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 №22

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

Львів 2024

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

Виконання роботи:

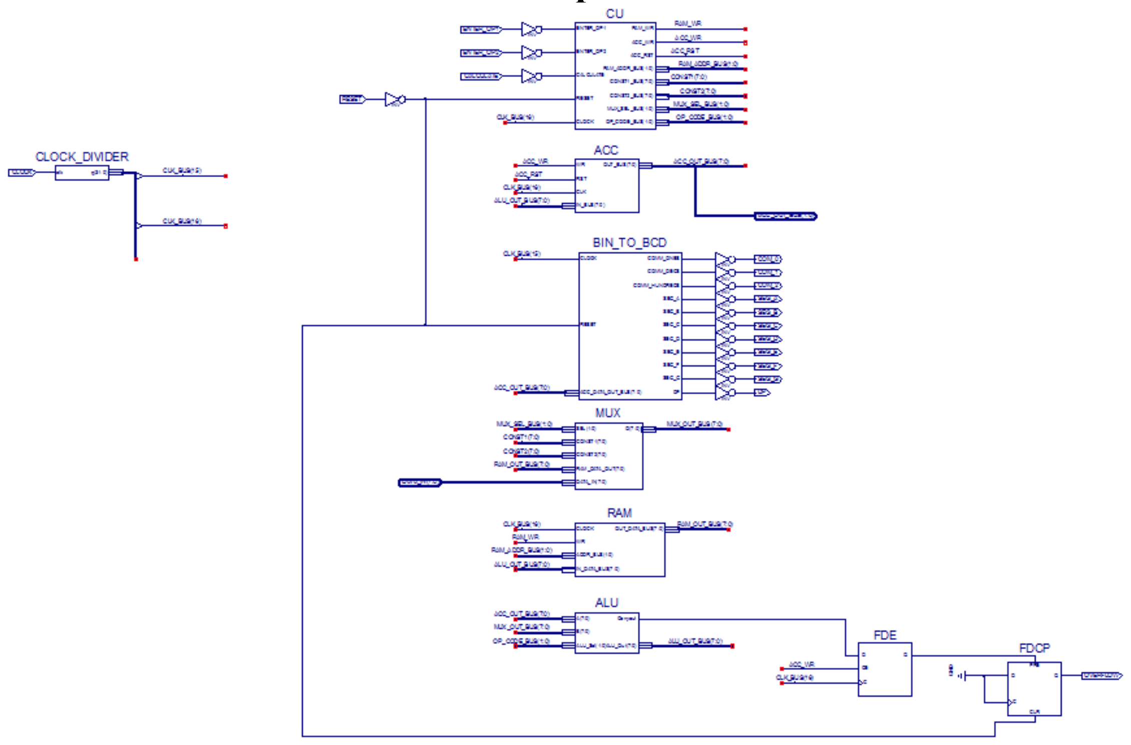


Рис. 1 – Top Level

Файл ACC.vhd

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

entity ACC is
   Port ( WR   : in  STD_LOGIC;
          RST  : in  STD_LOGIC;
          CLK  : in  STD_LOGIC;
          IN_BUS : in  STD_LOGIC_VECTOR (7 downto 0);
          OUT_BUS : 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 RST = '1' then
                DATA <= (others => '0');
            elsif WR = '1' then
                DATA <= IN_BUS;
            end if;
        end if;
    end process;


    OUT_BUS <= DATA;

end ACC_arch;
```

Файл ALU.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.NUMERIC_STD.all;
entity ALU is
    Port (
    A, B     : in  STD_LOGIC_VECTOR(7 downto 0);
    ALU_Sel  : in  STD_LOGIC_VECTOR(1 downto 0);
    ALU_Out   : out  STD_LOGIC_VECTOR(7 downto 0);
    Carryout : out std_logic
    );
end ALU;
architecture Behavioral of ALU is

signal ALU_Result : std_logic_vector (15 downto 0);

begin
   process(A,B,ALU_Sel)
 begin
  case(ALU_Sel) is
  when "01" =>
   ALU_Result <= ("00000000" & A) + ("00000000" & B);
  when "10" =>
   ALU_Result <=
std_logic_vector(to_unsigned((to_integer(unsigned(("00000000" & A))) *
to_integer(unsigned(("00000000" & B)))),16)) ;
  when "11" =>
   case(B) is
```

```
            when x"00"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 0);
            when x"01"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 1);
            when x"02"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 2);
            when x"03"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 3);
            when x"04"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 4);
            when x"05"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 5);
            when x"06"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 6);
            when x"07"   => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 7);
            when others  => ALU_Result <=
std_logic_vector(unsigned(("00000000" & A)) sll 0);
  end case;
  ALU_Result(15 downto 8) <= "00000000";
  when others => ALU_Result <= ("00000000" & B);
  end case;
 end process;
 ALU_Out <= ALU_Result(7 downto 0);
 Carryout <= ALU_Result(8) OR ALU_Result(9) OR ALU_Result(10) OR
ALU_Result(11) OR ALU_Result(12) OR ALU_Result(13) OR
ALU_Result(14) OR ALU_Result(15);
end Behavioral;
```

