Міністерство освіти і науки України
Національний університет „Львівська політехніка”
Кафедра ЕОМ

Звіт

З лабораторної роботи № 3

З дисципліни “Моделювання комп’ютерних систем”

На тему: “Поведінковий опис цифрового автомата. Перевірка роботи автомата за допомогою стенда”

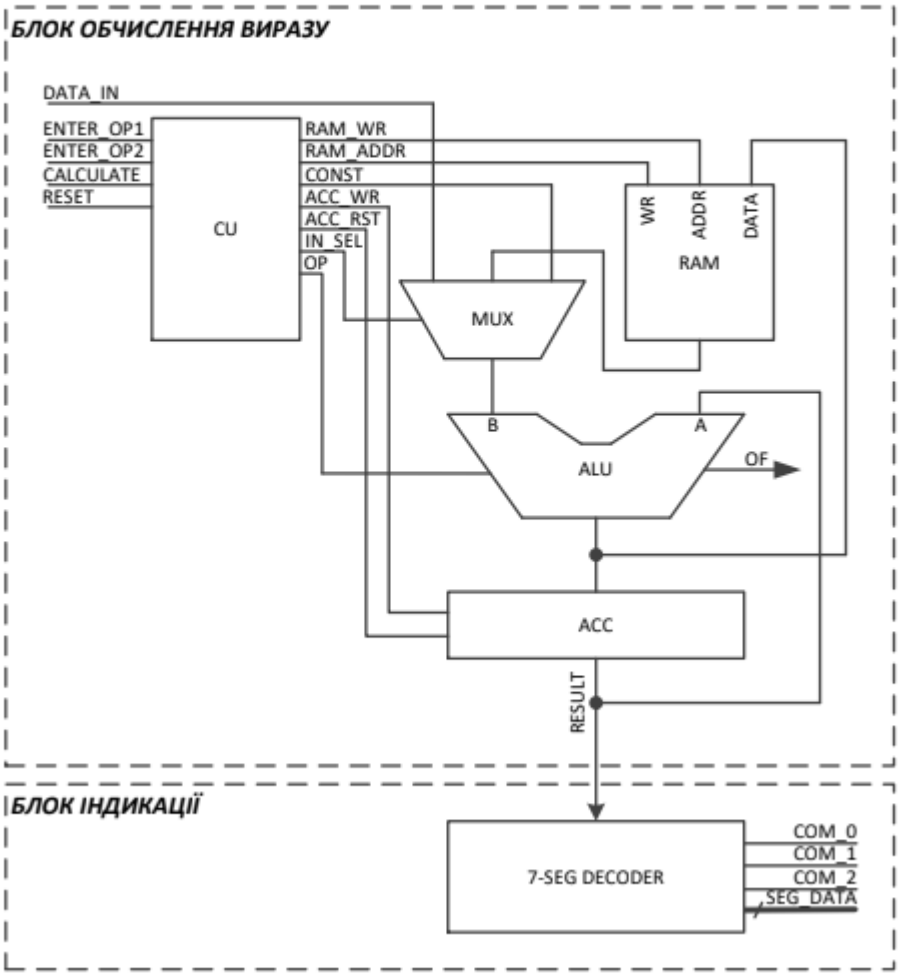
Варіант – 1

Виконав: ст.гр. КІ-202
Лозинський А.Я.
Перевірив:
Козак Н.Б.

Львів 2023

Мета роботи : На базі стенда Elbert V2 – Spartan 3A FPGA, реалізувати цифровий автомат для обчислення значення виразу дотримуючись наступних вимог:

1. Функціонал пристрою повинен бути реалізований згідно отриманого варіанту завдання. Дивись розділ ЗАВДАННЯ:.
2. Пристрій повинен бути ітераційним (АЛП *(ALU)* повинен виконувати за один такт одну операцію), та реалізованим згідно наступної структурної схеми (*Малюнок 1*):



Малюнок 1 - Структурна схема автомата.

