
63.             end if ;
64.             if bcd(11 downto 8) > "0100" then
65.                 bcd(11 downto 8) := bcd(11 downto 8) + "0011" ;
66.             end if ;
67.
68.             bcd := bcd(10 downto 0) & hex_src(hex_src'left) ; -- shift bcd + 1
    new entry
69.             hex_src := hex_src(hex_src'left - 1 downto hex_src'right) & '0' ; --
    shift src + pad with 0
70.         end loop ;
71.
72.         HONDREDS_BUS       <=  bcd (11 downto 8);
73.         DECS_BUS        <=  bcd (7 downto 4);
74.         ONES_BUS       <=  bcd (3 downto 0);
75.
76.     end process BIN_TO_BCD;
77.
78.          INDICATE : process(CLOCK)
79.                type DIGIT_TYPE is (ONES, DECS, HUNDREDS);
80.
81.                variable CUR_DIGIT     : DIGIT_TYPE := ONES;
82.                variable DIGIT_VAL      : STD_LOGIC_VECTOR(3 downto 0) := "0000";
83.                variable DIGIT_CTRL     : STD_LOGIC_VECTOR(6 downto 0) :=
    "0000000";
84.                variable COMMONS_CTRL  : STD_LOGIC_VECTOR(2 downto 0) := "000";
85.
86.                begin
87.                       if (rising_edge(CLOCK)) then
88.                              if(RESET = '0') then
89.                                     case CUR_DIGIT is
90.                                            when ONES =>
91.                                                   DIGIT_VAL := ONES_BUS;
92.                                                   CUR_DIGIT := DECS;
93.                                                   COMMONS_CTRL := "001";
94.                                            when DECS =>
95.                                                   DIGIT_VAL := DECS_BUS;
96.                                                   CUR_DIGIT := HUNDREDS;
97.                                                   COMMONS_CTRL := "010";
98.                                            when HUNDREDS =>
99.                                                   DIGIT_VAL := HONDREDS_BUS;
100.                                                         CUR_DIGIT := ONES;
101.                                                         COMMONS_CTRL :=
    "100";
102.                                                  when others =>
103.                                                        DIGIT_VAL :=
    ONES_BUS;
104.                                                        CUR_DIGIT := ONES;
105.                                                        COMMONS_CTRL :=
    "000";
106.                                           end case;
107.
108.                                           case DIGIT_VAL is              --
    abcdefg
109.                                                  when "0000" => DIGIT_CTRL :=
    "1111110";
110.                                                  when "0001" => DIGIT_CTRL :=
    "0110000";
111.                                                  when "0010" => DIGIT_CTRL :=
    "1101101";
112.                                                  when "0011" => DIGIT_CTRL :=
    "1111001";
113.                                                  when "0100" => DIGIT_CTRL :=
    "0110011";
114.                                                  when "0101" => DIGIT_CTRL :=
    "1011011";
```

```
115.                                                when "0110" => DIGIT_CTRL :=
    "1011111";
116.                                                when "0111" => DIGIT_CTRL :=
    "1110000";
117.                                                when "1000" => DIGIT_CTRL :=
    "1111111";
118.                                                when "1001" => DIGIT_CTRL :=
    "1111011";
119.                                                when others => DIGIT_CTRL :=
    "0000000";
120.                                        end case;
121.                                    else
122.                                        DIGIT_VAL := ONES_BUS;
123.                                        CUR_DIGIT := ONES;
124.                                        COMMONS_CTRL := "000";
125.                                    end if;
126.
127.                                    COMM_ONES       <= not COMMONS_CTRL(0);
128.                                    COMM_DECS       <= not COMMONS_CTRL(1);
129.                                    COMM_HUNDREDS <= not COMMONS_CTRL(2);
130.
131.                                    SEG_A <= not DIGIT_CTRL(6);
132.                                    SEG_B <= not DIGIT_CTRL(5);
133.                                    SEG_C <= not DIGIT_CTRL(4);
134.                                    SEG_D <= not DIGIT_CTRL(3);
135.                                    SEG_E <= not DIGIT_CTRL(2);
136.                                    SEG_F <= not DIGIT_CTRL(1);
137.                                    SEG_G <= not DIGIT_CTRL(0);
138.                                    DP     <= '1';
139.