Файл CPU.vhd

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



entity CU 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_BUS : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      CONST1_BUS : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
```

```
    CONST2_BUS : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    ACC_WR : OUT STD_LOGIC;
    ACC_RST : OUT STD_LOGIC;
    MUX_SEL_BUS : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
    OP_CODE_BUS : OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
end CU;

architecture CU_arch of CU 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
  CONST1_BUS <= "00000001";
  CONST2_BUS <= "00001010";


  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 (CUR_STATE, ENTER_OP1,
ENTER_OP2, CALCULATE)
   begin
    --declare default state for next_state to avoid latches
    NEXT_STATE <= CUR_STATE;  --default is to stay in current state
    --insert statements to decode next_state
    --below is a simple example
   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     =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST     <= '1';
        ACC_WR     <= '0';
      when IDLE     =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST     <= '0';
        ACC_WR      <= '0';
      when LOAD_OP1   =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '1';
        ACC_RST     <= '0';
        ACC_WR      <= '1';
      when LOAD_OP2   =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "01";
        RAM_WR      <= '1';
        ACC_RST     <= '0';
        ACC_WR      <= '1';
      when RUN_CALC0 =>
        MUX_SEL_BUS   <= "01";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "01";
        RAM_WR      <= '0';
        ACC_RST     <= '0';
        ACC_WR      <= '1';
      when RUN_CALC1 =>
        MUX_SEL_BUS   <= "01";
        OP_CODE_BUS   <= "10";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST     <= '0';
        ACC_WR      <= '1';
      when RUN_CALC2 =>
        MUX_SEL_BUS   <= "10";
        OP_CODE_BUS   <= "11";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST     <= '0';
        ACC_WR      <= '1';
      when RUN_CALC3 =>
        MUX_SEL_BUS   <= "01";
        OP_CODE_BUS   <= "01";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
```

```
        ACC_RST      <= '0';
        ACC_WR      <= '1';
      when RUN_CALC4 =>
        MUX_SEL_BUS   <= "11";
        OP_CODE_BUS   <= "01";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST      <= '0';
        ACC_WR      <= '1';
      when FINISH   =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST      <= '0';
        ACC_WR      <= '0';
      when others    =>
        MUX_SEL_BUS   <= "00";
        OP_CODE_BUS   <= "00";
        RAM_ADDR_BUS  <= "00";
        RAM_WR      <= '0';
        ACC_RST      <= '0';
        ACC_WR      <= '0';
    end case;
   end process;
end CU_arch;
```

```
Файл MUX.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX_Testbench is
end MUX_Testbench;

architecture Behavioral of MUX_Testbench is
    -- Constants

    -- Signals
    signal SEL : std_logic_vector(1 downto 0) := "00";
    signal CONST1 : std_logic_vector(7 downto 0) := (others => '0');
    signal CONST2 : std_logic_vector(7 downto 0) := (others => '1');
```

```
   signal RAM_DATA_OUT : std_logic_vector(7 downto 0) := (others =>
'0');
   signal DATA_IN : std_logic_vector(7 downto 0) := (others => '1');
   signal O : std_logic_vector(7 downto 0);

begin

   -- Stimulus process
   stimulus: process
   begin
      -- Test case 1
      SEL <= "00";
      CONST1 <= "00001100";
      CONST2 <= "01010101";
      RAM_DATA_OUT <= "11110000";
      DATA_IN <= "11111111";
      wait for 10 ms;
      -- Test case 2
      SEL <= "01";
      wait for 10 ms;

      -- Test case 3
      SEL <= "10";
      wait for 10 ms;

      -- Test case 4
      SEL <= "11";
      wait for 10 ms;

      -- Add more test cases as needed

      wait;
   end process;

