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 3A FPGA»

Варіант №6

Виконав:

ст. гр. КІ-201

Вітик С.А.

Прийняв:

ст. викладач   
каф. ЕОМ  
Козак Н. Б.

Львів 2024

**Мета роботи**: На базі стенда реалізувати цифровий автомат для обчислення значення виразів.

**Виконання роботи:**

**Зображення, що містить схема, текст, План, Креслення

Автоматично згенерований опис**

*Рис. 1 – Top Level*

|  |
| --- |
| Файл 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;  -- Uncomment the following library declaration if instantiating  -- any Xilinx primitives in this code.  --library UNISIM;  --use UNISIM.VComponents.all;  entity ACC is  Port ( WR : in STD\_LOGIC;  RESET : in STD\_LOGIC;  CLK : in STD\_LOGIC;  INPUT : in STD\_LOGIC\_VECTOR (7 downto 0);  OUTPUT : out STD\_LOGIC\_VECTOR (7 downto 0));  end ACC;  architecture ACC\_arch of ACC is  signal DATA : STD\_LOGIC\_VECTOR (7 downto 0);  begin  process (CLK)  begin  if rising\_edge(CLK) then  if RESET = '1' then  DATA <= (others => '0');  elsif WR = '1' then  DATA <= INPUT;  end if;  end if;  end process;    OUTPUT <= DATA;  end ACC\_arch; |

|  |
| --- |
| Файл 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 is  Port ( A : in STD\_LOGIC\_VECTOR(7 downto 0);  B : in STD\_LOGIC\_VECTOR(7 downto 0);  OP : in STD\_LOGIC\_VECTOR(1 downto 0);  OUTPUT : out STD\_LOGIC\_VECTOR(7 downto 0);  OVERFLOW: out STD\_LOGIC);  end ALU;  architecture ALU\_Behavioral of ALU is  signal ALUR: STD\_LOGIC\_VECTOR(15 downto 0) := (others => '0');  signal Carry: STD\_LOGIC := '0';  begin  process(A, B, OP)  begin  case (OP) is  when "01" => ALUR <= ("00000000" & A) + ("00000000" & B);  when "10" => ALUR <= ("00000000" & A) + ("11111111" & not B) + "0000000000000001";  when "11" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll to\_integer(unsigned(B))); ALUR(15 downto 8) <= "00000000";  when others => ALUR <= ("00000000" & B);  end case;  end process;  OUTPUT <= ALUR(7 downto 0);  OVERFLOW <= ALUR(8) OR ALUR(9) OR ALUR(10) OR ALUR(11) OR ALUR(12) OR ALUR(13) OR ALUR(14) OR ALUR(15);  end ALU\_Behavioral; |

|  |
| --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity CPU is  port( ENTER\_OP1 : IN STD\_LOGIC;  ENTER\_OP2 : IN STD\_LOGIC;  CALCULATE : IN STD\_LOGIC;  RESET : IN STD\_LOGIC;  CLOCK : IN STD\_LOGIC;  RAM\_WR : OUT STD\_LOGIC;  RAM\_ADDR : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0);  CONST : 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 : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0));  end CPU;    architecture CPU\_arch of CPU is  type STATE\_TYPE is (RST, IDLE, LOAD\_OP1, LOAD\_OP2, RUN\_CALC0, RUN\_CALC1, RUN\_CALC2, RUN\_CALC3, RUN\_CALC4, FINISH);  signal CUR\_STATE : STATE\_TYPE;  signal NEXT\_STATE : STATE\_TYPE;  begin  SYNC\_PROC: process (CLOCK)  begin  if (rising\_edge(CLOCK)) then  if (RESET = '1') then  CUR\_STATE <= RST;  else  CUR\_STATE <= NEXT\_STATE;  end if;  end if;  end process;      NEXT\_STATE\_DECODE: process (CLOCK, ENTER\_OP1, ENTER\_OP2, CALCULATE)  begin  NEXT\_STATE <= CUR\_STATE;    case(CUR\_STATE) is  when RST =>  NEXT\_STATE <= IDLE;  when IDLE =>  if (ENTER\_OP1 = '1') then  NEXT\_STATE <= LOAD\_OP1;  elsif (ENTER\_OP2 = '1') then  NEXT\_STATE <= LOAD\_OP2;  elsif (CALCULATE = '1') then  NEXT\_STATE <= RUN\_CALC0;  else  NEXT\_STATE <= IDLE;  end if;  when LOAD\_OP1 =>  NEXT\_STATE <= IDLE;  when LOAD\_OP2 =>  NEXT\_STATE <= IDLE;  when RUN\_CALC0 =>  NEXT\_STATE <= RUN\_CALC1;  when RUN\_CALC1 =>  NEXT\_STATE <= RUN\_CALC2;  when RUN\_CALC2 =>  NEXT\_STATE <= RUN\_CALC3;  when RUN\_CALC3 =>  NEXT\_STATE <= RUN\_CALC4;  when RUN\_CALC4 =>  NEXT\_STATE <= FINISH;  when FINISH =>  NEXT\_STATE <= FINISH;  when others =>  NEXT\_STATE <= IDLE;  end case;  end process;  OUTPUT\_DECODE: process (CUR\_STATE)  begin  case (CUR\_STATE) is  when RST =>  RAM\_WR <= '0';  RAM\_ADDR <= "00";  CONST <= "00000000";  ACC\_WR <= '0';  ACC\_RST <= '1';  IN\_SEL <= "00";  OP <= "00";  when LOAD\_OP1 =>  RAM\_WR <= '1';  RAM\_ADDR <= "00";  CONST <= "00000000";  ACC\_WR <= '0';  ACC\_RST <= '1';  IN\_SEL <= "00";  OP <= "00";  when LOAD\_OP2 =>  RAM\_WR <= '1';  RAM\_ADDR <= "01";  CONST <= "00000000";  ACC\_WR <= '0';  ACC\_RST <= '1';  IN\_SEL <= "00";  OP <= "00";  when RUN\_CALC0 =>  RAM\_WR <= '0';  RAM\_ADDR <= "00";  CONST <= "00000000";  ACC\_WR <= '1';  ACC\_RST <= '0';  IN\_SEL <= "01";  OP <= "01";  when RUN\_CALC1 =>  RAM\_WR <= '0';  RAM\_ADDR <= "01";  CONST <= "00000000";  ACC\_WR <= '1';  ACC\_RST <= '0';  IN\_SEL <= "01";  OP <= "01";  when RUN\_CALC2 =>  RAM\_WR <= '0';  RAM\_ADDR <= "01";  CONST <= "00001010";  ACC\_WR <= '1';  ACC\_RST <= '0';  IN\_SEL <= "10";  OP <= "01";  when RUN\_CALC3 =>  RAM\_WR <= '0';  RAM\_ADDR <= "01";  CONST <= "00000010";  ACC\_WR <= '1';  ACC\_RST <= '0';  IN\_SEL <= "10";  OP <= "10";  when RUN\_CALC4 =>  RAM\_WR <= '0';  RAM\_ADDR <= "01";  CONST <= "00000000";  ACC\_WR <= '1';  ACC\_RST <= '0';  IN\_SEL <= "01";  OP <= "11";  when IDLE =>  RAM\_WR <= '0';  RAM\_ADDR <= "00";  CONST <= "00000000";  ACC\_WR <= '0';  ACC\_RST <= '0';  IN\_SEL <= "00";  OP <= "00";  when others =>  RAM\_WR <= '0';  RAM\_ADDR <= "00";  CONST <= "00000000";  ACC\_WR <= '0';  ACC\_RST <= '0';  IN\_SEL <= "00";  OP <= "00";  end case;  end process;  end CPU\_arch; |

|  |
| --- |
| Файл FullAdder.vhd  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity FullAdder is  Port (A: in STD\_LOGIC;  B: in STD\_LOGIC;  Ci: in STD\_LOGIC;  S: out STD\_LOGIC;  Co: out STD\_LOGIC);  end FullAdder;  architecture Behavioral of FullAdder is  begin  S <= (A xor B) xor Ci;  Co <= (A and B) or ((A xor B) and Ci);  end Behavioral; |

|  |
| --- |
| Файл FullAdder8.vhd  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity FullAdder8 is  Port ( A: in STD\_LOGIC\_VECTOR (7 downto 0);  B: in STD\_LOGIC\_VECTOR (7 downto 0);  Ci: in STD\_LOGIC;  S: out STD\_LOGIC\_VECTOR (7 downto 0);  Co: out STD\_LOGIC);  end FullAdder8;  architecture Behavioral of FullAdder8 is  component FullAdder is  Port (A: in STD\_LOGIC;  B: in STD\_LOGIC;  Ci: in STD\_LOGIC;  S: out STD\_LOGIC;  Co: out STD\_LOGIC);  end component;  signal carry: STD\_LOGIC\_VECTOR(8 downto 0) := (others => '0');  begin  carry(0) <= Ci;  FullAdderGenerate: for i in 0 to 7 generate  adder: FullAdder port map(  A => A(i),  B => B(i),  Ci => carry(i),  S => S(i),  Co => carry(i + 1)  );  end generate FullAdderGenerate;  Co <= carry(8);  end Behavioral; |

|  |
| --- |
| Файл MUX.vhd  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity MUX is  PORT(  SEL: in STD\_LOGIC\_VECTOR(1 downto 0);  CONST: in STD\_LOGIC\_VECTOR(7 downto 0);  --CONST1: in STD\_LOGIC\_VECTOR()  DATA\_IN0: in STD\_LOGIC\_VECTOR(7 downto 0);  DATA\_IN1: in STD\_LOGIC\_VECTOR(7 downto 0);  OUTPUT: out STD\_LOGIC\_VECTOR(7 downto 0)  );  end MUX;  architecture Behavioral of MUX is  begin  process (SEL, DATA\_IN0, DATA\_IN1, CONST)  begin  if (SEL = "00") then  OUTPUT <= DATA\_IN0;  elsif (SEL = "01") then  OUTPUT <= DATA\_IN1;  else  OUTPUT <= CONST;  end if;  end process;  end Behavioral; |

|  |
| --- |
| Файл RAM.vhd  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity MUX is  PORT(  SEL: in STD\_LOGIC\_VECTOR(1 downto 0);  CONST: in STD\_LOGIC\_VECTOR(7 downto 0);  --CONST1: in STD\_LOGIC\_VECTOR()  DATA\_IN0: in STD\_LOGIC\_VECTOR(7 downto 0);  DATA\_IN1: in STD\_LOGIC\_VECTOR(7 downto 0);  OUTPUT: out STD\_LOGIC\_VECTOR(7 downto 0)  );  end MUX;  architecture Behavioral of MUX is  begin  process (SEL, DATA\_IN0, DATA\_IN1, CONST)  begin  if (SEL = "00") then  OUTPUT <= DATA\_IN0;  elsif (SEL = "01") then  OUTPUT <= DATA\_IN1;  else  OUTPUT <= CONST;  end if;  end process;  end Behavioral; |

|  |
| --- |
| Файл SEG\_DECODER.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;    ENTITY SegDecoderTest IS  END SegDecoderTest;    ARCHITECTURE behavior OF SegDecoderTest IS    -- Component Declaration for the Unit Under Test (UUT)    COMPONENT SEG\_DECODER  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 COMPONENT;    --Inputs  signal CLOCK : std\_logic := '0';  signal RESET : std\_logic := '0';  signal ACC\_DATA\_OUT\_BUS : std\_logic\_vector(7 downto 0) := (others => '0');  --Outputs  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;  -- Clock period definitions  constant CLKP : time := 2ps;    BEGIN    -- Instantiate the Unit Under Test (UUT)  uut: SEG\_DECODER PORT MAP (  CLOCK => CLOCK,  RESET => RESET,  ACC\_DATA\_OUT\_BUS => ACC\_DATA\_OUT\_BUS,  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  );  -- Clock process definitions  CLOCK\_process :process  begin  CLOCK <= '0';  wait for CLKP/2;  CLOCK <= '1';  wait for CLKP/2;  end process;    -- Stimulus process  stim\_proc: process  begin  RESET <= '1';  wait for 4 \* CLKP;  RESET <= '0';  ACC\_DATA\_OUT\_BUS <= "00001111";  wait for CLKP;  assert COMM\_ONES = '0' severity failure;  assert COMM\_DECS = '1' severity failure;  assert COMM\_HUNDREDS = '1' severity failure;  assert SEG\_A = '0' severity failure;  assert SEG\_B = '1' severity failure;  assert SEG\_C = '0' severity failure;  assert SEG\_D = '0' severity failure;  assert SEG\_E = '1' severity failure;  assert SEG\_F = '0' severity failure;  assert SEG\_G = '0' severity failure;  assert DP = '1' severity failure;  wait for CLKP;  assert COMM\_ONES = '1' severity failure;  assert COMM\_DECS = '0' severity failure;  assert COMM\_HUNDREDS = '1' severity failure;  assert SEG\_A = '1' severity failure;  assert SEG\_B = '0' severity failure;  assert SEG\_C = '0' severity failure;  assert SEG\_D = '1' severity failure;  assert SEG\_E = '1' severity failure;  assert SEG\_F = '1' severity failure;  assert SEG\_G = '1' severity failure;  assert DP = '1' severity failure;  -- insert stimulus here  wait;  end process;  END; |

Зображення, що містить знімок екрана, ряд, Мультимедійне програмне забезпечення

Автоматично згенерований опис

*Рис. 2 – Часова діаграма ACC*

*Зображення, що містить Мультимедійне програмне забезпечення, програмне забезпечення, Графічний редактор, знімок екрана

Автоматично згенерований опис*

*Рис. 3 – Часова діаграма ALU*

*Зображення, що містить знімок екрана, ряд, програмне забезпечення, текст

Автоматично згенерований опис*

*Рис. 4 – Часова діаграма MUX*

*Зображення, що містить Мультимедійне програмне забезпечення, програмне забезпечення, ряд, знімок екрана

Автоматично згенерований опис*

*Рис. 5 – Часова діаграма RAM*

*Зображення, що містить знімок екрана, Барвистість, Мультимедійне програмне забезпечення

Автоматично згенерований опис*

*Рис 6. – Часова діграма SEG\_DECODER*

*Зображення, що містить знімок екрана, монітор, Мультимедійне програмне забезпечення

Автоматично згенерований опис*

*Рис 7. – Часова діграма TopLevel*

|  |
| --- |
| *Файл TopLevelTest.vhd*  *LIBRARY ieee;*  *USE ieee.std\_logic\_1164.ALL;*  *USE ieee.numeric\_std.ALL;*  *LIBRARY UNISIM;*  *USE UNISIM.Vcomponents.ALL;*  *ENTITY TopLevel\_TopLevel\_sch\_tb IS*  *END TopLevel\_TopLevel\_sch\_tb;*  *ARCHITECTURE behavioral OF TopLevel\_TopLevel\_sch\_tb IS*  *COMPONENT TopLevel*  *PORT( CLOCK : IN STD\_LOGIC;*  *RESET : 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);*  *COMMON\_0\_OUT : OUT STD\_LOGIC;*  *COMMON\_1\_OUT : OUT STD\_LOGIC;*  *COMMON\_2\_OUT : OUT STD\_LOGIC;*  *TEST: OUT STD\_LOGIC\_VECTOR(7 downto 0);*  *A\_OUT : OUT STD\_LOGIC;*  *B\_OUT : OUT STD\_LOGIC;*  *C\_OUT : OUT STD\_LOGIC;*  *D\_OUT : OUT STD\_LOGIC;*  *E\_OUT : OUT STD\_LOGIC;*  *F\_OUT : OUT STD\_LOGIC;*  *G\_OUT : OUT STD\_LOGIC;*  *DP\_OUT : OUT STD\_LOGIC;*  *RAMOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0);*  *ALUOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0);*  *RAMA: OUT STD\_LOGIC\_VECTOR(1 downto 0);*  *RAMWR: OUT STD\_LOGIC;*  *OVERFLOW : OUT STD\_LOGIC);*  *END COMPONENT;*  *SIGNAL CLOCK : STD\_LOGIC := '0';*  *SIGNAL RESET : 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 COMMON\_0\_OUT : STD\_LOGIC;*  *SIGNAL COMMON\_1\_OUT : STD\_LOGIC;*  *SIGNAL COMMON\_2\_OUT : STD\_LOGIC;*  *SIGNAL A\_OUT : STD\_LOGIC;*  *SIGNAL B\_OUT : STD\_LOGIC;*  *SIGNAL C\_OUT : STD\_LOGIC;*  *SIGNAL D\_OUT : STD\_LOGIC;*  *SIGNAL E\_OUT : STD\_LOGIC;*  *SIGNAL F\_OUT : STD\_LOGIC;*  *SIGNAL G\_OUT : STD\_LOGIC;*  *SIGNAL DP\_OUT : STD\_LOGIC;*  *SIGNAL OVERFLOW : STD\_LOGIC;*  *SIGNAL TEST: STD\_LOGIC\_VECTOR(7 downto 0);*  *SIGNAL TEST2: STD\_LOGIC\_VECTOR(7 