140.                            end if;
141.                end process INDICATE;
142.
143.        end Behavioral;
```

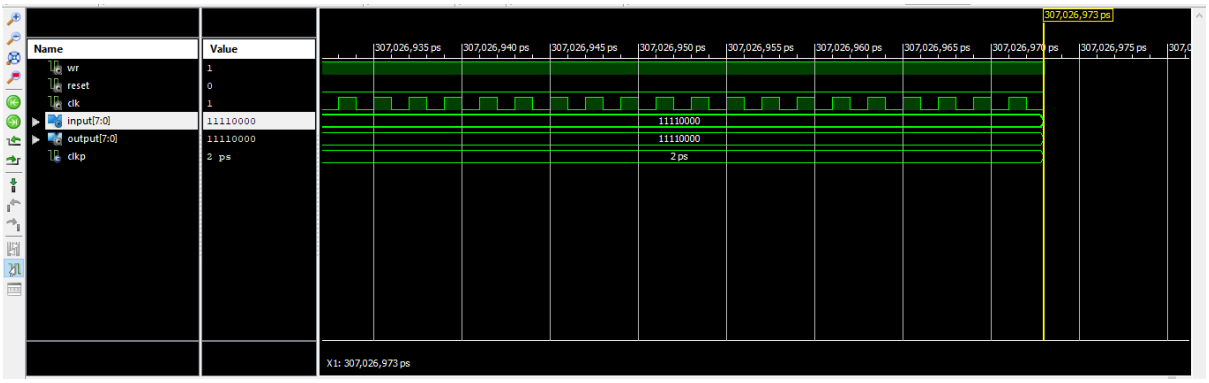

Рис. 2 – Часова діаграма ACC

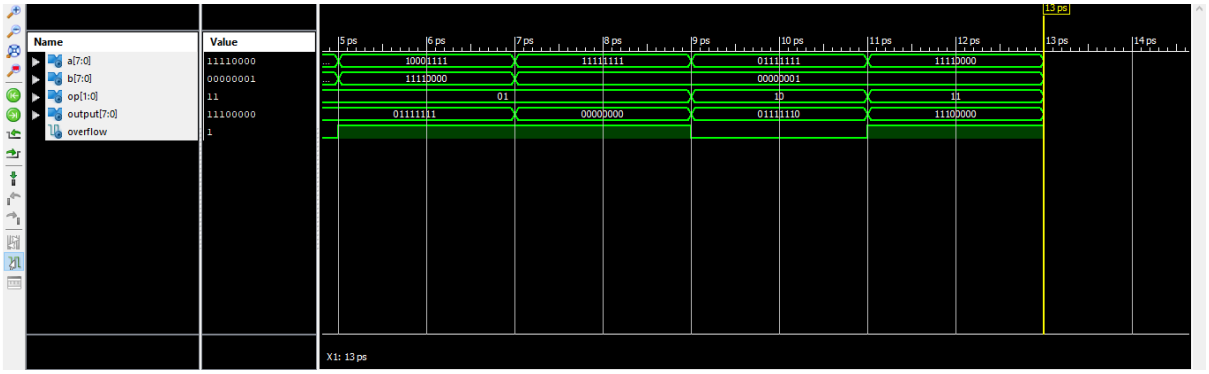

Рис. 3 – Часова діаграма ALU

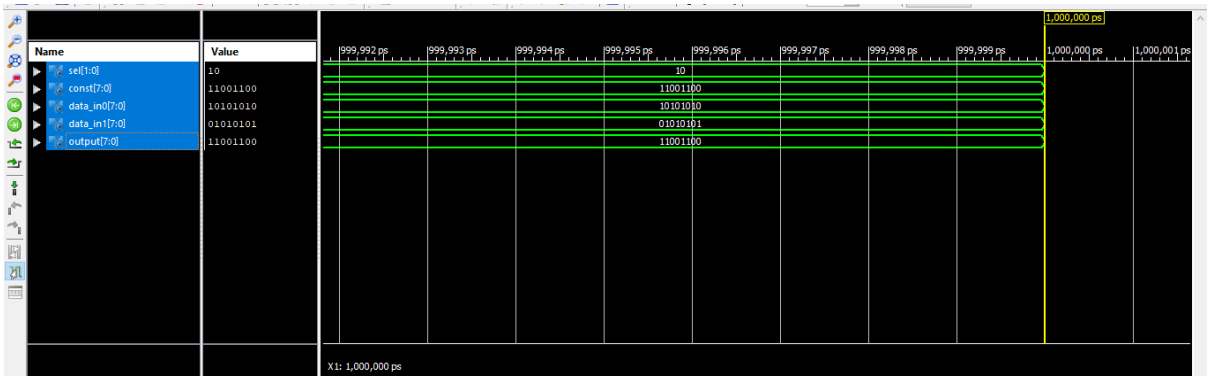

Рис. 4 – Часова діаграма MUX

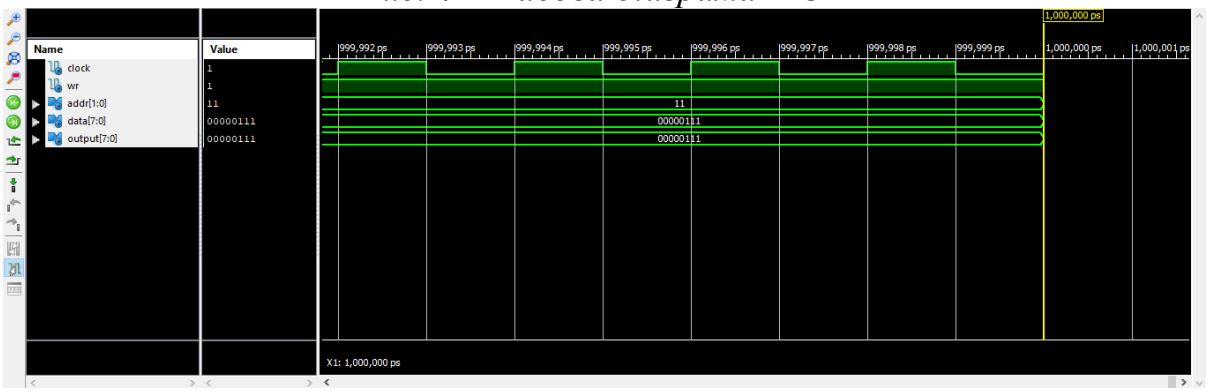

Рис. 5 – Часова діаграма RAM

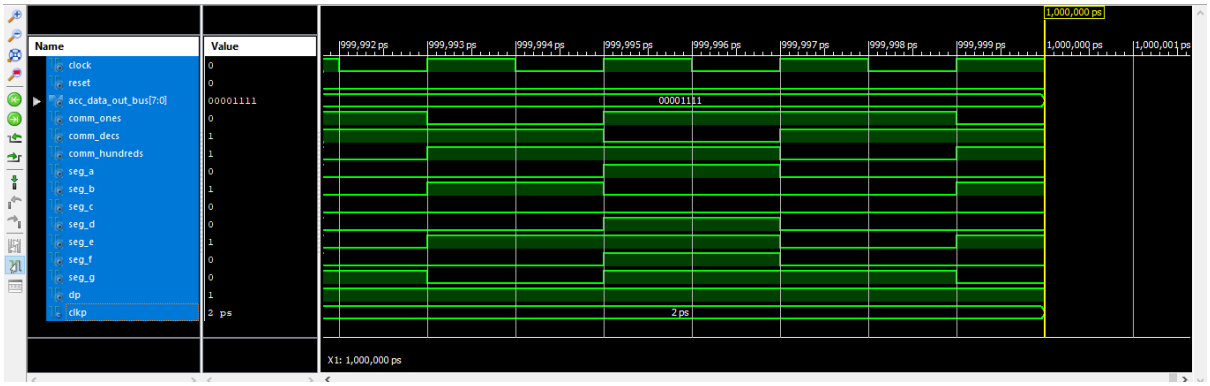

Рис 6. – Часова діграма SEG_DECODER

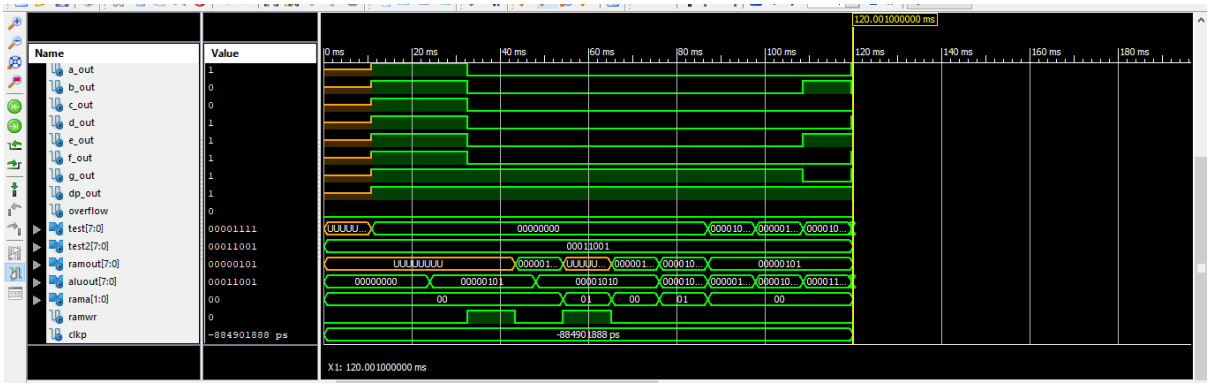

Рис 7. – Часова діграма TopLevel

Файл TopLevelTest.vhd

```
1.  -- Vhdl test bench created from schematic
    C:\Users\User\Documents\Lab_3\TopLevel.sch - Mon Apr 29 19:00:56 2024
2.  --
3.  -- Notes:
4.  -- 1) This testbench template has been automatically generated using types
5.  -- std_logic and std_logic_vector for the ports of the unit under test.
6.  -- Xilinx recommends that these types always be used for the top-level
7.  -- I/O of a design in order to guarantee that the testbench will bind
