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”

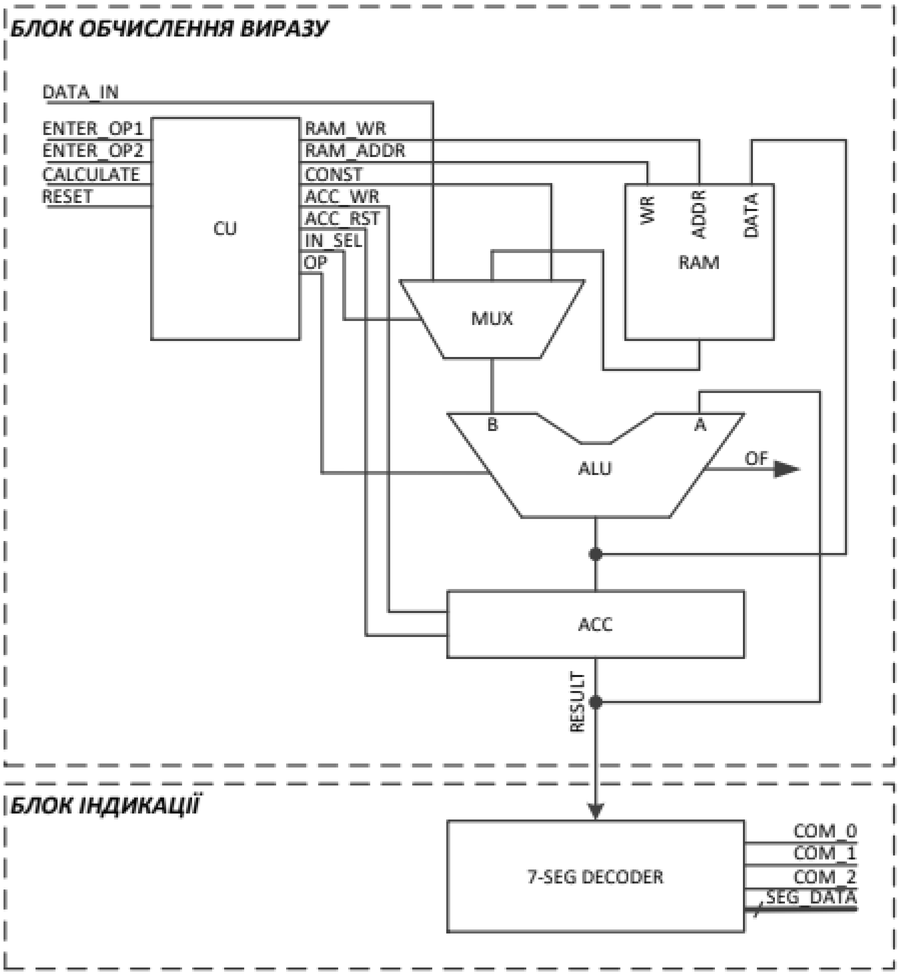
Варіант – 1

Виконав: ст. гр. КІ-201
Моравський О-Б.С.
Прийняв:
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:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:27:31 04/27/2023
-- Design Name:
-- Module Name:    CU - 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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity CU_intf is
	port(CLOCK		: IN STD_LOGIC;
		RESET		: IN STD_LOGIC;
		ENTER_OP1	: IN STD_LOGIC;
		ENTER_OP2	: IN STD_LOGIC;
		CALCULATE	: IN STD_LOGIC;

