HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING

[DIGITAL SYSTEM]

Instructor/ Lecturer: Ms. Phan Vo Kim Anh

Class/Group: TT04

Name:	**Student ID:**
Nguyễn Vũ Thanh Tùng	2051024
Nguyễn Doãn Khải	2010332

REPORT LAB 6

PROJECT

A/ VHDL CODE FOR LAB 6:

I) VHDL code describing simple elements in the circuit:

1.REGISTER:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY regn IS
generic (n: natural:= 9);
PORT (      D : IN STD_LOGIC_VECTOR(n-1 downto 0);
                  Clk, Reset, Load :IN STD_LOGIC;
                  Q : OUT STD_LOGIC_VECTOR(n-1 downto 0));
END regn;
ARCHITECTURE behavioral OF regn IS
BEGIN
PROCESS (Clk, Reset)
BEGIN
      IF (Reset = '0') THEN
            Q <= (others => '0');
      ELSIF Load = '1' AND rising_edge(Clk) THEN
            Q <= D;
      END IF;
END PROCESS;
END behavioral;
```

2.7 SEGMENT HEX:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY my_7seghex IS
PORT (      C: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            Z : OUT STD_LOGIC_VECTOR(0 TO 6));
END my_7seghex;

ARCHITECTURE dataflow OF my_7seghex IS
BEGIN
With C select
Z <=  NOT "1111110" when "0000",
            NOT "0110000" when "0001",
            NOT "1101101" when "0010",
            NOT "1111001" when "0011",
            NOT "0110011" when "0100",
            NOT "1011011" when "0101",
            NOT "1011111" when "0110",
            NOT "1110000" when "0111",
            NOT "1111111" when "1000",
            NOT "1111011" when "1001",
            NOT "1110111" when "1010",
            NOT "0011111" when "1011",
            NOT "1001110" when "1100",
            NOT "0111101" when "1101",
            NOT "1001111" when "1110",
            NOT "1000111" when "1111",
            NOT "0000000" when OTHERS;
END dataflow;
```

3.D FLIP FLOP POSITIVE EDGE:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mineDFFpositive_edge IS
PORT (D, Clk, RESET : IN STD_LOGIC;
            Q : OUT STD_LOGIC);
END mineDFFpositive_edge;

ARCHITECTURE behavior OF mineDFFpositive_edge IS
BEGIN
PROCESS (Clk, Reset)
BEGIN
      IF (Reset = '0') THEN
            Q <= '0';
      ELSIF rising_edge(Clk) THEN
            Q <= D;
      END IF;
END PROCESS;
END behavior;
```

4.DECODER 3 TO 8:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY dec3to8 IS
PORT (      W : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                  En : IN STD_LOGIC;
                  Y : OUT STD_LOGIC_VECTOR(0 TO 7));
END dec3to8;
ARCHITECTURE Behavior OF dec3to8 IS
BEGIN
PROCESS (W, En)
BEGIN
IF En = '1' THEN
      CASE W IS
            WHEN "000" => Y <= "10000000";
            WHEN "001" => Y <= "01000000";
            WHEN "010" => Y <= "00100000";
            WHEN "011" => Y <= "00010000";
            WHEN "100" => Y <= "00001000";
            WHEN "101" => Y <= "00000100";
            WHEN "110" => Y <= "00000010";
            wHEN "111" => Y <= "00000001";
      END CASE;
ELSE
            Y <= "00000000";
END IF;
END PROCESS;
END Behavior;
```

5.10 TO 1 MULTIPLEXER:

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
ENTITY mine10to1mux IS
PORT ( Din, R0, R1 , R2, R3, R4, R5, R6, R7, Gout : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
            sel : IN STD_LOGIC_VECTOR(0 TO 9);
            Outmux : OUT STD_LOGIC_VECTOR(8 DOWNTO 0));
```

```
END mine10to1mux;
architecture structural of mine10to1mux is

BEGIN
PROCESS (sel)
BEGIN
IF sel = "1000000000" THEN
      Outmux <= R0;
ELSIF sel = "0100000000" THEN
      Outmux <= R1;
ELSIF sel = "0010000000" THEN
      Outmux <= R2;
ELSIF sel = "0001000000" THEN
      Outmux <= R3;
ELSIF sel = "0000100000" THEN
      Outmux <= R4;
ELSIF sel = "0000010000" THEN
      Outmux <= R5;
ELSIF sel = "0000001000" THEN
      Outmux <= R6;
ELSIF sel = "0000000100" THEN
      Outmux <= R7;
ELSIF sel = "0000000010" THEN
      Outmux <= Gout;
ELSE
      Outmux <= Din;
END IF;
END PROCESS;
end structural;
```

6.PC:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;

ENTITY PC IS
PORT (      D: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                  CLK, Reset, E, L: IN STD_LOGIC;
                  Q: OUT STD_LOGIC_VECTOR(8 downto 0));
END PC;
ARCHITECTURE Behavioral OF PC IS

BEGIN
PROCESS(CLK, Reset)
VARIABLE count: STD_LOGIC_VECTOR (8 downto 0);
BEGIN
IF (Reset = '0') THEN
      Count := (others => '0');
ELSIF rising_edge(CLK) THEN
      IF E = '1' THEN
            count := count + "000000001";
      ELSIF L = '1' THEN
            count := D;
      END IF;
            Q <= count;
END IF;
END PROCESS;
END Behavioral;
```

7.MEMORY:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity memory is
generic(
            addr_width : integer := 128;
            addr_bits : integer := 7;
            data_width : integer := 9
);
port( DATA_IN : in std_logic_vector(data_width-1 downto 0);
            ADDR : in std_logic_vector(addr_bits-1 downto 0);
            Clk : in std_logic;
            Write_EN : in std_logic;
            DATA_OUT : out std_logic_vector(data_width-1 downto 0)
);
end memory;