   -- Instantiate the MUX
   UUT: entity work.MUX
      port map(
         SEL => SEL,
         CONST1 => CONST1,
         CONST2 => CONST2,
         RAM_DATA_OUT => RAM_DATA_OUT,
         DATA_IN => DATA_IN,
         O => O
```

```
     );

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( CLOCK : STD_LOGIC;
      WR : IN STD_LOGIC;
      ADDR_BUS : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
      IN_DATA_BUS : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      OUT_DATA_BUS : 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(CLOCK, ADDR_BUS, UNIT)
   begin
    if (rising_edge(CLOCK)) then
      if (WR = '1') then
        UNIT(conv_integer(ADDR_BUS)) <= IN_DATA_BUS;
      end if;
    end if;
    OUT_DATA_BUS <= UNIT(conv_integer(ADDR_BUS));
  end process;



end RAM_arch;
```

Файл SEG_DECODER.vhd
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity BIN_TO_BCD 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 BIN_TO_BCD;

architecture Behavioral of BIN_TO_BCD 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 ;

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

        COMM_ONES     <= COMMONS_CTRL(0);
        COMM_DECS     <= COMMONS_CTRL(1);
        COMM_HUNDREDS <= COMMONS_CTRL(2);

        SEG_A <= DIGIT_CTRL(6);
        SEG_B <= DIGIT_CTRL(5);
        SEG_C <= DIGIT_CTRL(4);
        SEG_D <= DIGIT_CTRL(3);
        SEG_E <= DIGIT_CTRL(2);
SEG_F <= DIGIT_CTRL(1);
        SEG_G <= DIGIT_CTRL(0);
        DP   <= '0';

      end if;
  end process INDICATE;

end Behavioral;
```

