Міністерство освіти і науки України

Національний університет „Львівська політехніка”



**ЛАБОРАТОРНА РОБОТА**

з дисципліни

**МОДЕЛЮВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ**

**Звіт з лабораторної роботи №3**

на тему:

«Поведінковий опис цифрового автомата Перевірка роботи автомата за допомогою стенда Elbert V2 - Spartan 3A FPGA»

**Виконав:**

**Прийняв:**

студент групи KI-202

Порубайміх О.Є.

Козак Н.Б.

Львів – 2023

**Мета:** на базі стенда Elbert V2 - Spartan 3A FPGA реалізувати цифровий автомат для обчислення значення виразу дотримуючись наступних вимог.

**Завдання**

1. Функціонал пристрою повинен бути реалізований згідно отриманого варіанту завдання
2. Пристрій повинен бути ітераційним АЛП повинен виконувати за один такт одну операцію та реалізованим згідно наступної структурної схеми

A diagram of a computer

Description automatically generated with low confidence

Рис. 1. Структурна схема

**Варіант**



Рис. 2. Варіант завдання

**Виконання роботи**

A picture containing text, diagram, plan, parallel

Description automatically generated

Рис. 3. Top level схема

Вміст файлів:

-- CU.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.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);

    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 STATES IS (S\_RST, S\_IDLE, S\_LO1, S\_LO2, S\_C0, S\_C1, SC2, S\_C3, S\_C4, S\_C5, S\_C6, S\_C7, S\_C8, S\_C9, S\_C10, S\_C11, S\_FINISH);

  SIGNAL cu\_cur\_state : STATES;

  SIGNAL cu\_next\_state : STATES;

BEGIN

  CLK : PROCESS (CLOCK)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

      IF (RESET = '1') THEN

        cu\_cur\_state <= S\_RST;

      ELSE

        cu\_cur\_state <= cu\_next\_state;

      END IF;

    END IF;

  END PROCESS CLK;

  NEXT\_STATE : PROCESS (cu\_cur\_state, ENTER\_OP1, ENTER\_OP2, CALCULATE)

  BEGIN

    cu\_next\_state <= cu\_cur\_state;

    CASE(cu\_cur\_state) IS

      WHEN S\_RST =>

      cu\_next\_state <= S\_IDLE;

      WHEN S\_IDLE =>

      IF (ENTER\_OP1 = '1') THEN

        cu\_next\_state <= S\_LO1;

      ELSIF (ENTER\_OP2 = '1') THEN

        cu\_next\_state <= S\_LO2;

      ELSIF (rising\_edge(CALCULATE)) THEN

        cu\_next\_state <= S\_C0;

      ELSE

        cu\_next\_state <= S\_IDLE;

      END IF;

      WHEN S\_LO1 =>

      cu\_next\_state <= S\_IDLE;

      WHEN S\_LO2 =>

      cu\_next\_state <= S\_IDLE;

      WHEN S\_C0 =>

      cu\_next\_state <= S\_C1;

      WHEN S\_C1 =>

      cu\_next\_state <= SC2;

      WHEN SC2 =>

      cu\_next\_state <= S\_C3;

      WHEN S\_C3 =>

      cu\_next\_state <= S\_C4;

      WHEN S\_C4 =>

      cu\_next\_state <= S\_C5;

      WHEN S\_C5 =>

      cu\_next\_state <= S\_C6;

      WHEN S\_C6 =>

      cu\_next\_state <= S\_C7;

      WHEN S\_C7 =>

      cu\_next\_state <= S\_C8;

      WHEN S\_C8 =>

      cu\_next\_state <= S\_C9;

      WHEN S\_C9 =>

      cu\_next\_state <= S\_C10;

      WHEN S\_C10 =>

      cu\_next\_state <= S\_C11;

      WHEN S\_C11 =>

      cu\_next\_state <= S\_FINISH;

      WHEN S\_FINISH =>

      cu\_next\_state <= S\_FINISH;

      WHEN OTHERS =>

      cu\_next\_state <= S\_IDLE;

    END CASE;

  END PROCESS NEXT\_STATE;

  OUTPUT : PROCESS (cu\_cur\_state)

  BEGIN

    CASE(cu\_cur\_state) IS

      WHEN S\_RST =>

      IN\_SEL <= "00";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "00";

      RAM\_WR <= '0';

      ACC\_RST <= '1';

      ACC\_WR <= '0';

      WHEN S\_IDLE =>

      IN\_SEL <= "00";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "00";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '0';

      WHEN S\_LO1 =>

      IN\_SEL <= "00";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '1';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN S\_LO2 =>

      IN\_SEL <= "00";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "10";

      RAM\_WR <= '1';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      -- OP 1

      WHEN S\_C0 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '0';

      WHEN S\_C1 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN SC2 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "00";