		RAM_WR : OUT STD_LOGIC;
		RAM_ADDR_BUS : OUT STD_LOGIC_VECTOR(1 downto 0);
```

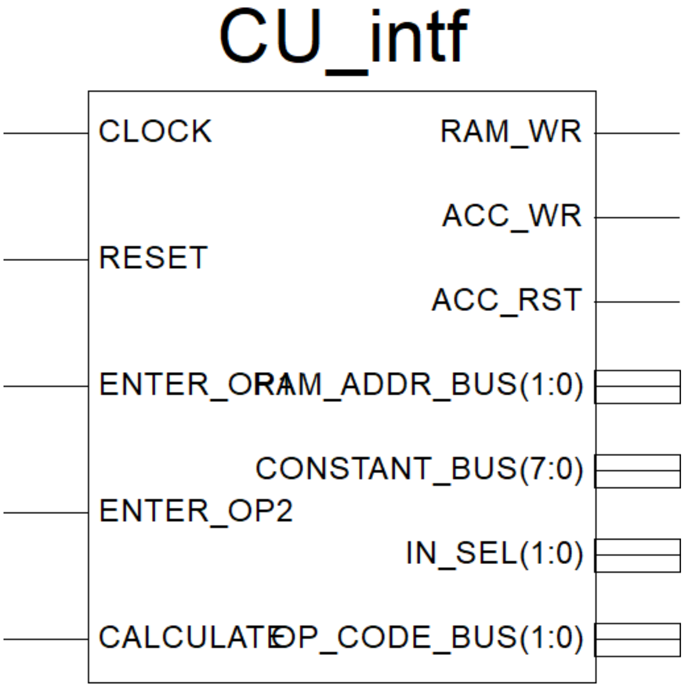
```
                CONSTANT_BUS       : OUT STD_LOGIC_VECTOR(7 downto 0):=
"00000010";
                ACC_WR : OUT STD_LOGIC;
                ACC_RST : OUT STD_LOGIC;
                IN_SEL : OUT STD_LOGIC_VECTOR(1 downto 0);
                OP_CODE_BUS : OUT STD_LOGIC_VECTOR(1 downto 0)
                );
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
    --declare default state for next_state to avoid latches
    cu_next_state <= cu_cur_state;  --default is to stay in current state
    --insert statements to decode next_state
    --below is a simple example
              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              <= "10";
        OP_CODE_BUS  <= "01";
        RAM_ADDR_BUS        <= "01";
        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;
end CU_arch;
```

Елемент CU:

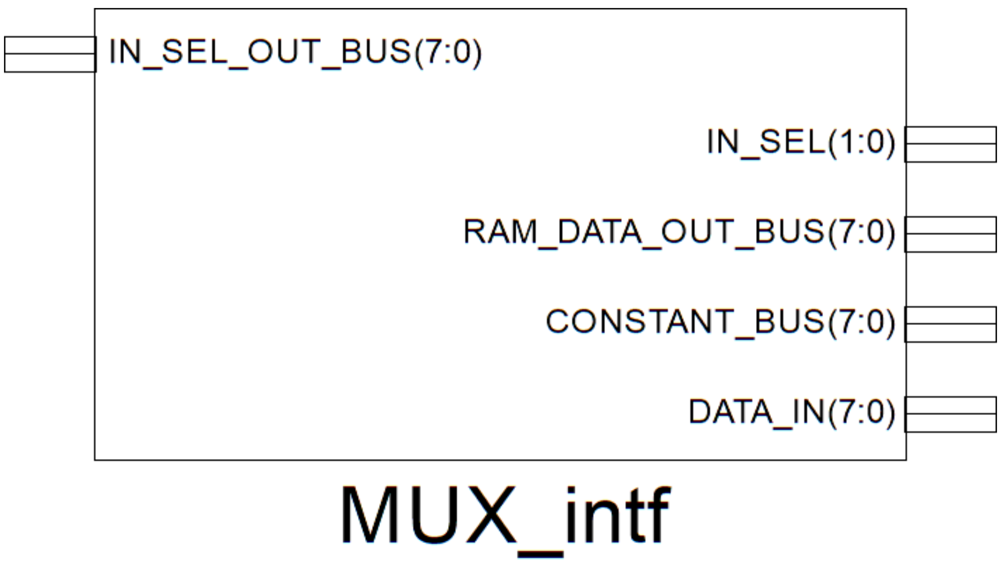


Файл MUX.vhd:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    15:06:55 04/27/2023
-- Design Name:
-- Module Name:    MUX - 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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity MUX_intf is
	port(
		DATA_IN			: IN STD_LOGIC_VECTOR(7 downto 0);
		CONSTANT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
		RAM_DATA_OUT_BUS: IN STD_LOGIC_VECTOR(7 downto 0);
		IN_SEL				: IN STD_LOGIC_VECTOR(1 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:

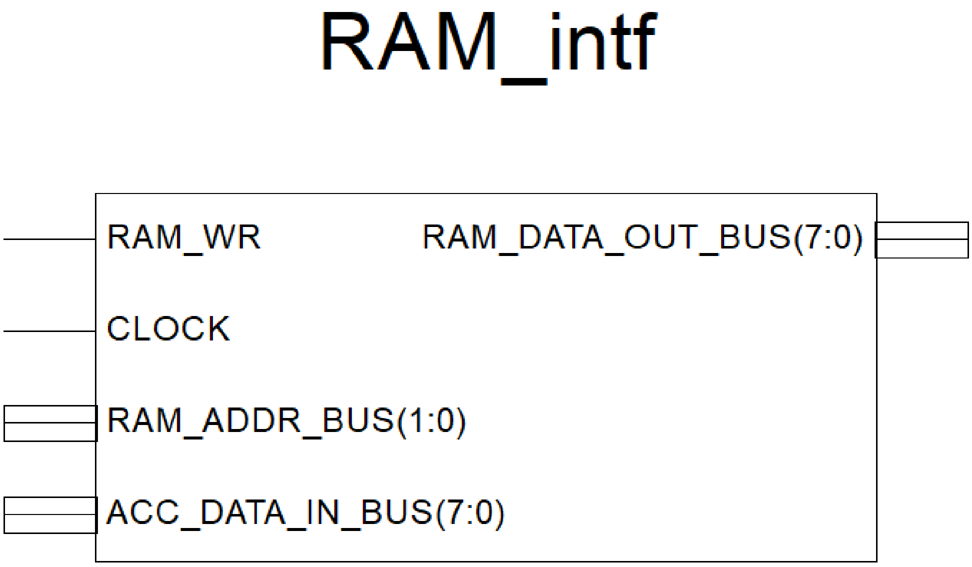


Файл RAM.vhd:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:49:14 04/27/2023
-- Design Name:
-- Module Name:    RAM - 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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity RAM_intf is
port(
		RAM_WR				: IN STD_LOGIC;
		RAM_ADDR_BUS			: IN STD_LOGIC_VECTOR(1 downto 
0);
		ACC_DATA_IN_BUS	: IN STD_LOGIC_VECTOR(7 downto 0);
		RAM_DATA_OUT_BUS: OUT STD_LOGIC_VECTOR(7 downto 0);
		CLOCK			: IN STD_LOGIC
		);
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;
signal RAM_DATA_IN_BUS	: STD_LOGIC_VECTOR(7 downto 0);

begin
	RAM_DATA_IN_BUS <= ACC_DATA_IN_BUS;

	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:



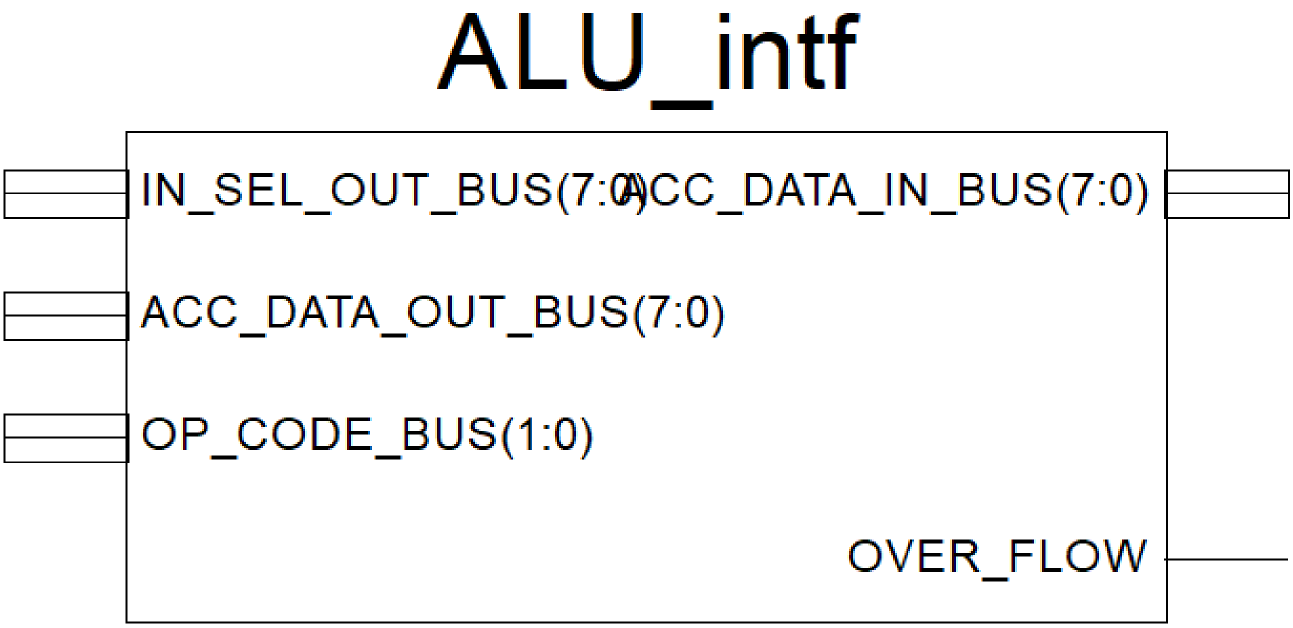
Файл ALU.vhd:

```
----------------------------------------------------------------------------------
-- Company: 
-- Engineer: 
-- 
-- Create Date:    16:13:46 04/27/2023 
-- Design Name: 
-- Module Name:    ALU - 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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ALU_intf is
port(
		IN_SEL_OUT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
		ACC_DATA_OUT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
		OP_CODE_BUS : IN STD_LOGIC_VECTOR(1 downto 0);
		ACC_DATA_IN_BUS : OUT STD_LOGIC_VECTOR(7 downto 0);
		OVER_FLOW : OUT STD_LOGIC
		--OF - overflow
		);
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 : std_logic_vector(8 downto 0);
      begin
            A := unsigned(ACC_DATA_OUT_BUS);
            B := unsigned(IN_SEL_OUT_BUS);

            if OP_CODE_BUS = "00" then
                  ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(B);
            elsif OP_CODE_BUS = "01" then
                  temp := STD_LOGIC_VECTOR('0' & A) +
STD_LOGIC_VECTOR('0' & B);
                        if (temp(8) = '1') then
                              OVER_FLOW <= '1';
                              else
                                    OVER_FLOW <= '0';
                                    end if;
                  ACC_DATA_IN_BUS <= temp(7 downto 0);
            elsif OP_CODE_BUS = "10" then
            ACC_DATA_IN_BUS <=
STD_LOGIC_VECTOR(RESIZE(unsigned(A*B(7 downto 0)), 8));
            elsif OP_CODE_BUS = "11" then
                   case(B) is --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;
            else
                  ACC_DATA_IN_BUS <= "00000000";
            end if;
      end process ALU;
```

end ALU_arch;

Елемент ALU:

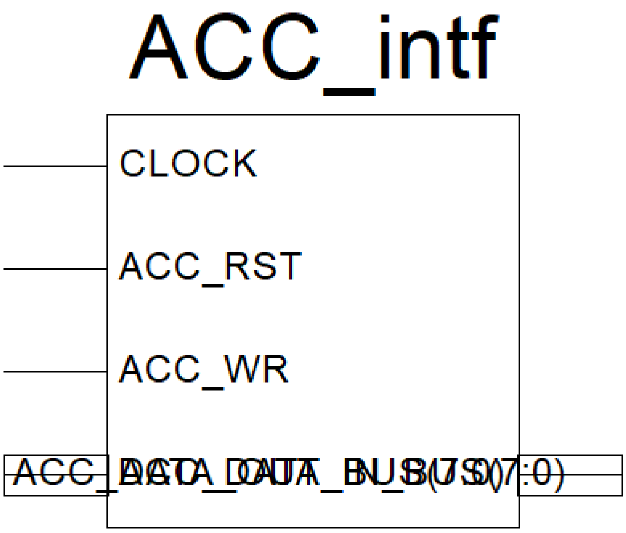


Файл ACC.vhd:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    15:27:57 04/27/2023
-- 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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ACC_intf is
port(
		CLOCK		: IN STD_LOGIC;
		ACC_RST			: IN STD_LOGIC;
		ACC_WR				: 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 SEGDEC.vhd:

```
----------------------------------------------------------------------------------
----------------------------------------------------------------------------------
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;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity SEGDEC_intf is
port(
		CLOCK			: IN STD_LOGIC;
		ACC_DATA_OUT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
		RESET			: IN STD_LOGIC;
		OverFlow_IN : IN STD_LOGIC;
		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;
		OverFlow_OUT : OUT STD_LOGIC := '0'
		);
end SEGDEC_intf;

architecture SEGDEC_arch of SEGDEC_intf is
signal ONES_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
signal DECS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
signal HONDREDS_BUS : STD_LOGIC_VECTOR(3 downto 0) := "0000";
begin


OVERFLOW_INDICATE : process(OverFlow_IN, RESET)
	begin
		--if rising_edge(CLOCK) then
		if (RESET = '1') then
```

```
                    OverFlow_OUT <= '0';
                    elsif (RESET = '0' and OverFlow_IN = '1') then
                        OverFlow_OUT <= '1';
                        end if;
                    --end if;
            end process OVERFLOW_INDICATE;

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 SEGDEC_arch;
```

Елемент SEGDEC:

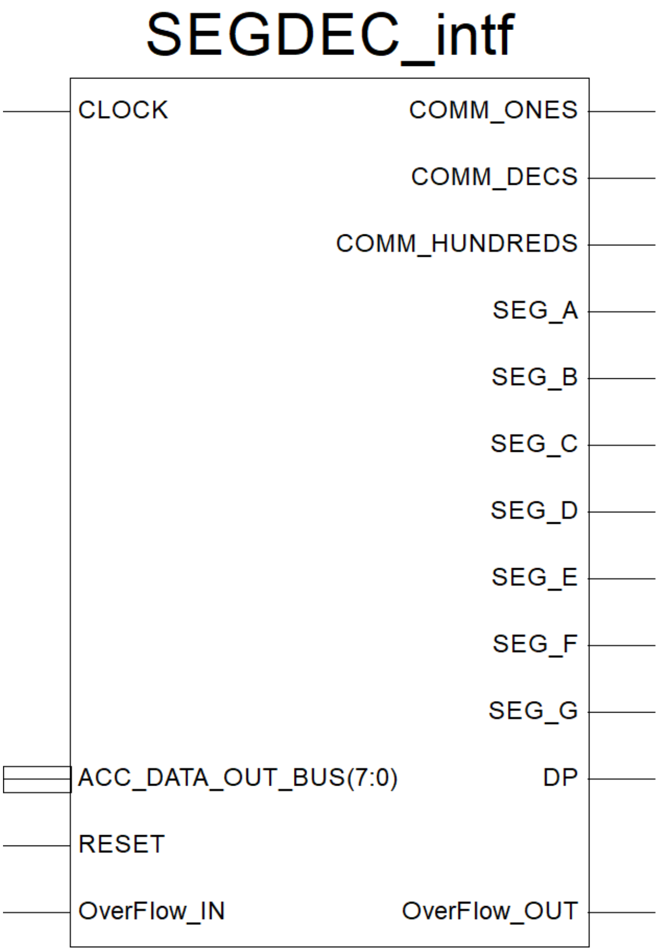
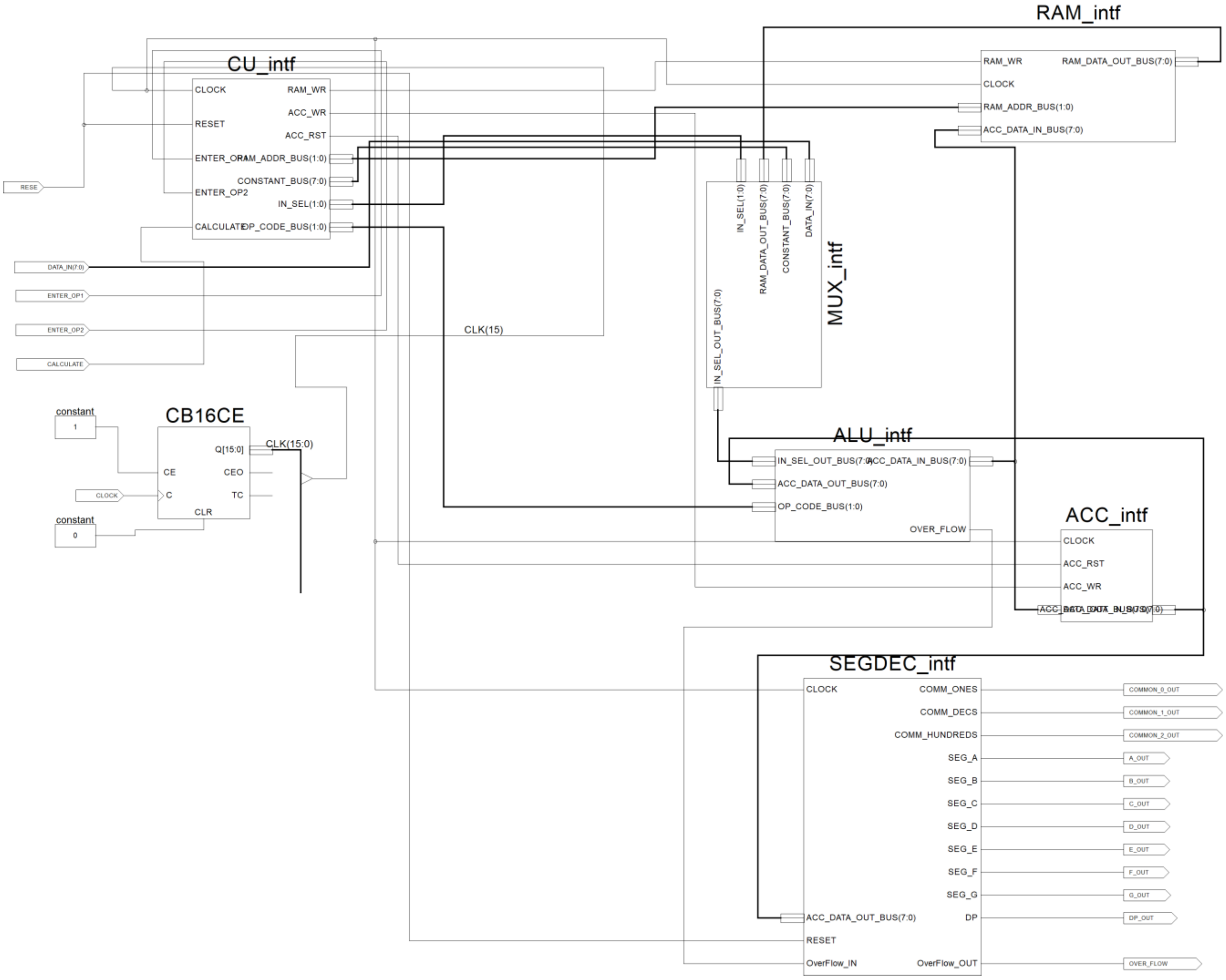


Схема для Top Level:



Файл Constraints.ucf:

#***#

\#   UCF for ElbertV2 Development Board

\#

#***#

CONFIG VCCAUX = "3.3" ;

 # Clock 12 MHz
 NET "CLOCK"   LOC = P129 | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;

##

\#   LED

##