8.  -- correctly to the timing (post-route) simulation model.
9.  -- 2) To use this template as your testbench, change the filename to any
10. -- name of your choice with the extension .vhd, and use the "Source->Add"
11. -- menu in Project Navigator to import the testbench. Then
12. -- edit the user defined section below, adding code to generate the
13. -- stimulus for your design.
14. --
15. LIBRARY ieee;
16. USE ieee.std_logic_1164.ALL;
17. USE ieee.numeric_std.ALL;
18. LIBRARY UNISIM;
19. USE UNISIM.Vcomponents.ALL;
20. ENTITY TopLevel_TopLevel_sch_tb IS
21. END TopLevel_TopLevel_sch_tb;
22. ARCHITECTURE behavioral OF TopLevel_TopLevel_sch_tb IS
23.
24.    COMPONENT TopLevel
25.    PORT( CLOCK :       IN      STD_LOGIC;
26.          RESET         :       IN      STD_LOGIC;
27.          ENTER_OP1     :       IN      STD_LOGIC;
28.          ENTER_OP2     :       IN      STD_LOGIC;
29.          CALCULATE     :       IN      STD_LOGIC;
30.          DATA_IN       :       IN      STD_LOGIC_VECTOR (7 DOWNTO 0);
31.          COMMON_0_OUT          :       OUT     STD_LOGIC;
32.          COMMON_1_OUT          :       OUT     STD_LOGIC;
33.          COMMON_2_OUT          :       OUT     STD_LOGIC;
34.                          TEST: OUT STD_LOGIC_VECTOR(7 downto 0);
35.          A_OUT         :       OUT     STD_LOGIC;
36.          B_OUT         :       OUT     STD_LOGIC;
37.          C_OUT         :       OUT     STD_LOGIC;
38.          D_OUT         :       OUT     STD_LOGIC;
39.          E_OUT         :       OUT     STD_LOGIC;
40.          F_OUT         :       OUT     STD_LOGIC;
41.          G_OUT         :       OUT     STD_LOGIC;
42.          DP_OUT        :       OUT     STD_LOGIC;
43.                          RAMOUT: OUT STD_LOGIC_VECTOR(7 downto 0);
44.                          ALUOUT: OUT STD_LOGIC_VECTOR(7 downto 0);
45.                          RAMA: OUT STD_LOGIC_VECTOR(1 downto 0);
46.                          RAMWR: OUT STD_LOGIC;
```

```
47.           OVERFLOW    :       OUT     STD_LOGIC);
48.    END COMPONENT;
49.