      RAM\_ADDR\_BUS <= "10";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      -- OP 2

      WHEN S\_C3 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "01";

      RAM\_ADDR\_BUS <= "10";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '0';

      WHEN S\_C4 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "01";

      RAM\_ADDR\_BUS <= "10";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN S\_C5 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "01";

      RAM\_ADDR\_BUS <= "10";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      -- OP 3

      WHEN S\_C6 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "10";

      RAM\_ADDR\_BUS <= "00";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '0';

      WHEN S\_C7 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "10";

      RAM\_ADDR\_BUS <= "00";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN S\_C8 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "10";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      -- OP 4

      WHEN S\_C9 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "11";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '0';

      WHEN S\_C10 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "11";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN S\_C11 =>

      IN\_SEL <= "01";

      OP\_CODE\_BUS <= "11";

      RAM\_ADDR\_BUS <= "01";

      RAM\_WR <= '0';

      ACC\_RST <= '0';

      ACC\_WR <= '1';

      WHEN S\_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 OUTPUT;

END CU\_ARCH;

-- RAM.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY RAM\_INTF IS

  PORT (

    CLOCK : IN STD\_LOGIC;

    WR : IN STD\_LOGIC;

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

    ADDRESS\_BUS : IN STD\_LOGIC\_VECTOR (1 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 := (OTHERS => (OTHERS => '0'));

BEGIN

  RAM\_IN : PROCESS (CLOCK)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

      IF (WR = '1') THEN

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

      END IF;

    END IF;

  END PROCESS RAM\_IN;

  RAM\_OUT : PROCESS (CLOCK)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

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

    END IF;

  END PROCESS RAM\_OUT;

END RAM\_ARCH;

-- MUX.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY MUX\_INTF IS

  PORT (

    SEL\_IN\_BUS : IN STD\_LOGIC\_VECTOR (1 DOWNTO 0);

    RAM\_DATA\_BUS : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0);

    DATA\_INPUT\_BUS : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0);

    DATA\_OUT : OUT STD\_LOGIC\_VECTOR (7 DOWNTO 0)

  );

END MUX\_INTF;

ARCHITECTURE MUX\_ARCH OF MUX\_INTF IS

  SIGNAL CONST : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

  CONST <= "00000000";

  MUX : PROCESS (SEL\_IN\_BUS, DATA\_INPUT\_BUS, RAM\_DATA\_BUS)

  BEGIN

    CASE (SEL\_IN\_BUS) IS

      WHEN "00" => DATA\_OUT <= DATA\_INPUT\_BUS;

      WHEN "01" => DATA\_OUT <= RAM\_DATA\_BUS;

      WHEN OTHERS => DATA\_OUT <= CONST;

    END CASE;

  END PROCESS;

END MUX\_ARCH;

-- ALU.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY ALU\_INTF IS

  PORT (

    OP\_CODE\_BUS : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

    MUX\_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 ALU\_INTF;

ARCHITECTURE ALU\_ARCH OF ALU\_INTF IS

BEGIN

  ALU : PROCESS (OP\_CODE\_BUS, MUX\_OUT\_BUS, ACC\_DATA\_OUT\_BUS)

    VARIABLE A : unsigned(7 DOWNTO 0);

    VARIABLE B : unsigned(7 DOWNTO 0);

    VARIABLE TEMP\_MUL : unsigned (15 DOWNTO 0);

  BEGIN

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

    B := unsigned(MUX\_OUT\_BUS);

    CASE(OP\_CODE\_BUS) IS

      WHEN "00" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(B);

      WHEN "01" => TEMP\_MUL := (A \* B);

      ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(TEMP\_MUL(7 DOWNTO 0));

      WHEN "10" => ACC\_DATA\_IN\_BUS <= STD\_LOGIC\_VECTOR(A SRL 1);

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

      WHEN OTHERS => ACC\_DATA\_IN\_BUS <= "00000000";

    END CASE;

  END PROCESS ALU;

END ALU\_ARCH;

-- ACC.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ACC\_INTF IS

  PORT (

    CLOCK : IN STD\_LOGIC;

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

    WR : IN STD\_LOGIC;

    RST : IN STD\_LOGIC;

    DATA\_OUT\_BUS : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));

END ACC\_INTF;

ARCHITECTURE ACC\_ARCH OF ACC\_INTF IS

  SIGNAL DATA : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

  ACC\_IN : PROCESS (CLOCK)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

      IF (RST = '1') THEN

        DATA <= "00000000";

      ELSIF (WR = '1') THEN

        DATA <= DATA\_IN\_BUS;

      END IF;

    END IF;

  END PROCESS ACC\_IN;

  ACC\_OUT : PROCESS (CLOCK)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

      DATA\_OUT\_BUS <= DATA;

    END IF;

  END PROCESS ACC\_OUT;

END ACC\_ARCH;

