**МІНІСТЕРСТО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**Кафедра ЕОМ**

**Лабораторної роботи №3**

**з дисципліни**

**«Моделювання комп’ютерних систем»**

**Варіант 9**

**Виконав:**

Студент гр КІ-201

Гординяк В. Р.

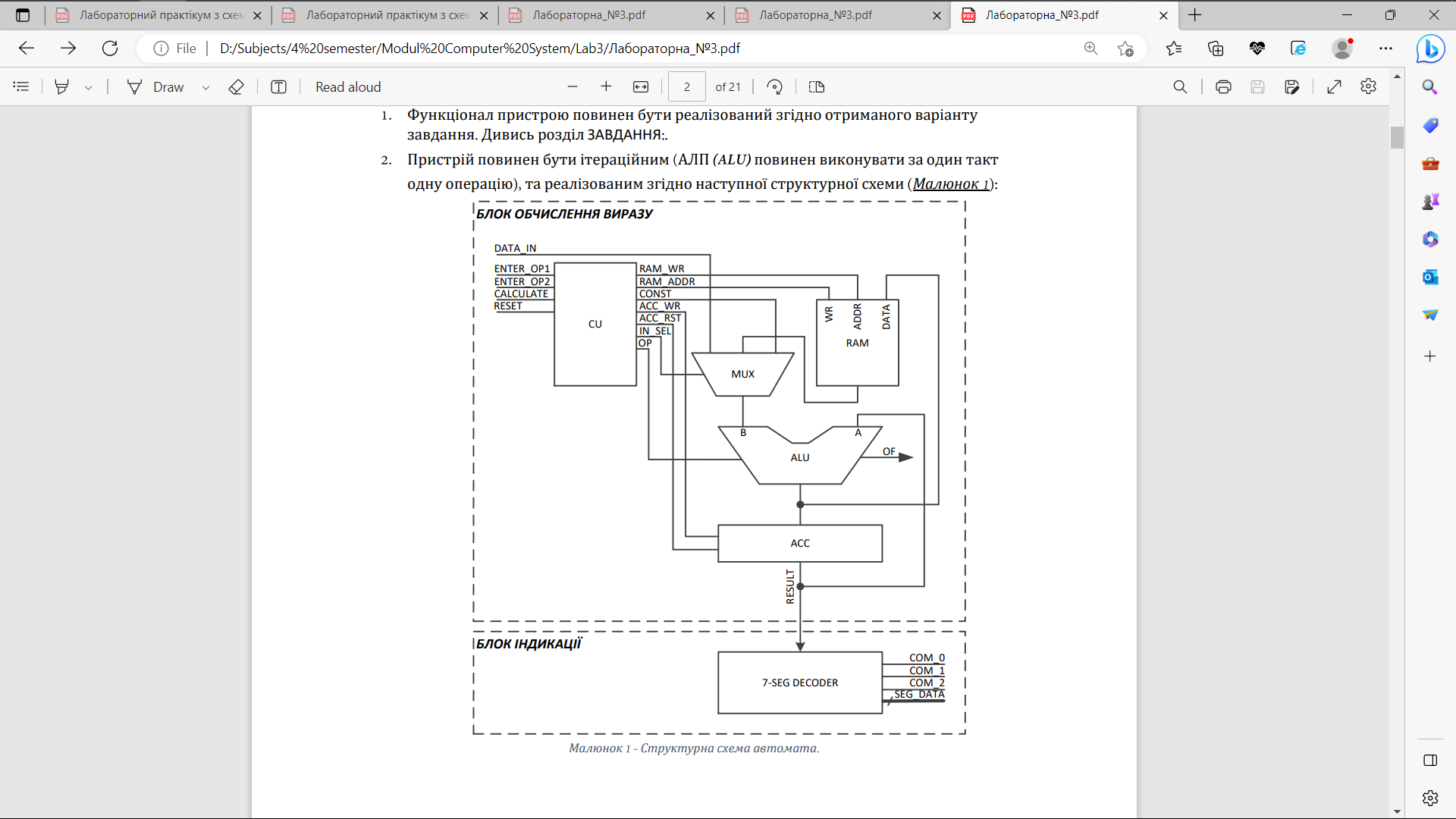
**Перевірив:**

Козак Н. Б.

Львів – 2023

**Тема роботи**. Поведінковий опис цифрового автомата. Перевірка роботи автомата за допомогою **Elbert V2 – Spartan 3A FPGA**.

