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 №8

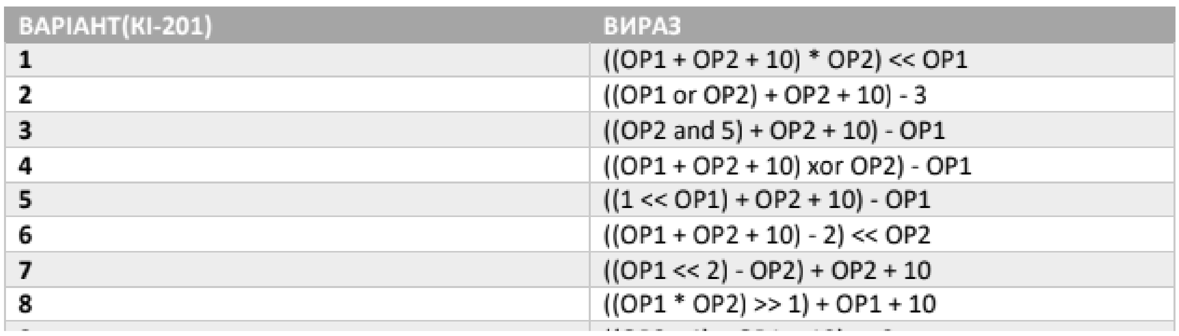
Виконав: ст. гр. КІ-201 Головко І.О.

Прийняв: ст. викладач каф. ЕОМ Козак Н.Б.

Львів - 2024

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

Мій варіант - №8



ВАРІАНТ(КІ-201)	ВИРАЗ
1	((OP1 + OP2 + 10) * OP2) << OP1
2	((OP1 or OP2) + OP2 + 10) - 3
3	((OP2 and 5) + OP2 + 10) - OP1
4	((OP1 + OP2 + 10) xor OP2) - OP1
5	((1 << OP1) + OP2 + 10) - OP1
6	((OP1 + OP2 + 10) - 2) << OP2
7	((OP1 << 2) - OP2) + OP2 + 10
8	((OP1 * OP2) >> 1) + OP1 + 10

Рис. 1 - скріншот заданого варіанту

Виконання роботи

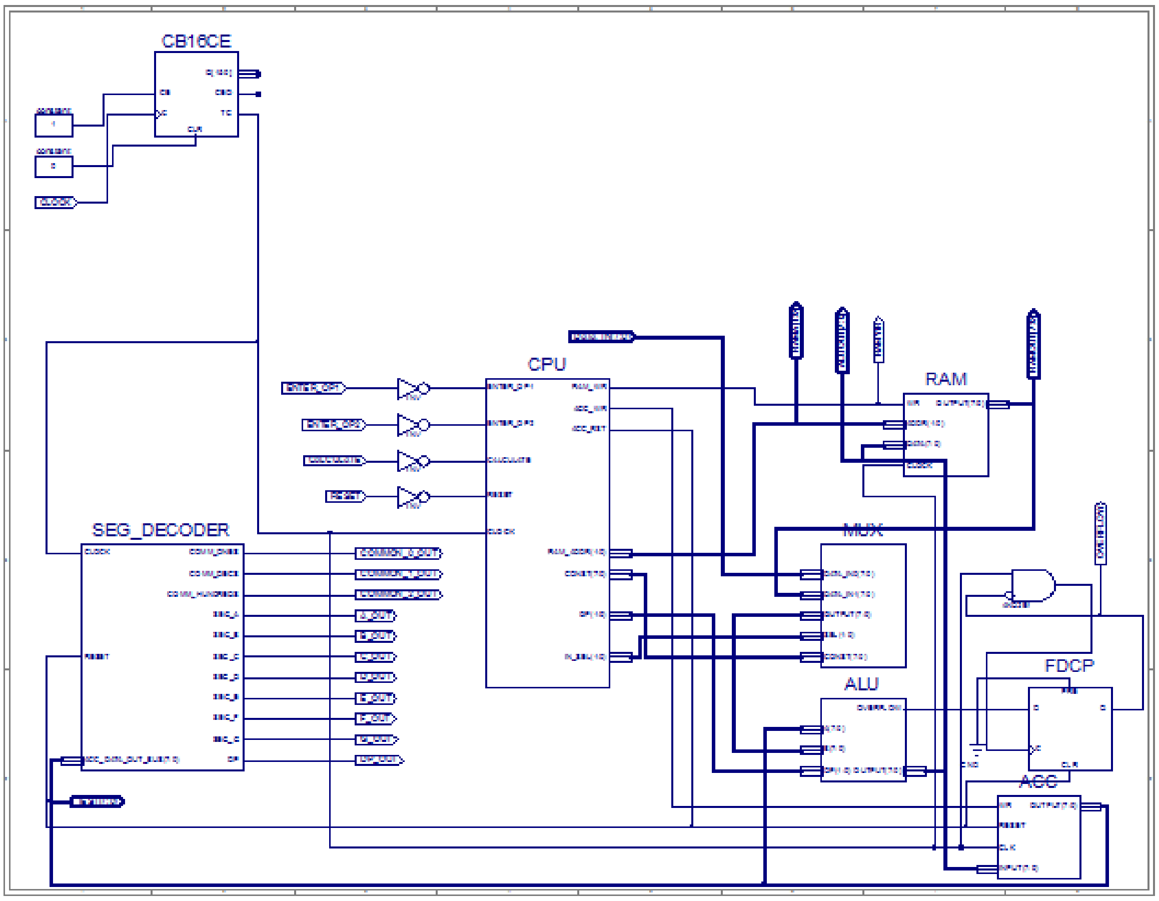


Рис. 2 - Top Level

Файл ACC.vhd:

```
library IEEE;
```

```vhdl
use IEEE.STD_LOGIC_1164.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:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.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 <=
std_logic_vector(to_unsigned((to_integer(unsigned(("00000
000" & A))) * to_integer(unsigned(("00000000" &
B)))),16));
               when "11" => ALUR <=
std_logic_vector(unsigned(("00000000" & A)) srl
to_integer(unsigned(B)));
               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;
```

Файл CPU.vhd:

```
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 =>
```

```vhdl
                    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 <= "00";
        when RUN_CALC1 =>
            RAM_WR <= '0';
            RAM_ADDR <= "01";
            CONST <= "00000000";
            ACC_WR <= '1';
            ACC_RST <= '0';
            IN_SEL <= "01";
            OP <= "10";
        when RUN_CALC2 =>
            RAM_WR <= '0';
            RAM_ADDR <= "01";
            CONST <= "00000001";
            ACC_WR <= '1';
            ACC_RST <= '0';
            IN_SEL <= "10";
            OP <= "11";
        when RUN_CALC3 =>
            RAM_WR <= '0';
            RAM_ADDR <= "00";
            CONST <= "00000000";
            ACC_WR <= '1';
            ACC_RST <= '0';
            IN_SEL <= "01";
            OP <= "01";
        when RUN_CALC4 =>
            RAM_WR <= '0';
            RAM_ADDR <= "00";
            CONST <= "00001010";
            ACC_WR <= '1';
```

```
                   ACC_RST <= '0';
                   IN_SEL <= "10";
                   OP <= "01";
              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 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);
```

```vhdl
          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:

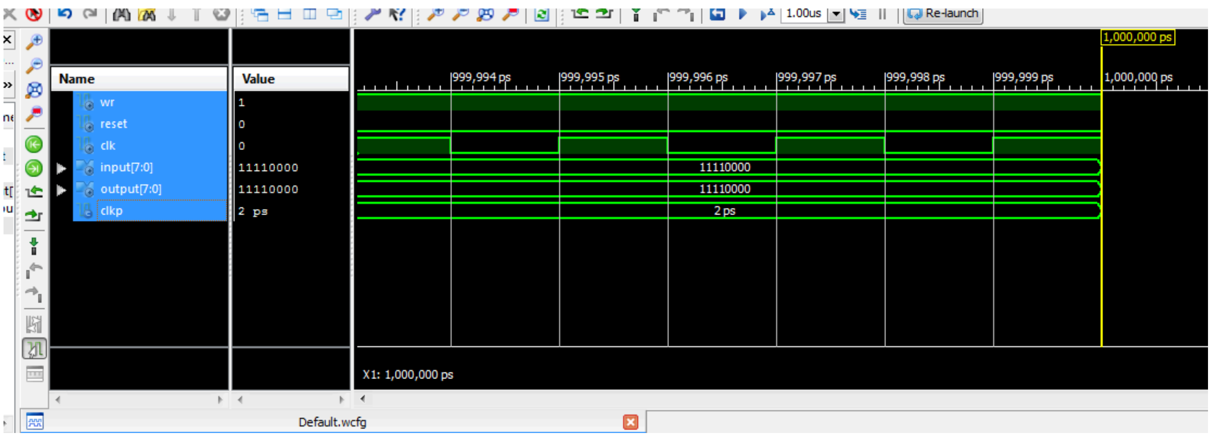
```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY RAMTest IS
END RAMTest;

ARCHITECTURE behavior OF RAMTest IS
    COMPONENT RAM
    PORT(
      CLOCK: IN std_logic;
         WR : IN  std_logic;
         ADDR : IN  std_logic_vector(1 downto 0);
         DATA : IN  std_logic_vector(7 downto 0);
         OUTPUT : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;

  signal CLOCK: std_logic := '0';
   signal WR : std_logic := '0';
   signal ADDR : std_logic_vector(1 downto 0) := (others
=> '0');
   signal DATA : std_logic_vector(7 downto 0) := (others
=> '0');
   signal OUTPUT : std_logic_vector(7 downto 0);

BEGIN

   uut: RAM PORT MAP (
       CLOCK => CLOCK,
          WR => WR,
          ADDR => ADDR,
          DATA => DATA,
          OUTPUT => OUTPUT
        );

  clock_proc: process
  begin
    CLOCK <= not CLOCK;
    wait for 1ps;
  end process;

   stim_proc: process
   begin
    WR <= '1';
      ADDR <= "00";
    DATA <= "11111111";
    wait for 2ps;
    WR <= '0';
    assert OUTPUT = DATA report "OUTPUT != DATA when ADDR
= 00 and WR = 0" severity FAILURE;
    DATA <= "00000000";
    wait for 2ps;
    assert OUTPUT = x"FF" report "OUTPUT != x'FF' when
ADDR = 00 and WR = 0" severity FAILURE;
    ADDR <= "01";
    WR <= '1';
    DATA <= x"AF";
    wait for 2ps;
    assert OUTPUT = x"AF" report "OUTPUT != x'AF' when
ADDR = 01 and WR = 1" severity FAILURE;
      wait;
   end process;

END;
```

Файл SEG_DECODER.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity SEG_DECODER 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;
```

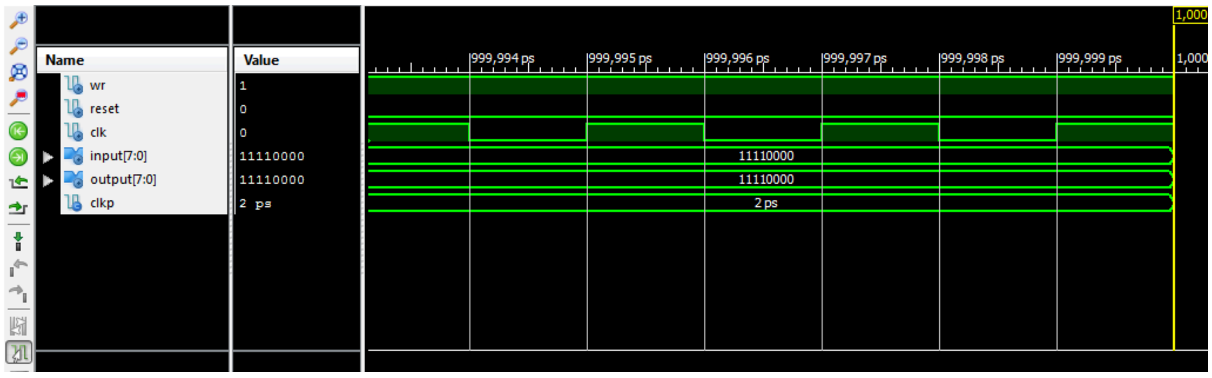
```vhdl
               SEG_D      : OUT STD_LOGIC;
               SEG_E      : OUT STD_LOGIC;
               SEG_F      : OUT STD_LOGIC;
               SEG_G      : OUT STD_LOGIC;
               DP         : OUT STD_LOGIC);
end SEG_DECODER;

architecture Behavioral of SEG_DECODER 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       <= not
COMMONS_CTRL(0);
                    COMM_DECS       <= not
COMMONS_CTRL(1);
                    COMM_HUNDREDS <= not COMMONS_CTRL(2);

                    SEG_A <= not DIGIT_CTRL(6);
                    SEG_B <= not DIGIT_CTRL(5);
                    SEG_C <= not DIGIT_CTRL(4);
                    SEG_D <= not DIGIT_CTRL(3);
                    SEG_E <= not DIGIT_CTRL(2);
                    SEG_F <= not DIGIT_CTRL(1);
                    SEG_G <= not DIGIT_CTRL(0);
                    DP   <= '1';

               end if;
     end process INDICATE;

end Behavioral;
```



