Міністерство освіти і науки України Національний університет „Львівська політехніка”

Кафедра ЕОМ



Звіт

з лабораторної роботи №3

з дисципліни: “Моделювання комп’ютерних систем” Варіант 10

Виконав: ст.гр. КІ-203 Замкова

Прийняв: Козак Н.Б.

Львів-2023

Тема роботи:

Поведінковий опис цифрового автомата Перевірка роботи автомата за допомогою стенда.

Мета роботи:

На базі стенда реалізувати цифровий автомат для обчислення значення виразу.

Завдання 9

|  |  |
| --- | --- |
| Варіант | Формула |
| 9 | ((OP2 - 4) + OP1) or 2 |

Методичні вказівки

В найрозповсюдженішому варіанті семи сегментний індикатор являє собою модуль який містить окремих елементів відображення:

* Сегментів розміщених у вигляді цифри включення яких у різних комбінаціях дозволяє відображати різні цифри Сегменти зазвичай позначаються латинськими літерами A,B,C,D,E,F,G.
* Десяткової крапки розміщеної праворуч від цифри внизу Крапка призначена для відображення дробових чисел.

Є основних підходи до використання сегментних індикаторів

* Метод статичної індикації .
* Метод динамічної індикації.

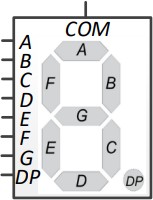
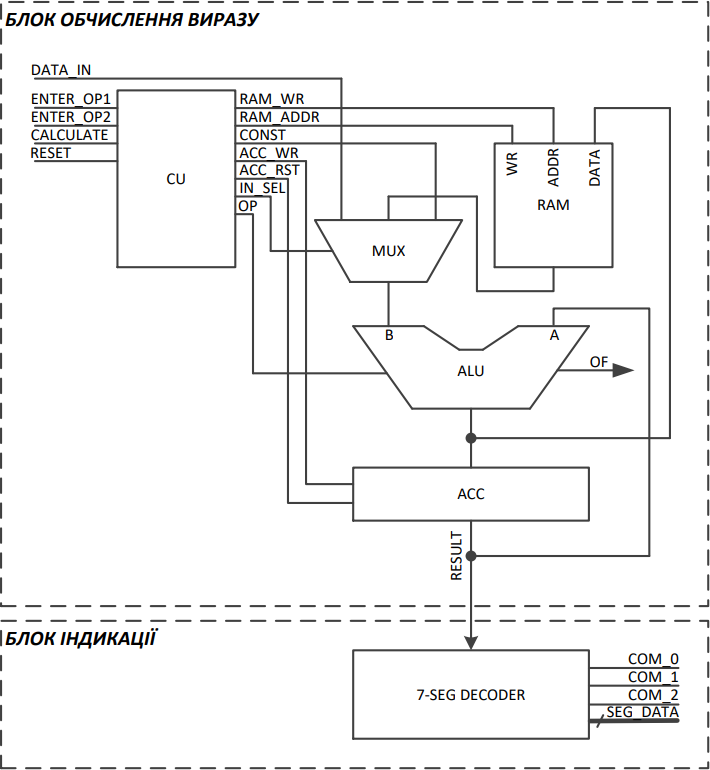


Рис.1 Модуль семи сегментного індикатора.



Код програми Файл CU.vhd

Рис. 2 Структурна схема автомата.

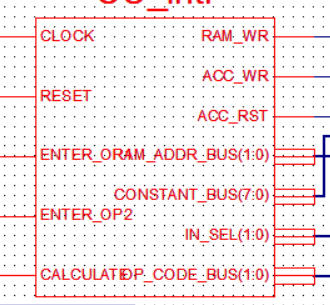


Рис. 3 Згенерований символ з файлу CU.vhd

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 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;

end CU\_intf;

RAM\_WR : OUT STD\_LOGIC;

RAM\_ADDR\_BUS : OUT STD\_LOGIC\_VECTOR(1 downto 0); CONSTANT\_BUS : OUT STD\_LOGIC\_VECTOR(7 downto 0):= "00000100"; 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)

);

architecture CU\_arch of CU\_intf is

-- cu\_state\_type перелік станів контролера

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 <= "00000100";