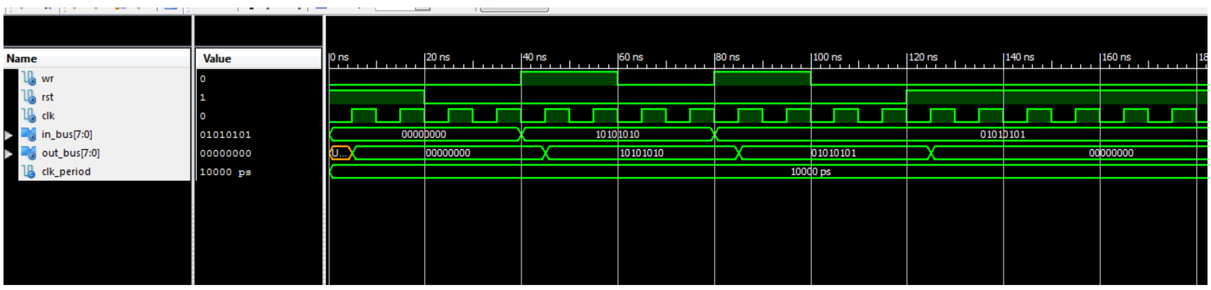


Рис. 2 – Часова діаграма ACC

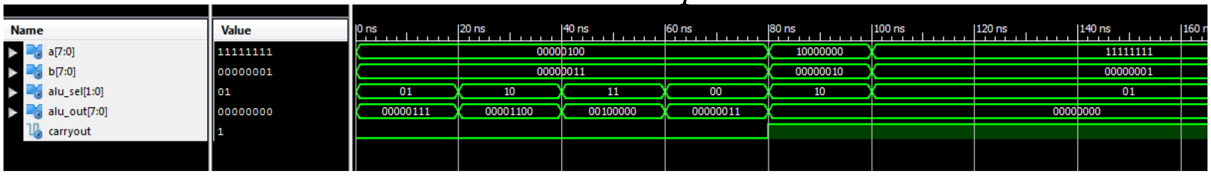


Рис. 3 – Часова діаграма ALU

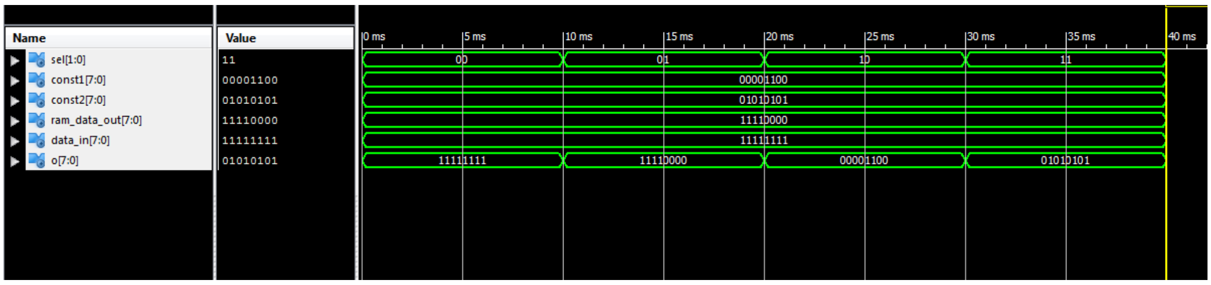


Рис. 4 – Часова діаграма MUX

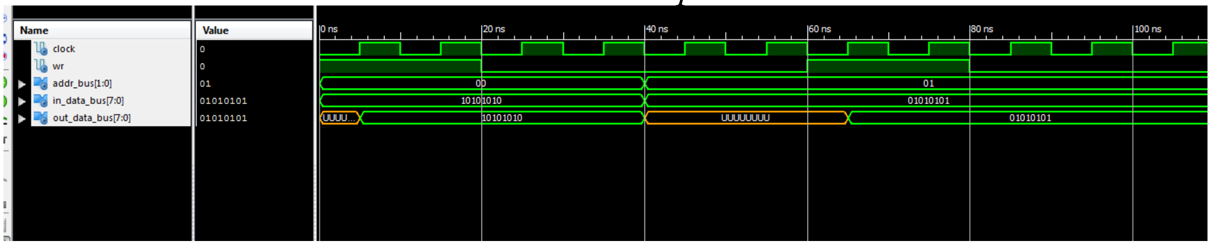


Рис. 5 – Часова діаграма RAM

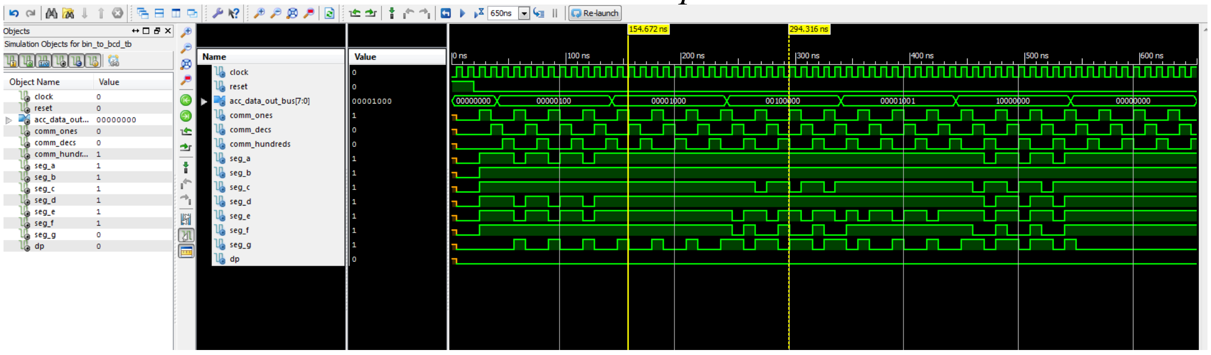


Рис 6. – Часова діграма SEG_DECODER

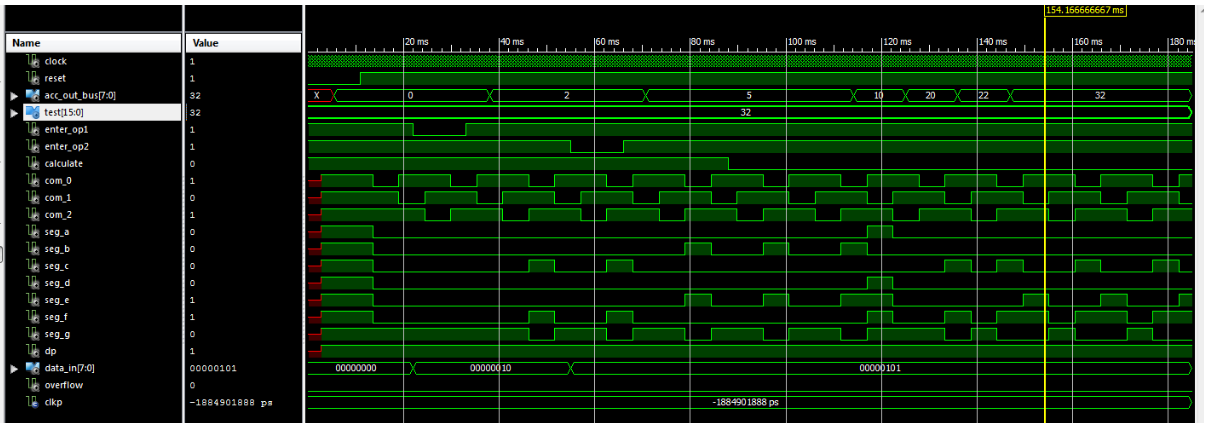


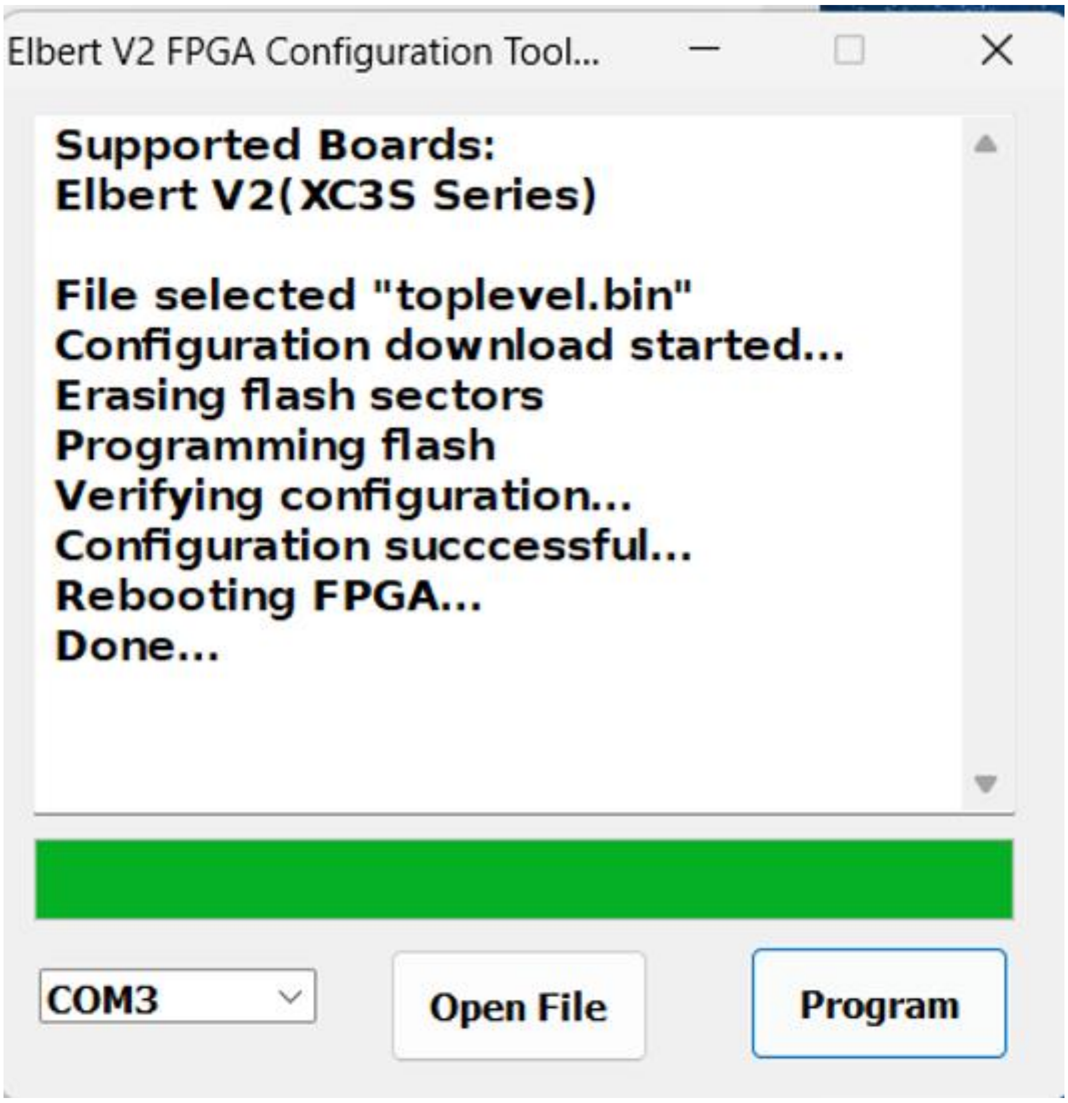
Рис 7. – Часова діграма TopLevel

Файл TopLevelTest.vhd

```
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;
         ENTER_OP1  :  IN  STD_LOGIC;
         ENTER_OP2  :  IN  STD_LOGIC;
         CALCULATE  :  IN  STD_LOGIC;
      ACC_OUT_BUS : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
         COM_0  :  OUT  STD_LOGIC;
         COM_1  :  OUT  STD_LOGIC;
         COM_2  :  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;
         DATA_IN  :  IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
         OVERFLOW  :  OUT  STD_LOGIC);
   END COMPONENT;
```

```
  SIGNAL CLOCK  :  STD_LOGIC;
  SIGNAL RESET  :  STD_LOGIC;
  SIGNAL ENTER_OP1  :  STD_LOGIC;
  SIGNAL ENTER_OP2  :  STD_LOGIC;
  SIGNAL CALCULATE  :  STD_LOGIC;
  SIGNAL COM_0  :  STD_LOGIC;
