NanoProcessor Design Competition (Lab 9-10)

CS1050 Computer Organization and Digital Design

Student Names

Subavarshana Arumugam (210621R)

Madhushika K.A.H.M. (210350J)

Assigned Lab Task

- This is an implementation of a 4- bit processor which is capable of executing 4 instructions.
- It is included with a 4-bit add/subtract unit, 3-bit adder, 3-bit program counter, k-way b-bit multiplexers, register bank, buses, program ROM, buses, slow clock, seven segment display and instruction decoder.

Assembly program and its machine code representation

==========

----Count 10 to 0

--"100010001010", --0 Move R1 10s

--"100100000001", --1 Move R2 01

--"010100000000", --2 Neg R2

--"000010100000", --3 R1<- R1+R2

--"110010000111", --4 JMP R1=0 PR7

--"110000000011", --5 JMP R0=0 PR3

--"110010000111", --6

--"110010000110" --7

==========

==========

--Add 1,2,3

"101110000001", -- 0-- MOVI R7,1

"101100000010", -- 1-- MOVI R6,2

"101010000011", -- 2-- MOVI R5,3

"001111100000", -- 3-- ADD R7,R6

"001111010000", -- 4-- ADD R7,R5

"110000000110", -- 5-- JZR R0,1

"110000000101", -- 6-- JZR R1,3

"110000000101" -- 7-- JZR R5,7

==========

All VHDL codes

*1. **Nano Processor***

==========

```
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 Nano_processor is
```

```
 Port (  res : in STD_LOGIC;
        Clk_in : in std_logic;
        OvrFlw : out STD_LOGIC;
        Zero : out STD_LOGIC;
        R7LED : out STD_LOGIC_VECTOR (3 downto 0);
        sev_out : out STD_LOGIC_VECTOR (6 downto 0);
        minus_carry: out STD_LOGIC_VECTOR(3 downto 0)
        );
end Nano_processor;

architecture Behavioral of Nano_processor is

component Instruction_decoder
 port(
    ROM_Instruction : in STD_LOGIC_VECTOR (11 downto 0);
    Select_A : out STD_LOGIC_VECTOR (2 downto 0);
    Select_B : out STD_LOGIC_VECTOR (2 downto 0);
    Enable_Register : out STD_LOGIC_VECTOR (2 downto 0);
    Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
    LoadSelect : out STD_LOGIC;
    AddSub : out STD_LOGIC;
    Jump_Flag : out STD_LOGIC;
    Jump_Address : out std_logic_vector (2 downto 0);
    Check_Jump : in STD_LOGIC_VECTOR (3 downto 0));
end component;

component Program_ROM
port(
    MemoryAddress : in STD_LOGIC_VECTOR (2 downto 0);
    Instruction: out STD_LOGIC_VECTOR (11 downto 0));
end component;

component Adder_3 --3bit Adder
   port(
        A : in STD_LOGIC_VECTOR (2 downto 0);
        S : out STD_LOGIC_VECTOR (2 downto 0);
        B : in std_logic);
end component;

component PC_3
   port(Mux_out : in STD_LOGIC_VECTOR (2 downto 0);
       Mem_S : out STD_LOGIC_VECTOR (2 downto 0);
       Res : in STD_LOGIC;
       Clk : in STD_LOGIC);
```

```
end component;


component Mux_2_way_3_bit
port(
    S : in STD_LOGIC;
    IN0 : in STD_LOGIC_VECTOR (2 downto 0);
    IN1 : in STD_LOGIC_VECTOR (2 downto 0);
    OP : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux_2_way_4_bit
port(
    S : in STD_LOGIC;
    IN0 : in STD_LOGIC_VECTOR (3 downto 0);
    IN1 : in STD_LOGIC_VECTOR (3 downto 0);
    OP : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Mux_8_way_4_bit
 port(
   S : in STD_LOGIC_VECTOR (2 downto 0);
   IN0 : in STD_LOGIC_VECTOR (3 downto 0);
   IN1 : in STD_LOGIC_VECTOR (3 downto 0);
   IN2 : in STD_LOGIC_VECTOR (3 downto 0);
   IN3 : in STD_LOGIC_VECTOR (3 downto 0);
   IN4 : in STD_LOGIC_VECTOR (3 downto 0);
   IN5 : in STD_LOGIC_VECTOR (3 downto 0);
   IN6 : in STD_LOGIC_VECTOR (3 downto 0);
   IN7 : in STD_LOGIC_VECTOR (3 downto 0);
   OP : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component AdderSubtractor_4  --Adder Substractor(4bit)
   Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
        B : in STD_LOGIC_VECTOR (3 downto 0);
        C_IN : in STD_LOGIC;
        S : inout STD_LOGIC_VECTOR (3 downto 0);
        Zero: out STD_LOGIC;
        C_OUT : out STD_LOGIC);
end component;

component Register_bank
port(
```

```
    clk  : in STD_LOGIC;
    sel  : in STD_LOGIC_VECTOR (2 downto 0);
    data : in STD_LOGIC_VECTOR (3 downto 0);
    res  : in STD_LOGIC;
    out0 : out STD_LOGIC_VECTOR (3 downto 0);
    out1 : out STD_LOGIC_VECTOR (3 downto 0);
    out2 : out STD_LOGIC_VECTOR (3 downto 0);
    out3 : out STD_LOGIC_VECTOR (3 downto 0);
    out4 : out STD_LOGIC_VECTOR (3 downto 0);
    out5 : out STD_LOGIC_VECTOR (3 downto 0);
    out6 : out STD_LOGIC_VECTOR (3 downto 0);
    out7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Slow_Clock
 port(
   clk_in : in STD_LOGIC;
   clk_out : out STD_LOGIC);
end component;

component Seven_Segment --7Segement
port(
   number : in STD_LOGIC_VECTOR (3 downto 0);
   output : out STD_LOGIC_VECTOR (6 downto 0));
end component;

signal slw_clk,OvrFlw0 : std_logic;
signal ProgramCounterOut : STD_LOGIC_VECTOR (2 downto 0);
signal RCA_3_bit_Out : std_logic_vector (2 downto 0); -- send to 2_way 3_bit mux
signal two_way_mux_out : std_logic_vector (2 downto 0); -- send to program counter
signal ROM_OUT : std_logic_vector (11 downto 0); -- send to instruction Decoder

--ROM signals
signal regSelmuxA,decoderVal,regSelmuxB,addressJMP : STD_LOGIC_VECTOR (2 downto 0);
signal Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal loadSel,addOrSub,jmpFlag : STD_LOGIC;

--Registers' signal and MUX output
signal
REGI_0,REGI_1,REGI_2,REGI_3,REGI_4,REGI_5,REGI_6,REGI_7,MUX_A,MUX_B,MUX_2_to_1_out,RCA
_out : STD_LOGIC_VECTOR (3 downto 0);
begin
--Mapping Start
```

```
Slow_Clock_0 : Slow_Clock
 port map(clk_in => Clk_in,
        clk_out => slw_clk);

PC_0 : PC_3  --Program counter
 port map (Mux_out=>two_way_mux_out,
          Mem_S=> ProgramCounterOut,
          res => res,
          Clk => slw_clk);

Adder_3_0 : Adder_3
 port map(
    A => ProgramCounterOut,
    S => RCA_3_bit_Out,
    B => '1');

Mux_2_way_3_bit_0 : Mux_2_way_3_bit
 port map(
    S => jmpFlag,
    IN0 => RCA_3_bit_Out,
    IN1 => addressJMP,
    OP => two_way_mux_out);

ROMS : Program_ROM
 port map(
 MemoryAddress => ProgramCounterOut,
 Instruction => ROM_OUT
 );

InstructionDecoder0: Instruction_Decoder
  port map(
    ROM_Instruction => ROM_OUT,
    Select_A => regSelmuxA,
    Select_B => regSelmuxB,
    Enable_Register => decoderVal,
    Immediate_Value => Immediate_Value,
    LoadSelect => loadSel,
    AddSub => addOrSub,
    Jump_Flag => jmpFlag,
    Jump_Address => addressJMP,
    Check_Jump => MUX_A);

Reg_bank0: Register_bank
 port map( clk => slw_clk,
```

```
        sel => decoderVal,
        data => MUX_2_to_1_out,
        res => res,
        out0 => REGI_0,
        out1 => REGI_1,
        out2 => REGI_2,
        out3 => REGI_3,
        out4 => REGI_4,
        out5 => REGI_5,
        out6 => REGI_6,
        out7 => REGI_7);

Mux_8_way_4_bit_A : Mux_8_way_4_bit
 port map(
 S => regSelmuxA,
 IN0 => REGI_0,
 IN1 => REGI_1,
 IN2 => REGI_2,
 IN3 => REGI_3,
 IN4 => REGI_4,
 IN5 => REGI_5,
 IN6 => REGI_6,
 IN7 => REGI_7,
 OP => MUX_A
 );

Mux_8_way_4_bit_B : Mux_8_way_4_bit
 port map(
 S => regSelmuxB,
 IN0 => REGI_0,
 IN1 => REGI_1,
 IN2 => REGI_2,
 IN3 => REGI_3,
 IN4 => REGI_4,
 IN5 => REGI_5,
 IN6 => REGI_6,
 IN7 => REGI_7,
 OP => MUX_B
 );

Mux_2_way_4_bit_0 : Mux_2_way_4_bit
   port map(
       S => loadSel,
       IN0 => RCA_out,
```

```
        IN1 => Immediate_Value,
        OP => MUX_2_to_1_out);

    RCA_0 : AdderSubtractor_4
     port map(
     A => MUX_B,
     B => MUX_A,
     C_IN => addOrSub,
     C_OUT => OvrFlw0,
     S => RCA_out,
     Zero => Zero
     );

    svenSeg_0 : Seven_Segment
     port map(
     number => REGI_7,
     output => sev_out );

    R7LED <= REGI_7;
    OvrFlw <= OvrFlw0;
    minus_carry<= "1110";
    end Behavioral;
```

