Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №3

з дисципліни: “Моделювання комп’ютерних систем”

на тему: “ Структурний опис цифрового автомата Перевірка роботи автомата за допомогою стенда ***Elbert V2 – Spartan 3A FPGA.***”

Варіант 21

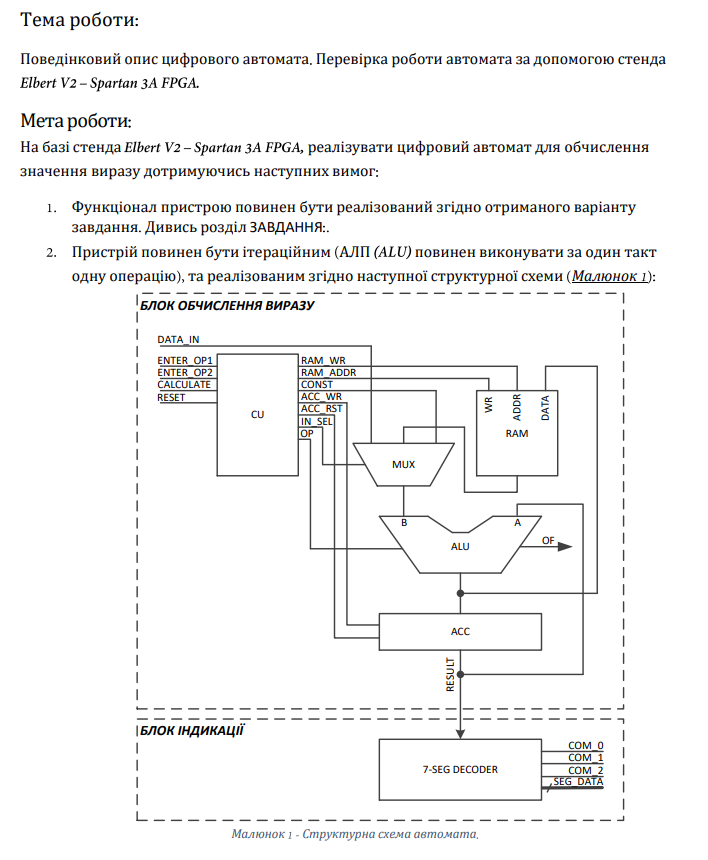
Виконав: ст. гр. КІ-22

Проць Ю.В.

Прийняв:

Козак Н.Б.

Львів – 2021



**Завдання:**



**Хід роботи:**

1. **VHDL код MUX.vhd.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity my\_MuX\_intf is

port(

DATA\_IN : in std\_logic\_vector(7 downto 0);

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

CONSTANT\_BUS1 : in std\_logic\_vector(7 downto 0);

CONSTANT\_BUS2 : 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 my\_MuX\_intf;

architecture my\_MuX\_arch of my\_MuX\_intf is

begin

INSEL\_A\_MUX : process(DATA\_IN, CONSTANT\_BUS1, CONSTANT\_BUS2, 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;

elsif(IN\_SEL = "10") then

IN\_SEL\_OUT\_BUS <= CONSTANT\_BUS1;

else

IN\_SEL\_OUT\_BUS <= CONSTANT\_BUS2;

end if;

end process INSEL\_A\_MUX;

end my\_MuX\_arch;

1. **VHDL код ACC.vhd.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity my\_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 my\_ACC\_intf;

architecture my\_ACC\_arch of my\_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 my\_ACC\_arch;

1. **VHDL код ALU.vhd.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity my\_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 my\_ALU\_intf;

architecture my\_ALU\_arch of my\_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);

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" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(A - B);

when "11" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(A or B);

when others => ACC\_DATA\_IN\_BUS <= "00000000";

end case;

end process ALU;

end my\_ALU\_arch;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity my\_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\_BUS1 : out std\_logic\_vector(7 downto 0);

CONSTANT\_BUS2 : 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 my\_CU\_intf;

architecture my\_CU\_arch of my\_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\_BUS1 <= "00000100";

CONSTANT\_BUS2 <= "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;