architecture arch of memory is
type ram_type is array (0 to addr_width-1) of std_logic_vector(data_width-1 downto 0);
signal user_RAM : ram_type;
attribute ram_init_file : string;
attribute ram_init_file of user_RAM : signal is "ram_data.mif";
begin
process(Clk, Write_EN)
begin
if Rising_edge(Clk) then
      if Write_EN = '1' then
            user_RAM(to_integer(unsigned(ADDR))) <= DATA_IN;
      end if;
end if;
end process;
      DATA_OUT   <=   user_RAM   (to_integer(unsigned(ADDR)))   WHEN   (Write_EN   =   '0')
ELSE"000000000";
end arch;
```

II) VHDL code for complex components:

1.FSM CONTROL UNIT:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
ENTITY FSM_controlunit IS
PORT (      clk, Reset, Run: IN std_logic;
                  I, G: IN std_logic_vector(8 DOWNTO 0);

                  G_O, Din_O: OUT std_logic;
                  R_O: OUT std_logic_vector(0 TO 7);

                  IR_I, ADD_SUB, A_I, G_I: OUT std_logic;
                  ADDR_I, DOUT_I: OUT std_logic;
                  R_I : OUT STD_LOGIC_VECTOR(0 TO 7);
                  Done, incr_PC, W_D: BUFFER  std_logic);
END FSM_controlunit;
ARCHITECTURE behavior OF FSM_controlunit IS
COMPONENT dec3to8
PORT (      W : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                  En : IN STD_LOGIC;
                  Y : OUT STD_LOGIC_VECTOR(0 TO 7));
END COMPONENT;
```

```
      TYPE state IS (IWAIT, INITIAL, MV, MVI1, MVI2, MVI3, ADD1, ADD2, ADD3, SUB1, SUB2,
SUB3, LD1, LD2, ST1, ST2, MVNZ);
      SIGNAL pr_state, nx_state: state;
      SIGNAL RX_TEMP, RY_TEMP, R7_TEMP: STD_LOGIC_VECTOR (0 TO 7);
      SIGNAL clear: STD_LOGIC;
BEGIN
clear <= reset or done;
cst_pr: PROCESS(clk, clear)
      BEGIN
            IF (clear = '0') THEN
                  pr_state <= IWAIT;
            ELSIF (rising_edge(clk)) THEN
                  pr_state <= nx_state;
            END IF;
END PROCESS cst_pr;

nxt_pr: PROCESS (RUN, I(8 DOWNTO 6), pr_state, Done, G)
BEGIN
      CASE pr_state IS
            WHEN IWAIT =>
                  IF RUN = '1' THEN
                  nx_state <= IWAIT;
                  ELSIF (RUN = '0') THEN
                  nx_state <= INITIAL;
                  END IF;

            WHEN INITIAL =>
                  IF (I(8 DOWNTO 6) ="000") THEN
                  nx_state <= MV;
                  ELSIF (I(8 DOWNTO 6) ="001") THEN
                  nx_state <= MVI1;
                  ELSIF (I(8 DOWNTO 6) ="010") THEN
                  nx_state <= ADD1;
                  ELSIF (I(8 DOWNTO 6) ="011") THEN
                  nx_state <= SUB1;
                  ELSIF (I(8 DOWNTO 6) ="100") THEN
                  nx_state <= LD1;
                  ELSIF (I(8 DOWNTO 6) ="101") THEN
                  nx_state <= ST1;
                  ELSIF (I(8 DOWNTO 6) ="110") THEN
                  nx_state <= MVNZ;
                  END IF;

            WHEN MV =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= MV;
            END IF;

            WHEN MVI1 =>
                        nx_state <= MVI2;
            WHEN MVI2 =>
                        nx_state <= MVI3;
            WHEN MVI3 =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= MVI3;
            END IF;

            WHEN ADD1 =>
                  nx_state <= ADD2;
```

```
            WHEN ADD2 =>
                  nx_state <= ADD3;
            WHEN ADD3 =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= ADD3;
            END IF;

            WHEN SUB1 =>
                  nx_state <= SUB2;
            WHEN SUB2 =>
                  nx_state <= SUB3;
            WHEN SUB3 =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= SUB3;
            END IF;

            WHEN LD1 =>
                  nx_state <= LD2;
            WHEN LD2 =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= LD2;
            END IF;

            WHEN ST1 =>
                  nx_state <= ST2;
            WHEN ST2 =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= sT2;
            END IF;

            WHEN MVNZ =>
                  IF (Done = '1') THEN
                        nx_state <= IWAIT;
                  ElSE
                     nx_state <= MVNZ;
            END IF;

            WHEN OTHERS =>
                  nx_state <= IWAIT;
            END CASE;
END PROCESS nxt_pr;

Idec3to8_X: dec3to8 PORT MAP ( I(5 DOWNTO 3), '1', RX_TEMP);
Idec3to8_Y: dec3to8 PORT MAP ( I(2 DOWNTO 0), '1', RY_TEMP);
Idec3to8_7: dec3to8 PORT MAP ( "111", '1', R7_TEMP);

out_pr: PROCESS(pr_state)
BEGIN
      CASE pr_state IS
            WHEN IWAIT =>
            IR_I <= '1';
            R_I <= "00000000";
            A_I <= '0';
            G_I <= '0';
            R_O <= R7_TEMP;
```

```
            G_O <= '0';
            Din_O <= '0';
            DONE <= '0';
            ADDR_I <= '1';
            DOUT_I <= '0';
            W_D <= '0';
            incr_PC <= '0';

            WHEN INITIAL =>
            IR_I <= '0';
            R_I <= "00000000";
            A_I <= '0';
            G_I <= '0';
            R_O <= "00000000";
            G_O <= '0';
            Din_O <= '0';
            DONE <= '0';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            incr_PC <= '0';

            WHEN MV =>
            IR_I <= '0';
            R_I <= RX_TEMP;
            A_I <= '0';
            G_I <= '0';
            R_O <= RY_TEMP;
            G_O <= '0';
            Din_O <= '0';
            DONE <= '1';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            IF (RX_TEMP = "00000001") THEN
                  incr_PC <= '0';
            ElSE
                  incr_PC <= '1';
      END IF;

            WHEN MVI1 =>
            IR_I <= '0';
            R_I <= "00000000";
         A_I <= '0';
            G_I <= '0';
            R_O <= "00000000";
            G_O <= '0';
            Din_O <= '0';
            DONE <= '0';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            incr_PC <= '1';
            WHEN MVI2 =>
            IR_I <= '0';
            R_I <= "00000000";
         A_I <= '0';
            G_I <= '0';
            R_O <= R7_TEMP;
            G_O <= '0';
            Din_O <= '0';
            DONE <= '0';
            ADDR_I <= '1';
```

```
            DOUT_I <= '0';
            W_D <= '0';
            incr_PC <= '0';
            WHEN MVI3 =>
            IR_I <= '0';
            R_I <= RX_TEMP;
         A_I <= '0';
            G_I <= '0';
            R_O <= "00000000";
            G_O <= '0';
            Din_O <= '1';
            DONE <= '1';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            IF (RX_TEMP = "00000001") THEN
                  incr_PC <= '0';
            ElSE
                  incr_PC <= '1';
      END IF;

            WHEN ADD1 =>
            IR_I <= '0';
         R_I <= "00000000";
            A_I <= '1';
            G_I <= '0';
            R_O <= RX_TEMP;
            G_O <= '0';
            Din_O <= '0';
            DONE <= '0';
      ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
      incr_PC <= '0';
            WHEN ADD2 =>
            IR_I <= '0';
            R_I <= "00000000";
            A_I <= '0';
            G_I <= '1';
            R_O <= RY_TEMP;
            Din_O <= '0';
            G_O <= '0';
            DONE <= '0';
            ADD_SUB <= '0';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
      incr_PC <= '0';
            WHEN ADD3 =>
            IR_I <= '0';
            R_I <= RX_TEMP;
            A_I <= '0';
            G_I <= '0';
            R_O <= "00000000";
            Din_O <= '0';
            G_O <= '1';
            DONE <= '1';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            IF (RX_TEMP = "00000001") THEN
                  incr_PC <= '0';
            ElSE
```

```
                incr_PC <= '1';
    END IF;

          WHEN SUB1 =>
          IR_I <= '0';
       R_I <= "00000000";
          A_I <= '1';
          G_I <= '0';
          R_O <= RX_TEMP;
          G_O <= '0';
          Din_O <= '0';
          DONE <= '0';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
    incr_PC <= '0';
          WHEN SUB2 =>
          IR_I <= '0';
          R_I <= "00000000";
          A_I <= '0';
          G_I <= '1';
          R_O <= RY_TEMP;
          Din_O <= '0';
          G_O <= '0';
          DONE <= '0';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
    incr_PC <= '0';
          ADD_SUB <= '1';
          WHEN SUB3 =>
          IR_I <= '0';
          R_I <= RX_TEMP;
          A_I <= '0';
          G_I <= '0';
          R_O <= "00000000";
          Din_O <= '0';
          G_O <= '1';
          DONE <= '1';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
          IF (RX_TEMP = "00000001") THEN
                incr_PC <= '0';
          ElSE
                incr_PC <= '1';
    END IF;

          WHEN LD1 =>
          IR_I <= '0';
          R_I <= "00000000";
          A_I <= '0';
          G_I <= '0';
          R_O <= RY_TEMP;
          Din_O <= '0';
          G_O <= '0';
          DONE <= '0';
          ADDR_I <= '1';
          DOUT_I <= '0';
          W_D <= '0';
          incr_PC <= '0';
          WHEN LD2 =>
          IR_I <= '0';
```

```
            R_I <= RX_TEMP;
            A_I <= '0';
            G_I <= '0';
            R_O <= "00000000";
            Din_O <= '1';
            G_O <= '0';
            DONE <= '1';
            ADDR_I <= '0';
            DOUT_I <= '0';
            W_D <= '0';
            IF (RX_TEMP = "00000001") THEN
                  incr_PC <= '0';
            ElSE
                  incr_PC <= '1';
      END IF;

            WHEN ST1 =>
            IR_I <= '0';
            R_I <= "00000000";
            A_I <= '0';
            G_I <= '0';
            R_O <= RX_TEMP;
            Din_O <= '0';
            G_O <= '0';
            DONE <= '0';
            ADDR_I <= '0';
            DOUT_I <= '1';
            W_D <= '0';
            incr_PC <= '0';
            WHEN ST2 =>
            IR_I <= '0';
            R_I <= "00000000";
            A_I <= '1';
            G_I <= '0';
            R_O <= RY_TEMP;
            Din_O <= '0';
            G_O <= '0';
            DONE <= '1';
            ADDR_I <= '1';
            DOUT_I <= '0';
            W_D <= '1';
            incr_PC <= '1';

            WHEN MVNZ =>
            IF (G = "00000000") THEN
                  IR_I <= '0';
                  R_I <= "00000000";
                  A_I <= '0';
                  G_I <= '0';
                  R_O <= "00000000";
                  G_O <= '0';
                  Din_O <= '0';
                  DONE <= '1';
                  ADDR_I <= '0';
                  DOUT_I <= '0';
                  W_D <= '0';
                  incr_PC <= '1';
            ElSE
                  IR_I <= '0';
                  R_I <= RX_TEMP;
                  A_I <= '0';
                  G_I <= '0';
                  R_O <= RY_TEMP;
```

```
                  G_O <= '0';
                  Din_O <= '0';
                  DONE <= '1';
                  ADDR_I <= '0';
                  DOUT_I <= '0';
                  W_D <= '0';
                  IF (RX_TEMP = "00000001") THEN
                        incr_PC <= '0';
                  ElSE
                        incr_PC <= '1';
                  END IF;
      END IF;

            WHEN OTHERS =>
            DONE <= '0';
      END CASE;
END PROCESS out_pr;
END behavior;
```

2.PROCESSOR:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;
USE ieee.numeric_std.all;
ENTITY Processor IS
PORT (            clk, Reset, Run: IN std_logic;
                        DIN : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                        HEX4, HEX5: OUT STD_LOGIC_VECTOR(0 TO 6);
                        Done: BUFFER STD_LOGIC;
                        W: BUFFER STD_LOGIC;
                        ADDR, DOUT: BUFFER STD_LOGIC_VECTOR(8 DOWNTO 0)
                        );
END Processor;
ARCHITECTURE Behavior OF Processor IS
COMPONENT FSM_controlunit
PORT (      clk, Reset, Run: IN std_logic;
                  I, G: IN std_logic_vector(8 DOWNTO 0);

                  G_O, Din_O: OUT std_logic;
                  R_O: OUT std_logic_vector(0 TO 7);

                  IR_I, ADD_SUB, A_I, G_I: OUT std_logic;
                  ADDR_I, DOUT_I: OUT std_logic;
                  R_I : OUT STD_LOGIC_VECTOR(0 TO 7);
                  Done, incr_PC, W_D: BUFFER  std_logic);
END COMPONENT;

COMPONENT regn
generic (n: natural:= 9);
PORT (      D : IN STD_LOGIC_VECTOR(n-1 downto 0);
                  Clk, Reset, Load :IN STD_LOGIC;
                  Q : OUT STD_LOGIC_VECTOR(n-1 downto 0));
END COMPONENT;

COMPONENT PC IS
PORT (      D: IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                  CLK, RESET, E, L: IN STD_LOGIC;
                  Q: OUT STD_LOGIC_VECTOR(8 downto 0));
END COMPONENT;

COMPONENT mineDFFpositive_edge IS
```

```
PORT (D, Clk, RESET : IN STD_LOGIC;
            Q : OUT STD_LOGIC);
END COMPONENT;

COMPONENT mine10to1mux
PORT ( Din, R0, R1 , R2, R3, R4, R5, R6, R7, Gout : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
            sel : IN STD_LOGIC_VECTOR(0 TO 9);
            Outmux : OUT STD_LOGIC_VECTOR(8 DOWNTO 0));
END COMPONENT;

COMPONENT my_7seghex IS
PORT (      C: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                  Z : OUT STD_LOGIC_VECTOR(0 TO 6));
END COMPONENT;

SIGNAL R_O, R_I: STD_LOGIC_VECTOR(0 TO 7);
SIGNAL G_O, Din_O, IR_I, ADD_SUB, A_I, G_I, E, L, ADDR_I, DOUT_I, W_D, incr_PC: STD_LOGIC;
SIGNAL  R_OUT_0,  R_OUT_1,  R_OUT_2,  R_OUT_3,  R_OUT_4,  R_OUT_5,  R_OUT_6,  PC_OUT  :
STD_LOGIC_VECTOR(8 DOWNTO 0);
SIGNAL A_OUT, ADD_SUB_RES, G_OUT, DIN_IR, BusWires: STD_LOGIC_VECTOR(8 DOWNTO 0);
SIGNAL COUT : STD_LOGIC;
SIGNAL Sel : STD_LOGIC_VECTOR(0 to 9);
--SIGNAL ADDR, DOUT: STD_LOGIC_VECTOR(8 DOWNTO 0);
BEGIN

iFSM_controlunit: FSM_controlunit PORT MAP(clk => clk,
Reset => Reset,
Run => Run,
I(8 DOWNTO 0) => DIN_IR(8 DOWNTO 0),
G(8 DOWNTO 0) => G_OUT(8 DOWNTO 0),
G_O => G_O,
Din_O => Din_O,
R_O(0 TO 7) => R_O(0 TO 7),
IR_I => IR_I,
ADD_SUB => ADD_SUB,
A_I => A_I,
G_I => G_I,
ADDR_I => ADDR_I,
DOUT_I => DOUT_I,
R_I(0 TO 7) => R_I(0 TO 7),
Done => Done,
incr_PC => incr_PC,
W_D => W_D);

iregn_0: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(0), R_OUT_0(8 DOWNTO 0));
iregn_1: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(1), R_OUT_1(8 DOWNTO 0));
iregn_2: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(2), R_OUT_2(8 DOWNTO 0));
iregn_3: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(3), R_OUT_3(8 DOWNTO 0));
iregn_4: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(4), R_OUT_4(8 DOWNTO 0));
iregn_5: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(5), R_OUT_5(8 DOWNTO 0));
iregn_6: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, R_I(6), R_OUT_6(8 DOWNTO 0));
iPC: PC PORT MAP(BusWires(8 DOWNTO 0), Clk, RESET, incr_PC, R_I(7), PC_OUT(8 DOWNTO 0));
iregn_A: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, A_I, A_OUT(8 DOWNTO 0));
iregn_G: regn PORT MAP(ADD_SUB_RES(8 DOWNTO 0), Clk, Reset, G_I, G_OUT(8 DOWNTO 0));
iregn_IR: regn PORT MAP(DIN(8 DOWNTO 0), Clk, Reset, IR_I, DIN_IR(8 DOWNTO 0));
iregn_ADDR: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, ADDR_I, ADDR(8 DOWNTO 0));
iregn_DOUT: regn PORT MAP(BusWires(8 DOWNTO 0), Clk, Reset, DOUT_I, DOUT(8 DOWNTO 0));
iregn_WD: mineDFFpositive_edge PORT MAP(W_D, clk, Reset,W);