==========

2. ***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_Clock is
       Port (  clk_in : in STD_LOGIC;
             clk_out : out STD_LOGIC);
```

```
end Slow_Clock;

architecture Behavioral of Slow_Clock 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 = 2 ) then
      --if (count = 25000000) then
         clk_status <= not clk_status;
         clk_out <= clk_status;
         count <= 1;
      end if;
    end if;
 end process;
end Behavioral;
```

==========

3. ***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_3 is
   Port ( Mux_out : in STD_LOGIC_VECTOR (2 downto 0):="000";--D
          Clk : in STD_LOGIC;
          Res : in STD_LOGIC;
          Mem_s : out STD_LOGIC_VECTOR (2 downto 0));--Q
```

```
end PC_3;

architecture Behavioral of PC_3 is
component D_FF
   Port ( D : in STD_LOGIC;
          Res : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Q : out STD_LOGIC;
          Qbar : out STD_LOGIC);
end component;
begin

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

D_FF1: D_FF
   port map(
   D=>Mux_out(1),
   Q=>Mem_s(1),
   Res=>Res,
   Clk=>clk
   );

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

end Behavioral;
```

==========

4. D Flip Flop

==========

```
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;
          Qbar : out STD_LOGIC);
end D_FF;

architecture Behavioral of D_FF is

begin
   process (Clk) begin
      if (rising_edge(Clk)) then
         if Res='1' then --reset
            Q<='0';
            Qbar<='1';
         else
            Q<=D; --delay flipflop
            Qbar<=not D;
         end if;
      end if;
   end process;

end Behavioral;
```

==========

5. 3 bit 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 Adder_3 is
   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
          B : in STD_LOGIC;
          S : out STD_LOGIC_VECTOR (2 downto 0));
end Adder_3;

architecture Behavioral of Adder_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,c : STD_LOGIC;
begin
FA_0 :FA
   port map(
      A => A(0),
      B => B,
      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
        );

    end Behavioral;
```

==========

6. 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 : out STD_LOGIC
    );
end component;

signal HA0_C,HA0_S,HA1_C,HA1_S : STD_LOGIC;

begin
HA_0 : HA
    port map(
        A => A,
        B => B,
        S => HA0_S,
        C_OUT => HA0_C
    );
HA_1 : HA
        port map(
            A => HA0_S,
            B => C_IN,
            S => HA1_S,
            C_OUT => HA1_C
        );
C_OUT <= HA0_C OR HA1_C;
S <= HA1_S;

end Behavioral;
```

==========

7. 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 : out STD_LOGIC
          );
end HA;

architecture Behavioral of HA is

begin
 S <= A XOR B;
 C_OUT <= A AND B;

end Behavioral;
```

==========

8. ***Mux-2 way 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 Mux_2_way_3_bit is
   Port ( IN0 : in STD_LOGIC_VECTOR (2 downto 0);
          IN1 : in STD_LOGIC_VECTOR (2 downto 0);
          S : in STD_LOGIC;
          OP : out STD_LOGIC_VECTOR (2 downto 0)
          );
```

```
end Mux_2_way_3_bit;

architecture Behavioral of Mux_2_way_3_bit is
component Mux_2_to_1
   Port ( D0 : in STD_LOGIC;
          D1 : in STD_LOGIC;
          S : in STD_LOGIC;
          Y : out STD_LOGIC
          );
end component;

begin
Mux_2_to_1_0 :Mux_2_to_1
   port map( D0 => IN0(0),
             D1 => IN1(0),
             S => S,
             Y => OP(0)
              );

Mux_2_to_1_1 :Mux_2_to_1
   port map( D0 => IN0(1),
             D1 => IN1(1),
             S => S,
             Y => OP(1)
              );

Mux_2_to_1_2 :Mux_2_to_1
   port map( D0 => IN0(2),
             D1 => IN1(2),
             S => S,
             Y => OP(2)
              );

end Behavioral;
```