CU\_SYNC\_PROC: process (CLOCK) begin

if (rising\_edge(CLOCK)) then if (RESET = '1') then

cu\_cur\_state <= cu\_rst; --поточний стан схеми обнуляється. else

cu\_cur\_state <= cu\_next\_state;--приймає значення наступного стану. end if;

end if; end process;

begin

CUNEXT\_STATE\_DECODE: process (cu\_cur\_state, ENTER\_OP1, ENTER\_OP2, CALCULATE)

--оголосити стан за замовчуванням для next\_state, щоб уникнути блокувань

cu\_next\_state <= cu\_cur\_state; --default is to stay in current state

--за умовчанням залишатися в поточному стані --insert оператори для декодування наступного стану 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

end if;

cu\_next\_state <= cu\_idle;

end case;

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 process;

CU\_OUTPUT\_DECODE: process (cu\_cur\_state) begin

case(cu\_cur\_state) is

when cu\_rst => --стан скидання IN\_SEL <= "00";

OP\_CODE\_BUS <= "00";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '1';

ACC\_WR <= '0';

when cu\_idle =>--стан очікування на операції IN\_SEL <= "00";

OP\_CODE\_BUS <= "00";

RAM\_ADDR\_BUS <= "00";

RAM\_WR <= '0';

ACC\_RST <= '0';

ACC\_WR <= '0';

when cu\_load\_op1 => --стан завантаження 1 операнду

IN\_SEL <= "00";--яка команда подається на мультиплексор OP\_CODE\_BUS <= "00";--яка команда подається на ALU (арифметичної логічної одиниці) RAM\_ADDR\_BUS <= "00";--адреса памяті

RAM\_WR <= '1'; --вхідні данні записуються в RAM

ACC\_RST <= '0'; --коли ресет 1 запис 0 в память

ACC\_WR <= '1'; --вхідні данні записуються в акумулятор

when cu\_load\_op2 => --стан завантаження 2 операнду

IN\_SEL <= "00";--яка команда подається на мультиплексор OP\_CODE\_BUS <= "00";--яка команда подається на ALU (арифметичної логічної одиниці) RAM\_ADDR\_BUS <= "01";--адреса памяті

RAM\_WR <= '1'; --вхідні данні записуються в RAM

ACC\_RST <= '0'; --коли ресет 1 запис 0 в память

ACC\_WR <= '1'; --вхідні данні записуються в акумулятор

when cu\_run\_calc0 => --запис в оп2 акум числа

IN\_SEL <= "01";--якщо 01 то читає з памяті сисло OP\_CODE\_BUS <= "00";--яка операція викрнується в ALU RAM\_ADDR\_BUS <= "01";--адреса памяті

RAM\_WR <= '0'; --запис в оперативку

ACC\_RST <= '0'; --ресет акум

ACC\_WR <= '1'; --запис в акум

when cu\_run\_calc1 => --операція з константою стан обчислення - етап 1 IN\_SEL <= "10";--береться константа OP\_CODE\_BUS <= "10";--операція віднімання

RAM\_ADDR\_BUS <= "00";--тут немає значення бо беретья константа RAM\_WR <= '0'; --запис в оперативку

ACC\_RST <= '0'; --ресет акум

ACC\_WR <= '1'; --запис в акум

памяті)

памяті)

when cu\_run\_calc2 =>----- стан обчислення - етап 2

IN\_SEL <= "01";--яка команда подається на мультиплексор (читає з

OP\_CODE\_BUS <= "01";--операція додавання RAM\_ADDR\_BUS <= "00";--адреса памяті

RAM\_WR <= '0'; --запис в оперативку

ACC\_RST <= '0'; --ресет акум

ACC\_WR <= '1'; --запис в акум

when cu\_run\_calc3 =>--- стан обчислення - етап 3

IN\_SEL <= "01";--яка команда подається на мультиплексор (читає з

OP\_CODE\_BUS <= "11";--операція або2 RAM\_ADDR\_BUS <= "00";--адреса памяті

RAM\_WR <= '0'; --запис в оперативку

ACC\_RST <= '0'; --ресет акум

ACC\_WR <= '1'; --запис в акум

when cu\_finish =>--- стан завершення обчислення

IN\_SEL <= "00";--яка команда подається на мультиплексор подає на Али OP\_CODE\_BUS <= "00";--повекртає тесаме значення

RAM\_ADDR\_BUS <= "00";--адреса памяті

RAM\_WR <= '0'; --запис в оперативку

ACC\_RST <= '0'; --ресет акум

ACC\_WR <= '0'; --запис в акум

when others =>

IN\_SEL <= "00";--яка команда подається на мультиплексор подає на Али OP\_CODE\_BUS <= "00";--повекртає тесаме значення

RAM\_ADDR\_BUS <= "00";--адреса памяті

RAM\_WR ACC\_RST ACC\_WR

<= '0'; --запис в оперативку

<= '0'; --ресет акум

<= '0'; --запис в акум

end case;

end process; end CU\_arch;

Файл ACC.vhd

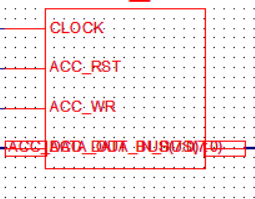


Рис. 4 Згенерований символ з файлу ACC.vhd

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 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); берігає 8-бітне значення.

begin

ACC : process(CLOCK, ACC\_DATA)

begin

if (rising\_edge(CLOCK)) then

if(ACC\_RST = '1') then

----коли ресет 1 запис 0 в память

ACC\_DATA <= "00000000";

elsif (ACC\_WR = '1') then

ACC\_DATA <= ACC\_DATA\_IN\_BUS; зберігає передане симло в ACC\_DATA

end if;

end if;

ACC\_DATA\_OUT\_BUS <= ACC\_DATA; останнє значеняя передаєть на визід

end process ACC; end ACC\_arch

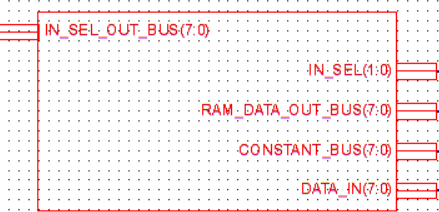
Файл MUX.vhd

Рис. 5 Згенерований символ з файлу MUX.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 MUX\_intf is

port(

DATA\_IN

: IN STD\_LOGIC\_VECTOR(7 downto 0);---вхыдны дані

CONSTANT\_BUS : IN STD\_LOGIC\_VECTOR(7 downto 0);---константа 4 RAM\_DATA\_OUT\_BUS: IN STD\_LOGIC\_VECTOR(7 downto 0);---даны з памяты

IN\_SEL : IN STD\_LOGIC\_VECTOR(1 downto 0);--запис в панять обо читання ||

константа

IN\_SEL\_OUT\_BUS : OUT std\_logic\_vector(7 downto 0)--повернення даних

);

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

--якщо 00 то подає на Али -> Ram і записує в тамять IN\_SEL\_OUT\_BUS <= DATA\_IN;

elsif(IN\_SEL = "01") then

--якщо 01 то читає з памяті

IN\_SEL\_OUT\_BUS <= RAM\_DATA\_OUT\_BUS;

else

---10 || 11 то береться константа

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

end if;

end process INSEL\_A\_MUX; end MUX\_arch;

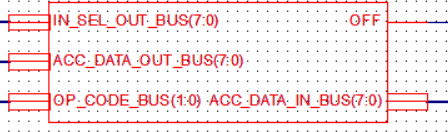
Файл ALU.vhd

Рис. 6 Згенерований символ з файлу ALU.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 ALU\_intf is port(

IN\_SEL\_OUT\_BUS : IN STD\_LOGIC\_VECTOR(7 downto 0);--будуть використовуватися для виконання операцій ALU. ACC\_DATA\_OUT\_BUS : IN STD\_LOGIC\_VECTOR(7 downto 0);--будуть використовуватися для виконання операцій

ALU.

OP\_CODE\_BUS : IN STD\_LOGIC\_VECTOR(1 downto 0);---яка операція виконується ACC\_DATA\_IN\_BUS : OUT STD\_LOGIC\_VECTOR(7 downto 0);

OFF : OUT STD\_LOGIC -- 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

OFF <='0';

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

-- overflow

--result:= unsigned(255, 8) - A;

--overflow case(OP\_CODE\_BUS) is

when "00" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(B); повекртає тесаме значення

when "01" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(A + B); додавання

if("11111111"-A<B) then

OFF <='1';

end if;

when "10" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(A - B); віднімання

if(A < B) then

OFF <= '1';

end if;

when "11" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR (A or "00000010"); операція або

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

end case;

--

-- if result >= B then

-- OFF <= '0';

-- else

-- OFF <= '1';

-- end if;

--

end process ALU; end ALU\_arch;

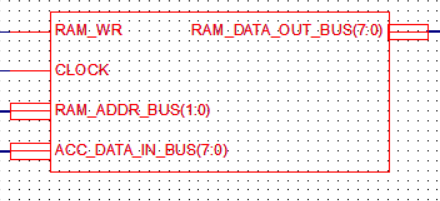
Файл RAM.vhd

Рис. 7 Згенерований символ з файлу RAM.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 RAM\_intf is port(

RAM\_WR : IN STD\_LOGIC; чи здійснюється запис в пам'ять

RAM\_ADDR\_BUS : IN STD\_LOGIC\_VECTOR(1 downto 0); адреса пам'яті,

ACC\_DATA\_IN\_BUS : IN STD\_LOGIC\_VECTOR(7 downto 0); дані, які будуть записані в пам'ять при записі.

RAM\_DATA\_OUT\_BUS: OUT STD\_LOGIC\_VECTOR(7 downto 0); дані, які будуть прочитані з пам'яті.

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);

----створюється сигналу до якого записуються дані з вхідного порту ACC\_DATA\_IN\_BUS 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

---вхідні данні записуються в RAM\_UNIT за переданою адресою RAM\_ADDR\_BUS RAM\_UNIT(conv\_integer(RAM\_ADDR\_BUS)) <= RAM\_DATA\_IN\_BUS;

end if;

end if;

----передається значення, пам'яті RAM\_UNIT за адресою порта який подається RAM\_ADDR\_BUS RAM\_DATA\_OUT\_BUS <= RAM\_UNIT(conv\_integer(RAM\_ADDR\_BUS));

end process RAM; end RAM\_arch;

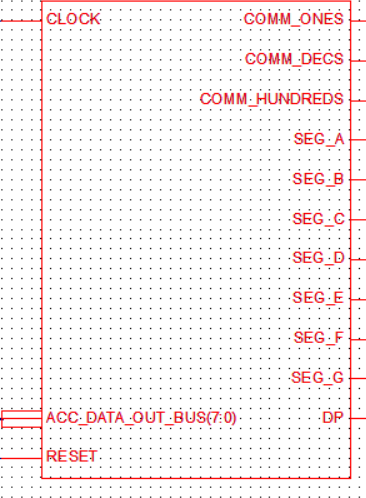
Файл SEGDEC.vhd

Рис. 8 Згенерований символ з файлу 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;

---OFF : 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 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) := "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);

розряду

variable CUR\_DIGIT : DIGIT\_TYPE := ONES;--- поточний розряд числа що відображується на дисплеї variable DIGIT\_VAL : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; зберігається значення для відповідного

variable DIGIT\_CTRL : STD\_LOGIC\_VECTOR(6 downto 0) := "0000000"; керування сегментами для

відображення числа на дисплеї

variable COMMONS\_CTRL : STD\_LOGIC\_VECTOR(2 downto 0) := "000"; --- курування роботою сегментів

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;

---if OFF = "1" then

---DIGIT\_CTRL := "1001111";

---end if

---else

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";

else

end if;

end case;

DIGIT\_VAL := ONES\_BUS; CUR\_DIGIT := ONES; COMMONS\_CTRL := "000";

COMM\_ONES <= COMMONS\_CTRL(0); COMM\_DECS <= COMMONS\_CTRL(1); COMM\_HUNDREDS <= COMMONS\_CTRL(2);

----вивід на дисплей числа

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;

Файл Constraints.vhd

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

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

# UCF for ElbertV2 Development Board # #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*# CONFIG VCCAUX = "3.3" ;

# Clock 12 MHz

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

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

NET "overfloat" LOC = P48 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P49 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P50 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P51 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P54 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P55 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P54 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12; NET "overfloat" LOC = P55 | 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;

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

Висновок

На даній лабораторній роботі я навчився реалізувати цифровий автомат для обчислення значення виразу.