Sel <= R_O & G_O & Din_O;
Imine10to1mux: mine10to1mux PORT MAP(DIN(8 DOWNTO 0) => DIN(8 DOWNTO 0),
 R0(8 DOWNTO 0) => R_OUT_0(8 DOWNTO 0),
 R1(8 DOWNTO 0) => R_OUT_1(8 DOWNTO 0),
```

```
 R2(8 DOWNTO 0) => R_OUT_2(8 DOWNTO 0),
 R3(8 DOWNTO 0) => R_OUT_3(8 DOWNTO 0),
 R4(8 DOWNTO 0) => R_OUT_4(8 DOWNTO 0),
 R5(8 DOWNTO 0) => R_OUT_5(8 DOWNTO 0),
 R6(8 DOWNTO 0) => R_OUT_6(8 DOWNTO 0),
 R7(8 DOWNTO 0) => PC_OUT(8 DOWNTO 0),
 Gout(8 DOWNTO 0) => G_OUT(8 DOWNTO 0),
 sel(0 TO 9) => sel (0 TO 9),
 Outmux(8 DOWNTO 0) => BusWires(8 DOWNTO 0));

alu: PROCESS (ADD_SUB, A_OUT, BusWires)
BEGIN
      IF ADD_SUB = '0' THEN
            ADD_SUB_RES <= A_OUT + BusWires;
      ELSE
            ADD_SUB_RES <= A_OUT - BusWires;
END IF;
END PROCESS;

imy_7seghex_4: my_7seghex PORT MAP (R_OUT_4(3 DOWNTO 0), HEX4(0 TO 6));
imy_7seghex_5: my_7seghex PORT MAP (R_OUT_5(3 DOWNTO 0), HEX5(0 TO 6));
END Behavior;
```

3.OVERALL CODE FOR LAB 6:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_signed.all;
USE ieee.numeric_std.all;
ENTITY lab6 IS
PORT (            clk, Reset, Run: IN std_logic;
                        Done: BUFFER STD_LOGIC;
                        HEX0, HEX1, HEX2, HEX3, HEX4, HEX5: OUT STD_LOGIC_VECTOR(0 TO 6);
                        LEDs: OUT STD_LOGIC_VECTOR(8 DOWNTO 0));
END lab6;
ARCHITECTURE Behavior OF lab6 IS

COMPONENT Processor
PORT (            clk, Reset, Run: IN std_logic;
                        DIN : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
                        HEX4, HEX5: OUT STD_LOGIC_VECTOR(0 TO 6);
                        Done: BUFFER STD_LOGIC;
                        W: BUFFER STD_LOGIC;
                        ADDR, DOUT: BUFFER STD_LOGIC_VECTOR(8 DOWNTO 0)
                        );
END COMPONENT;

COMPONENT memory
generic(
            addr_width : integer := 128;
            addr_bits : integer := 7;
            data_width : integer := 9
);
port( DATA_IN : in std_logic_vector(data_width-1 downto 0);
            ADDR : in std_logic_vector(addr_bits-1 downto 0);
            Clk : in std_logic;
            Write_EN : in std_logic;
            DATA_OUT : out std_logic_vector(data_width-1 downto 0)
);
end COMPONENT;
```

```
COMPONENT regn
generic (n: natural:= 9);
PORT (      D : IN STD_LOGIC_VECTOR(n-1 downto 0);
                  Clk, Reset, Load :IN STD_LOGIC;
                  Q : OUT STD_LOGIC_VECTOR(n-1 downto 0));
END COMPONENT;

COMPONENT my_7seghex IS
PORT (      C: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                  Z : OUT STD_LOGIC_VECTOR(0 TO 6));
END COMPONENT;

SIGNAL WR_en, ADDR7xnoR8, ADDRn7xnoR8, W, WRen, E: STD_LOGIC;
SIGNAL ADDR, DOUT, Din, DATA_OUT, DATA_IN: STD_LOGIC_VECTOR(8 DOWNTO 0);
SIGNAL ADDR_mem: STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
DATA_IN <= DOUT;
Din <= DATA_OUT;
ADDR_mem <= ADDR(6 DOWNTO 0);

ADDR7xnoR8 <= ADDR(7) xnor ADDR(8);
ADDRn7xnoR8 <= NOT(ADDR(7)) xnor ADDR(8);
WR_en <= W AND ADDR7xnoR8;
E <= W AND ADDRn7xnoR8;

iProcessor: Processor PORT MAP(clk, Reset, Run, Din, HEX4, HEX5, Done, W, ADDR, DOUT);
imemory: memory PORT MAP(DATA_IN, ADDR_mem, Clk, WR_en, DATA_OUT);
iregn: regn PORT MAP(DOUT, Clk, reset, E, LEDs);
imy_7seghex_0: my_7seghex PORT MAP (Din(3 DOWNTO 0), HEX0(0 TO 6));
imy_7seghex_1: my_7seghex PORT MAP (Din(7 DOWNTO 4), HEX1(0 TO 6));
imy_7seghex_2: my_7seghex PORT MAP ("000" & Din(8), HEX2(0 TO 6));
imy_7seghex_3: my_7seghex PORT MAP (ADDR(3 DOWNTO 0), HEX3(0 TO 6));
END Behavior;
```

III) VHDL code for testing instructions:

1.TESTING MVI – MVNZ – ADD – SUB:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000000011;
     2    :   110101100;
     3    :   000101100;
     4    :   010100101;
     5    :   110101100;
     6    :   011100101;
     [7..127]  :   000000000;
END;
```

2.TEST LOAD INSTRUCTION:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000000011;
     2    :   001101000;
     3    :   000001010;
     4    :   100100101;
     [5..9]  :   000000000;
     10   :   000001111;
     [11..127]  :   000000000;
END;
```

3.TEST STORE INSTRUCTION:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000000011;
     2    :   001101000;
     3    :   000000101;
     4    :   101100101;
     [5..127]  :   000000000;
END;
```

4.TEST THE LEDS:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000000011;
     2    :   001101000;
     3    :   010000101;
     4    :   101100101;
     [5..127]  :   000000000;
END;
```

5.TEST MVI MV OF R7:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000001010;
     2    :   000111100;
     [3..9]  :   000000000;
     10   :   001111000;
     11   :   000001111;
     [12..14]  :   000000000;
     15   :   000001000;
     [16..127]  :   000000000;
END;
```

6.TEST MVNZ, ADD, SUB OF R7:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000001010;
     2    :   110111100;
     3    :   010100000;
     4    :   110111100;
     [5..6]  :   000000000;
     7    :   000001000;
     [8..9]  :   000000000;
     10   :   001101000;
     11   :   000000101;
     12   :   011111101;
     [13..127]  :   000000000;
END;
```

7.TEST LOAD INSTRUCTIONS OF R7:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001100000;
     1    :   000001010;
     2    :   100111100;
     [3..9]  :   000000000;
     10   :   000001111;
     [11..14]  :   000000000;
     15   :   000001000;
     [16..127]  :   000000000;
END;
```

8.TEST LOOP:

```
WIDTH=9;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
     0    :   001010000;
     1    :   000000001;
     2    :   001100000;
     3    :   000000011;
     4    :   000101111;
     5    :   011100010;
     6    :   110111101;
     [7..127]  :   000000000;
END;
```

9.STATE OUTPUT CODE:

```
WHEN IWAIT =>
          IR_I <= '1';
          R_I <= "00000000";
          A_I <= '0';
          G_I <= '0';
          R_O <= R7_TEMP;
          G_O <= '0';
          Din_O <= '0';
          DONE <= '0';
          ADDR_I <= '1';
          DOUT_I <= '0';
          W_D <= '0';
```

```
        incr_PC <= '0';

        WHEN INITIAL =>
        IR_I <= '0';
        R_I <= "00000000";
        A_I <= '0';
        G_I <= '0';
        R_O <= "00000000";
        G_O <= '0';
        Din_O <= '0';
        DONE <= '0';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        incr_PC <= '0';

        WHEN MV =>
        IR_I <= '0';
        R_I <= RX_TEMP;
        A_I <= '0';
        G_I <= '0';
        R_O <= RY_TEMP;
        G_O <= '0';
        Din_O <= '0';
        DONE <= '1';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        IF (RX_TEMP = "00000001") THEN
             incr_PC <= '0';
        ElSE
             incr_PC <= '1';
    END IF;

        WHEN MVI1 =>
        IR_I <= '0';
        R_I <= "00000000";
      A_I <= '0';
        G_I <= '0';
        R_O <= "00000000";
        G_O <= '0';
        Din_O <= '0';
        DONE <= '0';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        incr_PC <= '1';
        WHEN MVI2 =>
        IR_I <= '0';
        R_I <= "00000000";
      A_I <= '0';
        G_I <= '0';
```

```
        R_O <= R7_TEMP;
        G_O <= '0';
        Din_O <= '0';
        DONE <= '0';
        ADDR_I <= '1';
        DOUT_I <= '0';
        W_D <= '0';
        incr_PC <= '0';
        WHEN MVI3 =>
        IR_I <= '0';
        R_I <= RX_TEMP;
      A_I <= '0';
        G_I <= '0';
        R_O <= "00000000";
        G_O <= '0';
        Din_O <= '1';
        DONE <= '1';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        IF (RX_TEMP = "00000001") THEN
             incr_PC <= '0';
        ElSE
             incr_PC <= '1';
   END IF;

        WHEN ADD1 =>
        IR_I <= '0';
     R_I <= "00000000";
        A_I <= '1';
        G_I <= '0';
        R_O <= RX_TEMP;
        G_O <= '0';
        Din_O <= '0';
        DONE <= '0';
    ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
    incr_PC <= '0';
        WHEN ADD2 =>
        IR_I <= '0';
        R_I <= "00000000";
        A_I <= '0';
        G_I <= '1';
        R_O <= RY_TEMP;
        Din_O <= '0';
        G_O <= '0';
        DONE <= '0';
        ADD_SUB <= '0';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
```

```
      incr_PC <= '0';
          WHEN ADD3 =>
          IR_I <= '0';
          R_I <= RX_TEMP;
          A_I <= '0';
          G_I <= '0';
          R_O <= "00000000";
          Din_O <= '0';
          G_O <= '1';
          DONE <= '1';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
          IF (RX_TEMP = "00000001") THEN
               incr_PC <= '0';
          ElSE
               incr_PC <= '1';
     END IF;