==========

9. Mux 2 to 1

==========

```
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_2_to_1 is
      Port ( D0 : in STD_LOGIC;
           D1 : in STD_LOGIC;
           S : in STD_LOGIC;
           Y : out STD_LOGIC);
end Mux_2_to_1;

architecture Behavioral of mux_2_to_1 is

begin
Y <= (D0 AND NOT(S)) OR (D1 AND S);
end Behavioral;
```

==========

10.Program 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 Program_ROM is
   Port ( MemoryAddress : in STD_LOGIC_VECTOR (2 downto 0);
        Instruction : out STD_LOGIC_VECTOR (11 downto 0));
```

```
end Program_ROM;

architecture Behavioral of Program_ROM is
type rom_type is array (0 to 7) of STD_LOGIC_VECTOR( 11 downto 0);
signal instructionROM : rom_type := (

   ----Count 10 to 0
   --"100010001010", --0  Move R1 10s
   --"100100000001", --1  Move R2 01
   --"010100000000", --2  Neg R2
   --"000010100000", --3  R1<- R1+R2
   --"110010000111", --4  JMP R1=0 PR7
   --"110000000011", --5  JMP R0=0 PR3
   --"110010000111", --6
   --"110010000110"  --7


      --Add 1,2,3
      "101110000001", -- 0-- MOVI R7,1
      "101100000010", -- 1-- MOVI R6,2
      "101010000011", -- 2-- MOVI R5,3
      "001111100000", -- 3-- ADD R7,R6
      "001111010000", -- 4-- ADD R7,R5
      "110000000110", -- 5-- JZR R0,1
      "110000000101", -- 6-- JZR R1,3
      "110000000101"  -- 7-- JZR R5,7
);

begin
Instruction <= instructionROM(to_integer(unsigned(MemoryAddress)));

end Behavioral;
```

==========

11.Instruction Decoder

==========

```
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 Instruction_decoder is
   Port ( ROM_Instruction : in STD_LOGIC_VECTOR (11 downto 0); -- instruction from program ROM
          Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);  -- register check for jump
          Select_A : out STD_LOGIC_VECTOR (2 downto 0);   --send the register to select to 1st MUX
          Select_B : out STD_LOGIC_VECTOR (2 downto 0);  --send the register to select to 1st MUX
          Enable_Register : out STD_LOGIC_VECTOR (2 downto 0);  --enable correct register
          AddSub : out STD_LOGIC; --if want to add enable 0, else if you want 2's complement enable 1
          Jump_Flag : out STD_LOGIC; --to indicate whether jump instruction is enabled or not
          Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);  --address to which we need to jump
          LoadSelect : out STD_LOGIC;  --enable the mux connected to register bank
          Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0));  --immediate value
end Instruction_decoder;

architecture Behavioral of Instruction_Decoder is

component Decoder_2_to_4
   Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
          EN : in STD_LOGIC;
          D : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal ADD, NEG, MOVI, JZR :STD_LOGIC;

begin
   Decoder_2_to_4_0: Decoder_2_to_4
   port map(
   I(0)=>ROM_Instruction(10),
   I(1)=>ROM_Instruction(11),
   EN=>'1',
   D(0)=>ADD,
   D(1)=>NEG,
   D(2)=>MOVI,
   D(3)=>JZR
   );

Select_A <= ROM_Instruction(9 downto 7);
Select_B <= ROM_Instruction(6 downto 4);
```

```
    Enable_Register <= ROM_Instruction(9 downto 7);
    AddSub <= NEG ;
    Jump_Flag <= JZR AND (NOT (Check_Jump(0) OR Check_Jump(1) OR Check_Jump(2) OR
    Check_Jump(3))) ;
    Jump_Address <= ROM_Instruction(2 downto 0);
    LoadSelect <= MOVI;
    Immediate_Value <= ROM_Instruction(3 downto 0);


    end Behavioral;
```

==========

12. 2-4 Decoder

==========

```
    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 Decoder_2_to_4 is
       Port ( I : in STD_LOGIC_vector(1 downto 0);
            EN : in STD_LOGIC;
            D : out STD_LOGIC_VECTOR (3 downto 0)
            );
    end Decoder_2_to_4;

    architecture Behavioral of Decoder_2_to_4 is

    begin

    D(0)<=not(I(1)) and not(I(0)) and EN;
    D(1)<=not(I(1)) and I(0) and EN;
    D(2)<=I(1) and not(I(0)) and EN;
    D(3)<=I(1) and I(0) and EN;
```

```
end Behavioral;
```

==========

13.Register 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 Register_bank is
 Port ( clk: in STD_LOGIC;
       sel : in STD_LOGIC_VECTOR (2 downto 0);
       data : in STD_LOGIC_VECTOR (3 downto 0);
       res : in STD_LOGIC;
       out0 : out STD_LOGIC_VECTOR (3 downto 0);
       out1 : out STD_LOGIC_VECTOR (3 downto 0);
       out2 : out STD_LOGIC_VECTOR (3 downto 0);
       out3 : out STD_LOGIC_VECTOR (3 downto 0);
       out4 : out STD_LOGIC_VECTOR (3 downto 0);
       out5 : out STD_LOGIC_VECTOR (3 downto 0);
       out6 : out STD_LOGIC_VECTOR (3 downto 0);
       out7 : out STD_LOGIC_VECTOR (3 downto 0)
       );
end Register_bank;

architecture Behavioral of Register_bank is
signal enable : std_logic_vector(7 downto 0);

component Decoder_3_to_8
port (
   I: in std_logic_vector(2 downto 0);
   D: out std_logic_vector(7 downto 0)
```

```
   );
end component;

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

begin
Decoder_3to_8_reg : Decoder_3_to_8
   port map (
      I => sel,
      D=> enable
      );

Reg_0 : Reg
   port map(
      D => "0000",
      En => '0', --only for read
      Q => out0,
      Res =>res,
      Clk => clk
      );

Reg_1 : Reg
   port map(
      D => data,
      En => enable(1),
      Q => out1,
      Res =>res,
      Clk => clk
      );

Reg_2 : Reg
   port map(
      D => data,
      En => enable(2),
      Q => out2,
      Res =>res,
```

```
      Clk => clk
      );

Reg_3 : Reg
   port map(
      D => data,
      En => enable(3),
      Q => out3,
      Res =>res,
      Clk => clk
      );

Reg_4 : REG
   port map(
      D => data,
      En => enable(4),
      Q => out4,
      Res =>res,
      Clk => clk
      );

Reg_5 : Reg
   port map(
      D => data,
      En => enable(5),
      Q => out5,
      Res =>res,
      Clk => clk
      );

Reg_6 : REG
   port map(
      D => data,
      En => enable(6),
      Q => out6,
      Res =>res,
      Clk => clk
      );

Reg_7 : Reg
   port map(
      D => data,
      En =>enable(7),
      Q => out7,
```

```
        Res =>res,
        Clk => clk
        );

    end Behavioral;
```