Завдання

1	((OP1 + 2) * OP2) << OP1

Виконання роботи:

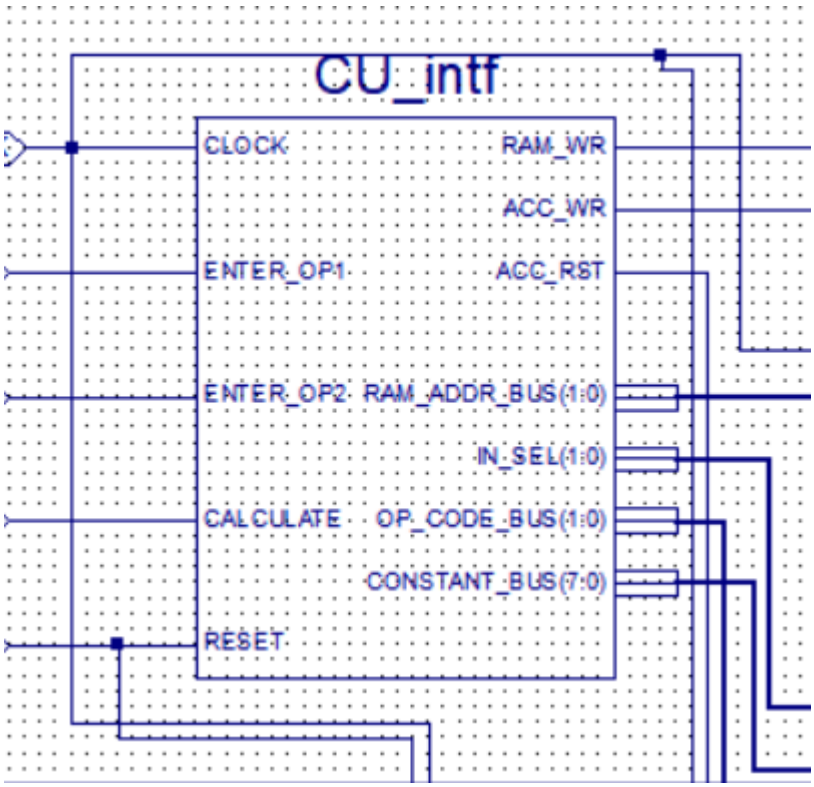
Файл CU.vhd:

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


entity CU_intf is
Port(
CLOCK       :  in std_logic;
ENTER_OP1    :  in std_logic;
ENTER_OP2    :  in std_logic;
CALCULATE    :  in std_logic;
RESET       :  in std_logic;



RAM_WR      : out std_logic;
RAM_ADDR_BUS : out std_logic_vector(1 downto 0);
CONSTANT_BUS : inout 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_CODE_BUS  : out std_logic_vector(1 downto 0)
);

end CU_intf;

architecture CU_arch of CU_intf is

type   cu_state_type is (cu_rst, cu_idle, cu_load_op1, cu_load_op2, cu_run_calc0, cu_run_calc1, cu_run_calc2,
cu_run_calc3, cu_finish);
signal cu_cur_state  : cu_state_type;
signal cu_next_state : cu_state_type;



begin
CONSTANT_BUS <= "00000010";


CU_SYNC_PROC: process (CLOCK)

  begin
    if (rising_edge(CLOCK)) then
      if (RESET = '1') then
        cu_cur_state <= cu_rst;
      else
        cu_cur_state <= cu_next_state;
      end if;
    end if;
  end process;

 CUNEXT_STATE_DECODE: process (cu_cur_state, ENTER_OP1, ENTER_OP2, CALCULATE)
  begin
```

```
  cu_next_state <= cu_cur_state;

 case(cu_cur_state) is
  when cu_rst     =>
    cu_next_state <= cu_idle;
  when cu_idle     =>
    if (ENTER_OP1 = '1') then
      cu_next_state <= cu_load_op1;
    elsif (ENTER_OP2 = '1') then
      cu_next_state <= cu_load_op2;
    elsif (CALCULATE = '1') then
      cu_next_state <= cu_run_calc0;
    else
      cu_next_state <= cu_idle;
    end if;

  when cu_load_op1   =>
    cu_next_state <= cu_idle;

  when cu_load_op2   =>
    cu_next_state <= cu_idle;

  when cu_run_calc0 =>
    cu_next_state <= cu_run_calc1;

  when cu_run_calc1 =>
    cu_next_state <= cu_run_calc2;

  when cu_run_calc2 =>
    cu_next_state <= cu_run_calc3;

  when cu_run_calc3 =>
    cu_next_state <= cu_finish;

  when cu_finish   =>
    cu_next_state <= cu_finish;

  when others      =>
    cu_next_state <= cu_idle;

 end case;
end process;