50.    SIGNAL CLOCK       :       STD_LOGIC := '0';
51.    SIGNAL RESET       :       STD_LOGIC;
52.    SIGNAL ENTER_OP1   :       STD_LOGIC;
53.    SIGNAL ENTER_OP2   :       STD_LOGIC;
54.    SIGNAL CALCULATE   :       STD_LOGIC;
55.    SIGNAL DATA_IN     :       STD_LOGIC_VECTOR (7 DOWNTO 0);
56.    SIGNAL COMMON_0_OUT        :       STD_LOGIC;
57.    SIGNAL COMMON_1_OUT        :       STD_LOGIC;
58.    SIGNAL COMMON_2_OUT        :       STD_LOGIC;
59.    SIGNAL A_OUT       :       STD_LOGIC;
60.    SIGNAL B_OUT       :       STD_LOGIC;
61.    SIGNAL C_OUT       :       STD_LOGIC;
62.    SIGNAL D_OUT       :       STD_LOGIC;
63.    SIGNAL E_OUT       :       STD_LOGIC;
64.    SIGNAL F_OUT       :       STD_LOGIC;
65.    SIGNAL G_OUT       :       STD_LOGIC;
66.    SIGNAL DP_OUT      :       STD_LOGIC;
67.    SIGNAL OVERFLOW    :       STD_LOGIC;
68.        SIGNAL TEST: STD_LOGIC_VECTOR(7 downto 0);
69.        SIGNAL TEST2: STD_LOGIC_VECTOR(7 downto 0);
70.        signal RAMOUT: STD_LOGIC_VECTOR(7 downto 0);
71.        signal ALUOUT: STD_LOGIC_VECTOR(7 downto 0);
72.        signal RAMA: STD_LOGIC_VECTOR(1 downto 0);
73.        signal RAMWR: STD_LOGIC;
74.
75. --     constant CLOCK_period : time := 166ns;
76.        constant CLKP: time := 12ms;--24ms;
77.
78. BEGIN
79.
80.    UUT: TopLevel PORT MAP(
81.                CLOCK => CLOCK,
82.                RESET => RESET,
83.                ENTER_OP1 => ENTER_OP1,
84.                ENTER_OP2 => ENTER_OP2,
85.                CALCULATE => CALCULATE,
86.                DATA_IN => DATA_IN,
87.                COMMON_0_OUT => COMMON_0_OUT,
88.                COMMON_1_OUT => COMMON_1_OUT,
89.                COMMON_2_OUT => COMMON_2_OUT,
90.                A_OUT => A_OUT,
91.                B_OUT => B_OUT,
92.                C_OUT => C_OUT,
93.                D_OUT => D_OUT,
94.                E_OUT => E_OUT,
95.                F_OUT => F_OUT,
96.                G_OUT => G_OUT,
97.                DP_OUT => DP_OUT,
98.                OVERFLOW => OVERFLOW,
99.                TEST => TEST,
100.                       RAMOUT => RAMOUT,
101.                       ALUOUT => ALUOUT,
102.                       RAMA => RAMA,
103.                       RAMWR => RAMWR
104.         );
105.
106.             CLOCK_process: process
107.         begin
108.                       CLOCK <= '0';
109.                       wait for 83ns;
110.                       CLOCK <= '1';
111.                       wait for 83ns;
112.        end process;
```

```
113.