==========

14. 3 to 8 Decoder

==========

```
    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 Decoder_3_to_8 is
       Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
            D : out STD_LOGIC_VECTOR (7 downto 0));
    end Decoder_3_to_8;

    architecture Behavioral of Decoder_3_to_8 is
    component Decoder_2_to_4
       Port ( I : in STD_LOGIC_vector(1 downto 0);
            EN : in STD_LOGIC;
            D : out STD_LOGIC_VECTOR (3 downto 0)
            );
    end component;
    signal EN0,EN1,EN:std_logic;
    begin
    Decoder_2_to_40:Decoder_2_to_4
          port map(I=>I(1 downto 0),
                EN=>EN0,
                D=>D(3 downto 0)
                );
```

```
    Decoder_2_to_41:Decoder_2_to_4
            port map(I=>I(1 downto 0),
                  EN=>EN1,
                  D=>D(7 downto 4)
                  );
    EN<='1';
    EN0<=EN and not(I(2));
    EN1<=EN and I(2);

    end Behavioral;
```

==========

15.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);
            Res : in STD_LOGIC;
            Clk : in STD_LOGIC;
            Q : out STD_LOGIC_VECTOR (3 downto 0);
            En : in STD_LOGIC
            );
    end Reg;

    architecture Behavioral of REG is
    component D_FF_EN
       Port ( D : in STD_LOGIC;
          Res : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Q : out STD_LOGIC;
```

```
        Qbar : out STD_LOGIC;
        En : in STD_LOGIC
        );
    end component;
    begin
    D_FF_EN0 : D_FF_EN
    port map(
        D => D(0),
        En => En,
        Clk => Clk,
        Res => Res,
        Q => Q(0)
        );

    D_FF_EN1 : D_FF_EN
    port map(
        D => D(1),
        En => En,
        Clk => Clk,
        Res => Res,
        Q => Q(1)
        );

    D_FF_EN2 : D_FF_EN
     port map(
        D => D(2),
        En => En,
        Clk => Clk,
        Res => Res,
        Q => Q(2)
        );

    D_FF_EN3 : D_FF_EN
     port map(
        D => D(3),
        En => En,
        Clk => Clk,
        Res => Res,
        Q => Q(3)
        );

    end Behavioral;
```

==========

16. 8 – way 4 bit Multiplexer

==========

```
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_8_way_4_bit is
   Port ( IN0 : in STD_LOGIC_VECTOR (3 downto 0);
          IN1 : in STD_LOGIC_VECTOR (3 downto 0);
          IN2 : in STD_LOGIC_VECTOR (3 downto 0);
          IN3 : in STD_LOGIC_VECTOR (3 downto 0);
          IN4 : in STD_LOGIC_VECTOR (3 downto 0);
          IN5 : in STD_LOGIC_VECTOR (3 downto 0);
          IN6 : in STD_LOGIC_VECTOR (3 downto 0);
          IN7 : in STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC_VECTOR (2 DOWNTO 0);
          OP : out STD_LOGIC_VECTOR (3 downto 0)
          );
end Mux_8_way_4_bit;
architecture Behavioral of Mux_8_way_4_bit is
component Mux_8_to_1
--    Port ( D0 : in STD_LOGIC;
--           D1 : in STD_LOGIC;
--           S : in STD_LOGIC;
--           Y : out STD_LOGIC);
  Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
         D : in STD_LOGIC_VECTOR (7 downto 0);
         Y : out STD_LOGIC
         );
end component;

begin
Mux_8_to_1_0 :Mux_8_to_1
```

```
   port map( D(0) => IN0(0),
             D(1) => IN1(0),
             D(2) => IN2(0),
             D(3) => IN3(0),
             D(4) => IN4(0),
             D(5) => IN5(0),
             D(6) => IN6(0),
             D(7) => IN7(0),
             S => S,
             Y => OP(0)
             );
Mux_8_to_1_1 :Mux_8_to_1
   port map( D(0) => IN0(1),
             D(1) => IN1(1),
             D(2) => IN2(1),
             D(3) => IN3(1),
             D(4) => IN4(1),
             D(5) => IN5(1),
             D(6) => IN6(1),
             D(7) => IN7(1),
             S => S,
             Y => OP(1)
             );
Mux_8_to_1_2 :Mux_8_to_1
   port map( D(0) => IN0(2),
             D(1) => IN1(2),
             D(2) => IN2(2),
             D(3) => IN3(2),
             D(4) => IN4(2),
             D(5) => IN5(2),
             D(6) => IN6(2),
             D(7) => IN7(2),
             S => S,
             Y => OP(2)
             );
Mux_8_to_1_3 :Mux_8_to_1
   port map( D(0) => IN0(3),
             D(1) => IN1(3),
             D(2) => IN2(3),
             D(3) => IN3(3),
             D(4) => IN4(3),
             D(5) => IN5(3),
             D(6) => IN6(3),
             D(7) => IN7(3),
```

```
            S => S,
            Y => OP(3)
            );

    end Behavioral;
```

==========

17. 8 to 1 Multiplexer

==========

```
    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_8_to_1 is
       Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
            D : in STD_LOGIC_VECTOR (7 downto 0);
            Y : out STD_LOGIC);
    end Mux_8_to_1;

    architecture Behavioral of Mux_8_to_1 is

    COMPONENT Decoder_3_to_8
       Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
            D : out STD_LOGIC_VECTOR (7 downto 0));
    end COMPONENT;

    SIGNAL S0 : STD_LOGIC_VECTOR (2 downto 0);
    SIGNAL Y0, F : STD_LOGIC_VECTOR (7 downto 0);

    begin
       Decoder_3_to_8_0 : Decoder_3_to_8
          PORT MAP(
             I => S,
```

```
            D => Y0
        );

    F(0) <= D(0) AND Y0(0);
    F(1) <= D(1) AND Y0(1);
    F(2) <= D(2) AND Y0(2);
    F(3) <= D(3) AND Y0(3);
    F(4) <= D(4) AND Y0(4);
    F(5) <= D(5) AND Y0(5);
    F(6) <= D(6) AND Y0(6);
    F(7) <= D(7) AND Y0(7);