-- DECODER.VHD

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.NUMERIC\_STD.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY 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 DECODER\_INTF;

ARCHITECTURE DECODER\_ARCH OF 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);

      hex\_src := hex\_src(hex\_src'left - 1 DOWNTO hex\_src'right) & '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 DECODER\_ARCH;

-- top\_level\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY testbench IS

END testbench;

ARCHITECTURE behavior OF testbench IS

  COMPONENT top\_level

    PORT (

      CLOCK : IN STD\_LOGIC;

      RESET : IN STD\_LOGIC;

      ENTER\_OP1 : IN STD\_LOGIC;

      ENTER\_OP2 : IN STD\_LOGIC;

      CALC : 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\_OUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

      RAM\_OUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

      ALU\_OUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

      RAM\_ADDR : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0);

      MUX\_OUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

      OP\_CODE : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0);

      IN\_SEL : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0);

      ACC\_WR : OUT STD\_LOGIC;

      RAM\_WR : OUT STD\_LOGIC);

  END COMPONENT;

  SIGNAL CLOCK : STD\_LOGIC;

  SIGNAL RESET : STD\_LOGIC;

  SIGNAL ENTER\_OP1 : STD\_LOGIC;

  SIGNAL ENTER\_OP2 : STD\_LOGIC;

  SIGNAL CALC : STD\_LOGIC;

  SIGNAL DATA\_IN : STD\_LOGIC\_VECTOR (7 DOWNTO 0);

  SIGNAL ACC\_OUT : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

  SIGNAL ACC\_WR : STD\_LOGIC;

  SIGNAL RAM\_OUT : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

  SIGNAL ALU\_OUT : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

  SIGNAL MUX\_OUT : STD\_LOGIC\_VECTOR(7 DOWNTO 0);

  SIGNAL RAM\_ADDR : STD\_LOGIC\_VECTOR(1 DOWNTO 0);

  SIGNAL OP\_CODE : STD\_LOGIC\_VECTOR(1 DOWNTO 0);

  SIGNAL IN\_SEL : STD\_LOGIC\_VECTOR(1 DOWNTO 0);

  SIGNAL RAM\_WR : STD\_LOGIC;

  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;

  CONSTANT PERIOD : TIME := 1 ms;

BEGIN

  UUT : top\_level PORT MAP(

    CLOCK => CLOCK,

    RESET => RESET,

    ENTER\_OP1 => ENTER\_OP1,

    ENTER\_OP2 => ENTER\_OP2,

    IN\_SEL => IN\_SEL,

    CALC => CALC,

    DATA\_IN => DATA\_IN,

    ACC\_OUT => ACC\_OUT,

    RAM\_OUT => RAM\_OUT,

    RAM\_WR => RAM\_WR,

    ALU\_OUT => ALU\_OUT,

    MUX\_OUT => MUX\_OUT,

    RAM\_ADDR => RAM\_ADDR,

    OP\_CODE => OP\_CODE,

    ACC\_WR => ACC\_WR,

    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

  );

  CLK : PROCESS

  BEGIN

    CLOCK <= '0';

    WAIT FOR PERIOD / 2;

    CLOCK <= '1';

    WAIT FOR PERIOD / 2;

  END PROCESS CLK;

  MAIN : PROCESS

  BEGIN

    RESET <= '1';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000000";

    WAIT FOR PERIOD \* 4;

    RESET <= '0';

    ENTER\_OP1 <= '1';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000011";

    WAIT FOR PERIOD \* 4;

   RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000011";

    WAIT FOR PERIOD \* 2;

    RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000100";

    WAIT FOR PERIOD \* 2;

    RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '1';

    CALC <= '0';

    DATA\_IN <= "00000100";

    WAIT FOR PERIOD \* 4;

   RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000000";

    WAIT FOR PERIOD \* 2;

    RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '1';

    DATA\_IN <= "00000000";

    WAIT FOR PERIOD \* 20;

    RESET <= '0';

    ENTER\_OP1 <= '0';

    ENTER\_OP2 <= '0';

    CALC <= '0';

    DATA\_IN <= "00000000";

    WAIT FOR PERIOD \* 10;

  END PROCESS MAIN;

END;

Робота схеми:

A picture containing text, screenshot, number, diagram

Description automatically generated

Рис. 4. Результат запуску testbench

Помаранчевим кольором підкреслено вхідні дані: 310 і 410, синім – результат: 910.

, з чого виходить, що результат обраховано правильно.

Також можна спостерігати виходи декодера. Для одиниць, сегменти, що загоряються – a, b, c, d, f, g, що відповідають цифрі «9»; для десятків та сотень – a, b, c, d, e, f, що відповідають цифрі «0», тобто, в результаті, на виході буде «009».

**Висновок:** я реалізував цифровий автомат, що обчислює вираз згідно заданого варіанту.