114.       -- *** Test Bench - User Defined Section ***
115.          tb : PROCESS
116.          BEGIN
117.                      lp1: for i in 4 to 4 loop
118.                             lp2: for j in 2 to 2 loop
119.                                    TEST2 <=
    std_logic_vector(to_unsigned(to_integer(signed(std_logic_vector(to_unsigned(j,
    8)) or std_logic_vector(to_unsigned(i, 8)))) + j + 10 - 3, 8));
120.                                    ENTER_OP1 <= '1';
121.                                    ENTER_OP2 <= '1';
122.                                    CALCULATE <= '1';
123.                                    DATA_IN <= (others => '0');
124.                                    RESET <= '0';
125.                                    wait for CLKP;
126.                                    RESET <= '1';
127.                                    wait for CLKP;
128.                                    DATA_IN <= std_logic_vector(to_unsigned(i,
    8)); -- A
129.                                    ENTER_OP1 <= '0';
130.                                    wait for CLKP;
131.                                    ENTER_OP1 <= '1';
132.                                    wait for CLKP;
133.                                    DATA_IN <= std_logic_vector(to_unsigned(j,
    8)); -- B
134.                                    ENTER_OP2 <= '0';
135.                                    wait for CLKP;
136.                                    ENTER_OP2 <= '1';
137.                                    wait for CLKP;
138.                                    CALCULATE <= '0'; -- START CALCULATION
139.                                    wait for CLKP* 7;
140.                                    assert TEST = TEST2 severity FAILURE;
141.                                    wait for CLKP;
142.                             end loop;
143.                      end loop;
144.
145.             WAIT; -- will wait forever
146.          END PROCESS;
147.       -- *** End Test Bench - User Defined Section ***
148.
149.       END;
```

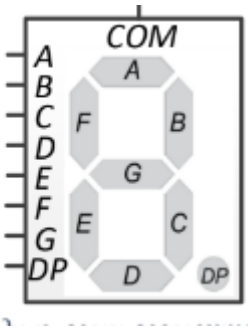

Рис.8 – 7-сегментний індикатор

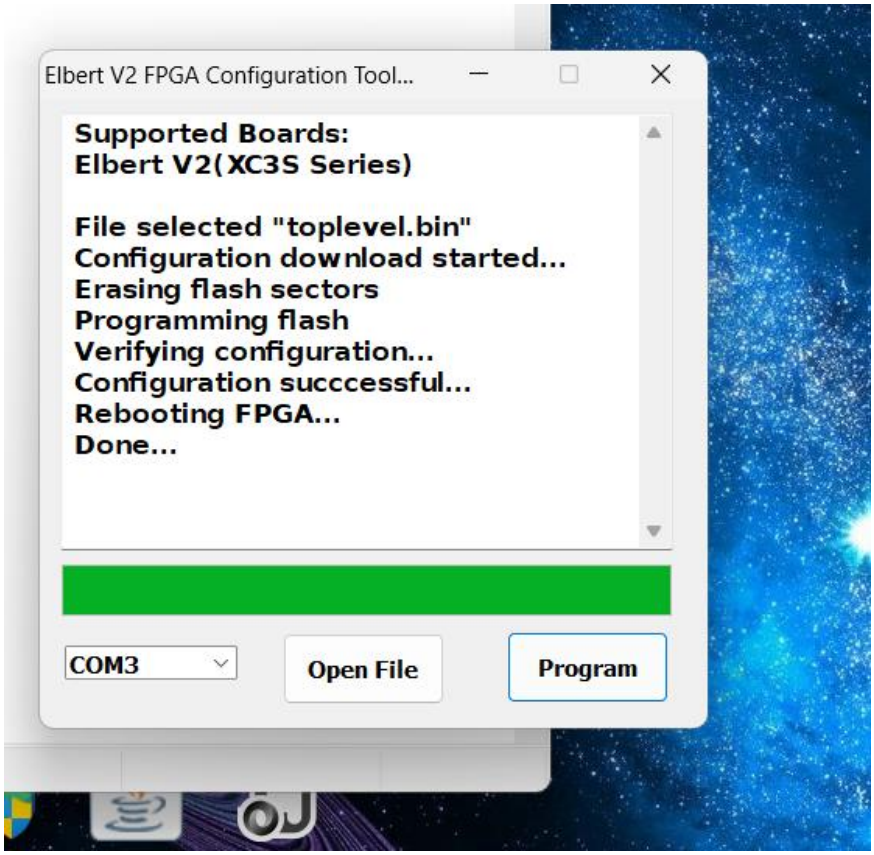

Рис.9 – Успішна прошивка

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