Y <=( F(0) OR F(1) OR F(2) OR F(3) OR F(4) OR F(5) OR F(6) OR F(7) );

end Behavioral;
```

==========

18. 2 way 4 bit Multiplexer

==========

```
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_2_way_4_bit is
    Port ( IN0 : in STD_LOGIC_VECTOR (3 downto 0);
           IN1 : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC;
           OP : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2_way_4_bit;
architecture Behavioral of Mux_2_way_4_bit is
component Mux_2_to_1
    Port ( D0 : in STD_LOGIC;
           D1 : in STD_LOGIC;
```

```
        S : in STD_LOGIC;
        Y : out STD_LOGIC);
    end component;
    begin
    Mux_2_to_1_0 :Mux_2_to_1
      port map( D0 => IN0(0),
            D1 => IN1(0),
            S => S,
            Y => OP(0)
              );

    Mux_2_to_1_1 :Mux_2_to_1
      port map( D0 => IN0(1),
            D1 => IN1(1),
            S => S,
            Y => OP(1)
              );

    Mux_2_to_1_2 :Mux_2_to_1
      port map( D0 => IN0(2),
            D1 => IN1(2),
            S => S,
            Y => OP(2)
              );
    Mux_2_to_1_3 :Mux_2_to_1
      port map( D0 => IN0(3),
            D1 => IN1(3),
            S => S,
            Y => OP(3)
              );

    end Behavioral;
```

==========

19.Adder/Subtractor

==========

```
    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 AdderSubtractor_4 is
   Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          C_IN : in STD_LOGIC;
          S : out STD_LOGIC_VECTOR (3 downto 0);
          Zero: out STD_LOGIC;
          C_OUT : out STD_LOGIC);
end AdderSubtractor_4;

architecture Behavioral of AdderSubtractor_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, B0,B1,B2,B3: STD_LOGIC;
signal S_out:STD_LOGIC_VECTOR (3 downto 0);

begin

FA_0: FA
   PORT MAP(
      A=>A(0),
      B=>B0,
      C_IN=>C_IN,
      S=>S_out(0),
      C_OUT=>FA0_C);

FA_1: FA
   PORT MAP(
      A=>A(1),
      B=>B1,
      C_IN=>FA0_C,
```

```
        S=>S_out(1),
        C_OUT=>FA1_C);

    FA_2: FA
      PORT MAP(
        A=>A(2),
        B=>B2,
        C_IN=>FA1_C,
        S=>S_out(2),
        C_OUT=>FA2_C);

    FA_3: FA
      PORT MAP(
        A=>A(3),
        B=>B3,
        C_IN=>FA2_C,
        S=>S_out(3),
        C_OUT=>FA3_C);

        B0<=B(0) XOR C_IN;
        B1<=B(1) XOR C_IN;
        B2<=B(2) XOR C_IN;
        B3<=B(3) XOR C_IN;
        C_OUT<=FA3_C ;
        Zero<= NOT(S_out(0) OR S_out(1) OR S_out(2) OR S_out(3));
    S<=S_out;
    end Behavioral;
```

==========

20.Seven 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 Seven_Segment is
 Port ( number : in STD_LOGIC_VECTOR (3 downto 0);
 output : out STD_LOGIC_VECTOR (6 downto 0);
 minus_carry : out std_logic);
end Seven_Segment;
architecture Behavioral of Seven_Segment 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

 output <= sevenSegment_ROM(to_integer(unsigned(number)));
 minus_carry <= number(3);

end Behavioral;
```

==========

Simulation Codes

1. ***Nano processor***

==========

```
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 Nano_processor
 Port ( res : in STD_LOGIC;
       Clk_in : in std_logic;
       OvrFlw : out STD_LOGIC;
       Zero : out STD_LOGIC;
       R7LED : out STD_LOGIC_VECTOR (3 downto 0);
       sev_out : out STD_LOGIC_VECTOR (6 downto 0);
       minus_carry : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal res,OvrFlw,Zero:std_logic;
signal minus_carry : STD_LOGIC_VECTOR (3 downto 0);
signal R7LED:STD_LOGIC_VECTOR (3 downto 0);
signal sev_out:STD_LOGIC_VECTOR (6 downto 0);
signal Clk_in:std_logic:='0';
begin
UUT: Nano_processor
   port map(
       res=>res,
       Clk_in=>Clk_in,
       Ovrflw=>Ovrflw,
       Zero=>Zero,
       R7LED=>R7LED,
       sev_out=>sev_out,
       minus_carry=>minus_carry
       );
```

```
    process
       begin
       wait for 5ns;
       Clk_in<=not(Clk_in);
    end process;
    process
       begin
       res<='1';
       wait for 100ns;
       res<='0';
       wait;

    end process;
    end Behavioral;
```

==========

2. ***Adder Subtractor***

==========

```
    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 AdderSubtractor_Sim is
    --  Port ( );
    end AdderSubtractor_Sim;

    architecture Behavioral of AdderSubtractor_Sim is
    component AdderSubtractor_4
       Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
            B : in STD_LOGIC_VECTOR (3 downto 0);
            C_IN : in STD_LOGIC;
            S : inout STD_LOGIC_VECTOR (3 downto 0);
            Zero: out STD_LOGIC;
```

```
        C_OUT : out STD_LOGIC);
end component;

signal A,B,S : STD_LOGIC_VECTOR (3 downto 0);
signal SET, OVERFLOW, ZERO : STD_LOGIC;

begin
UUT: AdderSubtractor_4
   PORT MAP(
   A=>A,
   B=>B,
   C_IN=>SET,
   S=>S,
   Zero=>ZERO,
   C_OUT=>OVERFLOW);

   PROCESS
   BEGIN
--   index number= 210621R
--   Binary form= 0011 0011 0110 1011 1101


   SET<='0';
   A<="0011";
   B<="0011";
   WAIT FOR 100NS;

   SET<='0';
   A<="0011";
   B<="0110";
   WAIT FOR 100NS;

   SET<='0';
   A<="0110";
   B<="1011";
   WAIT FOR 100NS;

   SET<='0';
   A<="1011";
   B<="1101";
   WAIT FOR 100NS;

   SET<='1';
   A<="0011";
```

```
    B<="0011";
    WAIT FOR 100NS;

    SET<='1';
    A<="0011";
    B<="0110";
    WAIT FOR 100NS;

    SET<='1';
    A<="0110";
    B<="1011";
    WAIT FOR 100NS;

    SET<='1';
    A<="1011";
    B<="1101";
    WAIT FOR 100NS;
    WAIT;
    END PROCESS;
end Behavioral;
```

==========

3. ***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 Adder_3_Sim is
--  Port ( );
end Adder_3_Sim;

architecture Behavioral of Adder_3_Sim is
```

```
    component Adder_3
      port(
        A : in STD_LOGIC_VECTOR (2 downto 0);
        B : in STD_LOGIC;
        S : out STD_LOGIC_VECTOR (2 downto 0)
      );
    end component;
    signal b : STD_LOGIC :='1';
    signal a,s :STD_LOGIC_VECTOR (2 downto 0);
    begin
    UUT : Adder_3
      port map(
        A => a,
        B => b,
        S => s
      );
      process
      begin
        a <= "000";
        wait for 100ns;
        a <= "001";
        wait for 100ns;
        a <= "010";
        wait for 100ns;
        a <= "011";
        wait for 100ns;
        a <= "100";
        wait for 100ns;
        a <= "101";
        wait for 100ns;
        a <= "110";
        wait for 100ns;
        a <= "111";
        wait ;
      end process;
    end Behavioral;
```

==========

4. Instruction Decoder

==========

```
    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_Sim is
--  Port ( );
end InstructionDecoder_Sim;

architecture Behavioral of InstructionDecoder_Sim is
component Instruction_decoder
   Port ( ROM_Instruction : in STD_LOGIC_VECTOR (11 downto 0);
          Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
          Select_A : out STD_LOGIC_VECTOR (2 downto 0);
          Select_B : out STD_LOGIC_VECTOR (2 downto 0);
          Enable_Register : out STD_LOGIC_VECTOR (2 downto 0);
          AddSub : out STD_LOGIC;
          Jump_Flag : out STD_LOGIC;
          Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);
          LoadSelect : out STD_LOGIC;
          Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal ROM_Instruction : STD_LOGIC_VECTOR (11 downto 0);
signal Check_Jump, Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Select_A, Select_B, Enable_Register, Jump_Address : STD_LOGIC_VECTOR (2 downto 0);
signal AddSub, Jump_Flag,LoadSelect : STD_LOGIC;

begin
UUT: Instruction_decoder
   PORT MAP(  ROM_Instruction=>ROM_Instruction,
              Check_Jump=>Check_Jump,
              Select_A=>Select_A,
              Select_B =>Select_B,
              Enable_Register=>Enable_Register,
              AddSub=>AddSub,
              Jump_Flag=>Jump_Flag,
```

```
            Jump_Address=>Jump_Address,
            LoadSelect=>LoadSelect,
            Immediate_Value=>Immediate_Value
        );

    process
    begin
        Check_Jump<="1110";
        ROM_Instruction<="001011110000";
        wait for 50ns;

        Check_Jump<="1010";
        ROM_Instruction<="011110000000";
        wait for 50ns;

        Check_Jump<="1011";
        ROM_Instruction<="101100000111";
        wait for 50ns;

        Check_Jump<="0000";
        ROM_Instruction<="110110000100";
        wait for 50ns;

        Check_Jump<="1010";
        ROM_Instruction<="111110000100";

        wait for 50ns;
    wait;

    end process;
    end Behavioral;
```

==========

5. ***Instruction Decoder***

==========

```
    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_Sim is
--  Port ( );
end InstructionDecoder_Sim;

architecture Behavioral of InstructionDecoder_Sim is
component Instruction_decoder
   Port ( ROM_Instruction : in STD_LOGIC_VECTOR (11 downto 0);
          Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
          Select_A : out STD_LOGIC_VECTOR (2 downto 0);
          Select_B : out STD_LOGIC_VECTOR (2 downto 0);
          Enable_Register : out STD_LOGIC_VECTOR (2 downto 0);
          AddSub : out STD_LOGIC;
          Jump_Flag : out STD_LOGIC;
          Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);
          LoadSelect : out STD_LOGIC;
          Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal ROM_Instruction : STD_LOGIC_VECTOR (11 downto 0);
signal Check_Jump, Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Select_A, Select_B, Enable_Register, Jump_Address : STD_LOGIC_VECTOR (2 downto 0);
signal AddSub, Jump_Flag,LoadSelect : STD_LOGIC;

begin
UUT: Instruction_decoder
   PORT MAP(  ROM_Instruction=>ROM_Instruction,
              Check_Jump=>Check_Jump,
              Select_A=>Select_A,
              Select_B =>Select_B,
              Enable_Register=>Enable_Register,
              AddSub=>AddSub,
              Jump_Flag=>Jump_Flag,
              Jump_Address=>Jump_Address,
              LoadSelect=>LoadSelect,
              Immediate_Value=>Immediate_Value
   );
```

```
    process
    begin
       Check_Jump<="1110";
       ROM_Instruction<="001011110000";
       wait for 50ns;

       Check_Jump<="1010";
       ROM_Instruction<="011110000000";
       wait for 50ns;

       Check_Jump<="1011";
       ROM_Instruction<="101100000111";
       wait for 50ns;

       Check_Jump<="0000";
       ROM_Instruction<="110110000100";
       wait for 50ns;

       Check_Jump<="1010";
       ROM_Instruction<="111110000100";

       wait for 50ns;
    wait;

    end process;
    end Behavioral;
```

==========

6. 2 way 3 bit Multiplexer

==========

```
    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_2_way_3_bit_Sim is
--  Port ( );
end Mux_2_way_3_bit_Sim;

architecture Behavioral of Mux_2_way_3_bit_Sim is
COMPONENT Mux_2_way_3_bit
   Port ( IN0 : in STD_LOGIC_VECTOR (2 downto 0);
          IN1 : in STD_LOGIC_VECTOR (2 downto 0);
          S : in STD_LOGIC;
          OP : out STD_LOGIC_VECTOR (2 downto 0));
END COMPONENT;

SIGNAL IN0,IN1,OP :STD_LOGIC_VECTOR (2 downto 0);
SIGNAL S : STD_LOGIC;

begin
UUT : Mux_2_way_3_bit
   PORT MAP(
          IN0 => IN0,
          IN1 => IN1,
          S => S,
          OP => OP
          );
PROCESS
 BEGIN
   IN0 <= "000";
   IN1 <= "001";
   S <= '0';
   WAIT FOR 50ns;

   IN0 <= "100";
   IN1 <= "101";
   S <= '1';
   WAIT FOR 50ns;

   S <= '0';
   WAIT FOR 50ns;

   IN0 <= "110";
   IN1 <= "010";
   S <= '0';
   WAIT FOR 50ns;
```

```
    END PROCESS;

    end Behavioral;
```

==========

7. ***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_Sim is
    --  Port ( );
    end PC_Sim;

    architecture Behavioral of PC_Sim is
    Component PC_3
       Port ( Mux_out : in STD_LOGIC_VECTOR (2 downto 0):="000";--D
              Clk : in STD_LOGIC;
              Res : in STD_LOGIC;
              Mem_s : out STD_LOGIC_VECTOR (2 downto 0));--Q
    end component;

    signal Mux_out,Mem_s:std_logic_vector(2 downto 0);
    signal Clk:std_logic:='0';
    signal Res:std_logic;

    begin
    UUT: PC_3
       port map(
          Mux_out=>Mux_out,
          Clk=>Clk,
          Res=>Res,
```

```
      Mem_s=>Mem_s
   );

process
 begin
   wait for 20ns;
   Clk<=not(Clk);
end process;

process
 begin
   --wait for 20ns;
   Res<='1';
   wait for 40ns;
   Res<='0';
   Mux_out<="001";
   wait for 40ns;

   Mux_out<="010";
   wait for 40ns;

   mux_out<="011";
   wait for 40ns;

   Mux_out<="100";
   wait for 40ns;

   Mux_out<="101";
   wait for 40ns;

   Mux_out<="110";
   wait for 40ns;

   Mux_out<="111";
   wait for 40ns;

   Mux_out<="000";
   wait for 40ns;

   Mux_out<="001";
   wait for 40ns;

   Mux_out<="010";
   wait for 40ns;
```

```
        mux_out<="011";
        wait for 40ns;

        Res<='1';
        wait for 40ns;

        Res<='0';
        Mux_out<="001";
        wait for 40ns;

        Mux_out<="010";
        wait for 40ns;

        mux_out<="011";
        wait;

    end process;

    end Behavioral;
```

==========

8. Program 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 ProgramROM_Sim is
    --  Port ( );
    end ProgramROM_Sim;

    architecture Behavioral of ProgramROM_Sim is
```

```
    component Program_Rom
        Port ( MemoryAddress : in STD_LOGIC_VECTOR (2 downto 0);
               Instruction : out STD_LOGIC_VECTOR (11 downto 0));
    end component;

    signal Mem :STD_LOGIC_VECTOR (2 downto 0);
    signal Ins :STD_LOGIC_VECTOR (11 downto 0);

    begin
    UUT : Program_Rom
       port map(
          MemoryAddress => Mem,
          Instruction => Ins
       );
    process
       begin
       Mem <= "000";
       wait for 100ns;
       Mem <= "001";
       wait for 100ns;
       Mem <= "010";
       wait for 100ns;
       Mem <= "011";
       wait for 100ns;
       Mem <= "100";
       wait for 100ns;
       Mem <= "101";
       wait for 100ns;
       Mem <= "110";
       wait for 100ns;
       Mem <= "111";
       wait;

    end process;

    end Behavioral;
```

==========

9. ***Register Bank***

==========

```
    library IEEE;
```

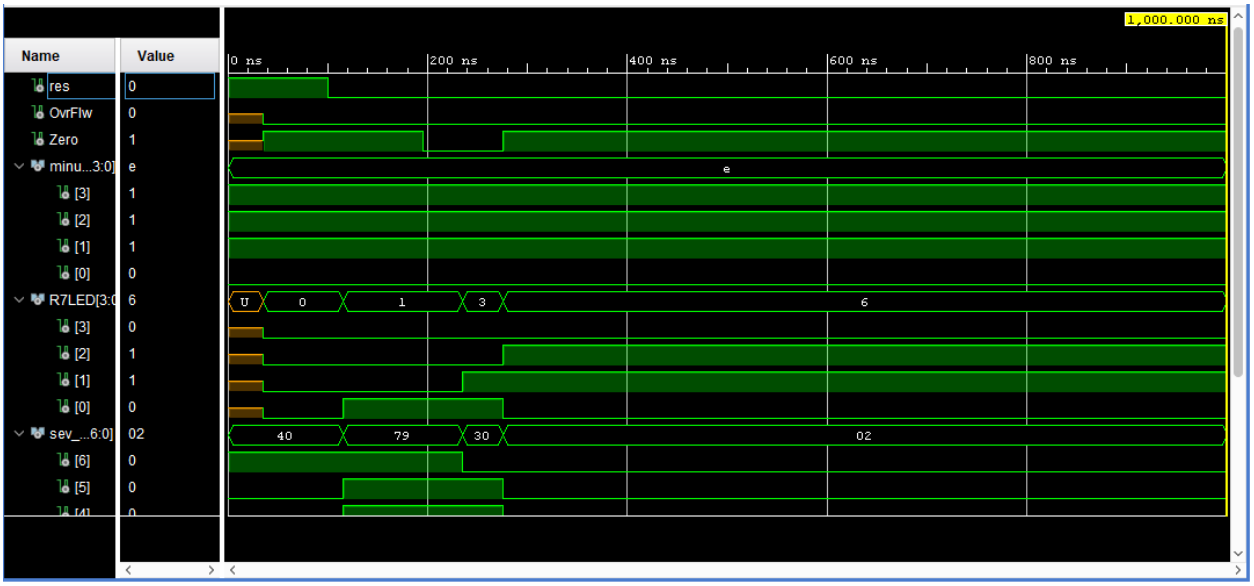
```
use IEEE.STD_LOGIC_1164.ALL;

-- 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_Sim is
--  Port ( );
end RegisterBank_Sim;

architecture Behavioral of RegisterBank_Sim is

component Register_bank
 Port ( clk: in STD_LOGIC;
      sel : in STD_LOGIC_VECTOR (2 downto 0);
      data : in STD_LOGIC_VECTOR (3 downto 0);
      res : in STD_LOGIC;
      out0 : out STD_LOGIC_VECTOR (3 downto 0);
      out1 : out STD_LOGIC_VECTOR (3 downto 0);
      out2 : out STD_LOGIC_VECTOR (3 downto 0);
      out3 : out STD_LOGIC_VECTOR (3 downto 0);
      out4 : out STD_LOGIC_VECTOR (3 downto 0);
      out5 : out STD_LOGIC_VECTOR (3 downto 0);
      out6 : out STD_LOGIC_VECTOR (3 downto 0);
      out7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal res: std_logic;
signal clk:std_logic:='0';
signal sel:std_logic_vector(2 downto 0);
signal data:std_logic_vector(3 downto 0);
signal out0,out1,out2,out3,out4,out5,out6,out7:std_logic_vector(3 downto 0);