CU_OUTPUT_DECODE: process (cu_cur_state)
begin
 case(cu_cur_state) is
  when cu_rst     =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '1';
    ACC_WR       <= '0';
  when cu_idle     =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
```

```
    ACC_RST      <= '0';
    ACC_WR       <= '0';
   when cu_load_op1   =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '1';
    ACC_RST      <= '0';
    ACC_WR       <= '1';
   when cu_load_op2   =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "01";
    RAM_WR       <= '1';
    ACC_RST      <= '0';
    ACC_WR       <= '1';


                        when cu_run_calc0 =>
    IN_SEL       <= "01";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '1';
   when cu_run_calc1 =>
    IN_SEL       <= "11";
    OP_CODE_BUS   <= "01";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '1';

   when cu_run_calc2 =>
    IN_SEL       <= "01";
    OP_CODE_BUS   <= "10";
    RAM_ADDR_BUS  <= "01";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '1';

   when cu_run_calc3 =>
    IN_SEL       <= "01";
    OP_CODE_BUS   <= "11";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '1';

   when cu_finish   =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '0';
   when others    =>
    IN_SEL       <= "00";
    OP_CODE_BUS   <= "00";
    RAM_ADDR_BUS  <= "00";
    RAM_WR       <= '0';
    ACC_RST      <= '0';
    ACC_WR       <= '0';
 end case;
end process;
```

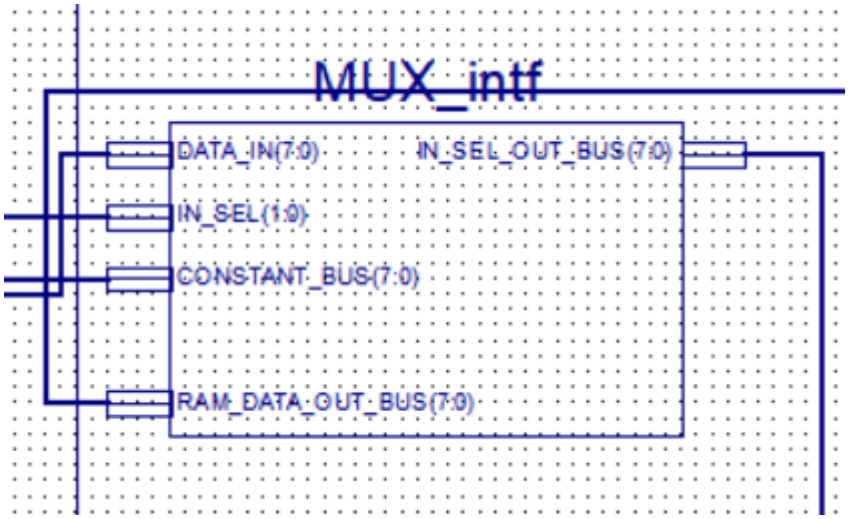
Елемент CU:



Файл MUX.vhd:

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


entity MUX_intf is
Port(
        DATA_IN         :  in std_logic_vector(7 downto 0);
        IN_SEL          :  in  std_logic_vector(1 downto 0);
        CONSTANT_BUS    :  in  std_logic_vector(7 downto 0);
        RAM_DATA_OUT_BUS :  in  std_logic_vector(7 downto 0);
        IN_SEL_OUT_BUS   :  out std_logic_vector(7 downto 0)
        );

end MUX_intf;

 architecture MUX_arch of MUX_intf is

begin

   INSEL_A_MUX : process(DATA_IN, CONSTANT_BUS,RAM_DATA_OUT_BUS, IN_SEL)
        begin
                if(IN_SEL = "00") then
                        IN_SEL_OUT_BUS <= DATA_IN;
                elsif(IN_SEL = "01") then
                        IN_SEL_OUT_BUS <= RAM_DATA_OUT_BUS;
                else
                        IN_SEL_OUT_BUS <= CONSTANT_BUS;
                end if;
        end process INSEL_A_MUX;


end MUX_arch;
```