          WHEN SUB1 =>
          IR_I <= '0';
        R_I <= "00000000";
          A_I <= '1';
          G_I <= '0';
          R_O <= RX_TEMP;
          G_O <= '0';
          Din_O <= '0';
          DONE <= '0';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
      incr_PC <= '0';
          WHEN SUB2 =>
          IR_I <= '0';
          R_I <= "00000000";
          A_I <= '0';
          G_I <= '1';
          R_O <= RY_TEMP;
          Din_O <= '0';
          G_O <= '0';
          DONE <= '0';
          ADDR_I <= '0';
          DOUT_I <= '0';
          W_D <= '0';
      incr_PC <= '0';
          ADD_SUB <= '1';
          WHEN SUB3 =>
          IR_I <= '0';
          R_I <= RX_TEMP;
          A_I <= '0';
          G_I <= '0';
          R_O <= "00000000";
```

```
        Din_O <= '0';
        G_O <= '1';
        DONE <= '1';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        IF (RX_TEMP = "00000001") THEN
             incr_PC <= '0';
        ElSE
             incr_PC <= '1';
    END IF;

        WHEN LD1 =>
        IR_I <= '0';
        R_I <= "00000000";
        A_I <= '0';
        G_I <= '0';
        R_O <= RY_TEMP;
        Din_O <= '0';
        G_O <= '0';
        DONE <= '0';
        ADDR_I <= '1';
        DOUT_I <= '0';
        W_D <= '0';
        incr_PC <= '0';
        WHEN LD2 =>
        IR_I <= '0';
        R_I <= RX_TEMP;
        A_I <= '0';
        G_I <= '0';
        R_O <= "00000000";
        Din_O <= '1';
        G_O <= '0';
        DONE <= '1';
        ADDR_I <= '0';
        DOUT_I <= '0';
        W_D <= '0';
        IF (RX_TEMP = "00000001") THEN
             incr_PC <= '0';
        ElSE
             incr_PC <= '1';
    END IF;

        WHEN ST1 =>
        IR_I <= '0';
        R_I <= "00000000";
        A_I <= '0';
        G_I <= '0';
        R_O <= RX_TEMP;
        Din_O <= '0';
        G_O <= '0';
        DONE <= '0';
```

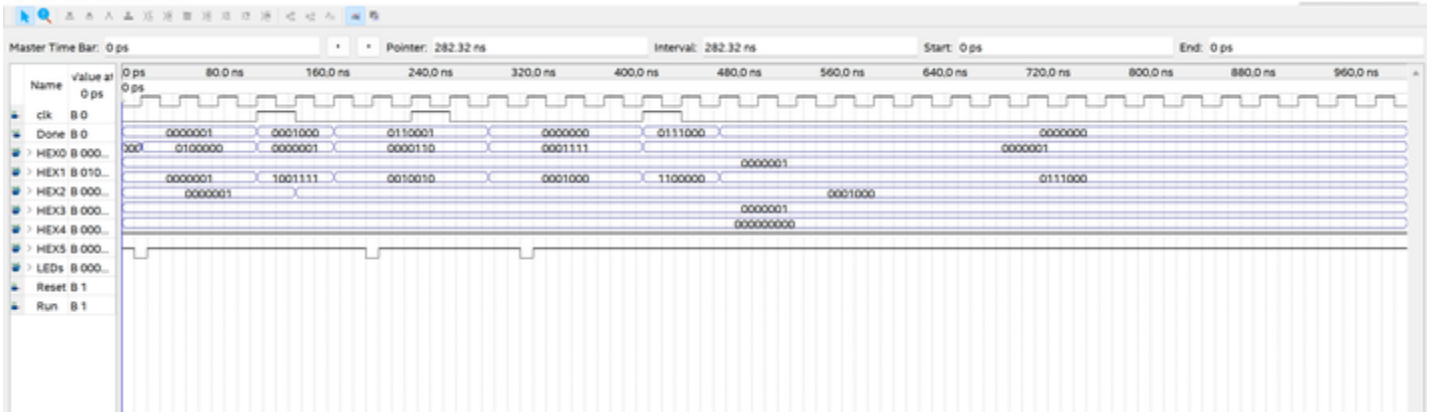
```
        ADDR_I <= '0';
        DOUT_I <= '1';
        W_D <= '0';
        incr_PC <= '0';
        WHEN ST2 =>
        IR_I <= '0';
        R_I <= "00000000";
        A_I <= '1';
        G_I <= '0';
        R_O <= RY_TEMP;
        Din_O <= '0';
        G_O <= '0';
        DONE <= '1';
        ADDR_I <= '1';
        DOUT_I <= '0';
        W_D <= '1';
        incr_PC <= '1';