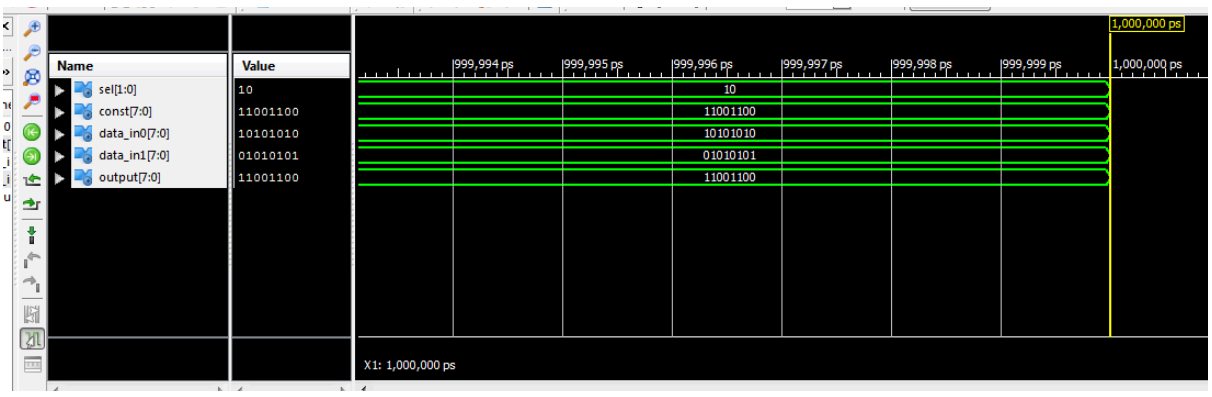
Name	Value
wr	1
reset	0
clk	0
input[7:0]	11110000
output[7:0]	11110000
clkp	2 ps

Рис. 3 - часова діаграма ACC



Name	Value
wr	1
reset	0
clk	0
input[7:0]	11110000
output[7:0]	11110000
clkp	2 ps

Рис. 4 - часова діаграма ALU



Name	Value
sel[1:0]	10
const[7:0]	11001100
data_in0[7:0]	10101010
data_in1[7:0]	01010101
output[7:0]	11001100