```
   NET "OVERFLOW"            LOC = P46   | IOSTANDARD = LVCMOS33 |
SLEW = SLOW | DRIVE = 12;

######################################################################
#################################
#                          Seven Segment Display
######################################################################
#################################

   NET "A_OUT"    LOC = P117  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "B_OUT"    LOC = P116  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "C_OUT"    LOC = P115  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "D_OUT"    LOC = P113  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "E_OUT"    LOC = P112  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "F_OUT"    LOC = P111  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "G_OUT"    LOC = P110  | IOSTANDARD = LVCMOS33 | SLEW = SLOW
| DRIVE = 12;
   NET "DP_OUT"   LOC = P114  | IOSTANDARD = LVCMOS33 | SLEW =
SLOW | DRIVE = 12;

   NET "COMMON_2_OUT"         LOC = P124  | IOSTANDARD = LVCMOS33 |
SLEW = SLOW | DRIVE = 12;
   NET "COMMON_1_OUT"         LOC = P121  | IOSTANDARD = LVCMOS33 |
SLEW = SLOW | DRIVE = 12;
   NET "COMMON_0_OUT"         LOC = P120  | IOSTANDARD = LVCMOS33 |
SLEW = SLOW | DRIVE = 12;
######################################################################
#################################
#                             DP Switches
######################################################################
#################################

   NET "DATA_IN(0)"       LOC = P70   | PULLUP  | IOSTANDARD =
LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA_IN(1)"       LOC = P69   | PULLUP  | IOSTANDARD =
LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA_IN(2)"       LOC = P68   | PULLUP  | IOSTANDARD =
LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA_IN(3)"       LOC = P64   | PULLUP  | IOSTANDARD =
LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

NET "DATA_IN(4)"   LOC = P63 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "DATA_IN(5)"   LOC = P60 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "DATA_IN(6)"   LOC = P59 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "DATA_IN(7)"   LOC = P58 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##
Switches
##

NET "ENTER_OP1"   LOC = P80 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "ENTER_OP2"   LOC = P79 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "CALCULATE"   LOC = P78 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "RESET"   LOC = P75 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##

Файл TestTopLevel.vhd:

-- Vhdl test bench created from schematic D:\Lab_3_Example\TopLevel.sch - Mon May 01 21:40:52 2023
--
-- 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 TopLevel_TopLevel_sch_tb IS
END TopLevel_TopLevel_sch_tb;
ARCHITECTURE behavioral OF TopLevel_TopLevel_sch_tb IS