        WHEN MVNZ =>
        IF (G = "00000000") THEN
             IR_I <= '0';
             R_I <= "00000000";
             A_I <= '0';
             G_I <= '0';
             R_O <= "00000000";
             G_O <= '0';
             Din_O <= '0';
             DONE <= '1';
             ADDR_I <= '0';
             DOUT_I <= '0';
             W_D <= '0';
             incr_PC <= '1';
        ElSE
             IR_I <= '0';
             R_I <= RX_TEMP;
             A_I <= '0';
             G_I <= '0';
             R_O <= RY_TEMP;
             G_O <= '0';
             Din_O <= '0';
             DONE <= '1';
             ADDR_I <= '0';
             DOUT_I <= '0';
             W_D <= '0';
             IF (RX_TEMP = "00000001") THEN
                  incr_PC <= '0';
             ElSE
                  incr_PC <= '1';
             END IF;
    END IF;
```

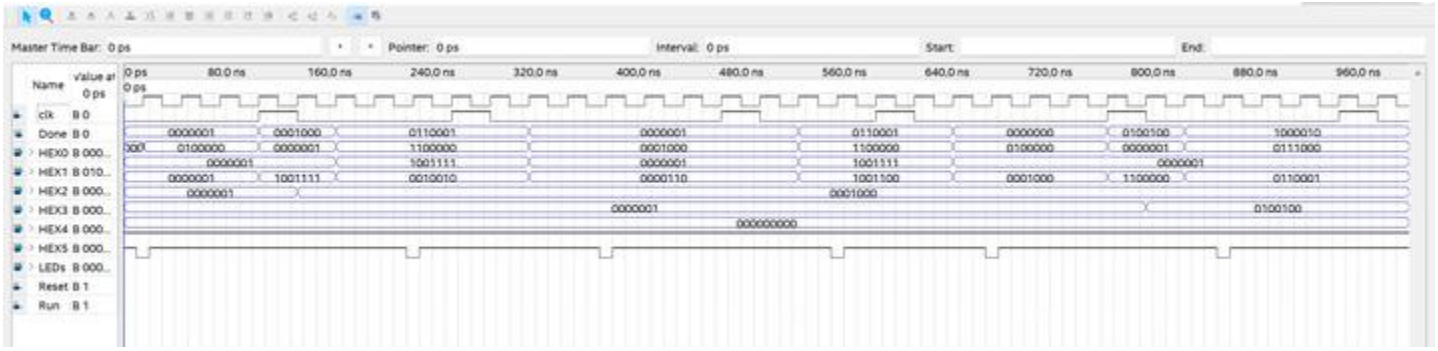
B/WAVEFORM OBTAINED FROM EXPERIMENT AND STATE DIAGRAM:

I)Waveform of each code state when running:

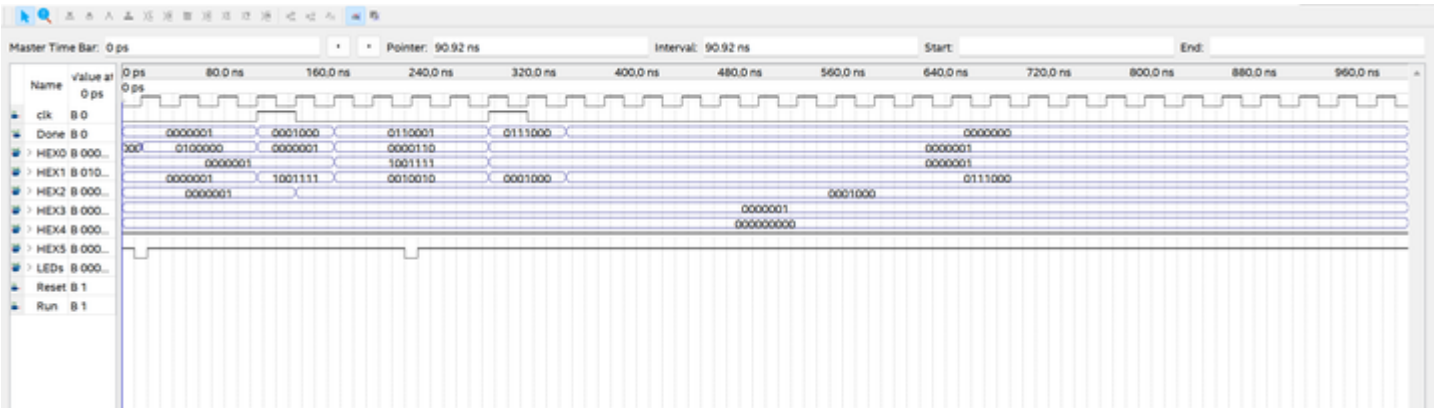
1.Waveform obtained when running the code to check MV,MVI of R7:



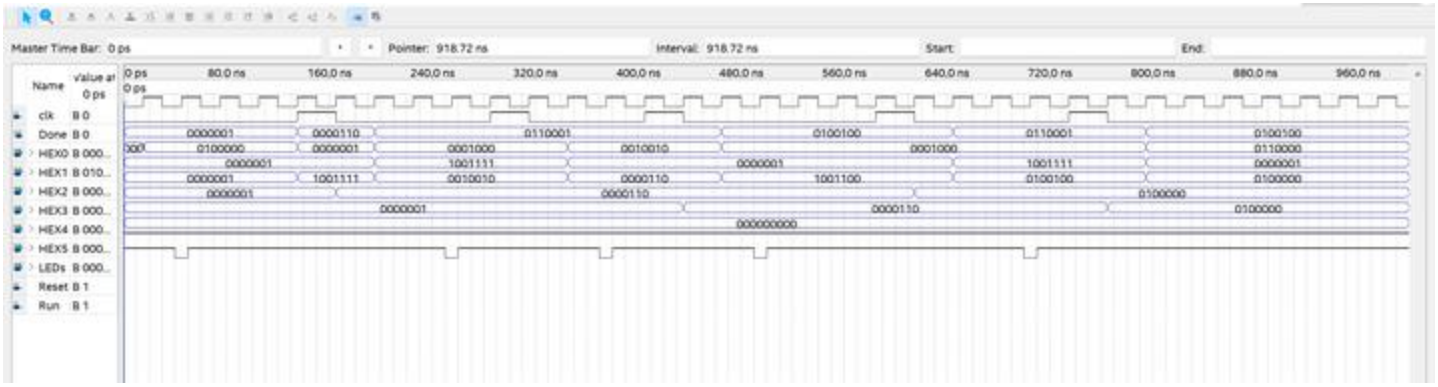