Рис. 5 - часова діаграма MUX

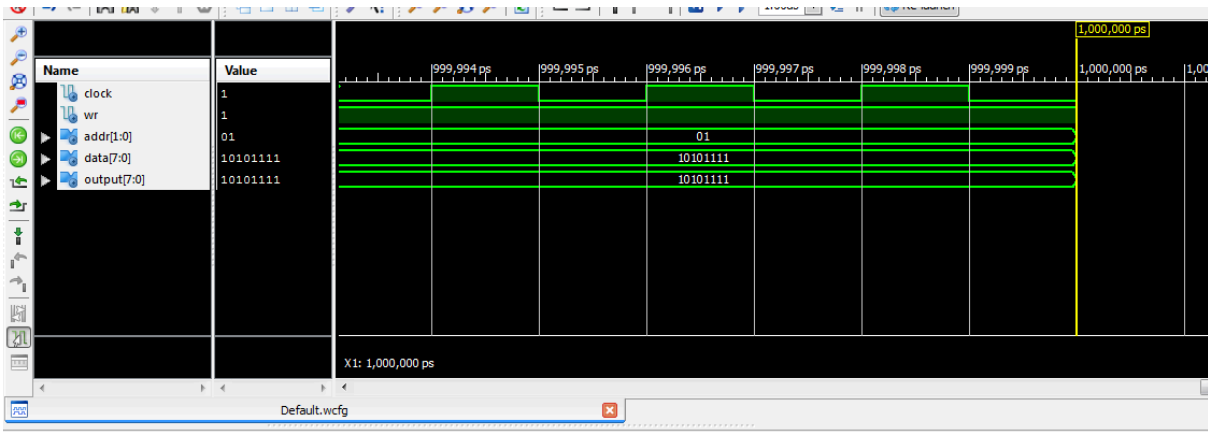


Рис. 6 - часова діаграма RAM

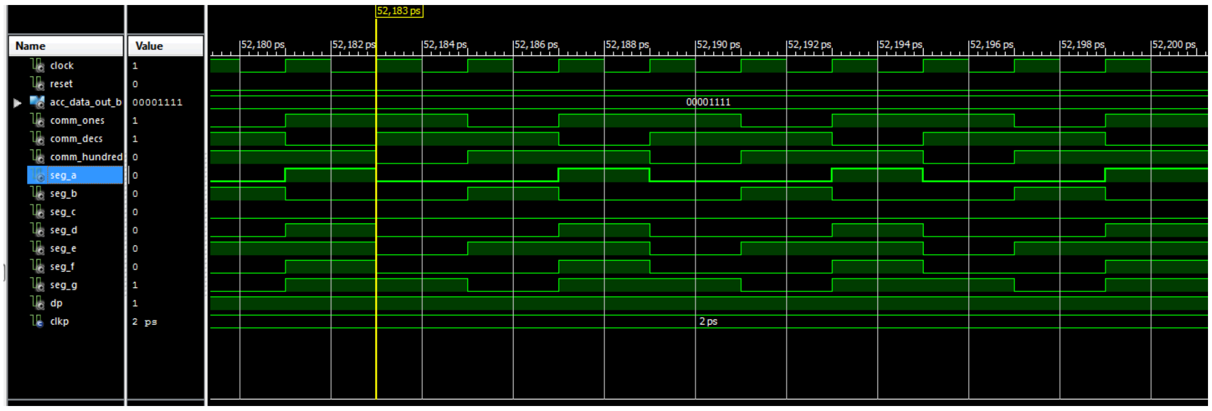


Рис. 7 - часова діаграма SEG_DECODER

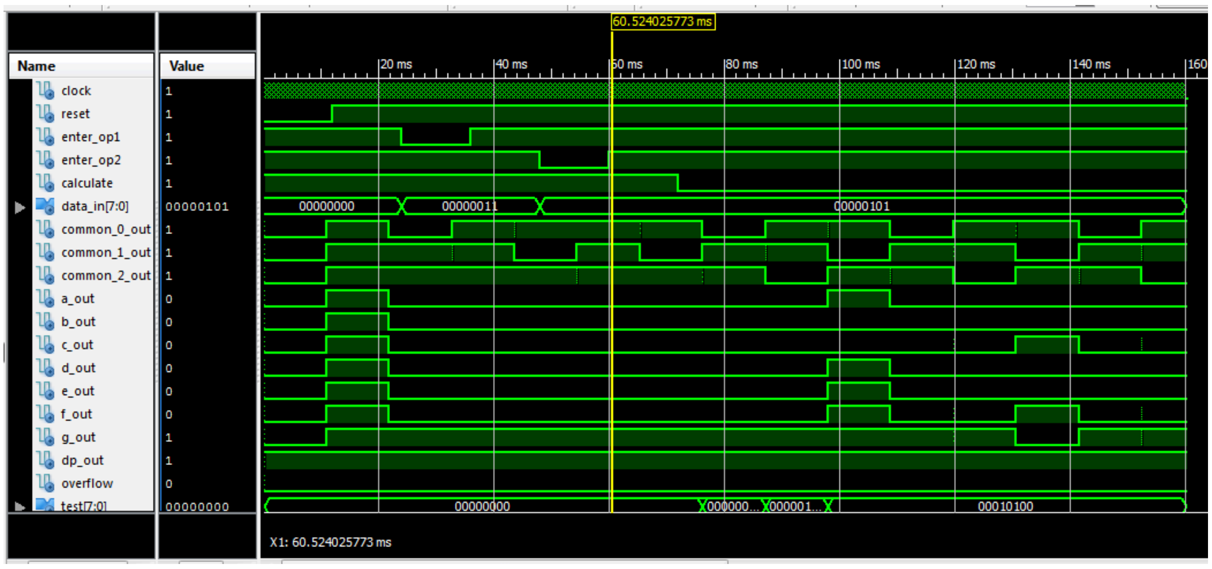


Рис. 8 - часова діаграма TopLevel

Файл TopLevelTest.vhd:

```vhdl
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 4 loop
              lp2: for j in 2 to 2 loop
                   TEST2 <=
std_logic_vector((to_unsigned(i * j, 8) srl 1) + i + 10);
                   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;
```

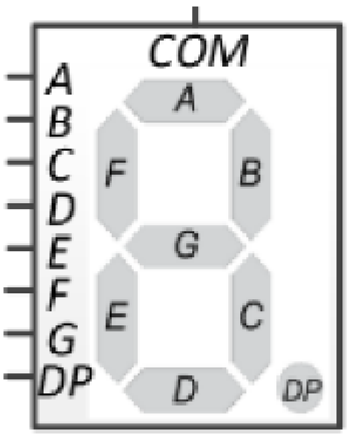


Рис. 9 - семисегментний індикатор

Вхідне число: $020_{10} = 10100_2$

Сотні: g = 1 (a,b,c,d,e,f = 0)

Десятки: c = 1, f = 1 (a,b,d,e,g = 0)

Одиниці: g = 1 (a,b,c,d,e,f = 0)

Висновок: виконуючи дану неймовірну лабораторну роботу та оформлюючи звіт до неї, я навчився реалізовувати цифровий автомат для обчислення значення виразів використовуючи засоби VHDL.