   COMPONENT TopLevel
   PORT( RESE	:	IN	STD_LOGIC;
          ENTER_OP1	:	IN	STD_LOGIC;
          ENTER_OP2	:	IN	STD_LOGIC;
          CALCULATE	:	IN	STD_LOGIC;
          COMMON_0_OUT	:	OUT	STD_LOGIC;
          COMMON_1_OUT	:	OUT	STD_LOGIC;
          COMMON_2_OUT	:	OUT	STD_LOGIC;
          A_OUT	:	OUT	STD_LOGIC;
          B_OUT	:	OUT	STD_LOGIC;
          C_OUT	:	OUT	STD_LOGIC;
          D_OUT	:	OUT	STD_LOGIC;
          E_OUT	:	OUT	STD_LOGIC;
          F_OUT	:	OUT	STD_LOGIC;
          G_OUT	:	OUT	STD_LOGIC;
          DP_OUT	:	OUT	STD_LOGIC;
          CLOCK	:	IN	STD_LOGIC;
          DATA_IN	:	IN	STD_LOGIC_VECTOR (7 DOWNTO 0);
          OVER_FLOW	:	OUT	STD_LOGIC);
   END COMPONENT;

          signal op1 : STD_LOGIC_VECTOR(7 DOWNTO 0);
          signal op2 : STD_LOGIC_VECTOR(7 DOWNTO 0);
   SIGNAL RESE	:	STD_LOGIC;
   SIGNAL ENTER_OP1	:	STD_LOGIC;
   SIGNAL ENTER_OP2	:	STD_LOGIC;
   SIGNAL CALCULATE	:	STD_LOGIC;
   SIGNAL COMMON_0_OUT	:	STD_LOGIC;
   SIGNAL COMMON_1_OUT	:	STD_LOGIC;
   SIGNAL COMMON_2_OUT	:	STD_LOGIC;
   SIGNAL A_OUT	:	STD_LOGIC;
   SIGNAL B_OUT	:	STD_LOGIC;
   SIGNAL C_OUT	:	STD_LOGIC;
   SIGNAL D_OUT	:	STD_LOGIC;
   SIGNAL E_OUT	:	STD_LOGIC;
   SIGNAL F_OUT	:	STD_LOGIC;
   SIGNAL G_OUT	:	STD_LOGIC;
   SIGNAL DP_OUT	:	STD_LOGIC;
   SIGNAL CLOCK	:	STD_LOGIC;
   SIGNAL DATA_IN	:	STD_LOGIC_VECTOR (7 DOWNTO 0);
   SIGNAL OVER_FLOW	:	STD_LOGIC;
```

```
        constant CLK_period: time := 1 us;
         constant TC_period: time := 65536 us;

BEGIN

  UUT: TopLevel PORT MAP(
                RESE => RESE,
                ENTER_OP1 => ENTER_OP1,
                ENTER_OP2 => ENTER_OP2,
                CALCULATE => CALCULATE,
                COMMON_0_OUT => COMMON_0_OUT,
                COMMON_1_OUT => COMMON_1_OUT,
                COMMON_2_OUT => COMMON_2_OUT,
                A_OUT => A_OUT,
                B_OUT => B_OUT,
                C_OUT => C_OUT,
                D_OUT => D_OUT,
                E_OUT => E_OUT,
                F_OUT => F_OUT,
                G_OUT => G_OUT,
                DP_OUT => DP_OUT,
                CLOCK => CLOCK,
                DATA_IN => DATA_IN,
                OVER_FLOW => OVER_FLOW
  );

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
        RESE <= '1';
        ENTER_OP1 <= '0';
        ENTER_OP2 <= '0';
  CALCULATE <= '0';
        DATA_IN   <=(others => '0');

        wait for 2*CLK_period;
        RESE <='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 8*TC_period;
         wait;
        end process stim_proc; --1.835 s
END;
```

Перевірка результату

OP1=00000001;

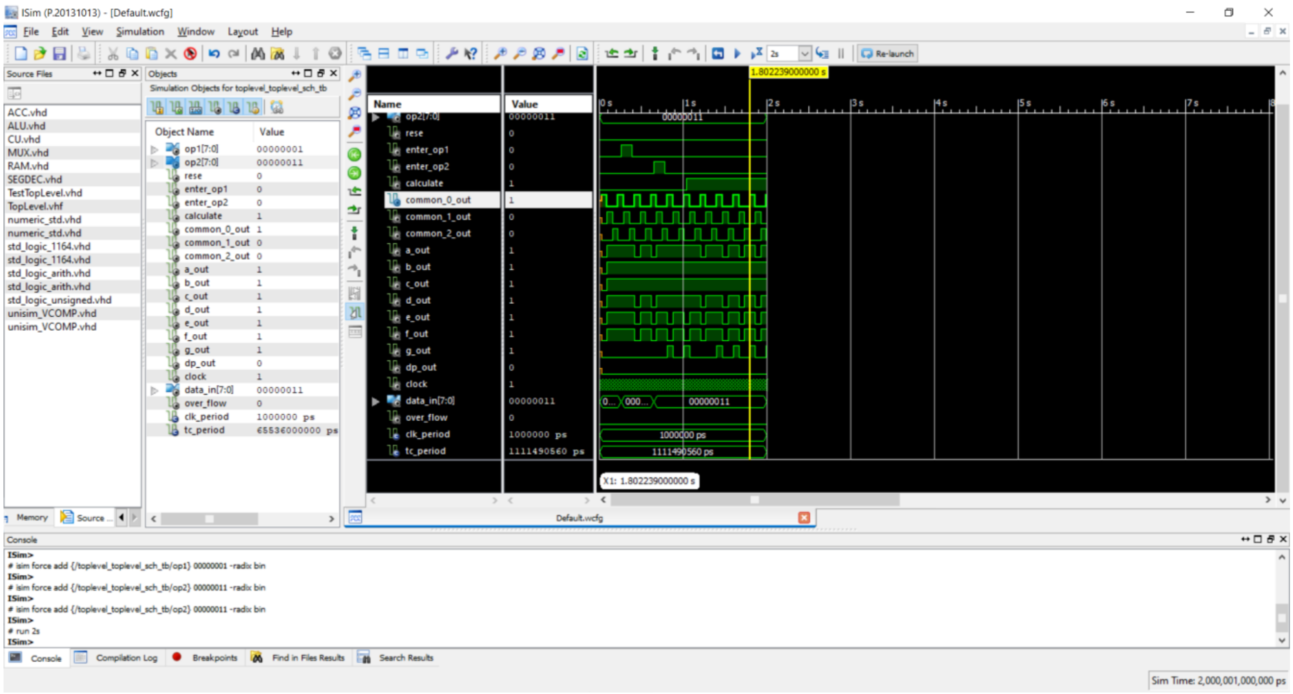
OP2=00000011;

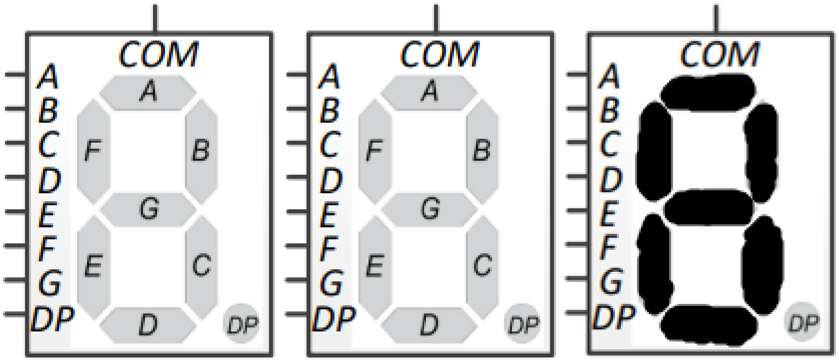
((OP1 + 2) * OP2) << OP1 = 00010010;

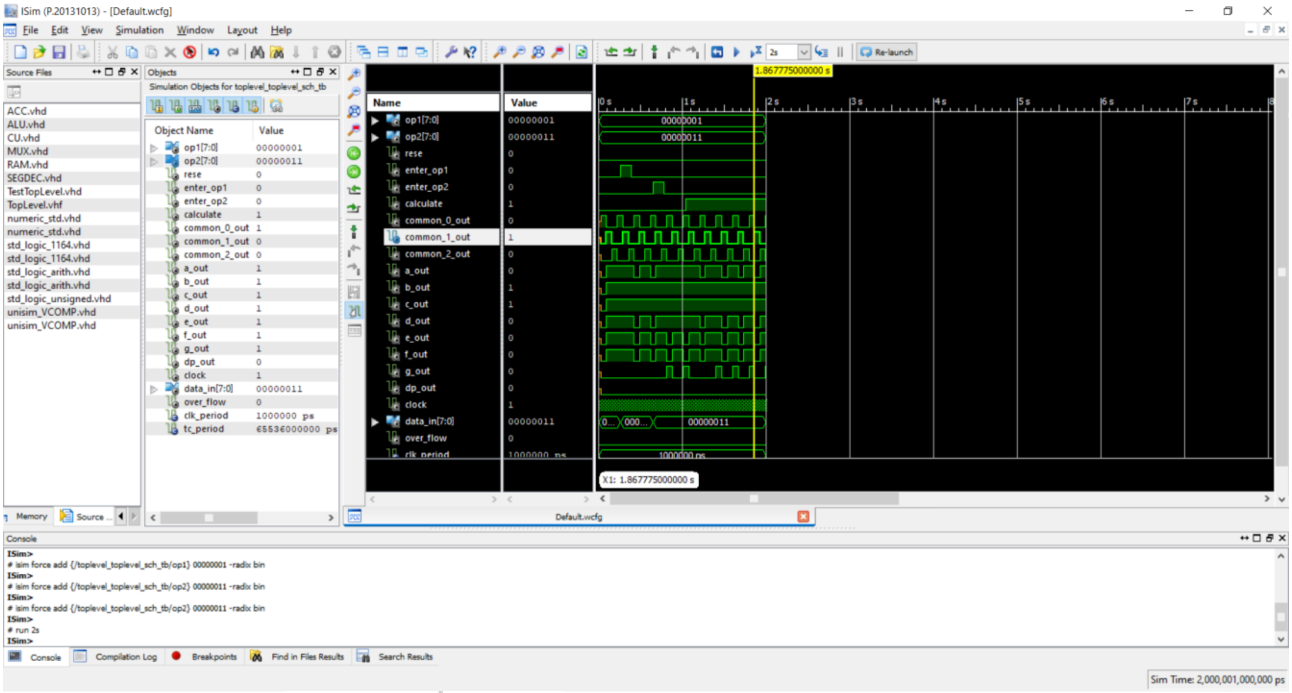
1) OP1 + 2 = 00000001 + 00000010 = 00000011;

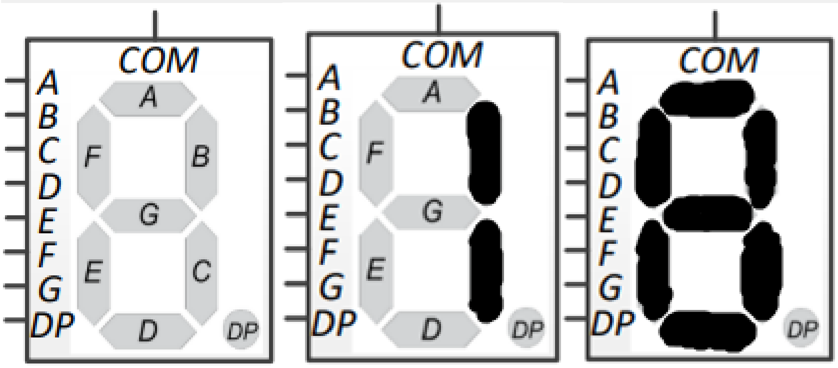
2) (OP1 + 2) * OP2 = 00000011 * 00000011 = 00001001;

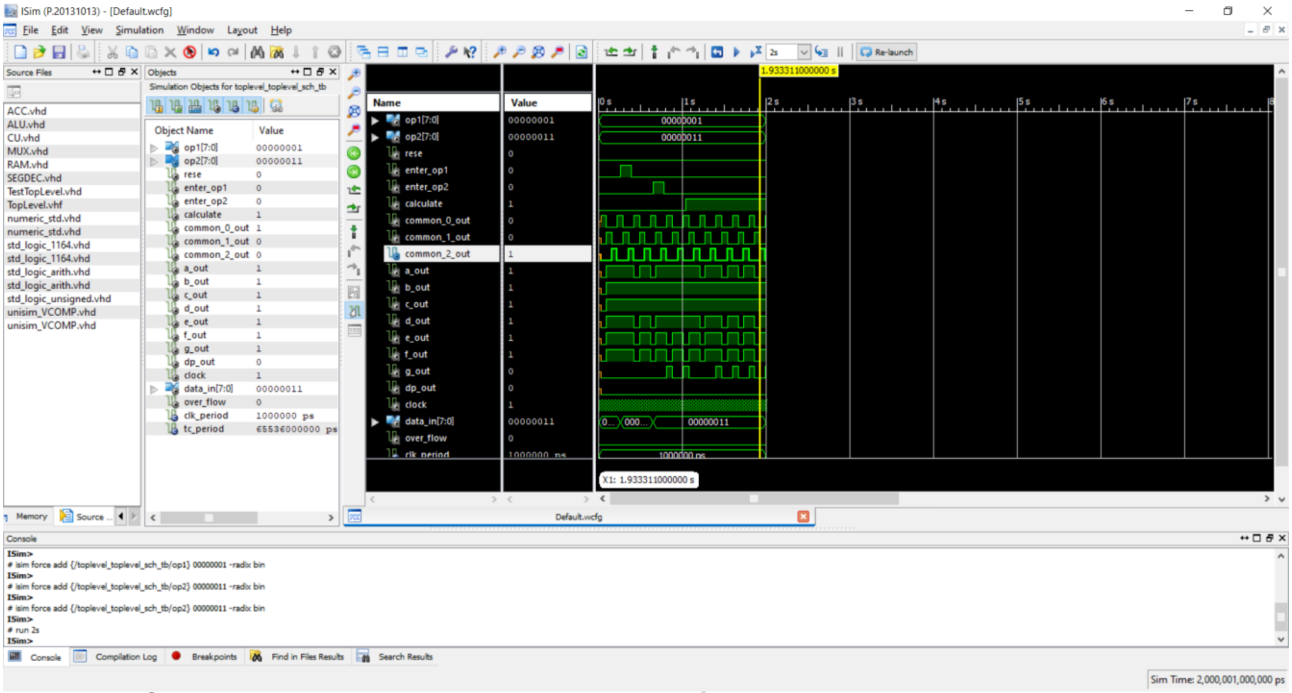
3) ((OP1 + 2) * OP2) << OP1 = 00001001 << 00000001 = 00010010;

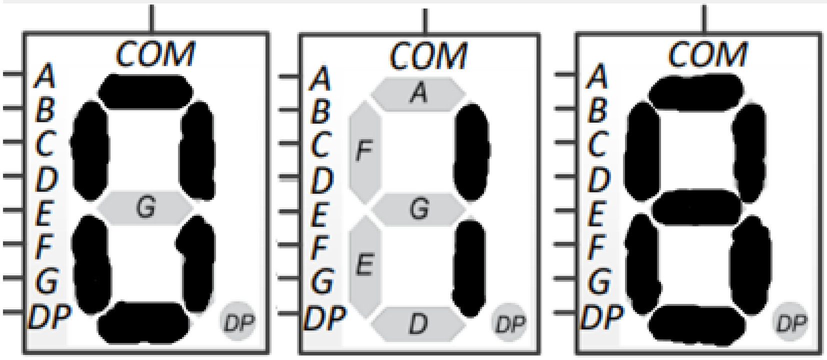



COM
A B C D E F G DP


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Source Files
Objects
Name
Value
Default.wcfg
Console
Sim Time: 2,000,001,000,000 ps


COM
A B C D E F G DP





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