downto 0);*  *signal RAMOUT: STD\_LOGIC\_VECTOR(7 downto 0);*  *signal ALUOUT: STD\_LOGIC\_VECTOR(7 downto 0);*  *signal RAMA: STD\_LOGIC\_VECTOR(1 downto 0);*  *signal RAMWR: STD\_LOGIC;*    *-- constant CLOCK\_period : time := 166ns;*  *constant CLKP: time := 12ms;--24ms;*  *BEGIN*  *UUT: TopLevel PORT MAP(*  *CLOCK => CLOCK,*  *RESET => RESET,*  *ENTER\_OP1 => ENTER\_OP1,*  *ENTER\_OP2 => ENTER\_OP2,*  *CALCULATE => CALCULATE,*  *DATA\_IN => DATA\_IN,*  *COMMON\_0\_OUT => COMMON\_0\_OUT,*  *COMMON\_1\_OUT => COMMON\_1\_OUT,*  *COMMON\_2\_OUT => COMMON\_2\_OUT,*  *A\_OUT => A\_OUT,*  *B\_OUT => B\_OUT,*  *C\_OUT => C\_OUT,*  *D\_OUT => D\_OUT,*  *E\_OUT => E\_OUT,*  *F\_OUT => F\_OUT,*  *G\_OUT => G\_OUT,*  *DP\_OUT => DP\_OUT,*  *OVERFLOW => OVERFLOW,*  *TEST => TEST,*  *RAMOUT => RAMOUT,*  *ALUOUT => ALUOUT,*  *RAMA => RAMA,*  *RAMWR => RAMWR*  *);*    *CLOCK\_process: process*  *begin*  *CLOCK <= '0';*  *wait for 83ns;*  *CLOCK <= '1';*  *wait for 83ns;*  *end process;*  *-- \*\*\* Test Bench - User Defined Section \*\*\**  *tb : PROCESS*  *BEGIN*  *lp1: for i in 4 to 6 loop*  *lp2: for j in 2 to 2 loop*  *TEST2 <= std\_logic\_vector(to\_unsigned(i + j + 10 - 2, 8) sll j);*  *ENTER\_OP1 <= '1';*  *ENTER\_OP2 <= '1';*  *CALCULATE <= '1';*  *DATA\_IN <= (others => '0');*  *RESET <= '0';*  *wait for CLKP;*  *RESET <= '1';*  *wait for CLKP;*  *DATA\_IN <= std\_logic\_vector(to\_unsigned(i, 8)); -- A*  *ENTER\_OP1 <= '0';*  *wait for CLKP;*  *ENTER\_OP1 <= '1';*  *wait for CLKP;*  *DATA\_IN <= std\_logic\_vector(to\_unsigned(j, 8)); -- B*  *ENTER\_OP2 <= '0';*  *wait for CLKP;*  *ENTER\_OP2 <= '1';*  *wait for CLKP;*  *CALCULATE <= '0'; -- START CALCULATION*  *wait for CLKP\* 7;*  *assert TEST = TEST2 severity FAILURE;*  *wait for CLKP;*  *end loop;*  *end loop;*    *WAIT; -- will wait forever*  *END PROCESS;*  *-- \*\*\* End Test Bench - User Defined Section \*\*\**  *END;* |

*Зображення, що містить текст, знімок екрана, схема, Шрифт

Автоматично згенерований опис*

*Рис.8 – 7-сегментний індикатор*

Переглянемо часову діаграму 7seg decoder.

Бачимо, що для вхідного числа 00001111(2) = 15(10)

Ми отримуємо такі значення(якщо значення не вказане, вважаюмо його за одиницю):

COMM\_ONES = 0: A,C,D, F, G = 0, що відповідає числу 5  
COMM\_DECS = 0: B,C = 0, що відповідає числу 1

COMM\_HUNDREDS = 0: A,B,C,D,E,F = 0, що відповідає числу 0

Отримуємо значення 015, що збігається з даним йому.

**Висновок:** Виконуючи дану лабораторну роботу я навчився реалізовувати цифровий автомат для обчислення значення виразів використовуючи засоби VHDL.