# Lab 9-10 – Nano-processor Design Competition

CS1050 Computer Organization and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

**Presented By –** S.A.C.H.Gunapala 210190R

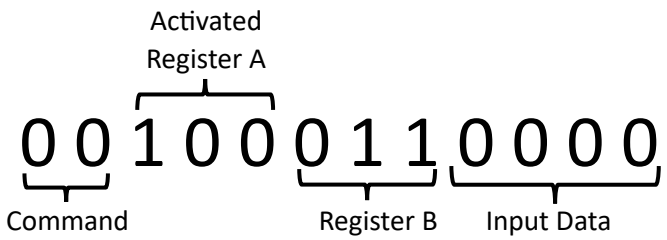
G.V.H.H.B.Jayarathna 210242F

## Assigned Lab Task.

This lab in particular is a team project. Teams of two are assigned to design and test a 4-bit processor on Basys3 FPGA which is capable of executing 4 instructions using a Programmable ROM.

## Assembly program and its machine code representation.

Given a 12-bit instruction as follows,




Let's see some examples of machine code representation of some assembly language code,

| Machine Code | Assembly Language |
|---|---|
| 100010000011 | MOV R1, 3 |
| 101110000001 | MOV R7, 1 |
| 001110010000 | ADD R7, R1 |
| 011110000000 | NEG R7 |
| 110010000110 | JZR R1, 6 |

Possible commands and other useful commands,

| Command | Assembly Language Command |
|---|---|
| 00 | ADD (adds the data in two registers) |
| 10 | MOVI (moves data to a register) |
| 01 | NEG (gets the two's compliment) |
| 11 | JZR (jump command if the given register has 0000 to an instruction) |

Register Selection,

| Command | Selected Register |
|---|---|
| 000 | R0 |
| 001 | R1 |
| 010 | R2 |
| 011 | R3 |
| 100 | R4 |
| 101 | R5 |
| 110 | R6 |
| 111 | R7 |

Last 4 digits display the immediate data input to the nanoprocessor.

## VHDL Codes.

- **Instructions Decorder.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity InstructionDecoder is
   Port ( I : in STD_LOGIC_VECTOR (11 downto 0);
          Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
          Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
          Reg_Sel0 : out STD_LOGIC_VECTOR (2 downto 0);
```

```
        Reg_Sel1 : out STD_LOGIC_VECTOR (2 downto 0);
        Address : out STD_LOGIC_VECTOR (2 downto 0);
        d_value : out STD_LOGIC_VECTOR (3 downto 0);
        Jump : out STD_LOGIC;
        Load_Sel : out STD_LOGIC;
        Add_Sub : out STD_LOGIC);
end InstructionDecoder;

architecture Behavioral of InstructionDecoder is

signal Jump_Temp : std_logic;

begin

   d_value <= I(3 downto 0);
   Reg_EN <= I(9 downto 7);
   Reg_Sel1 <= I(9 downto 7);
   Reg_Sel0 <= I(6 downto 4);
   Add_Sub <= (not I(11)) and I(10);
   Jump_Temp <= I(11) and I(10);
   Load_Sel <= I(11) and (not I(10));
   Address <= I(2 downto 0);
   Jump <= Jump_Temp and (not (Reg_Check(0) or Reg_Check(1) or Reg_Check(2) or Reg_Check(3)));

end Behavioral;
```

- **Mux 2way 4-bit.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Mux_2way_4bit is
   Port ( S : in STD_LOGIC;
          D0 : in STD_LOGIC_VECTOR (3 downto 0);
          D1 : in STD_LOGIC_VECTOR (3 downto 0);
          Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2way_4bit;

architecture Behavioral of Mux_2way_4bit is

signal S_vec :std_logic_vector(3 downto 0);

begin

   S_vec <= (others => S); -- Assigning S to every element of S_vec
   Y <= (D0 and not S_vec) or (D1 and S_vec);

end Behavioral;
```

- **LUT for 7-Segment Display.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity LUT_16_7 is
   Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
          data : out STD_LOGIC_VECTOR (6 downto 0));
end LUT_16_7;

architecture Behavioral of LUT_16_7 is

type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);

 signal sevenSegment_ROM : rom_type := (
   "1000000", --0
   "1111001", --1
   "0100100", --2
   "0110000", --3
   "0011001", --4
   "0010010", --5
   "0000010", --6
   "1111000", --7
   "0000000", --8
   "0010000", --9
   "0001000", --a
   "0000011", --b
   "1000110", --c
   "0100001", --d
   "0000110", --e
   "0001110" --f
 );

begin

   data <= sevenSegment_ROM(to_integer(unsigned(address)));

end Behavioral;
```

- **Programmable ROM.**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PROM is
   Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
          I : out STD_LOGIC_VECTOR (11 downto 0));
end PROM;

architecture Behavioral of PROM is

type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);

 signal instruction_ROM : rom_type := (
 --CHANGE INSTRUCTIONS HERE
    "101110000000", --MOV R7, 0
    "100010000001", --MOV R1, 1
    "100100000010", --MOV R2, 2
    "100110000011", --MOV R3, 3
    "001110010000", --ADD R7, R1
    "001110100000", --ADD R7, R2
    "001110110000", --ADD R7, R3
```

```
    "110000000111"  --JZR R0, 7
);

begin

   I <= instruction_ROM(to_integer(unsigned(Mem_Sel)));

end Behavioral;
```

- **Program Counter.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PC is
   Port ( Jump : in STD_LOGIC;
          Address : in STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          Res : in STD_LOGIC;
          M : out STD_LOGIC_VECTOR (2 downto 0));
end PC;

architecture Behavioral of PC is
```

```
component RCA_3
   Port ( A : in STD_LOGIC_VECTOR (2 DOWNTO 0);
          S : out STD_LOGIC_VECTOR (2 DOWNTO 0);
          C_out : out STD_LOGIC);
end component;

component Mux_2way_3bit
   Port ( S : in STD_LOGIC;
          D0 : in STD_LOGIC_VECTOR (2 downto 0);
          D1 : in STD_LOGIC_VECTOR (2 downto 0);
          Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component PCReg
   Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
          Res  : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (2 downto 0));
end component;

SIGNAL PC_out, adder_out,Mux_out : STD_LOGIC_VECTOR(2 downto 0);
SIGNAL C_out_adder: STD_LOGIC;

begin
   adder_3_bit: RCA_3
   port map(
      A => PC_out,
      S => adder_out,
      C_out => C_out_adder);
```

```
    Mux_2way_3bit_0: Mux_2way_3bit
    port map(
      S => Jump,
      D0 => adder_out,
      D1 => Address,
      Y => Mux_out);

    ProgramCounter: PCReg
    port map(
      D => Mux_out,
      Res => Res,
      Clk => Clk,
      Q => PC_out);

     M <= PC_out;

  end Behavioral;
```

- **Program Counter Register(PC_reg).**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity PCReg is
   Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
          Res  : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (2 downto 0));
end PCReg;

architecture Behavioral of PCReg is

component D_FF
port (
   D : in STD_LOGIC;
   Res: in STD_LOGIC;
   Clk : in STD_LOGIC;
   Q : out STD_LOGIC);
end component;


begin

D_FF0 : D_FF
port map (
   D => D(0),
   Res => Res,
   Clk => Clk,
   Q => Q(0));

D_FF1 : D_FF
   port map (
      D => D(1),
```

```
      Res => Res,
      Clk => Clk,
      Q => Q(1));

  D_FF2 : D_FF
  port map (
    D => D(2),
    Res => Res,
    Clk => Clk,
    Q => Q(2));

  end Behavioral;
```

- **D_FF**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity D_FF is
   Port ( D : in STD_LOGIC;
          Res : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Q : out STD_LOGIC);
end D_FF;

architecture Behavioral of D_FF is
signal Q_reg : std_logic := '0';

begin
   process (Clk) begin
      if (rising_edge(Clk)) then
         if Res = '1' then
            Q_reg <= '0';
```

```
        else
          Q_reg <= D;
        end if;
      end if;
      Q <= Q_reg;
   end process;
end Behavioral;
```

- **Mux_2way_3bit**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Mux_2way_3bit is
   Port ( S : in STD_LOGIC;
          D0 : in STD_LOGIC_VECTOR (2 downto 0);
          D1 : in STD_LOGIC_VECTOR (2 downto 0);
          Y : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2way_3bit;

architecture Behavioral of Mux_2way_3bit is

signal S_vec :std_logic_vector(2 downto 0);

begin

   S_vec <= (others => S); -- Assigning S to every element of S_vec
   Y <= (D0 and not S_vec) or (D1 and S_vec);

end Behavioral;
```

- **Adder_3_bit**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
```

```
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RCA_3 is
   Port ( A : in STD_LOGIC_VECTOR (2 DOWNTO 0);
          S : out STD_LOGIC_VECTOR (2 DOWNTO 0);
          C_out : out STD_LOGIC);
end RCA_3;

architecture Behavioral of RCA_3 is
   component FA
          port (
          A: in STD_LOGIC;
          B: in STD_LOGIC;
          C_in: in STD_LOGIC;
          S: out STD_LOGIC;
          C_out: out STD_LOGIC);
   end component;

SIGNAL FA0_C,FA1_C : STD_LOGIC;

begin

   FA_0 : FA
      port map(
      A => A(0),
      B => '1',
      C_in => '0',
      S => S(0),
      C_out => FA0_C);

   FA_1 : FA
          port map(
          A => A(1),
          B => '0',
          C_in => FA0_C,
          S => S(1),
          C_out => FA1_C);

   FA_2 : FA
          port map(
          A => A(2),
          B => '0',
          C_in => FA1_C,
          S => S(2),
          C_out => C_out);

end Behavioral;
```

- **Add/Sub Unit.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Add_Sub_unit is
   Port ( D : in STD_LOGIC_VECTOR (31 downto 0);
          RegSel0 : in STD_LOGIC_VECTOR (2 downto 0);
          RegSel1 : in STD_LOGIC_VECTOR (2 downto 0);
          AddSub : in STD_LOGIC; -- 0 is adder, 1 is substracter
          Overflow : out STD_LOGIC;
          Zero : out STD_LOGIC;
          Output : out STD_LOGIC_VECTOR (3 downto 0);
          RegCheck : out STD_LOGIC_VECTOR (3 downto 0));
end Add_Sub_unit;

architecture Behavioral of Add_Sub_unit is

   component RCAS_4
      Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
             B : in STD_LOGIC_VECTOR(3 downto 0);
             Func : in STD_LOGIC; -- 0 is adder, 1 is substracter
```

```
        S : out STD_LOGIC_VECTOR(3 downto 0);
        C_out : out STD_LOGIC);
    end component;

    component Mux_8way_4bit
      Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
        D : in STD_LOGIC_VECTOR (31 downto 0);
        Y : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

 SIGNAL Mux0_out, Mux1_out, result: STD_LOGIC_VECTOR(3 downto 0);
 SIGNAL Overflow_flag: STD_LOGIC;

begin

    Mux_8way_4bit_0 : Mux_8way_4bit
    port map(
    S => RegSel0,
    D => D,
    Y => Mux0_out);

    Mux_8way_4bit_1 : Mux_8way_4bit
    port map(
    S => RegSel1,
    D => D,
    Y => Mux1_out);

    RCAS_0 : RCAS_4
    port map(
    A => Mux0_out,
    B => Mux1_out,
```

```
    Func => AddSub,
    S => result,
    C_out => Overflow_flag);

    Overflow <= Overflow_flag;
    Output <= result;
    RegCheck <= Mux0_out;
    Zero <= NOT( result(0) OR result(1) OR result(2) OR result(3) OR Overflow_flag);

end Behavioral;
```

- **Mux_8way_4bit.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Mux_8way_4bit is
   Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
          D : in STD_LOGIC_VECTOR (31 downto 0);
          Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8way_4bit;

architecture Behavioral of Mux_8way_4bit is

component Decoder_3_to_8
 port (
    address: in std_logic_vector(2 downto 0);
    data: out std_logic_vector(7 downto 0));
end component;

signal decoded_S :std_logic_vector(7 downto 0);
signal S0_vec :std_logic_vector(3 downto 0);
signal S1_vec :std_logic_vector(3 downto 0);
signal S2_vec :std_logic_vector(3 downto 0);
signal S3_vec :std_logic_vector(3 downto 0);
```

```
signal S4_vec :std_logic_vector(3 downto 0);
signal S5_vec :std_logic_vector(3 downto 0);
signal S6_vec :std_logic_vector(3 downto 0);
signal S7_vec :std_logic_vector(3 downto 0);

begin

  LUT_Decoder : Decoder_3_to_8
       port map(
       address => S,
       data => decoded_S);


  S0_vec <= (others => decoded_S(0)); -- Assigning decoded_s(0) to every element of S0_vec
  S1_vec <= (others => decoded_S(1));
  S2_vec <= (others => decoded_S(2));
  S3_vec <= (others => decoded_S(3));
  S4_vec <= (others => decoded_S(4));
  S5_vec <= (others => decoded_S(5));
  S6_vec <= (others => decoded_S(6));
  S7_vec <= (others => decoded_S(7));

  Y <= (D(3 downto 0) and S0_vec) or (D(7 downto 4) and S1_vec) or (D(11 downto 8) and S2_vec) or
(D(15 downto 12) and S3_vec) or (D(19 downto 16) and S4_vec) or (D(23 downto 20) and S5_vec) or (D(27
downto 24) and S6_vec) or (D(31 downto 28) and S7_vec);

end Behavioral;
```

- **RCAS (Ripple carry Adder/Subtracter).**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RCAS_4 is
  Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
       B : in STD_LOGIC_VECTOR(3 downto 0);
       Func : in STD_LOGIC; -- 0 is adder, 1 is substracter
       S : out STD_LOGIC_VECTOR(3 downto 0);
       C_out : out STD_LOGIC);
end RCAS_4;
```

```
architecture Behavioral of RCAS_4 is
   component FA
         port (
         A: in STD_LOGIC;
         B: in STD_LOGIC;
         C_in: in STD_LOGIC;
         S: out STD_LOGIC;
         C_out: out STD_LOGIC);
   end component;

SIGNAL FA0_C,FA1_C,FA2_C,FA3_C : STD_LOGIC;
SIGNAL BS : STD_LOGIC_VECTOR(3 downto 0);

begin

   FA_0 : FA
      port map(
      A => A(0),
      B => BS(0),
      C_in => Func,
      S => S(0),
      C_out => FA0_C);

   FA_1 : FA
         port map(
         A => A(1),
         B => BS(1),
         C_in => FA0_C,
         S => S(1),
         C_out => FA1_C);

   FA_2 : FA
         port map(
         A => A(2),
         B => BS(2),
         C_in => FA1_C,
         S => S(2),
         C_out => FA2_C);

   FA_3 : FA
         port map(
         A => A(3),
         B => BS(3),
         C_in => FA2_C,
         S => S(3),
         C_out => C_out);

   BS(0) <= B(0) xor Func;
   BS(1) <= B(1) xor Func;
```

```
   BS(2) <= B(2) xor Func;
   BS(3) <= B(3) xor Func;

end Behavioral;
```

- **FA (Full Adder).**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity FA is
   Port ( A : in STD_LOGIC;
        B : in STD_LOGIC;
        C_in : in STD_LOGIC;
        S : out STD_LOGIC;
        C_out : out STD_LOGIC);
end FA;

architecture Behavioral of FA is

component HA
   port (
   A : in STD_LOGIC;
   B : in STD_LOGIC;
   S : out STD_LOGIC;
   C : out STD_LOGIC);
end component;

SIGNAL HA0_S, HA0_C, HA1_S, HA1_C : STD_LOGIC;

begin
HA_0 : HA
     port map(
     A => A,
     B => B,
     S => HA0_S,
     C => HA0_C);

HA_1 : HA
     port map(
```

```
        A => HA0_S,
        B => C_in,
        S => HA1_S,
        C => HA1_C);

    S <= HA1_S;
    C_out <= HA0_C OR HA1_C;

    end Behavioral;
```

- **HA (Half Adder).**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity HA is
   Port ( A : in STD_LOGIC;
          B : in STD_LOGIC;
          S : out STD_LOGIC;
          C : out STD_LOGIC);
end HA;

architecture Behavioral of HA is

begin

S <= A XOR B;
C <= A AND B;

end Behavioral;
```

- **Reg Bank.**

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values
```

```
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity RegisterBank is
    Port ( R_EN : in STD_LOGIC_VECTOR (2 downto 0);
           Res  : in STD_LOGIC;
           Clk  : in STD_LOGIC;
           Input : in STD_LOGIC_VECTOR (3 downto 0);
           D : out STD_LOGIC_VECTOR (31 downto 0));
end RegisterBank;

architecture Behavioral of RegisterBank is

component Reg
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Res  : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
 end component;

 component Decoder_3_to_8
     Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
            data : out STD_LOGIC_VECTOR (7 downto 0));
 end component;

signal EN_vec : std_logic_vector(7 downto 0);
```

```
begin

  LUT_Decoder : Decoder_3_to_8
   PORT MAP(
      address => R_EN,
      data => EN_vec
   );

  Reg_0 : Reg
   PORT MAP(
       D => "0000",
       En => '1' ,
       Res => '0',
       Clk => Clk,
       Q => D(3 downto 0)
    );

  Reg_1 : Reg
     PORT MAP(
         D => Input,
         En => EN_vec(1) ,
         Res => Res,
         Clk => Clk,
         Q => D(7 downto 4)
     );

  Reg_2 : Reg
       PORT MAP(
          D => Input,
          En => EN_vec(2) ,
```

```
        Res => Res,
        Clk => Clk,
        Q => D(11 downto 8)
    );

Reg_3 : Reg
 PORT MAP(
    D => Input,
    En => EN_vec(3) ,
    Res => Res,
    Clk => Clk,
    Q => D(15 downto 12)
 );

Reg_4 : Reg
 PORT MAP(
    D => Input,
    En => EN_vec(4) ,
    Res => Res,
    Clk => Clk,
    Q => D(19 downto 16)
 );

Reg_5 : Reg
 PORT MAP(
    D => Input,
    En => EN_vec(5) ,
    Res => Res,
    Clk => Clk,
    Q => D(23 downto 20)
 );
```

```
  Reg_6 : Reg
   PORT MAP(
      D => Input,
      En => EN_vec(6) ,
      Res => Res,
      Clk => Clk,
      Q => D(27 downto 24)
    );

  Reg_7 : Reg
   PORT MAP(
      D => Input,
      En => EN_vec(7) ,
      Res => Res,
      Clk => Clk,
      Q => D(31 downto 28)
    );

end Behavioral;
```

- **Decoder_3_to_8(LUT).**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
```

```
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Decoder_3_to_8 is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (7 downto 0));
end Decoder_3_to_8;

architecture Behavioral of Decoder_3_to_8 is

type rom_type is array (0 to 7) of std_logic_vector(7 downto 0);

 signal decoder_ROM : rom_type := (
    "00000001", --0
    "00000010", --1
    "00000100", --2
    "00001000", --3
    "00010000", --4
    "00100000", --5
    "01000000", --6
    "10000000"  --7
 );

begin

    data <= decoder_ROM(to_integer(unsigned(address)));

end Behavioral;
```

- **Register.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Reg is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Res  : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Reg;

architecture Behavioral of Reg is

begin
 process (Clk,Res)
 begin
 if (rising_edge(Clk)) then -- respond when clock rises
    if En = '1' then -- Enable should be set
        Q <= D;
    end if;
    if (Res = '1') then
        Q <= "0000";
```

```
    end if;
  end if;

end process;

end Behavioral;
```

- **Slow Clock.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;

architecture Behavioral of Slow_Clk is

signal count: integer := 1;
signal clk_status: std_logic := '0';

begin
```

```
    process(Clk_in) begin
    if (rising_edge(Clk_in)) then
        count <= count + 1;
        --if (count = 5) then
        if(count = 50000000) then
            clk_status <= not clk_status;
            Clk_out <= clk_status;
            count <= 1;
        end if;
    end if;
end process;

end Behavioral;
```

- **NanoProcessor.**

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity NanoProcessor is
    Port ( Clk : in STD_LOGIC;
           Seg7input : out std_logic_vector (3 downto 0);
```

```
        Reset : in STD_LOGIC;
        OverFlow : out STD_LOGIC;
        Zero : out STD_LOGIC;
        Seg7output : out STD_LOGIC_VECTOR (6 downto 0);
        Anode : out std_logic_vector (3 downto 0));
end NanoProcessor;

architecture Behavioral of NanoProcessor is

component Slow_Clk
   Port ( Clk_in : in STD_LOGIC;
        Clk_out : out STD_LOGIC);
end component;

component LUT_16_7
   Port ( result : in STD_LOGIC_VECTOR (3 downto 0);
        data : out STD_LOGIC_VECTOR (6 downto 0));
end component;

component RegisterBank
   Port ( R_EN : in STD_LOGIC_VECTOR (2 downto 0);
        Res  : in STD_LOGIC;
        Clk  : in STD_LOGIC;
        Input : in STD_LOGIC_VECTOR (3 downto 0);
        D : out STD_LOGIC_VECTOR (31 downto 0));
end component;

component Add_Sub_unit
   Port ( D : in STD_LOGIC_VECTOR (31 downto 0);
        RegSel0 : in STD_LOGIC_VECTOR (2 downto 0);
        RegSel1 : in STD_LOGIC_VECTOR (2 downto 0);
```

```
        AddSub : in STD_LOGIC; -- 0 is adder, 1 is substracter
        Overflow : out STD_LOGIC;
        Zero : out STD_LOGIC;
        Output : out STD_LOGIC_VECTOR (3 downto 0);
        RegCheck : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component PC
    Port ( Jump : in STD_LOGIC;
        Address : in STD_LOGIC_VECTOR (2 downto 0);
        Clk : in STD_LOGIC;
        Res : in STD_LOGIC;
        M : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component PROM
    Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
        I : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component Mux_2way_4bit
    Port ( S : in STD_LOGIC;
        D0 : in STD_LOGIC_VECTOR (3 downto 0);
        D1 : in STD_LOGIC_VECTOR (3 downto 0);
        Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component InstructionDecoder
    Port ( I : in STD_LOGIC_VECTOR (11 downto 0);
        Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
        Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
```

```
        Reg_Sel0 : out STD_LOGIC_VECTOR (2 downto 0);
        Reg_Sel1 : out STD_LOGIC_VECTOR (2 downto 0);
        Address : out STD_LOGIC_VECTOR (2 downto 0);
        d_value : out STD_LOGIC_VECTOR (3 downto 0);
        Jump : out STD_LOGIC;
        Load_Sel : out STD_LOGIC;
        Add_Sub : out STD_LOGIC);
end component;

signal slow_clk_sig, AddSub,Jump, Load_sel : std_logic;
signal Reg_En,RegSel0, RegSel1, JumpAddress,M : std_logic_vector(2 downto 0);
signal Regbank_input, AddSub_out, RegCheck, d_value : std_logic_vector(3 downto 0);
signal I : std_logic_vector(11 downto 0);
signal D : std_logic_vector(31 downto 0);

begin
   Slow_Clk_0: Slow_Clk
   port map(
      Clk_in => Clk,
      Clk_out => slow_clk_sig
   );

   Reg_bank: RegisterBank
   port map(
      R_EN => Reg_En,
      Res => Reset,
      Clk => slow_clk_sig,
      Input => Regbank_input,
      D => D
   );
```

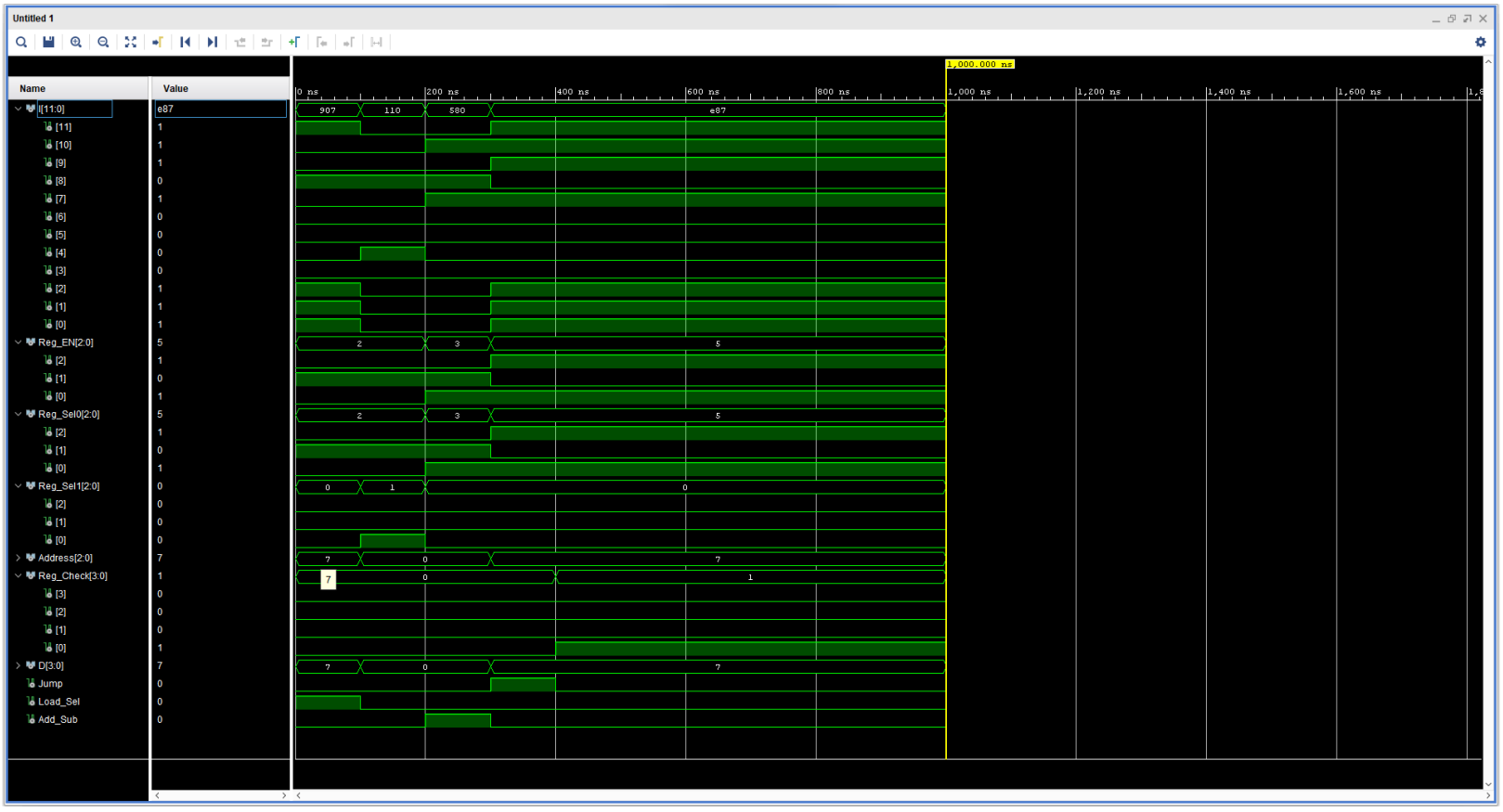
```
AddSubUnit: Add_Sub_unit
port map(
   D => D,
   RegSel0 => RegSel0,
   RegSel1 => RegSel1,
   AddSub => AddSub,
   OverFlow => overflow,
   Zero => Zero,
   Output => AddSub_out,
   RegCheck => RegCheck
);

ProgrammeCounter: PC
port map(
   Jump => Jump,
   Address => JumpAddress,
   Clk => slow_clk_sig,
   Res => Reset,
   M => M
);

ProgramROM : PROM
port map(
   Mem_Sel => M,
   I => I
);

LUT_16_7_0: LUT_16_7
port map(
   result => D(31 downto 28),
   data => Seg7output
```

```
);

Mux: Mux_2way_4bit
port map(
   S => Load_sel,
   D0 => AddSub_out,
   D1 => d_value,
   Y => Regbank_input
);

Instruction_decoder: InstructionDecoder
port map(
   I => I,
   Reg_Check => RegCheck,
   Reg_EN => Reg_En,
   Reg_sel0 => RegSel0,
   Reg_sel1 => RegSel1,
   Address => JumpAddress,
   d_value => d_value,
   Jump => Jump,
   Load_Sel => Load_sel,
   Add_Sub => AddSub
);

Anode <= "1110";
Seg7input <= D(31 downto 28);
```

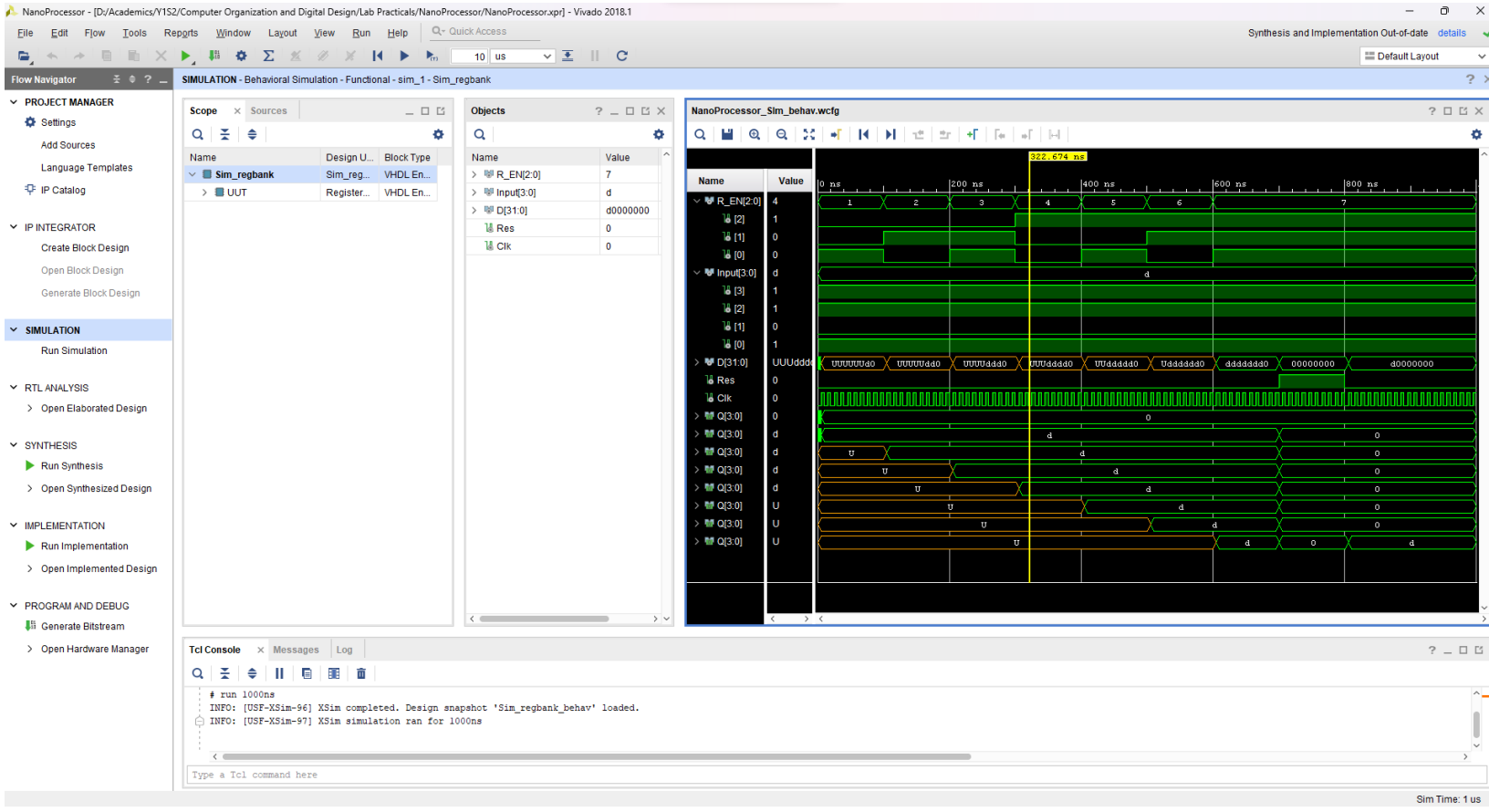
## Timing Diagrams.

- **Nano-Processor.**

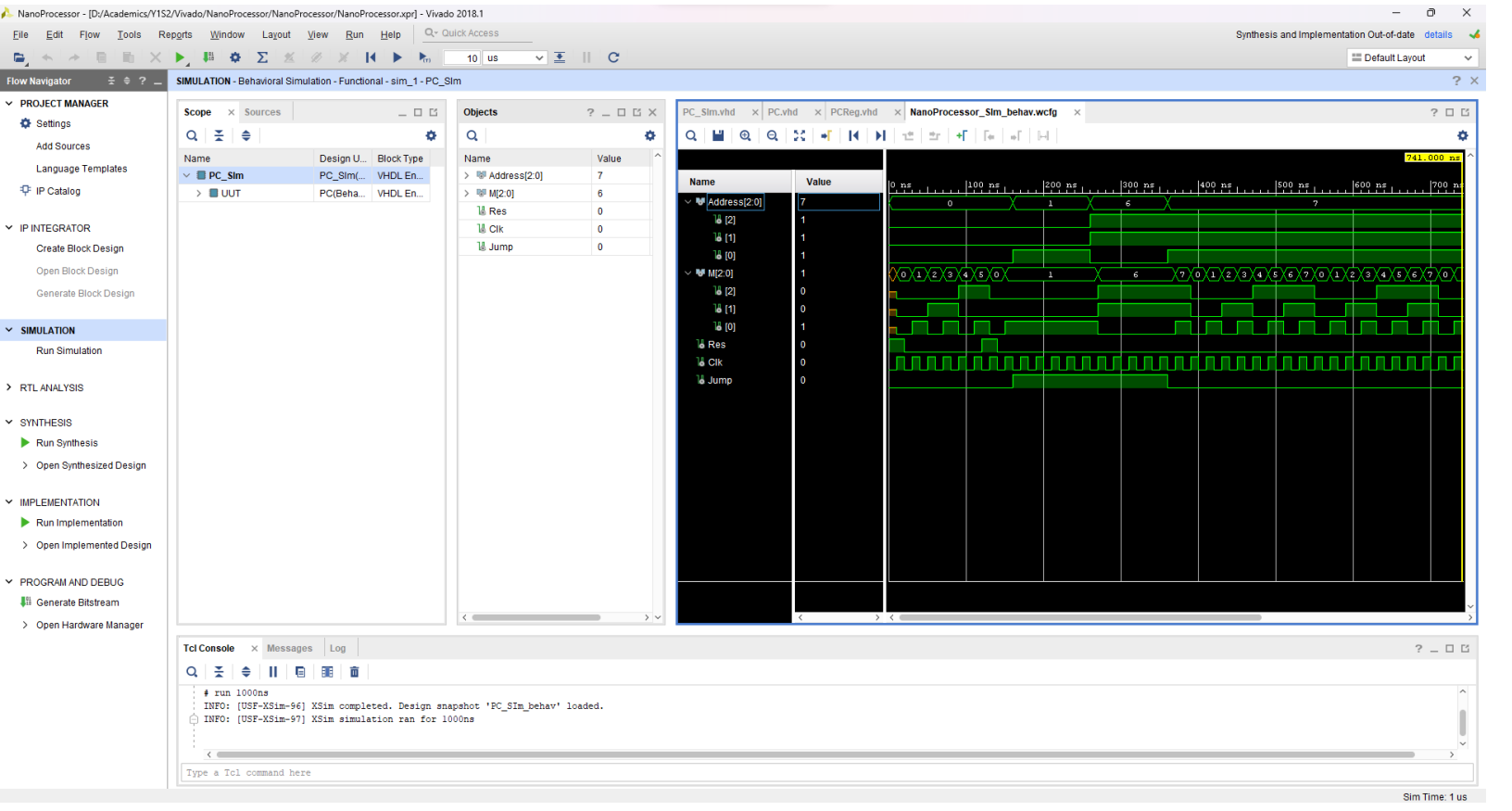

- **Instructions Decoder.**



- **Reg Bank.**




- **Program Counter.**



## Conclusion.

We were able to practice team-working skills and improve communication, coordination, sharing responsibilities and integrating components developed by different team members. We were able to design our very own 4-bit arithmetic unit that can add and subtract signed integers, decode instructions to activate different components of the processor and to verify the functionality of the nano-processor.

## Contribution.

| S.A.C.H.Gunapala<br>210190R | G.V.H.H.B.Jayarathna<br>210242F |
|---|---|
| Designed the Register Bank | Designed the 4-bit Add/Sub unit |
| Designed all the Multiplexers inside the nano-processor | Designed 3-bit adder |
| Designed the Programmable ROM | Designed the Instruction Decoder |
| Designed part of Program Counter | Designed part of Program Counter. |
| Simulated the related components | Simulated the related components |

Designing each component and connecting them took about 12 hours. Creating Test Bench files and simulating each component separately took about 6 hours. Connecting each component and simulating the Nano-Processor took 2 hours. Fixing bugs and optimizing the components for less logic gates took about another 10 – 12 hours. Spent about 30 – 32 hours completing the Nano-Processor.

## NanoProcessor simulation VHDL.

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.
```

```
--library UNISIM;
--use UNISIM.VComponents.all;

entity NanoProcessor_SIm is
--  Port ( );
end NanoProcessor_SIm;

architecture Behavioral of NanoProcessor_SIm is
component NanoProcessor
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Seg7input : out std_logic_vector (3 downto 0);
           OverFlow : out STD_LOGIC;
           Zero : out STD_LOGIC;
           Seg7output : out STD_LOGIC_VECTOR (6 downto 0);
           Anode : out std_logic_vector (3 downto 0));
end component;

signal Clk, Reset, OverFlow, Zero : std_logic;
signal Seg7output : std_logic_vector (6 downto 0);
signal Seg7input,Anode : std_logic_vector (3 downto 0);

begin
UUT: NanoProcessor port map (
    Clk => Clk,
    Reset => Reset,
    Seg7input => Seg7input,
    OverFlow => OverFlow,
    Zero => Zero,
```

```
    Seg7output => Seg7output,
    Anode => Anode);

Clk_process : process
begin
Clk <= '0';
wait for 5 ns;
Clk <= '1';
wait for 5 ns;
end process;

processor : process
begin
Reset <= '0';
wait for 250 ns;
Reset <= '1';
wait for 30 ns;
Reset <= '0';
wait;
end process;

end Behavioral;
```