Міністерство освіти і науки України

Національний університет „Львівська політехніка”



**ЛАБОРАТОРНА РОБОТА**

з дисципліни

**МОДЕЛЮВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ**

**Звіт з лабораторної роботи №1**

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, parallel, plan

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 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

  CLK : 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 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 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 NEXT\_STATE;

  OUTPUT : 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 <= "01";

      OP\_CODE\_BUS <= "01";

      RAM\_ADDR\_BUS <= "01";

      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 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;

BEGIN

  RAM : PROCESS (CLOCK, ADDRESS\_BUS, DATA\_IN\_BUS)

  BEGIN

    IF (rising\_edge(CLOCK)) THEN

      IF (WR = '1') THEN

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

      END IF;

    END IF;

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

  END PROCESS RAM;

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 : process(CLOCK, DATA)

   begin

    if (rising\_edge(CLOCK)) then

      if(RST = '1') then

        DATA <= "00000000";

      elsif (WR = '1') then

        DATA <= DATA\_IN\_BUS;

      end if;

    end if;

    DATA\_OUT\_BUS <= DATA;

   end process ACC;

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;

Робота схеми:

A screenshot of a computer

Description automatically generated with medium confidence

Рис. 4. Результат запуску testbench

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