  SIGNAL COM_1  :  STD_LOGIC;
  SIGNAL COM_2  :  STD_LOGIC;
  SIGNAL SEG_A  :  STD_LOGIC;
  SIGNAL SEG_B  :  STD_LOGIC;
  SIGNAL SEG_C  :  STD_LOGIC;
  SIGNAL SEG_D  :  STD_LOGIC;
  SIGNAL SEG_E  :  STD_LOGIC;
  SIGNAL SEG_F  :  STD_LOGIC;
  SIGNAL SEG_G  :  STD_LOGIC;
  SIGNAL DP  :  STD_LOGIC;
  SIGNAL DATA_IN  :  STD_LOGIC_VECTOR (7 DOWNTO 0);
 SIGNAL ACC_OUT_BUS  :  STD_LOGIC_VECTOR (7 DOWNTO 0);
 SIGNAL TEST : STD_LOGIC_VECTOR (15 DOWNTO 0);
  SIGNAL OVERFLOW  :  STD_LOGIC;

 constant CLKP: time := 11 ms;
BEGIN

  UUT: TOP_SCHEME PORT MAP(
   CLOCK => CLOCK,
   RESET => RESET,
   ENTER_OP1 => ENTER_OP1,
   ENTER_OP2 => ENTER_OP2,
   CALCULATE => CALCULATE,
   COM_0 => COM_0,
   COM_1 => COM_1,
   COM_2 => COM_2,
   SEG_A => SEG_A,
   SEG_B => SEG_B,
   SEG_C => SEG_C,
   SEG_D => SEG_D,
   SEG_E => SEG_E,
   SEG_F => SEG_F,
   SEG_G => SEG_G,
   DP => DP,
   DATA_IN => DATA_IN,
    ACC_OUT_BUS => ACC_OUT_BUS,
    OVERFLOW => OVERFLOW
   );

  CLOCK_process: process
   begin
     CLOCK <= '0';
     wait for 41500 ps;
     CLOCK <= '1';
     wait for 41500 ps;
    end process;

-- * Test Bench - User Defined Section *
   tb : PROCESS
   BEGIN

     lp1: for i in 2 to 2 loop
      lp2: for j in 5 to 5 loop
     TEST <=
std_logic_vector(to_unsigned(((to_integer(unsigned(std_logic_vector(to_unsi
gned(i, 8))))* 2 * to_integer(unsigned(std_logic_vector(to_unsigned(j, 8)))))
+ to_integer(unsigned(std_logic_vector(to_unsigned(i, 8)))) + 10), 16));
       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 * 2;
       DATA_IN <= (std_logic_vector(to_unsigned(j, 8))); -- B
       ENTER_OP2 <= '0';
       wait for CLKP;
       ENTER_OP2 <= '1';
       wait for CLKP * 2;
       CALCULATE <= '0'; -- START CALCULATION
     REPORT "OP1 = (" & integer'image(i) & ") and OP2 = (" &
integer'image(j) & ") calculation started" SEVERITY NOTE;
```

```
       wait for CLKP * 7;
     assert ACC_OUT_BUS = TEST(7 DOWNTO 0) severity FAILURE;
     REPORT "OP1 = (" & integer'image(i) & ") and OP2 = (" &
integer'image(j) & ") calculation finished" SEVERITY NOTE;
     wait for CLKP;
      end loop;
     end loop;
     WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```



```
Supported Boards:
Elbert V2(XC3S Series)

File selected "toplevel.bin"
Configuration download started...
Erasing flash sectors
Programming flash
Verifying configuration...
Configuration succcessful...
Rebooting FPGA...
Done...
```

Рис.9 – Успішна прошивка

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