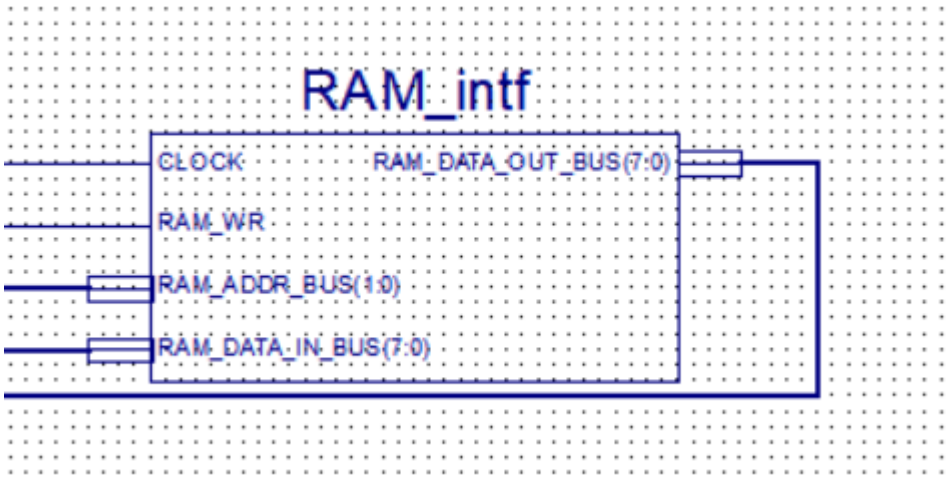
Елемент MUX:



Файл RAM.vhd:

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



entity RAM_intf is
port(
CLOCK           : in  std_logic;
RAM_WR          : in  std_logic;
RAM_ADDR_BUS    : in  STD_LOGIC_VECTOR(1 downto 0);
RAM_DATA_IN_BUS  : in  STD_LOGIC_VECTOR(7 downto 0);
RAM_DATA_OUT_BUS : out STD_LOGIC_VECTOR(7 downto 0)
);

end RAM_intf;

architecture RAM_arch of RAM_intf is
type ram_type is array (3 downto 0) of STD_LOGIC_VECTOR(7 downto 0);
signal RAM_UNIT                 : ram_type;
begin



RAM : process(CLOCK, RAM_ADDR_BUS, RAM_UNIT)
        begin
                if (rising_edge(CLOCK)) then
                        if (RAM_WR = '1') then
                                RAM_UNIT(conv_integer(RAM_ADDR_BUS)) <= RAM_DATA_IN_BUS;
                        end if;
                end if;
                RAM_DATA_OUT_BUS <= RAM_UNIT(conv_integer(RAM_ADDR_BUS));
        end process RAM;



end RAM_arch;
```

Елемент RAM:

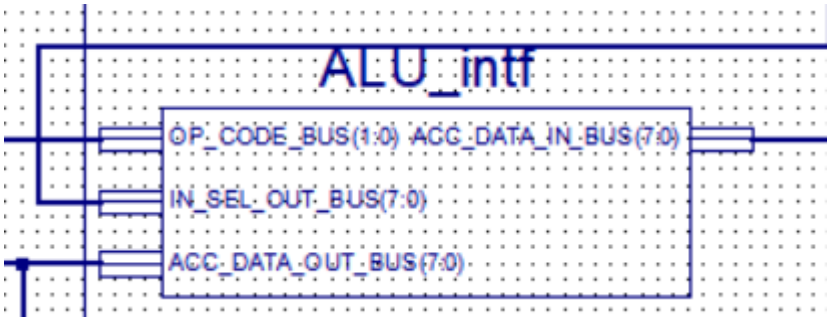


Файл ALU.vhd:

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

entity ALU_intf is
port(
	OP_CODE_BUS      : in  std_logic_vector(1 downto 0);
	IN_SEL_OUT_BUS   : in  std_logic_vector(7 downto 0);
	ACC_DATA_OUT_BUS : in  std_logic_vector(7 downto 0);
	ACC_DATA_IN_BUS  : out std_logic_vector(7 downto 0)
);

end ALU_intf;

architecture ALU_arch of ALU_intf is

begin

 ALU : process(OP_CODE_BUS, IN_SEL_OUT_BUS, ACC_DATA_OUT_BUS)
		variable A : unsigned(7 downto 0);
		variable B : unsigned(7 downto 0);
		variable TEMP_MUL : unsigned (15 downto 0);
	 begin
		A := unsigned(ACC_DATA_OUT_BUS);
		B := unsigned(IN_SEL_OUT_BUS);

		case(OP_CODE_BUS) is
			when "00"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(B);
			when "01"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A + B);
			when "10"	=> TEMP_MUL:=(A * B);
								ACC_DATA_IN_BUS <=
STD_LOGIC_VECTOR(TEMP_MUL(7 downto 0));
			when "11"	=>
				case(B) is
					when x"00"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 0);
					when x"01"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 1);
					when x"02"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 2);
					when x"03"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 3);
					when x"04"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 4);
					when x"05"	=> ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 5);
```

```
                                        when x"06"      => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 6);
                                        when x"07"      => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A
sll 7);
                                        when others  => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll
0);
                                end case;
                        when others => ACC_DATA_IN_BUS <= "00000000";
                end case;
                        end process ALU;

end ALU_arch;
```

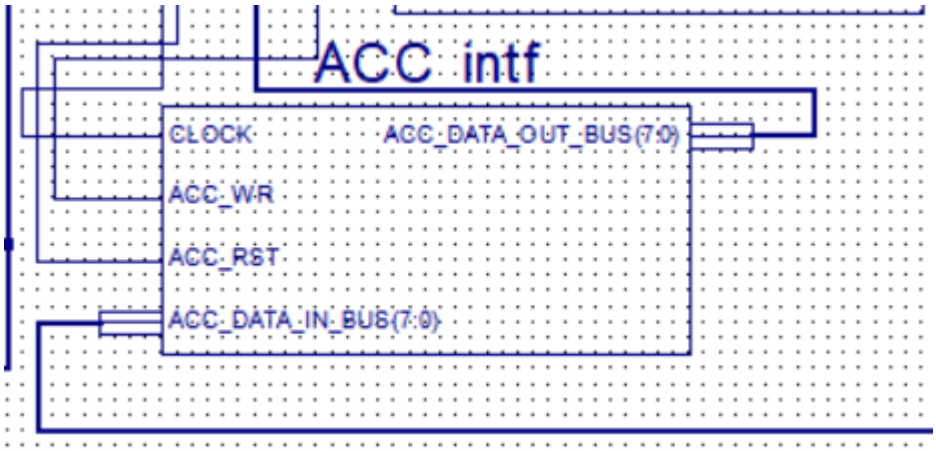
Елемент ALU:



Файл ACC.vhd:

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


entity ACC_intf is
port(
        CLOCK           : in  std_logic;
        ACC_WR          : in  std_logic;
        ACC_RST         : in  std_logic;
        ACC_DATA_IN_BUS  : in  std_logic_vector(7 downto 0);
        ACC_DATA_OUT_BUS : out std_logic_vector(7 downto 0)
);

end ACC_intf;

architecture ACC_arch of ACC_intf is
signal ACC_DATA : std_logic_vector(7 downto 0);
begin
ACC : process(CLOCK, ACC_DATA)
        begin
                if (rising_edge(CLOCK)) then
                        if(ACC_RST = '1') then
                                ACC_DATA <= "00000000";
                        elsif (ACC_WR = '1') then
                                ACC_DATA <= ACC_DATA_IN_BUS;
                        end if;
                end if;
                ACC_DATA_OUT_BUS <= ACC_DATA;
        end process ACC;



end ACC_arch;
```

Елемент ACC:



Файл DEC.vhd:

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



entity DEC_intf 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 DEC_intf;

architecture DEC_arch of DEC_intf 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 DEC_arch;
```

Елемент DEC:

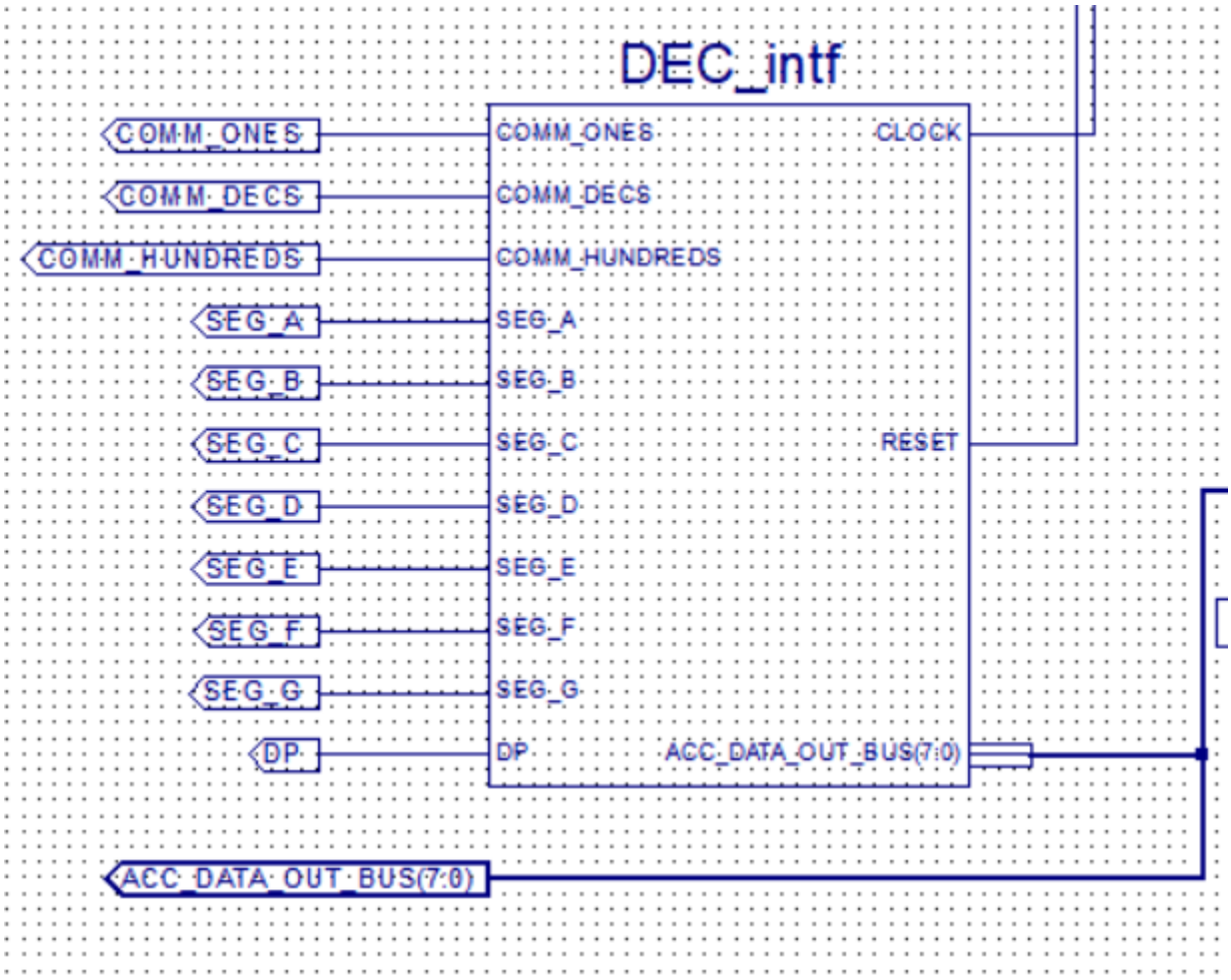
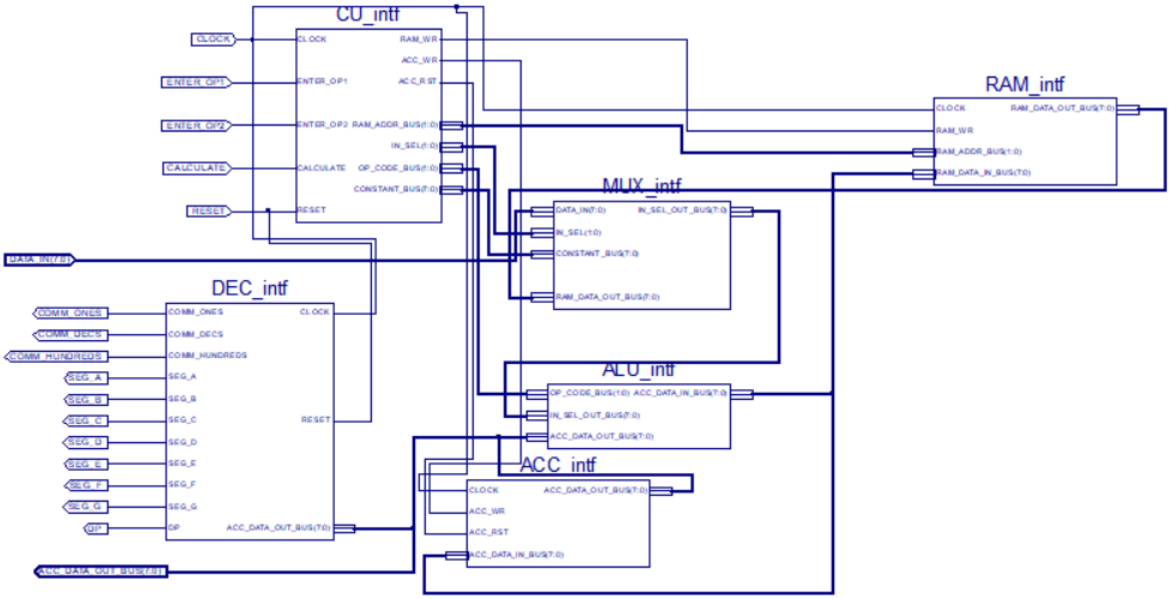


Схема для Top Level:

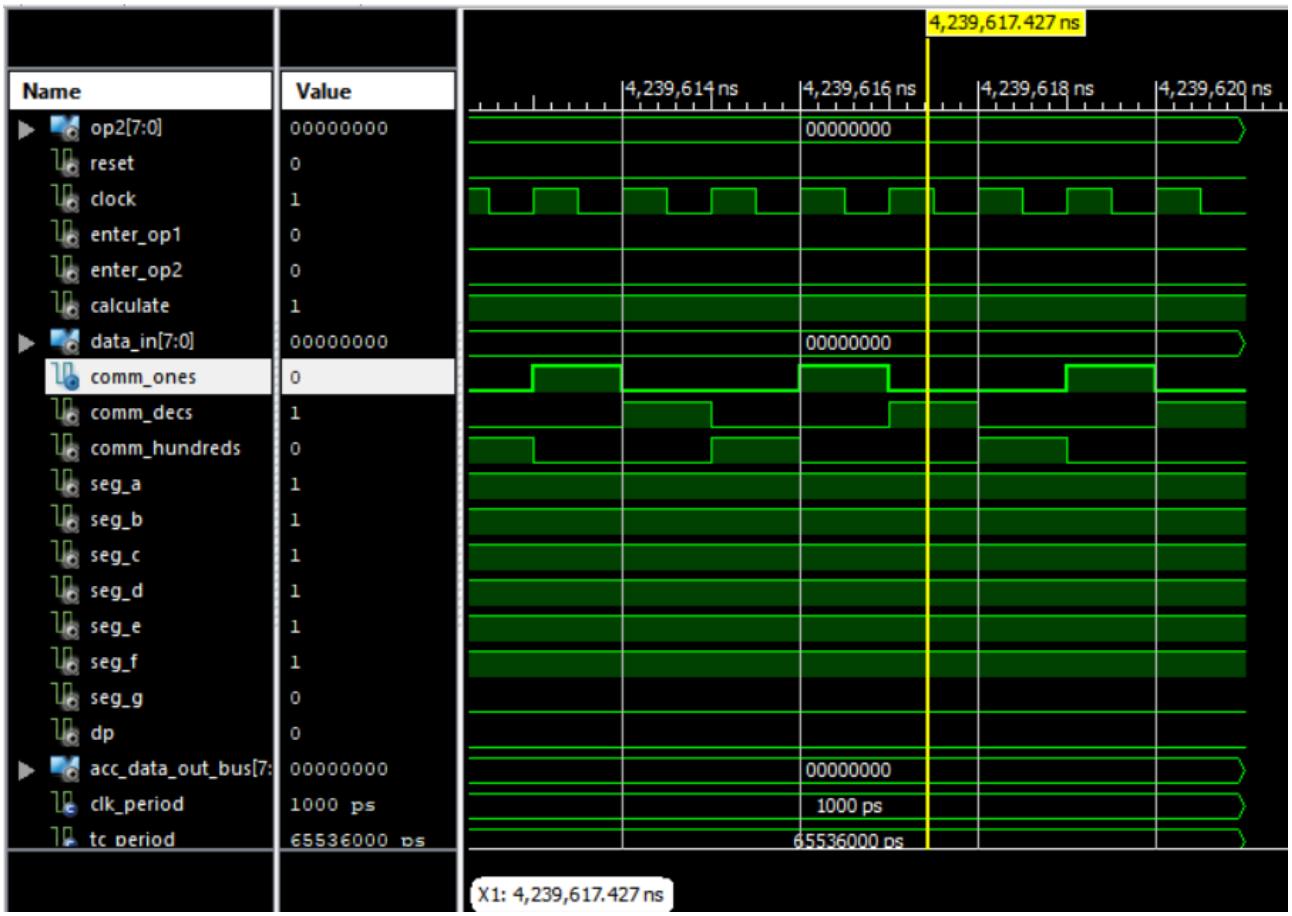


Файл TestTopLevel.vhd:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
use std.textio.all;
use ieee.std_logic_textio.all;
use IEEE.std_logic_signed.all;



entity TB_TOPLEVEL_intf is
end TB_TOPLEVEL_intf;

architecture TB_TOPLEVEL_arch of TB_TOPLEVEL_intf is


  COMPONENT TopLevel
  PORT(  RESET  :  IN  STD_LOGIC;
         CLOCK  :  IN  STD_LOGIC;
         ENTER_OP1  :  IN  STD_LOGIC;
         ENTER_OP2  :  IN  STD_LOGIC;
         CALCULATE  :  IN  STD_LOGIC;

         DATA_IN  :  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;
       ACC_DATA_OUT_BUS : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
       );
```

```
  END COMPONENT;


  signal op1 : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
  signal op2 : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";

  signal RESET :    STD_LOGIC;
  signal CLOCK :    STD_LOGIC;
  signal ENTER_OP1 :    STD_LOGIC;
  signal ENTER_OP2 :    STD_LOGIC;
  signal CALCULATE :    STD_LOGIC;
  signal DATA_IN :    STD_LOGIC_VECTOR (7 DOWNTO 0);
  signal COMM_ONES :    STD_LOGIC;
  signal COMM_DECS :    STD_LOGIC;
  signal COMM_HUNDREDS :    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 ACC_DATA_OUT_BUS :  STD_LOGIC_VECTOR(7 DOWNTO 0);


   constant CLK_period: time := 1 ns;
   constant TC_period: time := 65536 ns;



BEGIN

   UUT: TopLevel
  PORT MAP(
    RESET => RESET,
    CLOCK => CLOCK,
    ENTER_OP1 => ENTER_OP1,
    ENTER_OP2 => ENTER_OP2,
    CALCULATE => CALCULATE,
    DATA_IN => DATA_IN,
    COMM_ONES => COMM_ONES,
    COMM_DECS => COMM_DECS,
    COMM_HUNDREDS => COMM_HUNDREDS,
    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,
    ACC_DATA_OUT_BUS=>ACC_DATA_OUT_BUS

   );

 CLK_process : process
            begin
                        CLOCK <= '1';
                        wait for CLK_period/2;
                        CLOCK <= '0';
```

```
                wait for CLK_period/2;
end process CLK_process;


  stim_proc: process

  begin

  wait for 2*CLK_period;
  RESET <= '1';
  ENTER_OP1 <= '0';
  ENTER_OP2 <= '0';
  CALCULATE <= '0';
  DATA_IN   <=(others => '0');

  wait for 2*TC_period;
  RESET <='0';

  wait for 4*TC_period;
  ENTER_OP1 <='1';
  DATA_IN   <= op1;


  wait for 2*TC_period;
  ENTER_OP1 <='0';


    wait for 4*TC_period;
  ENTER_OP2 <= '1';
  DATA_IN   <= op2;

   wait for 2*TC_period;
  ENTER_OP2 <= '0';

  wait for 4*TC_period;
  CALCULATE <= '1';



  wait for 5*TC_period;
    wait;
end process stim_proc;



end TB_TOPLEVEL_arch;
```



Висновок: Під час даної лабораторної роботи, я на базі стенда Elbert V2 – Spartan 3A FPGA, реалізував цифровий автомат для обчислення значення заданого виразу.