2.Waveform obtained when running the code to check add,sub, and MVNZ of R7:



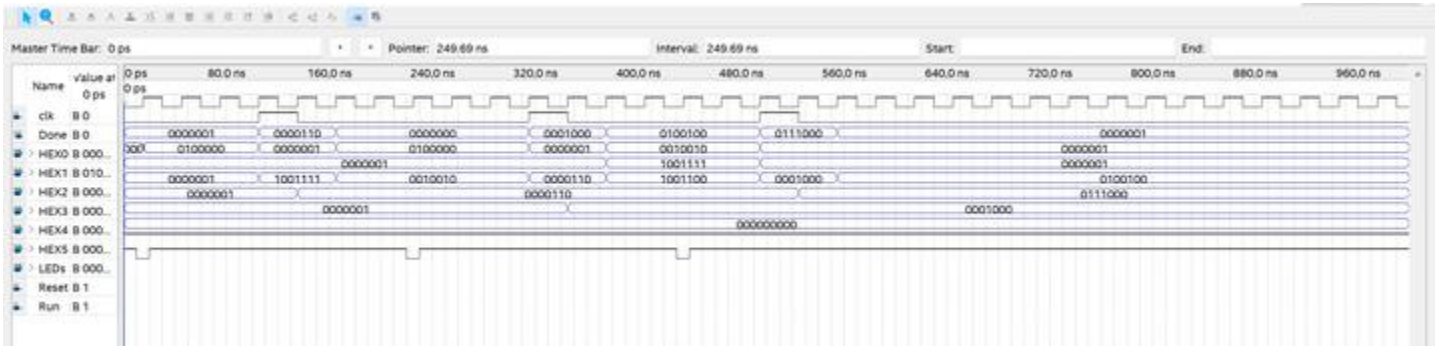
3.Waveform obtained when running the code to check LOAD of R7:



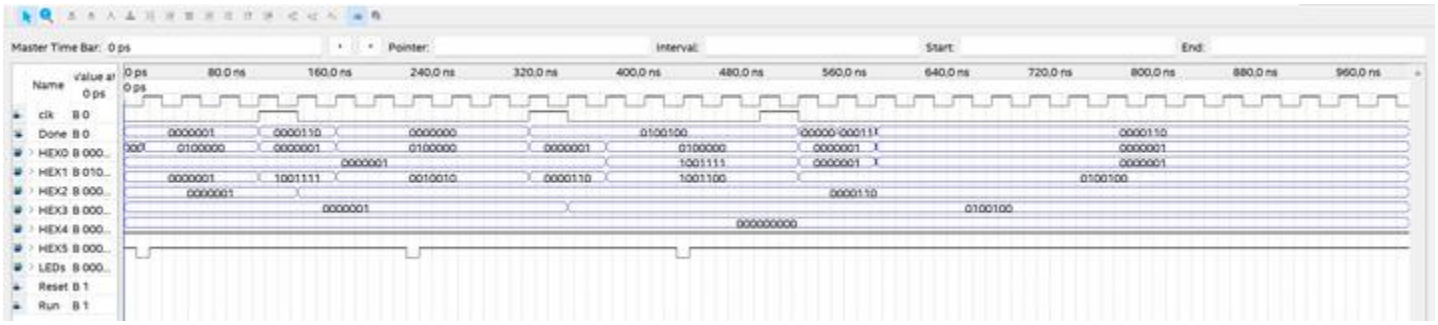
4.Waveform obtained when running the code to check mvi,mvnz,mv,add,sub:



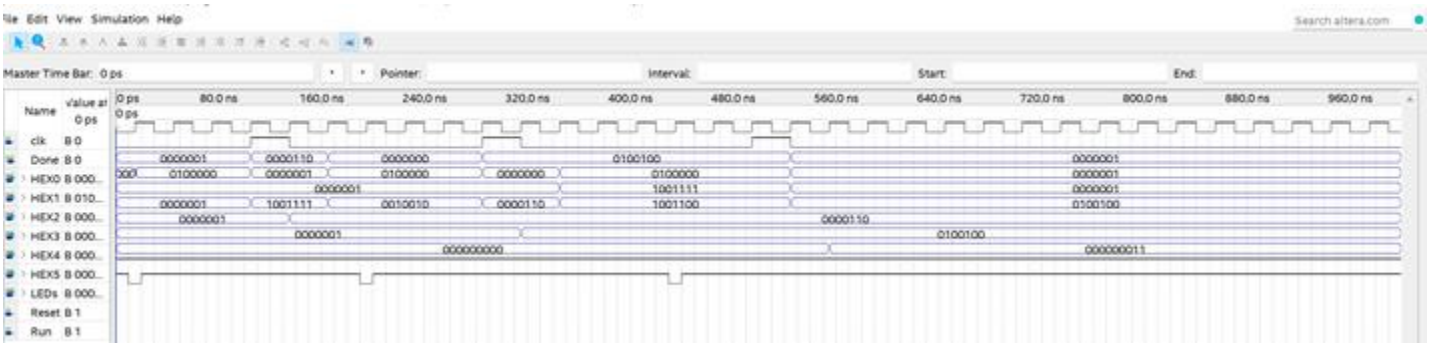
5.Waveform obtained when running the code to check LOAD instruction:



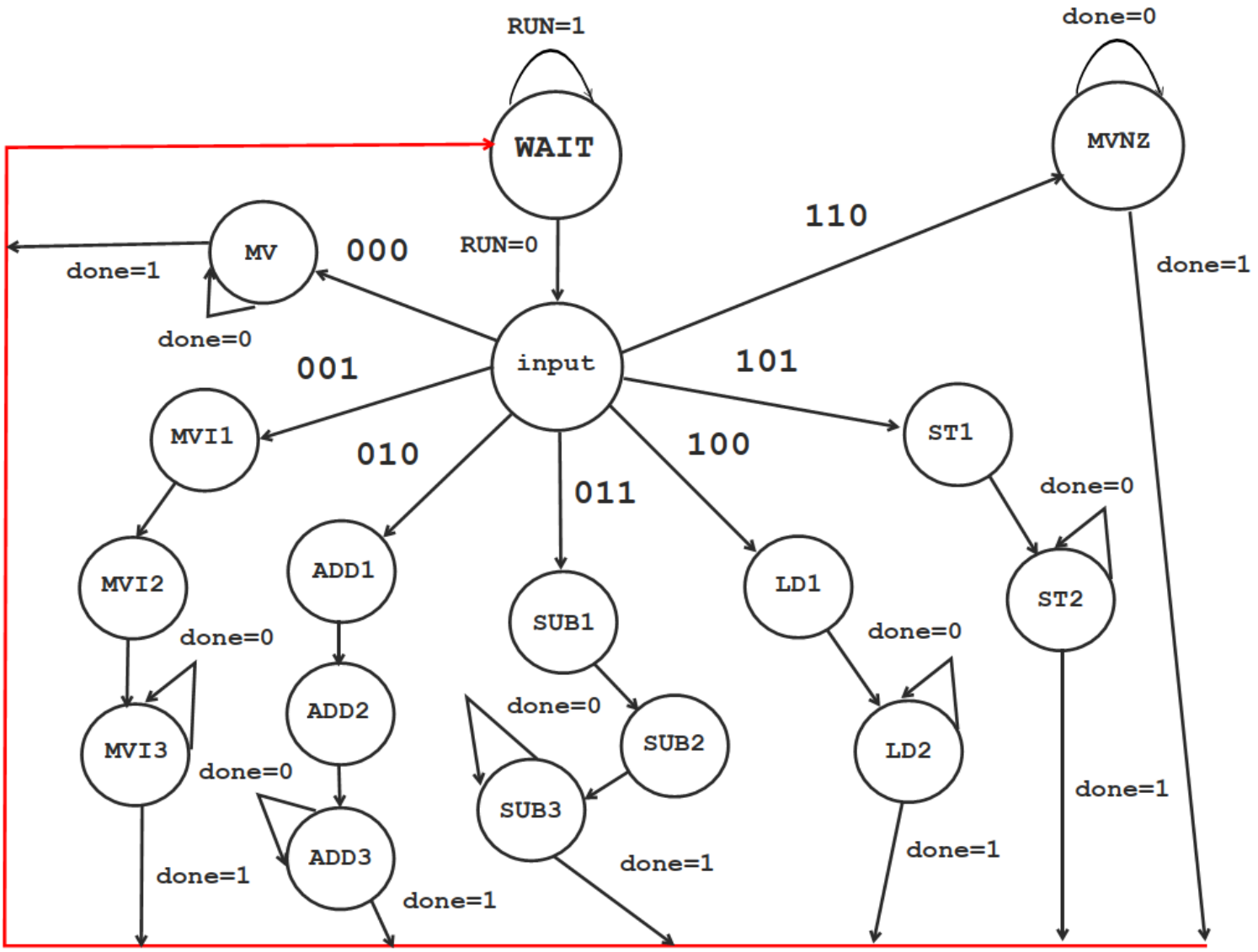
6.Waveform obtained when running the code to check STORE instruction:



7.Waveform obtained when running the code to check the LEDS:

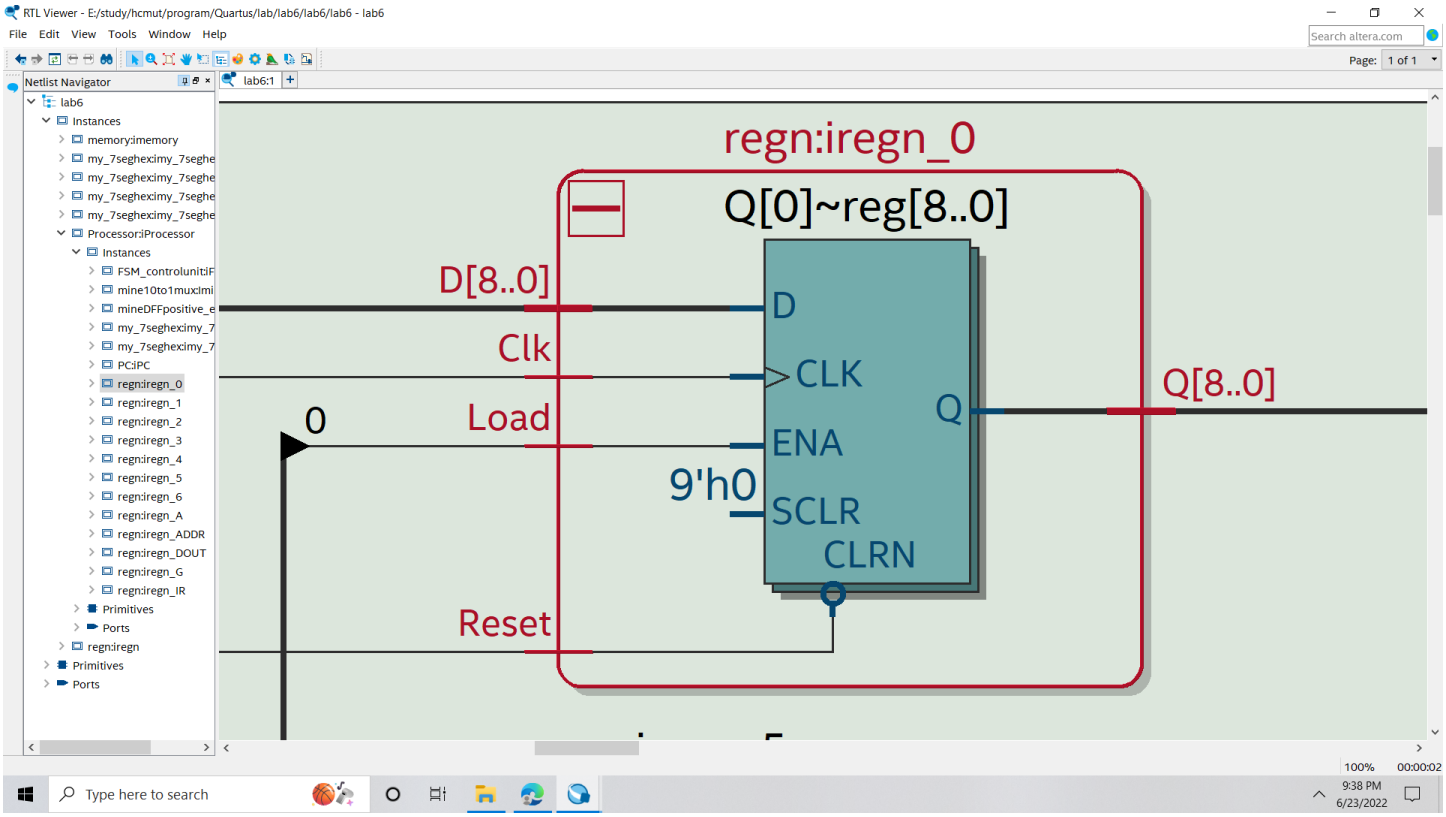


