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

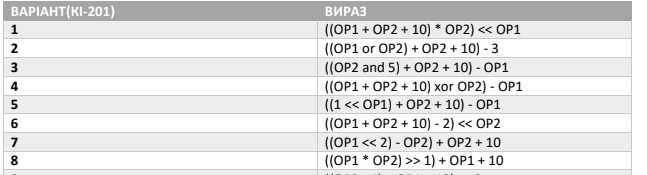


Рис. 1 - скріншот заданого варіанту

**Виконання роботи**

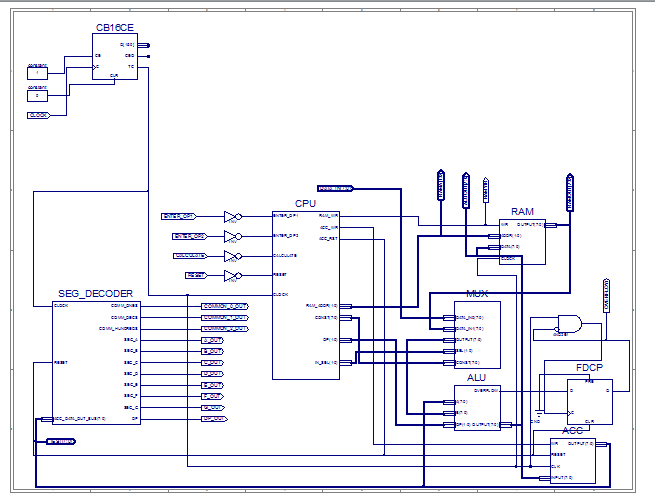
****

Рис. 2 - Top Level

Файл ACC.vhd:

| library IEEE; 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:

| 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(("00000000" & 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 =>  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);  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 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;  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; |
| --- |

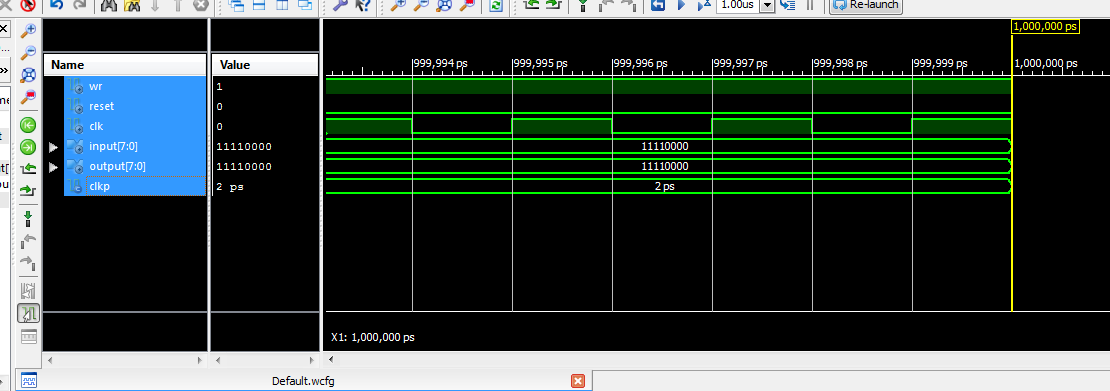


Рис. 3 - часова діаграма ACC

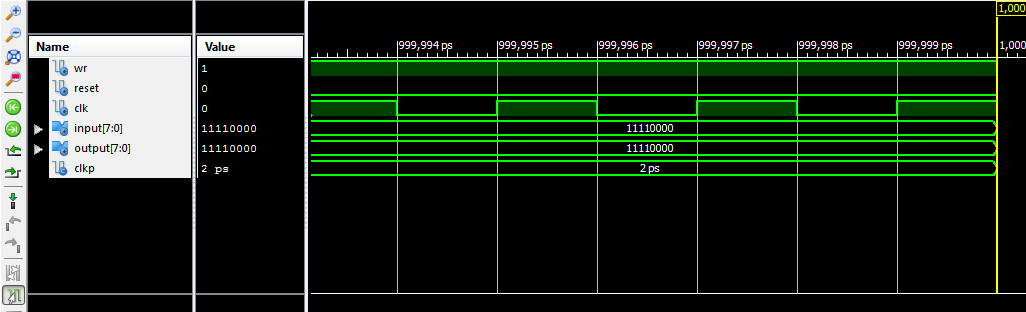


Рис. 4 - часова діаграма ALU

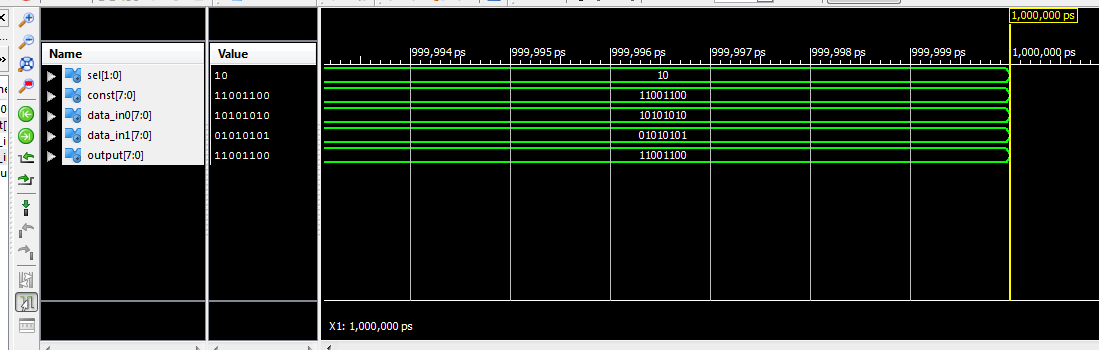


Рис. 5 - часова діаграма MUX

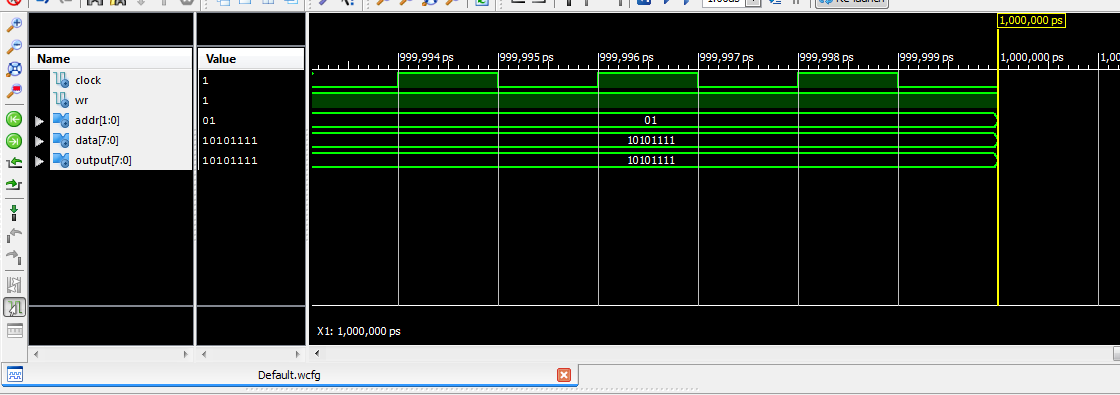


Рис. 6 - часова діаграма RAM

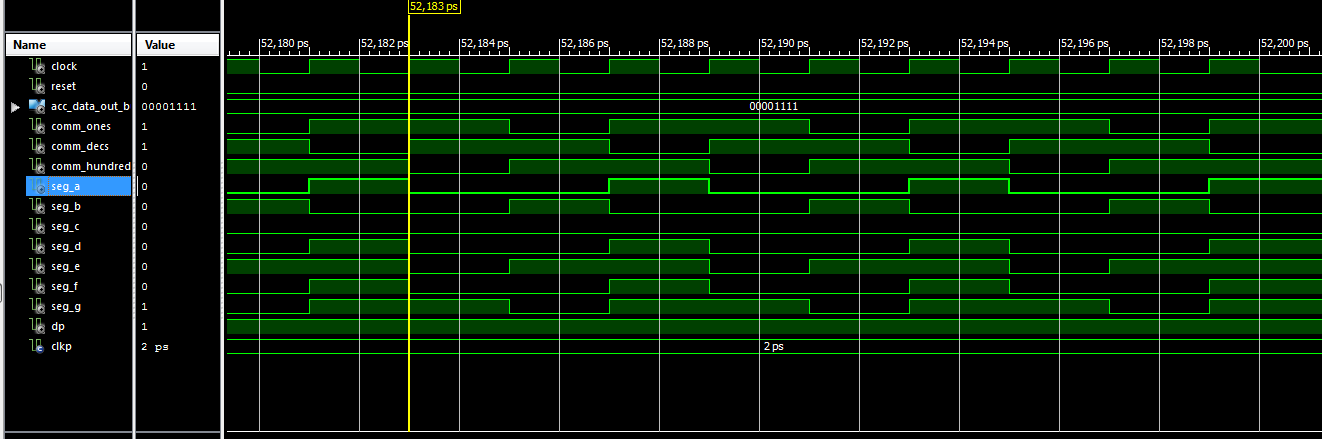


Рис. 7 - часова діаграма SEG\_DECODER

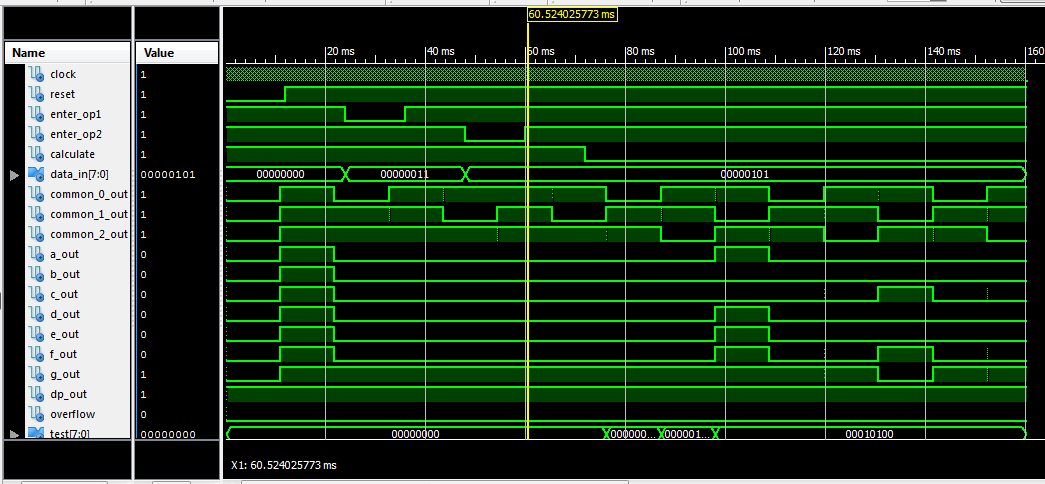


Рис. 8 - часова діаграма TopLevel

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 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; |
| --- |

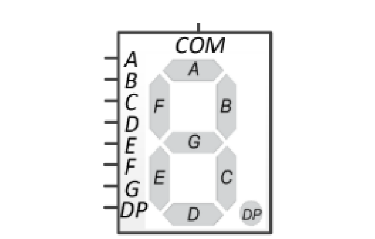


Рис. 9 - семисегментний індикатор

Вхідне число: 02010 = 101002

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.