**Мета роботи**. На базі стенда реалізувати цифровий автомат для обчислення значення виразу дотримуючись наступних вимог.

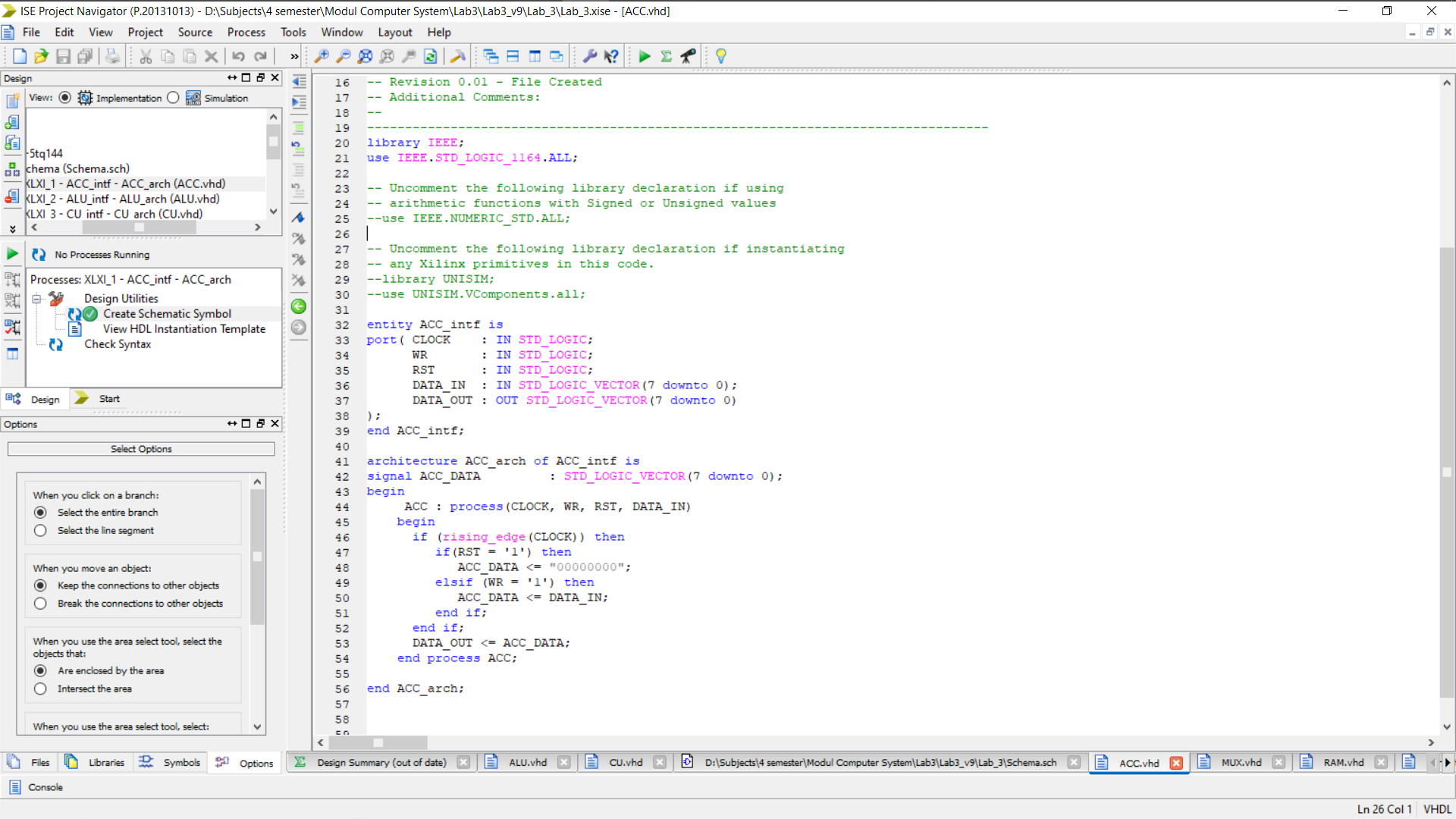


**Завдання:**





**Код на VHDL, який описує роботу ACC у файлі ACC.vhdl:**



**Код на VHDL, який описує роботу ALU у файлі ALU.vhdl:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity ALU\_intf is

port (

OP\_CODE\_BUS : in std\_logic\_vector(1 downto 0);

INPUT\_A : in std\_logic\_vector(7 downto 0);

INPUT\_B : in std\_logic\_vector(7 downto 0);

RESULT : out std\_logic\_vector(7 downto 0);

OVERFLOW : out std\_logic

);

end ALU\_intf;

architecture ALU\_arch of ALU\_intf is

signal RES\_ADD : unsigned(8 downto 0);

begin

ALU : process(OP\_CODE\_BUS, INPUT\_A, INPUT\_B)

variable A : unsigned(7 downto 0);

variable B : unsigned(7 downto 0);

begin

A := unsigned(INPUT\_A);

B := unsigned(INPUT\_B);

case OP\_CODE\_BUS is

when "00" =>

RESULT <= INPUT\_B;

OVERFLOW <= '0';

when "01" =>

RESULT <= std\_logic\_vector(A - B);

if A < B then

OVERFLOW <= '1';

else

OVERFLOW <= '0';

end if;

when "10" =>

RES\_ADD <= unsigned('0' & A) + unsigned('0' & B);

RESULT <= std\_logic\_vector(A + B);

if RES\_ADD(8) = '1' then

OVERFLOW <= '1';

else

OVERFLOW <= '0';

end if;

when "11" =>

RESULT <= std\_logic\_vector(A or B);

OVERFLOW <= '0';

when others =>

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

OVERFLOW <= '0';

end case;

end process ALU;

end ALU\_arch;

**Код на VHDL, який описує роботу CU у файлі CU.vhdl:**

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 02:54:27 06/01/2023

-- Design Name:

-- Module Name: CU\_intf - CU\_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;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity 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);

CONST : out STD\_LOGIC\_VECTOR(7 downto 0);

ACC\_WR : out std\_logic;

ACC\_RST : out std\_logic;

IN\_SEL : out std\_logic\_vector(1 downto 0);

OP\_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\_run\_calc4, cu\_finish);

signal cu\_cur\_state : cu\_state\_type;

signal cu\_next\_state : cu\_state\_type;

begin

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\_run\_calc4;

when cu\_run\_calc4 =>

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";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '1';

ACC\_WR <= '0';

when cu\_idle =>

IN\_SEL <= "00";

OP\_CODE\_BUS <= "00";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '0';

when cu\_load\_op1 =>

IN\_SEL <= "00";

OP\_CODE\_BUS <= "00";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '1';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_load\_op2 =>

IN\_SEL <= "00";

OP\_CODE\_BUS <= "00";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "01";

RAM\_WR <= '1';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc0 =>

IN\_SEL <= "01";

OP\_CODE\_BUS <= "00";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "01";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc1 =>

IN\_SEL <= "10";

OP\_CODE\_BUS <= "01";

CONST <= "00000100";

RAM\_ADDR\_BUS <= "01";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc2 =>

IN\_SEL <= "01";

OP\_CODE\_BUS <= "10";

CONST <= "00000000";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc3 =>

IN\_SEL <= "10";

OP\_CODE\_BUS <= "10";

CONST <= "00001010";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc4 =>

IN\_SEL <= "10";

OP\_CODE\_BUS <= "11";

CONST <= "00000010";

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;

**Код на VHDL, який описує роботу MUX у файлі MUX.vhdl:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity MUX\_intf is

port (

S : in std\_logic\_vector(1 downto 0);

X0, X1, CONST: in std\_logic\_vector(7 downto 0);

O : out std\_logic\_vector(7 downto 0));

end MUX\_intf;

architecture MUX\_arch of MUX\_intf is

begin

INSEL\_A\_MUX : process(X0, X1, CONST, S)

begin

if( S = "00") then

O <= X0 ;

elsif(S = "01") then

O <= X1;

else

O <= CONST;

end if;

end process INSEL\_A\_MUX;

end MUX\_arch;

**Код на VHDL, який описує роботу** RAM **у файлі** RAM**.vhdl:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity RAM\_intf is

port(CLOCK : IN STD\_LOGIC;

WR : in std\_logic;

ADDR : IN STD\_LOGIC\_VECTOR(1 downto 0);

DATA\_IN : IN STD\_LOGIC\_VECTOR(7 downto 0);

DATA\_OUT : 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, ADDR, RAM\_UNIT, WR , DATA\_IN)

begin

if (rising\_edge(CLOCK)) then

if (WR = '1') then

RAM\_UNIT(conv\_integer(ADDR)) <= DATA\_IN;

end if;

end if;

DATA\_OUT<= RAM\_UNIT(conv\_integer(ADDR));

end process RAM;

end RAM\_arch;

**Код на VHDL, який описує роботу RES\_DECODER у файлі RES\_DECODER.vhdl:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

entity RES\_DECODER\_intf is

port (

CLOCK : in STD\_LOGIC;

INPUT\_NUM : in STD\_LOGIC\_VECTOR(7 downto 0);

RESET : 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

);

end RES\_DECODER\_intf;

architecture RES\_DECODER\_arch of RES\_DECODER\_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 (INPUT\_NUM)

variable hex\_src : STD\_LOGIC\_VECTOR(7 downto 0) ;

variable bcd : STD\_LOGIC\_VECTOR(11 downto 0) ;

begin

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

hex\_src := INPUT\_NUM;

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);

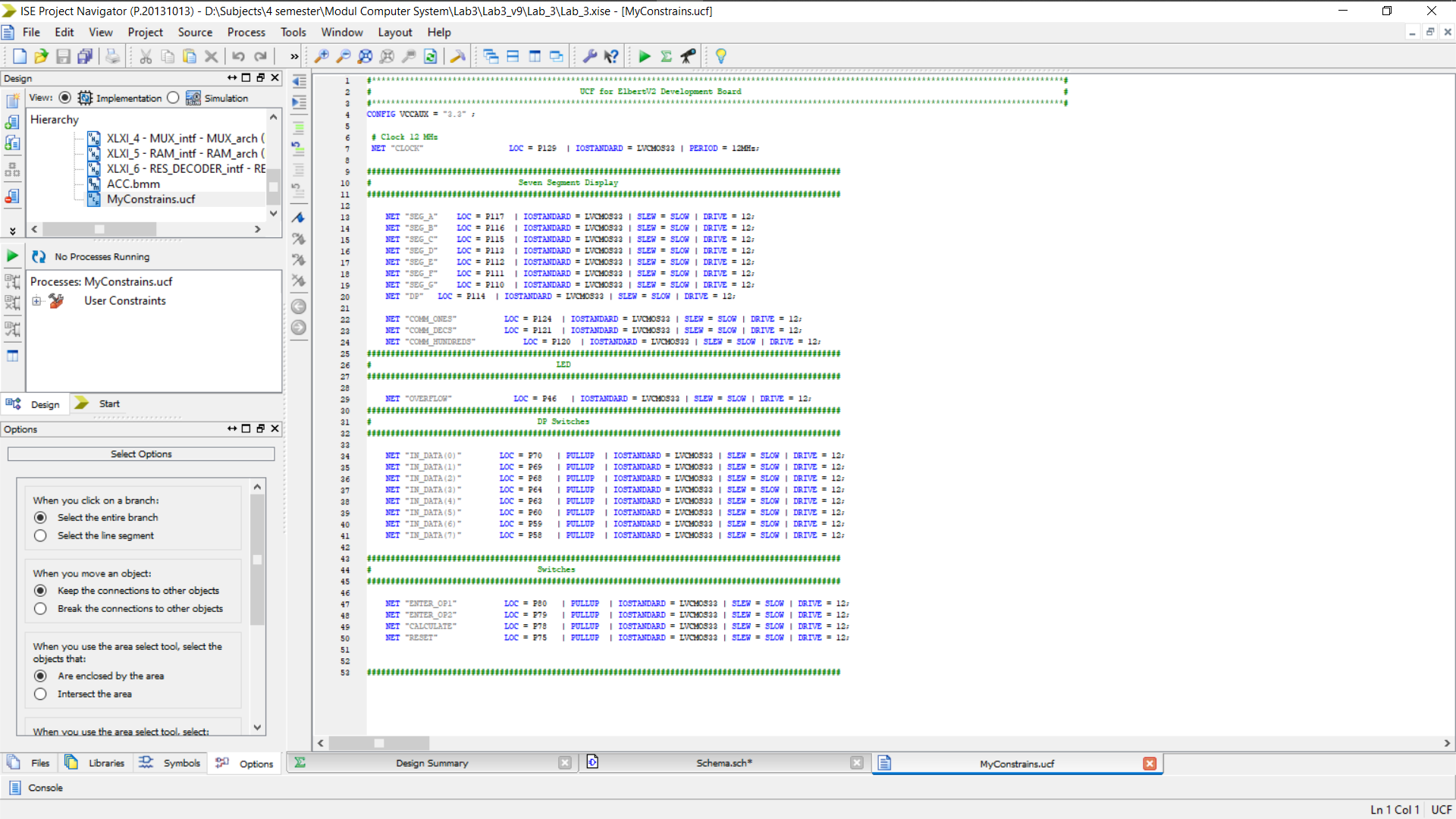
DP <= '0';

end if;

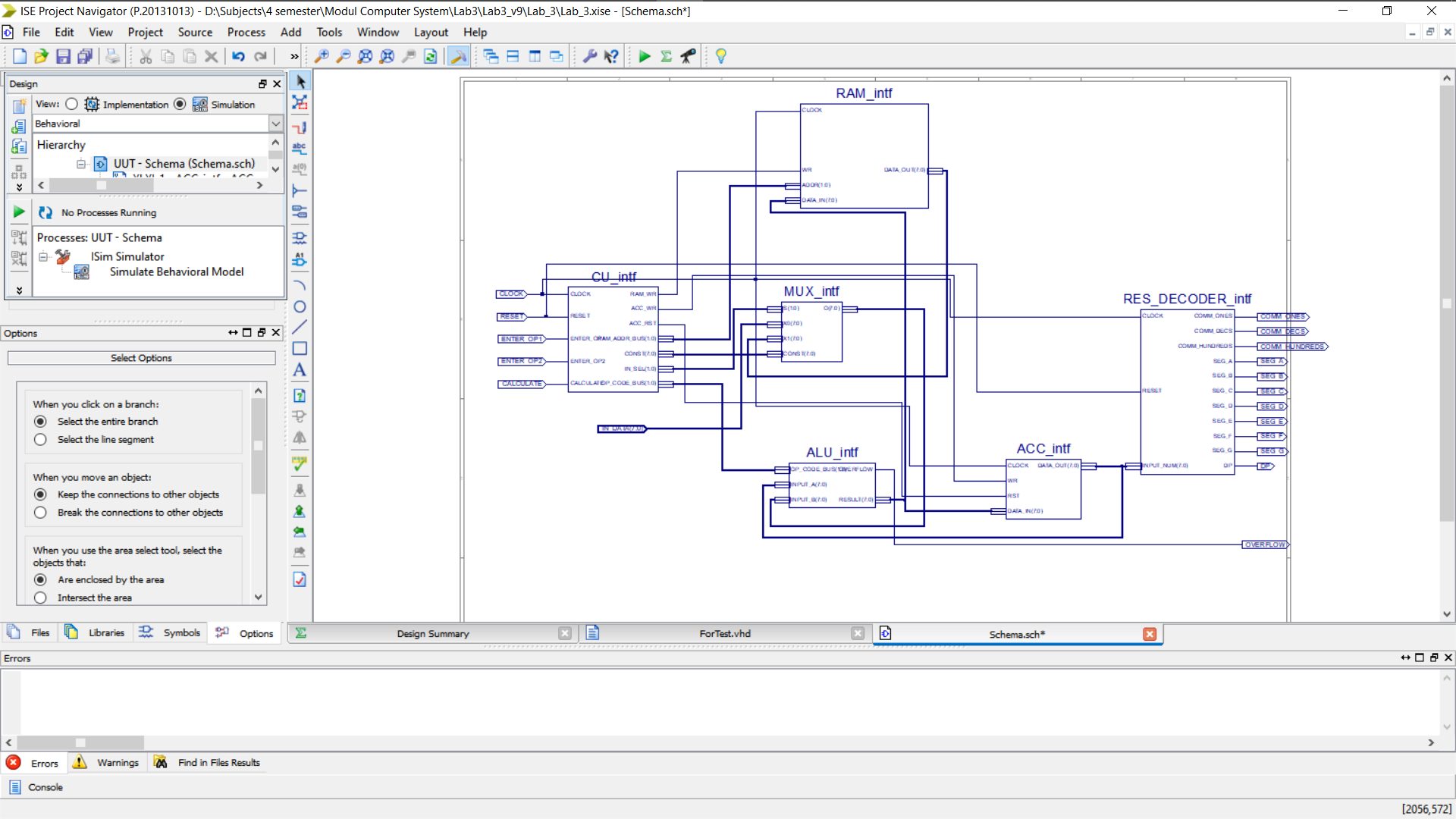
end process INDICATE;

end RES\_DECODER\_arch;

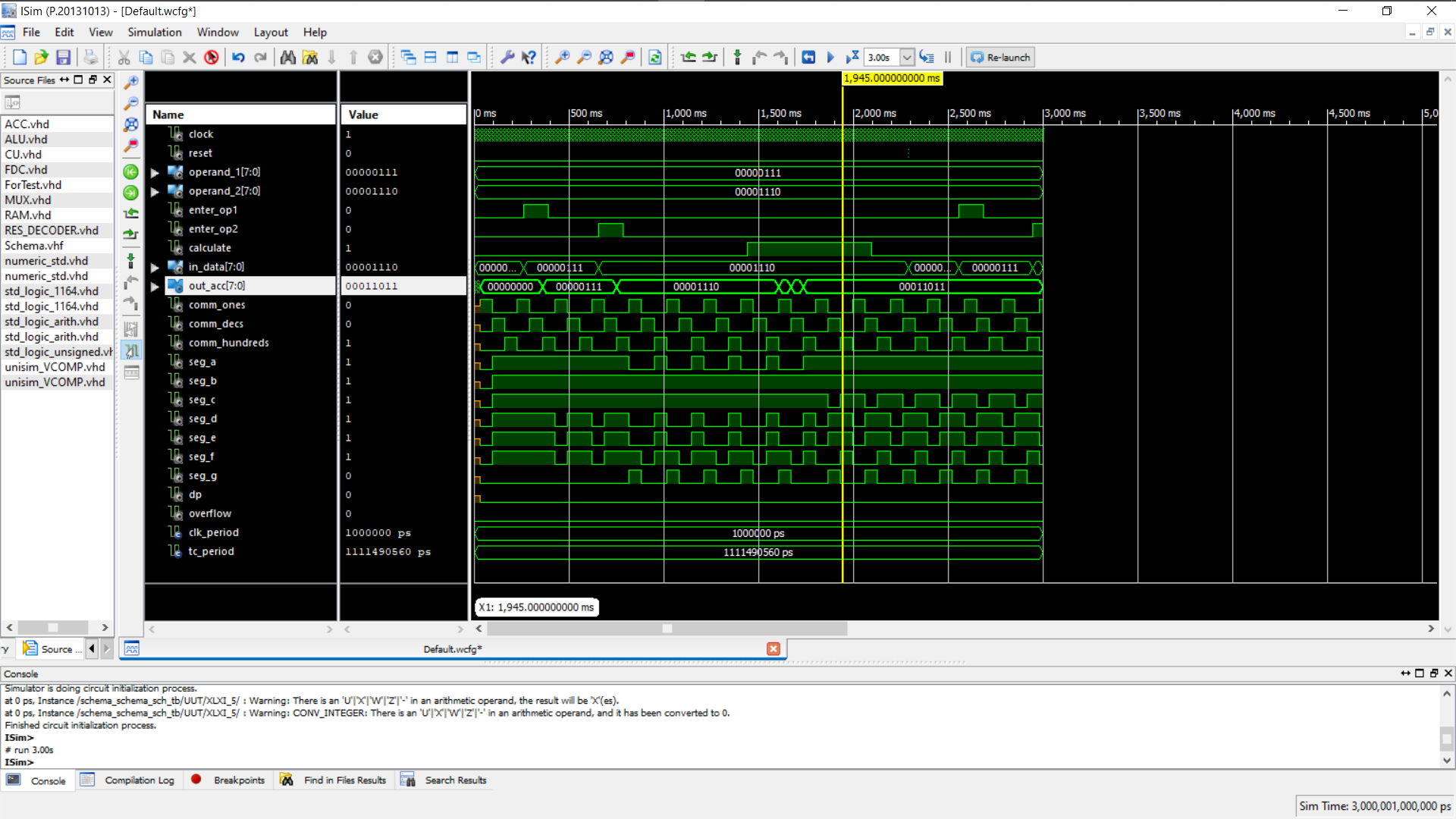
**Constrain для моделювання на стенді:**



**Схема пристрою:**



**Результат перевірки роботи схеми за допомогою симулятора ISim**:



**Висновок:** Я на базі стенда реалізував ітераційний цифровий автомат для обчислення значення виразу згідно варіанту завдання та структурної схеми автомату.