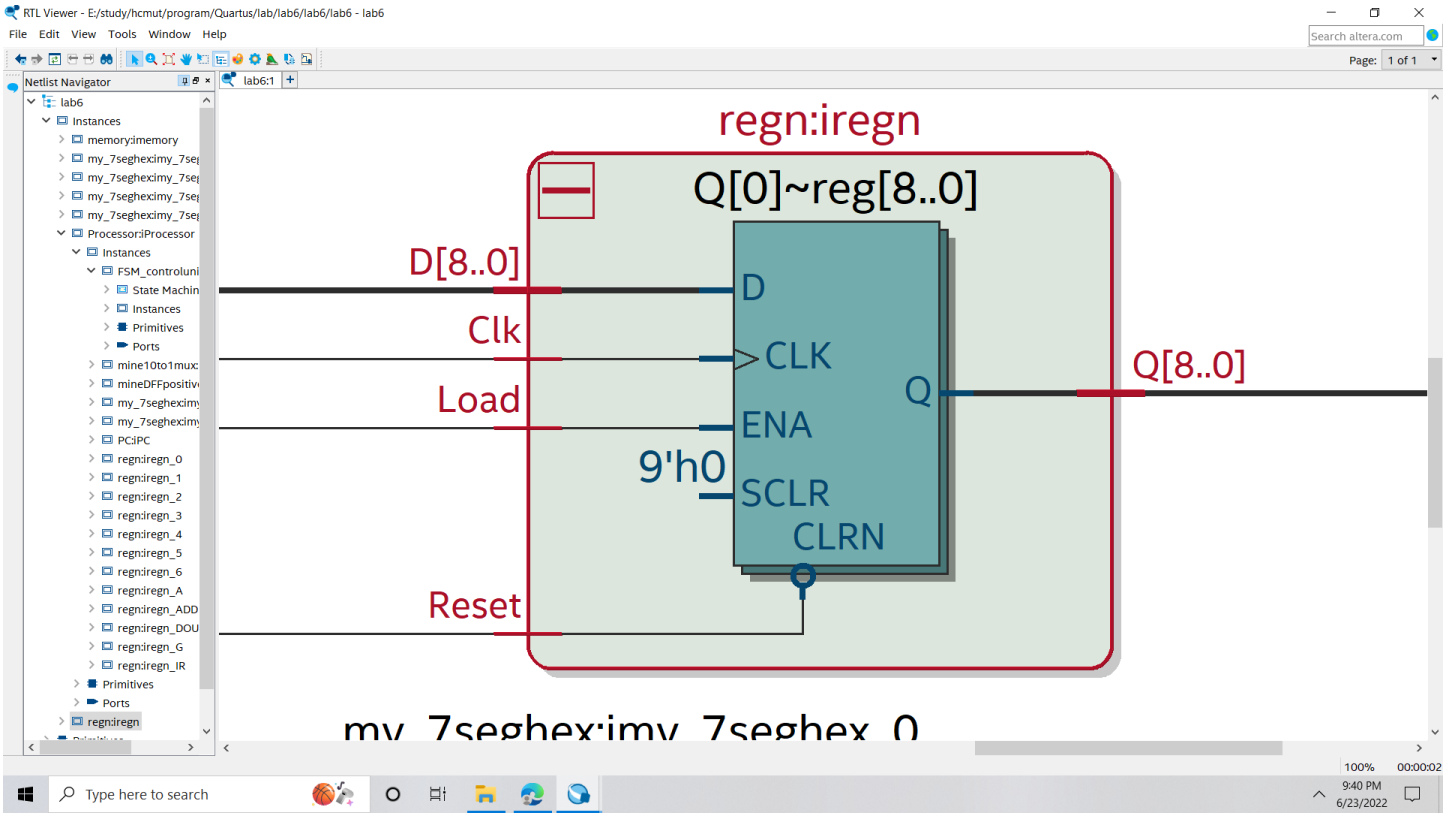
II)State diagram of the FSM Control Unit:



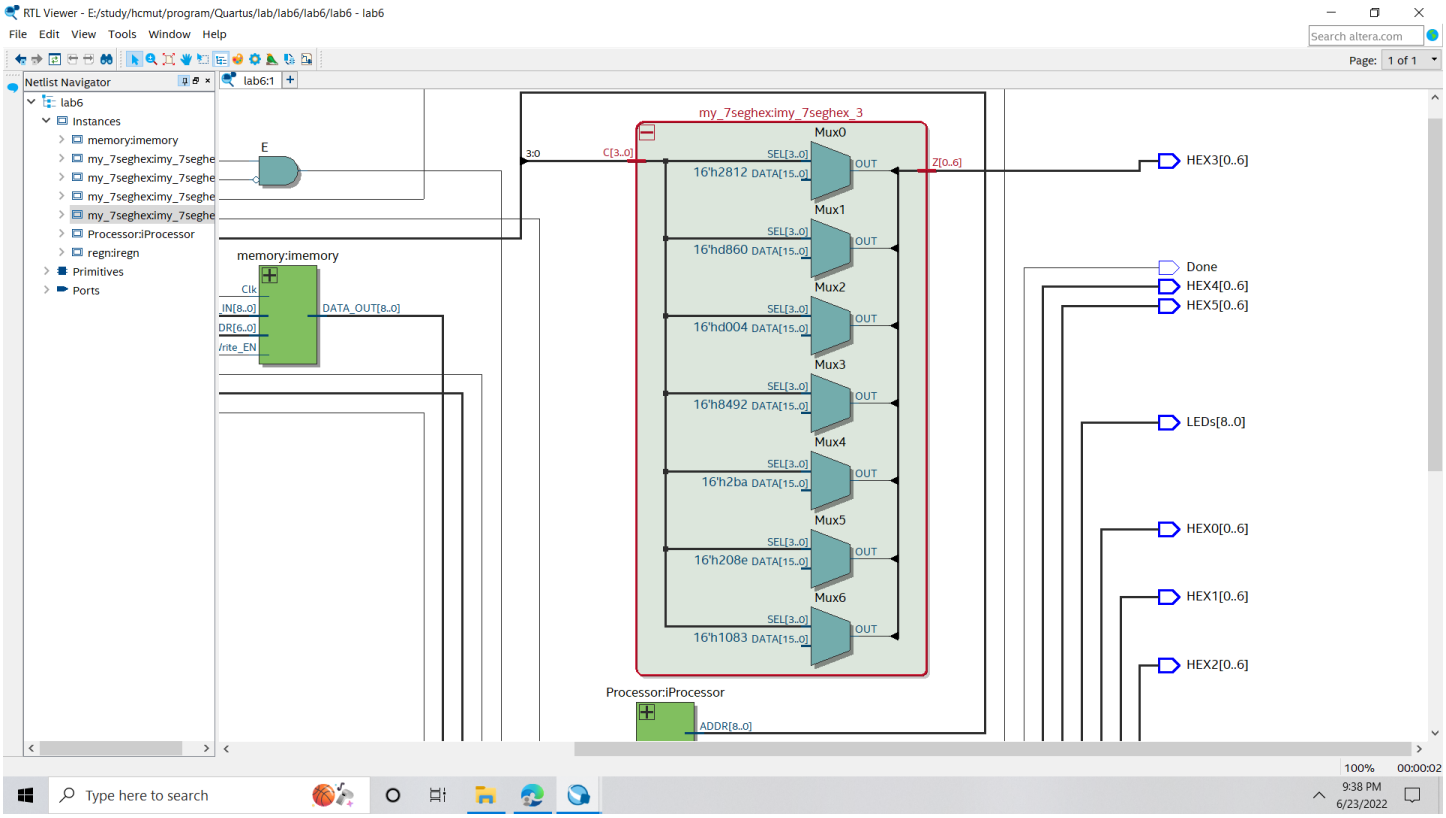
C/RTL view – digital circuit obtained:

1.Register:

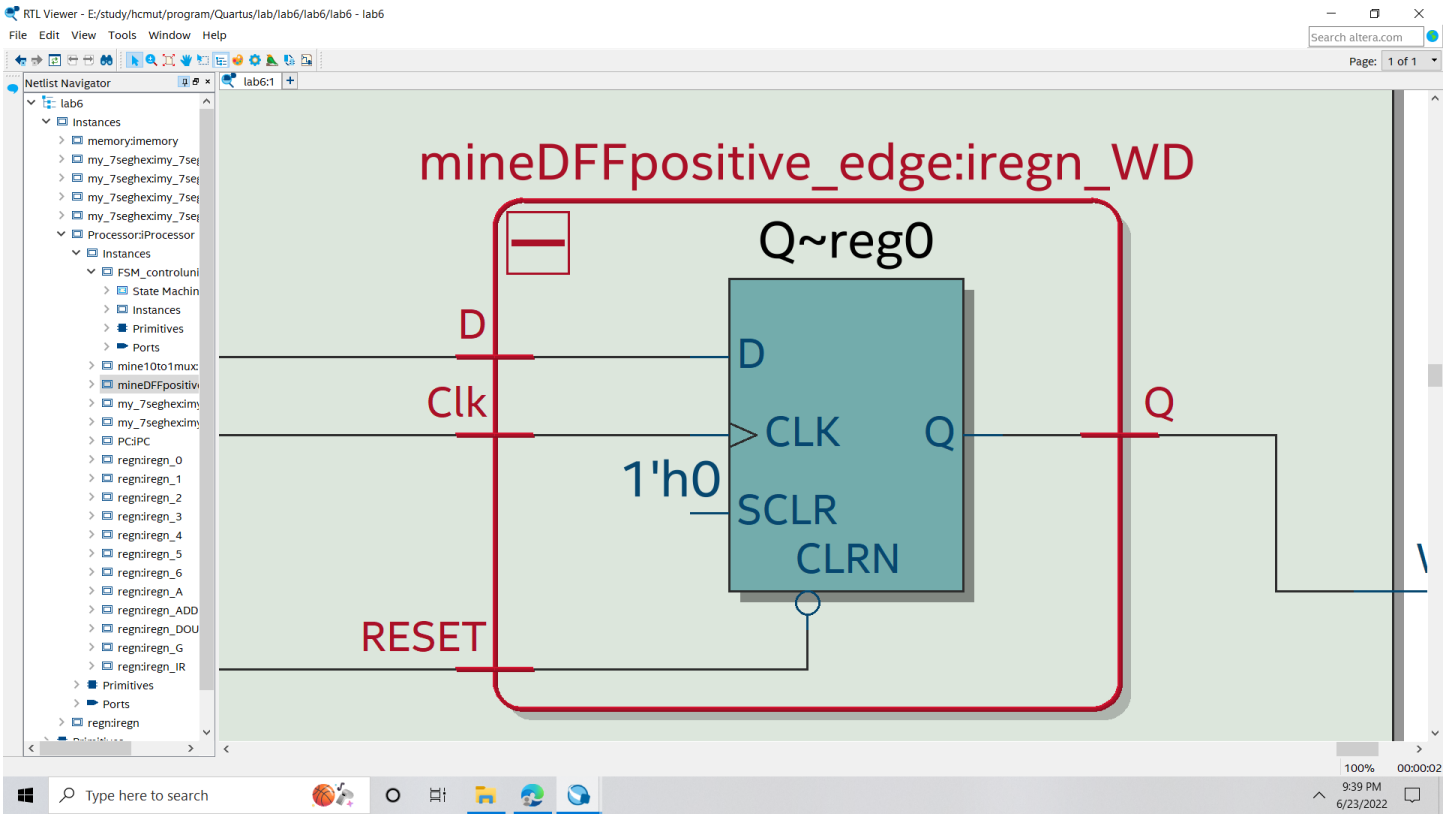




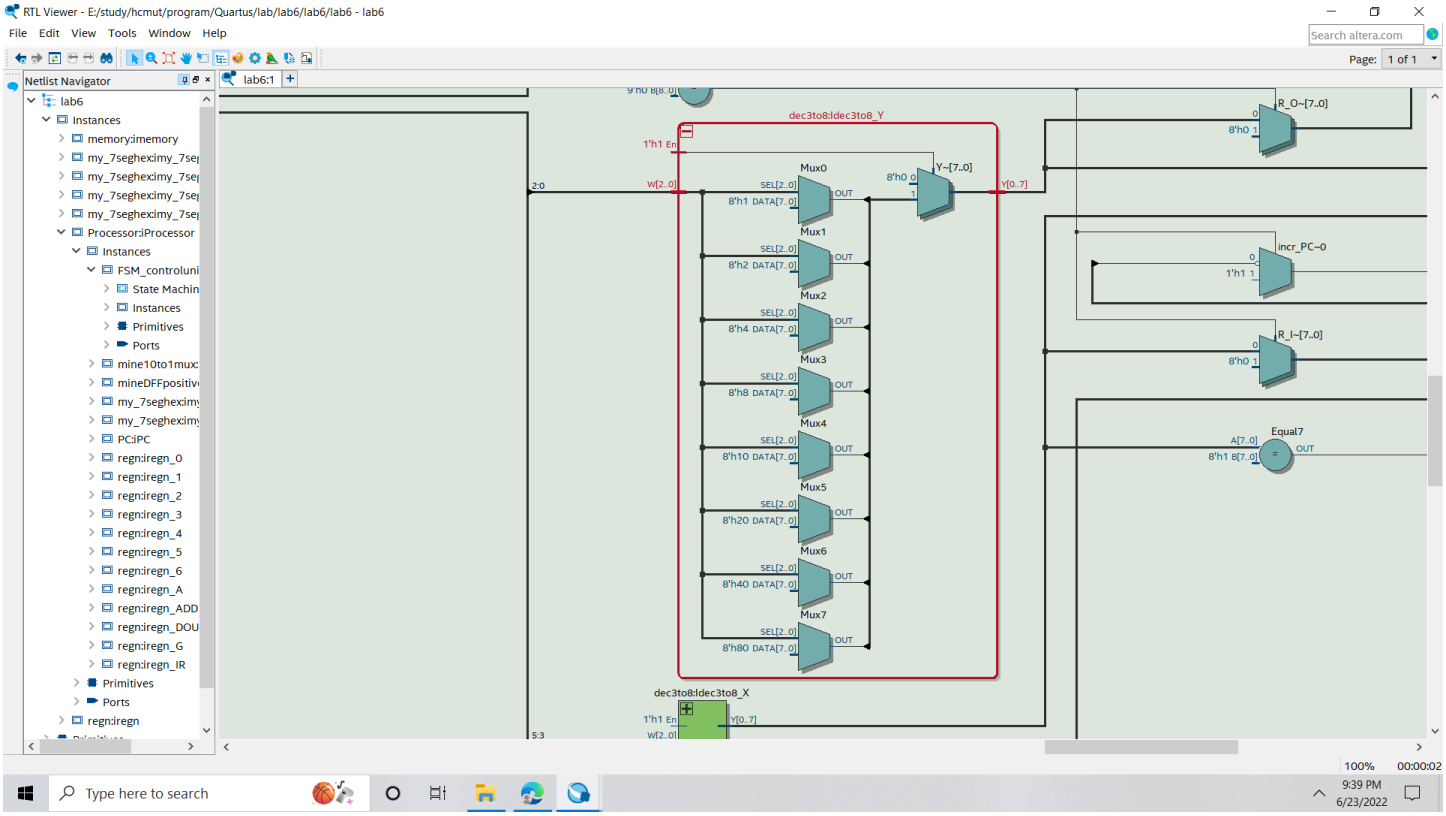
2)7 segment HEX:



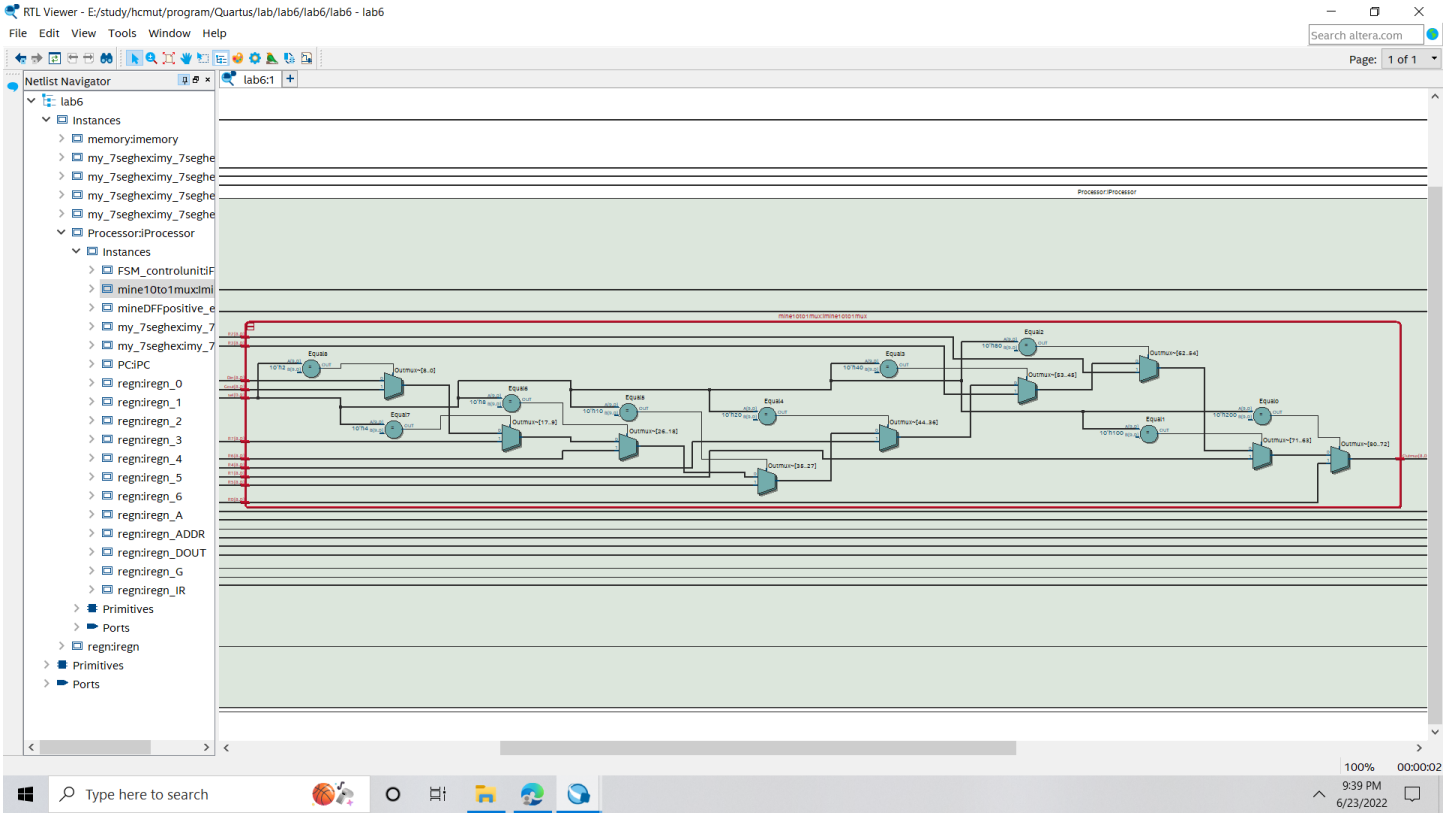
3)DFF positive edge:



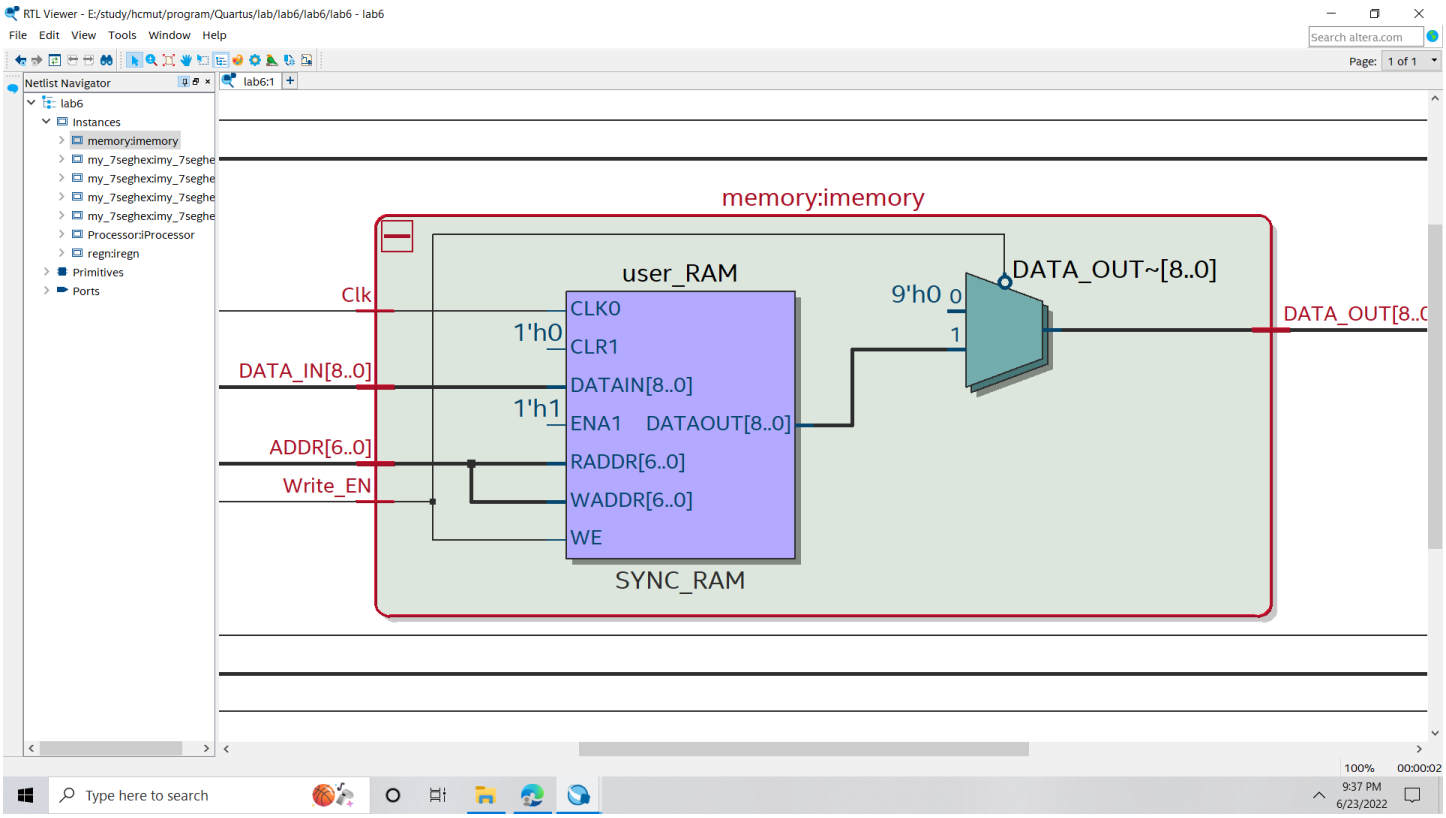
4)Decoder 3to8:



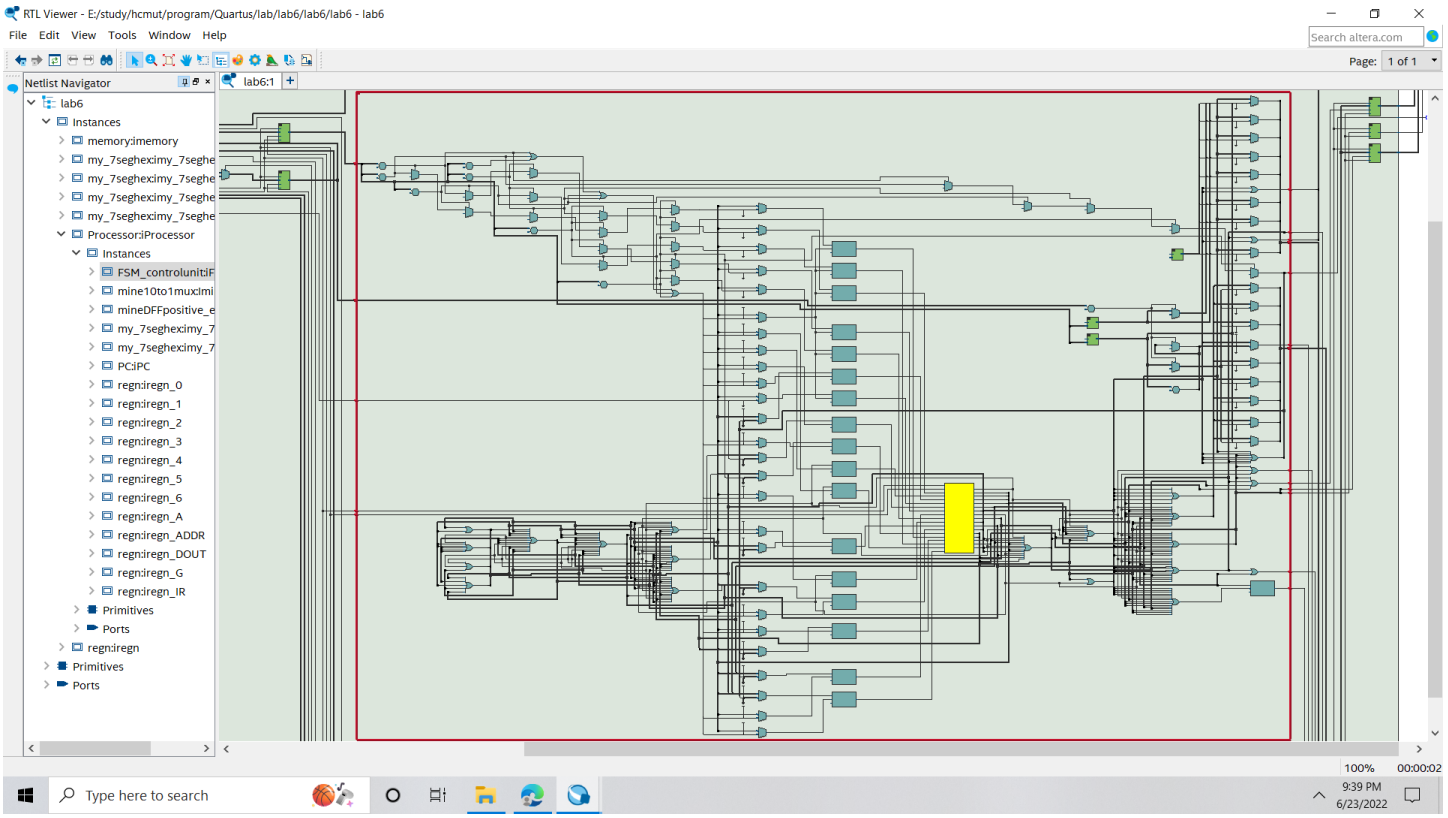
5)10to1 multiplexer:



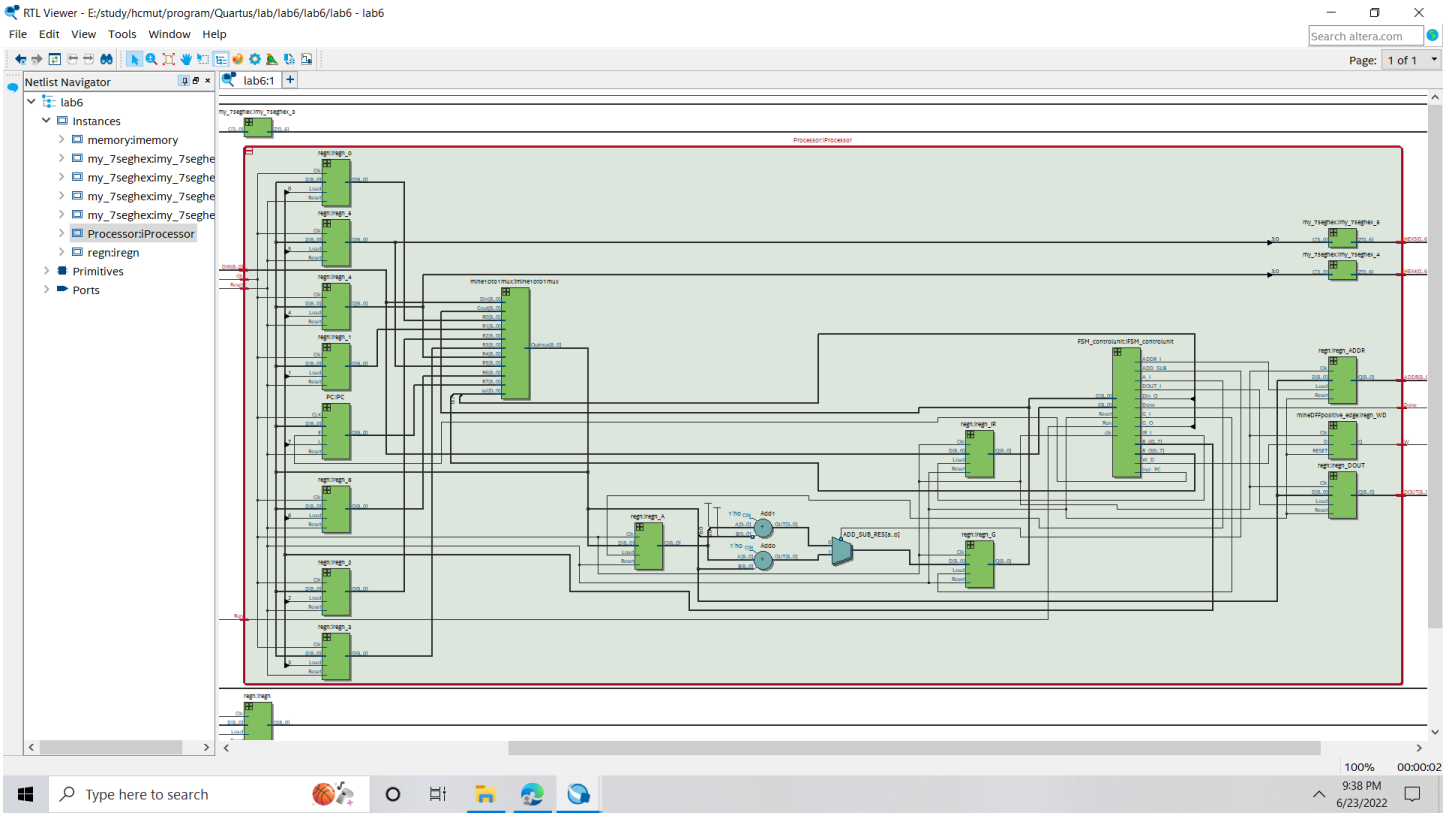
6) Memory:



7) FSM control unit:



8) Processor:



9) Overall LAB6: