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** 3**A FPGA.»**

**Варіант № 2**

**Виконав:**

ст.гр. КІ-202

Бажулін С.В.

Перевірив:

Козак Н.Б

**Львів 2023**

**Мета роботи:** на базі стенда **Elbert V2 – Spartan** 3**A FPGA** реалізувати цифровий автомат для обчислення значення виразу згідно вимог.

**Завдання згідно з варіантом:**

****

**Виконання роботи:**

Код MUX:

library IEEE;

use IEEE.STD\_LOGIC\_1164.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;

Код ACC:

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;

Код ALU:

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);

RESET : IN STD\_LOGIC;

ACC\_DATA\_IN\_BUS : OUT STD\_LOGIC\_VECTOR(7 downto 0);

OVERFLOW : OUT STD\_LOGIC := '0'

--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);

begin

A := unsigned(ACC\_DATA\_OUT\_BUS);

B := unsigned(IN\_SEL\_OUT\_BUS);

if(RESET = '1')then

OVERFLOW <= '0';

end if;

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);

if (A > "11111101") then

OVERFLOW <= '1';

end if;

when "10" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(RESIZE(unsigned(A\*B(7 downto 0)), 8));

when "11" =>

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;

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

end case;

end process ALU;

end ALU\_arch;

Код CU:

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

-- 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 <= "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;

end CU\_arch;

Код RAM:

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

--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 RAM\_arch;

Код SEVEN\_SEG\_DECODER:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity SEVEN\_SEG\_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 SEVEN\_SEG\_DECODER\_intf;

architecture SEVEN\_SEG\_DECODER\_arch of SEVEN\_SEG\_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';

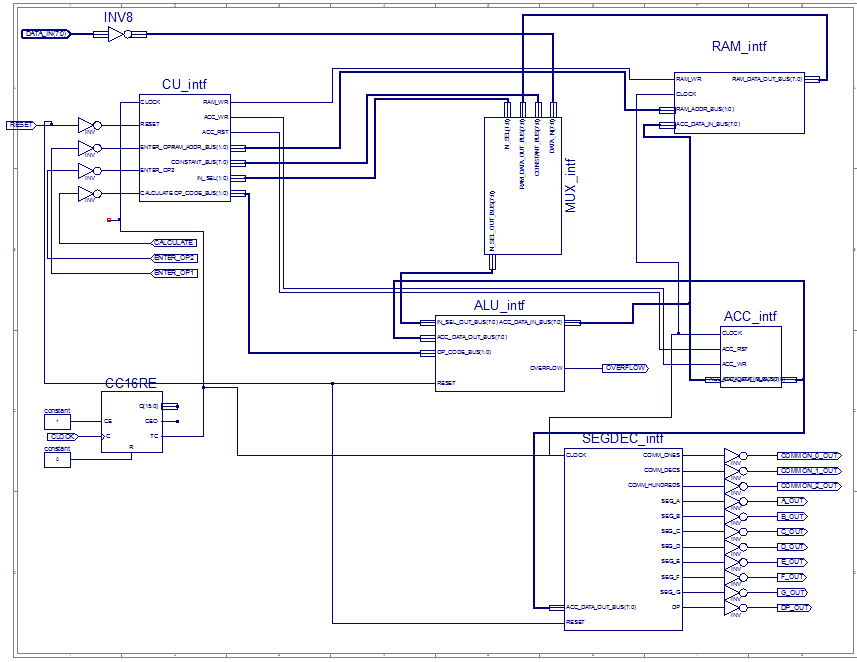
end if;

end process INDICATE;

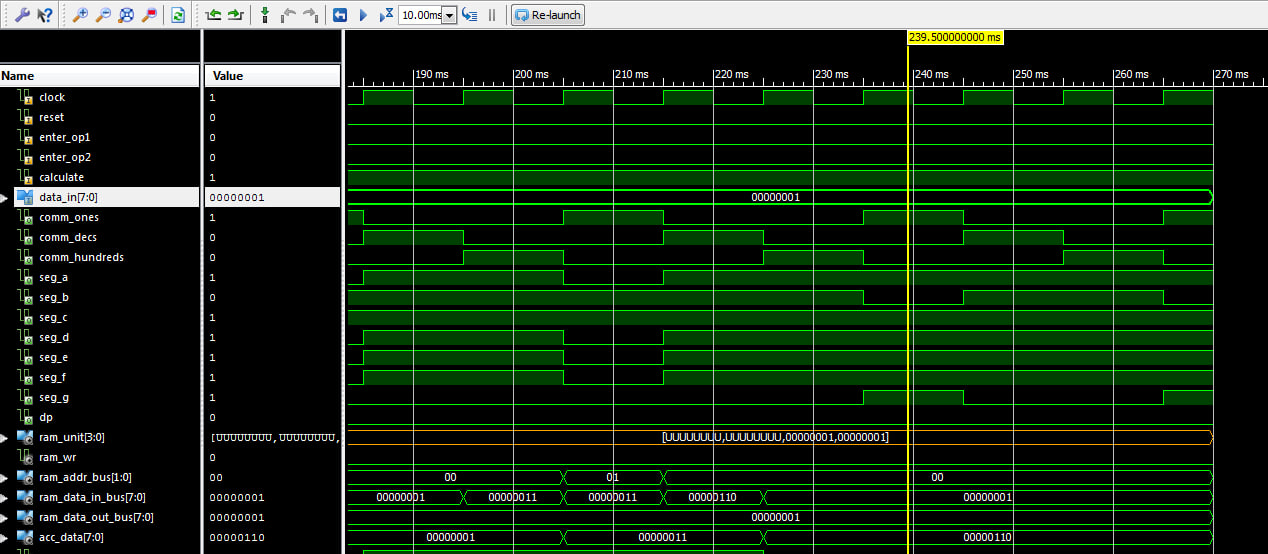
end SEVEN\_SEG\_DECODER\_arch;

Згенерував Schematic файли для реалізованих елементів.

Створив Schematic файл TopLevel, виконав в ньому інтеграцію компонентів системи між собою. Перевірив роботу системи за допомогою симулятора ISim.



*Рис. 7. Схема TopLevel*

*Рис. 8. Діаграма проведеної симуляції для TopLevel*

Створив Constraints файл, зв’язав в ньому виводи схеми та фізичні виводи плати.

Вміст Constraints.ucf:

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

# UCF for ElbertV2 Development Board #

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

CONFIG VCCAUX = "3.3" ;

# Clock 12 MHz

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

####################################################################################################

# Seven Segment Display

####################################################################################################

NET "SEG\_A" LOC = P117 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_B" LOC = P116 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_C" LOC = P115 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_D" LOC = P113 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_E" LOC = P112 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_F" LOC = P111 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SEG\_G" LOC = P110 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DP" LOC = P114 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "COMM\_HUNDREDS" LOC = P124 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "COMM\_DECS" LOC = P121 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "COMM\_ONES" LOC = P120 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

####################################################################################################

# DP Switches

####################################################################################################

NET "DATA(0)" LOC = P70 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(1)" LOC = P69 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(2)" LOC = P68 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(3)" LOC = P64 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(4)" LOC = P63 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(5)" LOC = P60 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(6)" LOC = P59 | PULLUP | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "DATA(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;

####################################################################################################

# LED

####################################################################################################

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

Згенерував бінарний файл для розробленого цифрового автомата.

**Висновок:** виконавши лабораторну роботу, здобуто навики реалізації цифрових автоматів для обчислення значення заданого виразу.