1. **VHDL код CU.vhd.**

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 <= "01";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc1 =>

IN\_SEL <= "10";

OP\_CODE\_BUS <= "10";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc2 =>

IN\_SEL <= "01";

OP\_CODE\_BUS <= "01";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '1';

when cu\_run\_calc3 =>

IN\_SEL <= "11";

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 my\_CU\_arch;

1. **VHDL код RAM.vhd.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity my\_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 my\_RAM\_intf;

architecture my\_RAM\_arch of my\_RAM\_intf is

type ram\_type is array (3 downto 0) of STD\_LOGIC\_VECTOR(7 downto 0);

signal RAM\_UNIT : ram\_type;

begin

--when reset will init const

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 my\_RAM\_arch;

1. **VHDL код OUT\_PUT\_DECODER.vhd.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity OUT\_PUT\_DECODER\_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 OUT\_PUT\_DECODER\_intf;

architecture OUT\_PUT\_DECODER\_arch of OUT\_PUT\_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 (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);

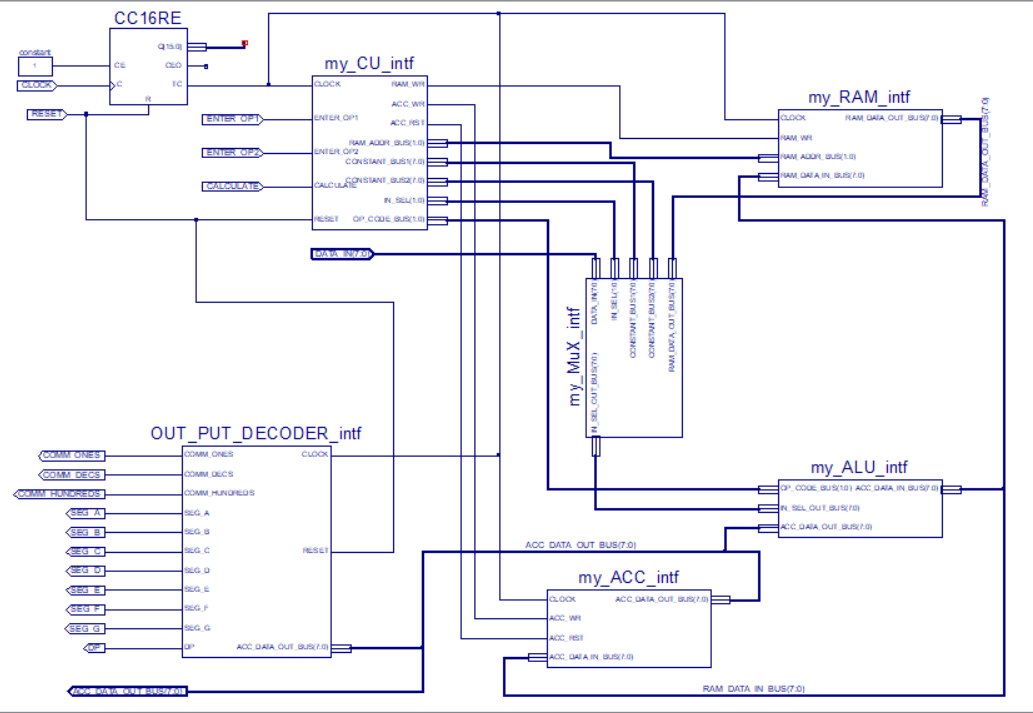
DP <= '0';

end if;

end process INDICATE;

end OUT\_PUT\_DECODER\_arch;

1. **Схема Top\_level.sch.**



1. **Test bench TB\_TOPLEVEL.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 Top\_level\_Top\_level\_sch\_tb IS

END Top\_level\_Top\_level\_sch\_tb;

ARCHITECTURE behavioral OF Top\_level\_Top\_level\_sch\_tb IS

COMPONENT Top\_level

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):="00000001";

signal op2 : STD\_LOGIC\_VECTOR(7 DOWNTO 0):="00000100";

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 us;

constant TC\_period: time := 65536 us;

BEGIN

UUT: Top\_level

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

RESET <= '1';

ENTER\_OP1 <= '0';

ENTER\_OP2 <= '0';

CALCULATE <= '0';

DATA\_IN <=(others => '0');

wait for 2\*CLK\_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 8\*TC\_period;

wait;

end process stim\_proc; --1.835 s

END ;

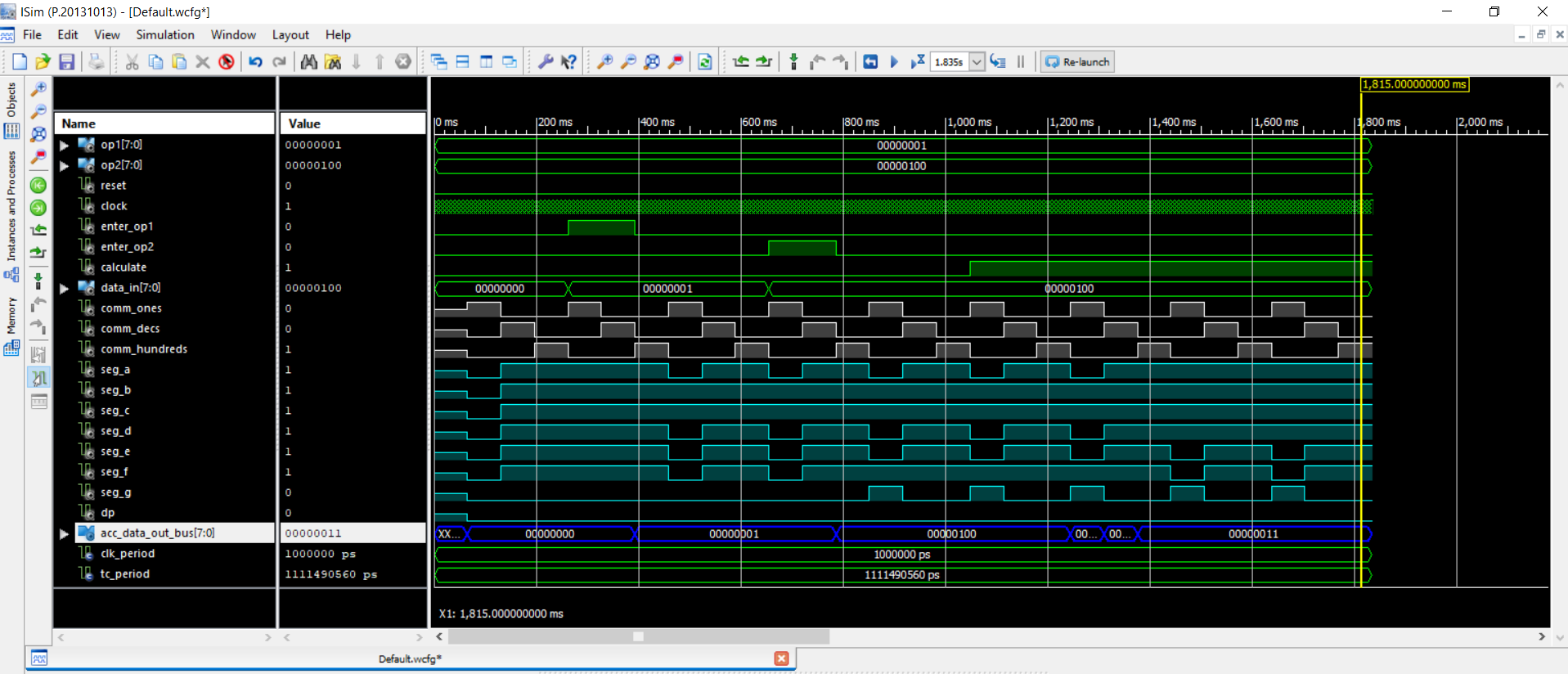


Рис.1. Часова діаграма.

**Висновок:** на цій лабораторній роботі я реалізував цифровий автомат для обчислення значення виразу.