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** 3**A FPGA.**»

Варіант № 2

Виконав:
ст.гр. КІ-202
Бажулін С.В.
Перевірив:
Козак Н.Б

Львів 2023

Мета роботи: на базі стенда **Elbert V2 – Spartan** 3**A FPGA** реалізувати цифровий автомат для обчислення значення виразу згідно вимог.

Завдання згідно з варіантом:

$$((OP1 + 2) * OP2) << OP1$$

Виконання роботи:

Код MUX:

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

entity MUX_intf is
port(
        DATA_IN         : in  std_logic_vector(7 downto 0);
        IN_SEL          : in  std_logic_vector(1 downto 0);
        CONSTANT_BUS    : in  std_logic_vector(7 downto 0);
        RAM_DATA_OUT_BUS : in  std_logic_vector(7 downto 0);
        IN_SEL_OUT_BUS   : out std_logic_vector(7 downto 0)
        );
end MUX_intf;

architecture MUX_arch of MUX_intf is

begin
  INSEL_A_MUX : process(DATA_IN, CONSTANT_BUS,  RAM_DATA_OUT_BUS,  IN_SEL)
        begin
                if(IN_SEL = "00") then
                        IN_SEL_OUT_BUS <= DATA_IN;
                elsif(IN_SEL = "01") then
                        IN_SEL_OUT_BUS <= RAM_DATA_OUT_BUS;
                else
                        IN_SEL_OUT_BUS <= CONSTANT_BUS;
                end if;
        end process INSEL_A_MUX;
end MUX_arch;
```

Код ACC:

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


entity ACC_intf is
port(
        CLOCK           : in  std_logic;
        ACC_WR          : in  std_logic;
        ACC_RST         : in  std_logic;
        ACC_DATA_IN_BUS  : in  std_logic_vector(7 downto 0);
        ACC_DATA_OUT_BUS  : out std_logic_vector(7 downto 0)
);
end ACC_intf;

architecture ACC_arch of ACC_intf is

signal ACC_DATA  : std_logic_vector(7 downto 0);

begin

ACC : process(CLOCK, ACC_DATA)
        begin
                if (rising_edge(CLOCK))  then
                        if(ACC_RST = '1') then
```

```
                                        ACC_DATA <= "00000000";
                                elsif (ACC_WR = '1') then
                                        ACC_DATA <= ACC_DATA_IN_BUS;
                                end if;
                        end if;
                        ACC_DATA_OUT_BUS <= ACC_DATA;
                end process ACC;

end ACC_arch;
```

Код ALU:

```
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_intf is
port(
                        IN_SEL_OUT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
                        ACC_DATA_OUT_BUS : IN STD_LOGIC_VECTOR(7 downto 0);
                        OP_CODE_BUS : IN STD_LOGIC_VECTOR(1 downto 0);
                        RESET : IN STD_LOGIC;
                        ACC_DATA_IN_BUS : OUT STD_LOGIC_VECTOR(7 downto 0);
                        OVERFLOW : OUT STD_LOGIC := '0'
                        --OF - overflow
                        );
end ALU_intf;

architecture ALU_arch of ALU_intf is

begin
ALU : process(OP_CODE_BUS, IN_SEL_OUT_BUS, ACC_DATA_OUT_BUS)
                        variable A : unsigned(7 downto 0);
                        variable B : unsigned(7 downto 0);
                begin
                        A := unsigned(ACC_DATA_OUT_BUS);
                        B := unsigned(IN_SEL_OUT_BUS);

                        if(RESET = '1')then
                                OVERFLOW <= '0';
                        end if;

                        case(OP_CODE_BUS) is
                                when "00"               => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(B);
                                when "01"               => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A + B);
                                 if (A > "11111101") then
          OVERFLOW <= '1';
        end if;
                                when "10"               => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(RESIZE(unsigned(A*B(7 downto 0)),
8));
                                when "11"               =>
                                        case(B) is --case(B) is
                                                when x"00"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 0);
                                                when x"01"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 1);
                                                when x"02"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 2);
                                                when x"03"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 3);
                                                when x"04"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 4);
                                                when x"05"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 5);
                                                when x"06"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 6);
                                                when x"07"              => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 7);
                                                when others => ACC_DATA_IN_BUS <= STD_LOGIC_VECTOR(A sll 0);
                                        end case;
                                when others => ACC_DATA_IN_BUS <= "00000000";
                        end case;
                end process ALU;

end ALU_arch;
```

Код CU:

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    16:27:31 04/27/2023
-- Design Name:
-- Module Name:    CU - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
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 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);
             CONSTANT_BUS : OUT STD_LOGIC_VECTOR(7 downto 0):= "00000010";
             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
CONSTANT_BUS    <= "00000010";
CU_SYNC_PROC: 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;

        CUNEXT_STATE_DECODE: process (cu_cur_state, ENTER_OP1, ENTER_OP2, CALCULATE)
  begin
    --declare default state for next_state to avoid latches
    cu_next_state <= cu_cur_state;  --default is to stay in current state
    --insert statements to decode next_state
    --below is a simple example
                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;


CU_OUTPUT_DECODE:  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                          <= "10";
                        OP_CODE_BUS     <= "01";
                        RAM_ADDR_BUS    <= "00";

                        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;
end CU_arch;
```

Код RAM:

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

entity RAM_intf is
port(
CLOCK           : in  std_logic;
RAM_WR          : in  std_logic;
RAM_ADDR_BUS    : in  STD_LOGIC_VECTOR(1  downto 0);
RAM_DATA_IN_BUS  : in  STD_LOGIC_VECTOR(7  downto 0);
RAM_DATA_OUT_BUS  : 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
--when reset will init const
RAM : process(CLOCK, RAM_ADDR_BUS, RAM_UNIT)
        begin
                if (rising_edge(CLOCK))  then
                        if (RAM_WR  = '1') then
                                RAM_UNIT(conv_integer(RAM_ADDR_BUS))  <= RAM_DATA_IN_BUS;
                        end if;
                end if;
                RAM_DATA_OUT_BUS  <= RAM_UNIT(conv_integer(RAM_ADDR_BUS));
        end process RAM;


end RAM_arch;
```

Код SEVEN_SEG_DECODER:

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

entity SEVEN_SEG_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 SEVEN_SEG_DECODER_intf;

architecture SEVEN_SEG_DECODER_arch  of SEVEN_SEG_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) ; -- 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       <= 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 SEVEN_SEG_DECODER_arch;
```

Згенерував Schematic файли для реалізованих елементів.
Створив Schematic файл TopLevel, виконав в ньому інтеграцію компонентів системи між собою. Перевірив роботу системи за допомогою симулятора ISim.

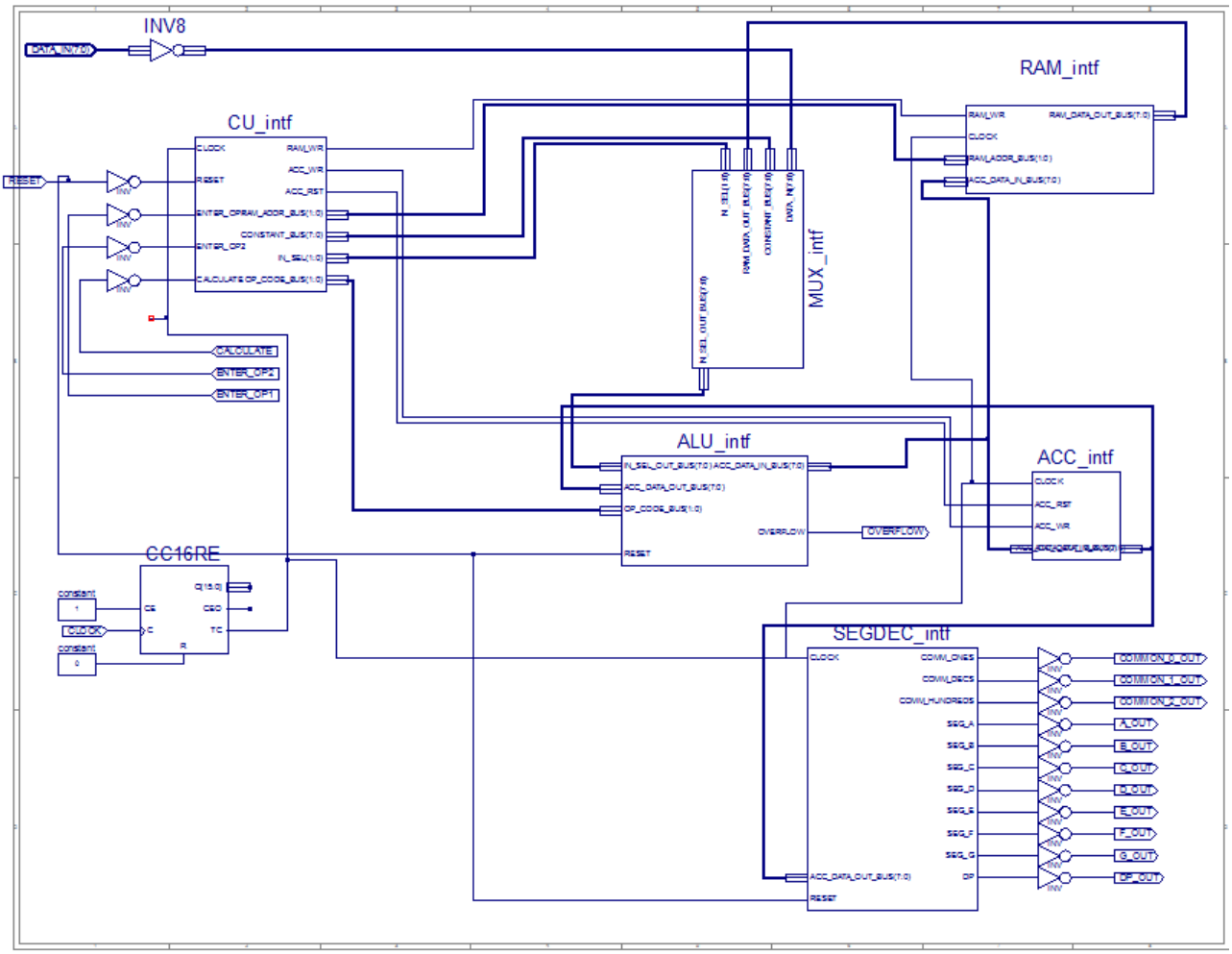

Рис. 7. Схема TopLevel

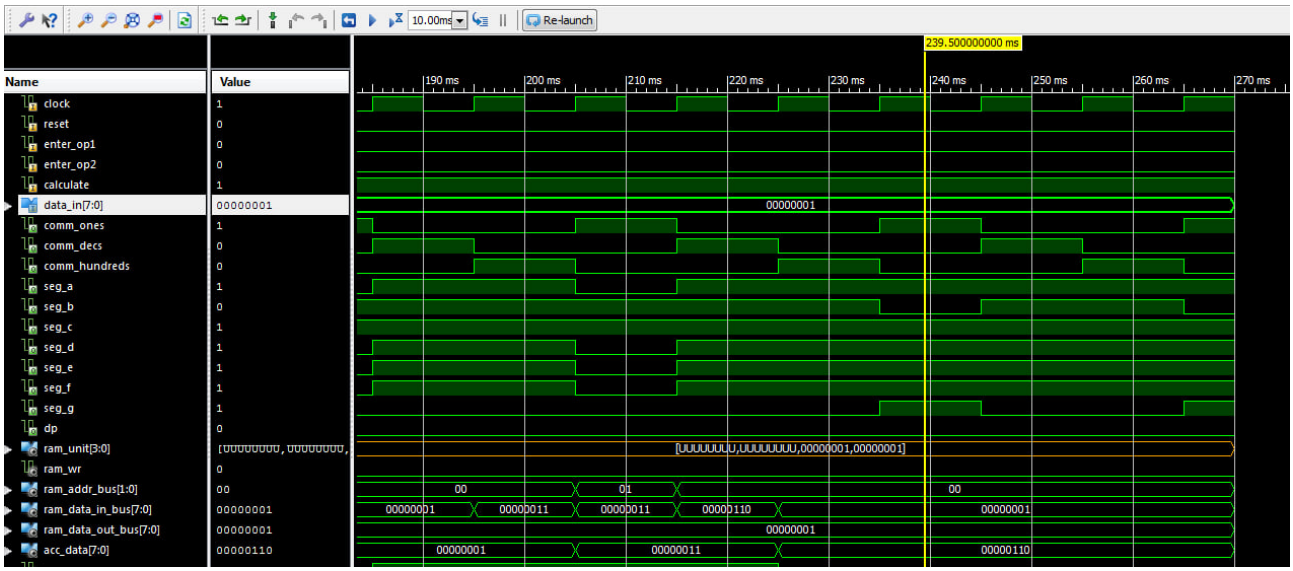

Рис. 8. Діаграма проведеної симуляції для TopLevel

Створив Constraints файл, зв’язав в ньому виводи схеми та фізичні виводи плати.

Вміст Constraints.ucf:

```
#**************************************************************************************************************************
**************************#
#                                UCF for Elbert V2 Development Board                                                      #
#**************************************************************************************************************************
**************************#
CONFIG VCCAUX = "3.3" ;

 # Clock 12 MHz
 NET "CLOCK"               LOC = P129  | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;

##########################################################################################################
#                          Seven Segment Display
##########################################################################################################

   NET "SEG_A"    LOC = P117  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_B"    LOC = P116  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_C"    LOC = P115  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_D"    LOC = P113  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_E"    LOC = P112  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_F"    LOC = P111  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "SEG_G"    LOC = P110  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DP"           LOC = P114  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

   NET "COMM_HUNDREDS"       LOC = P124  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "COMM_DECS"        LOC = P121  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "COMM_ONES"        LOC = P120  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
##########################################################################################################
#                          DP Switches
##########################################################################################################

   NET "DATA(0)"      LOC = P70   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(1)"      LOC = P69   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(2)"      LOC = P68   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(3)"      LOC = P64   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(4)"      LOC = P63   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(5)"      LOC = P60   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(6)"      LOC = P59   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "DATA(7)"      LOC = P58   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##########################################################################################################
#                          Switches
##########################################################################################################

   NET "ENTER_OP1"       LOC = P80   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "ENTER_OP2"       LOC = P79   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "CALCULATE"          LOC = P78   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
   NET "RESET"              LOC = P75   | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

##########################################################################################################
#                          LED
##########################################################################################################

   NET "OVERFLOW"          LOC = P46   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

Згенерував бінарний файл для розробленого цифрового автомата.

Висновок: виконавши лабораторну роботу, здобуто навики реалізації цифрових автоматів для обчислення значення заданого виразу.