begin
UUT: Register_bank
   port map(
      clk=>clk,
      res=>res,
      data=>data,
```

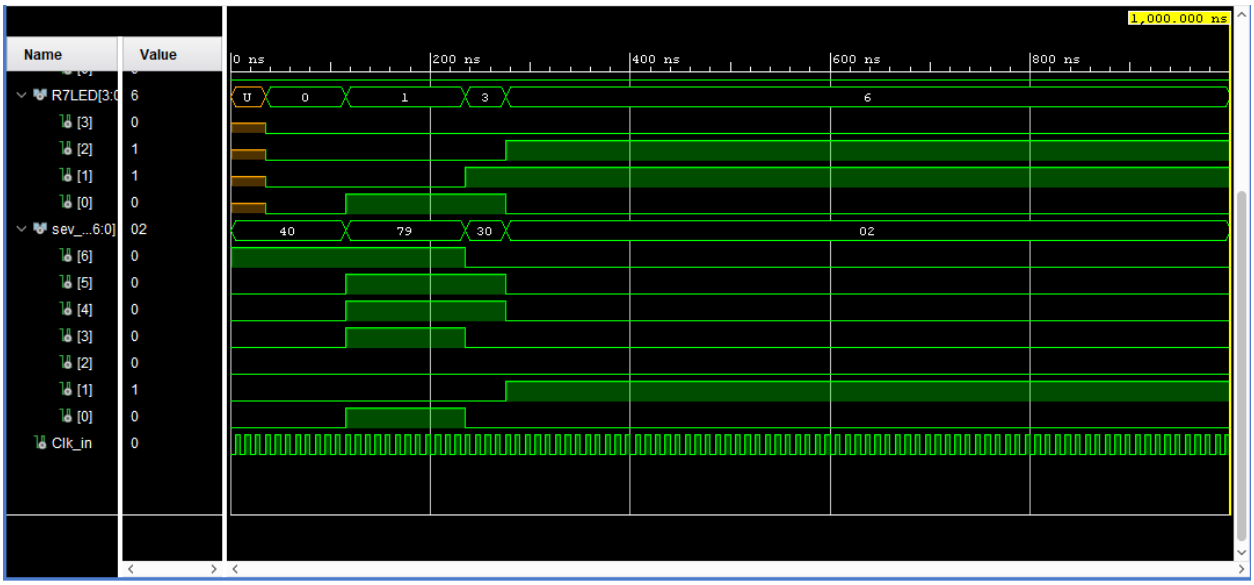
```
        sel=>sel,
        out0=>out0,
        out1=>out1,
        out2=>out2,
        out3=>out3,
        out4=>out4,
        out5=>out5,
        out6=>out6,
        out7=>out7
    );
process
begin
        wait for 20ns;
        Clk<=not(Clk);
end process;

process
 begin
    res<='1';
    wait for 40ns;
    res<='0';
    data<="0101";
    sel<="000";
    wait for 40ns;
    sel<="001";
    wait for 40ns;
    sel<="010";
    wait for 40ns;
    sel<="011";
    wait for 40ns;
    sel<="100";
    wait for 40ns;
    sel<="101";
    wait for 40ns;
    sel<="111";
    wait;

end process;

end Behavioral;
```

==========

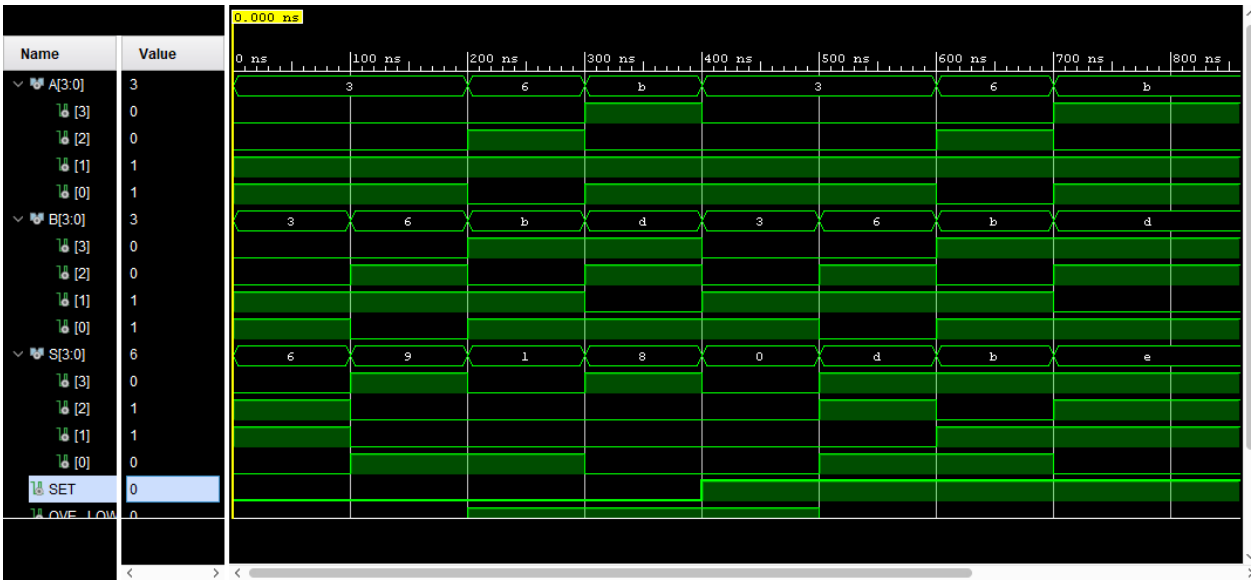
Timing Diagrams

1. *Nano Processor*

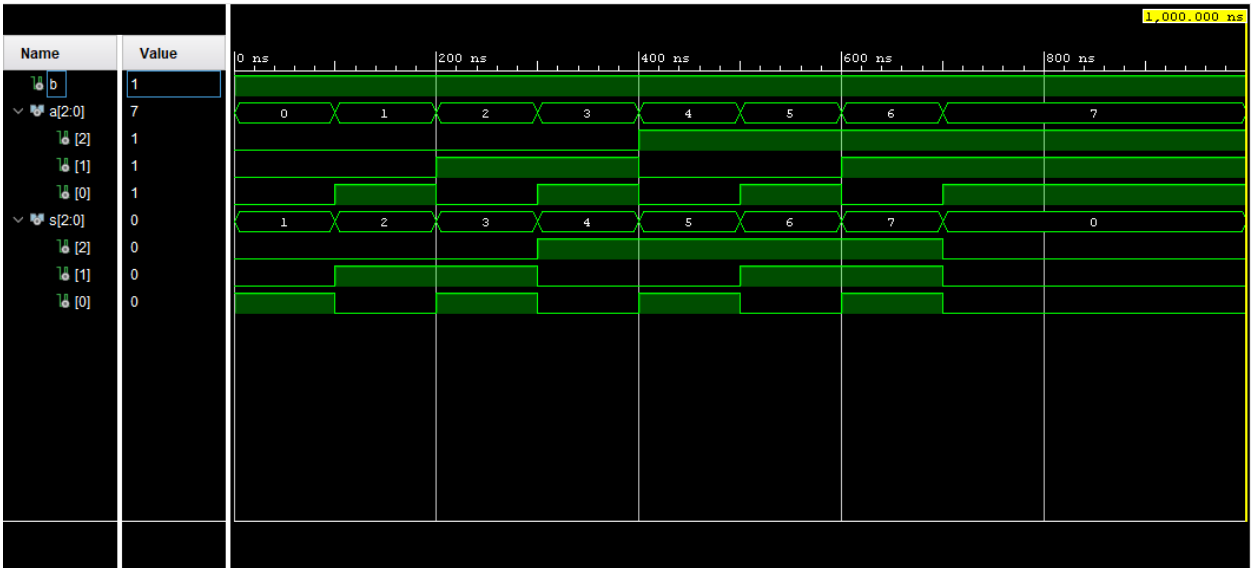




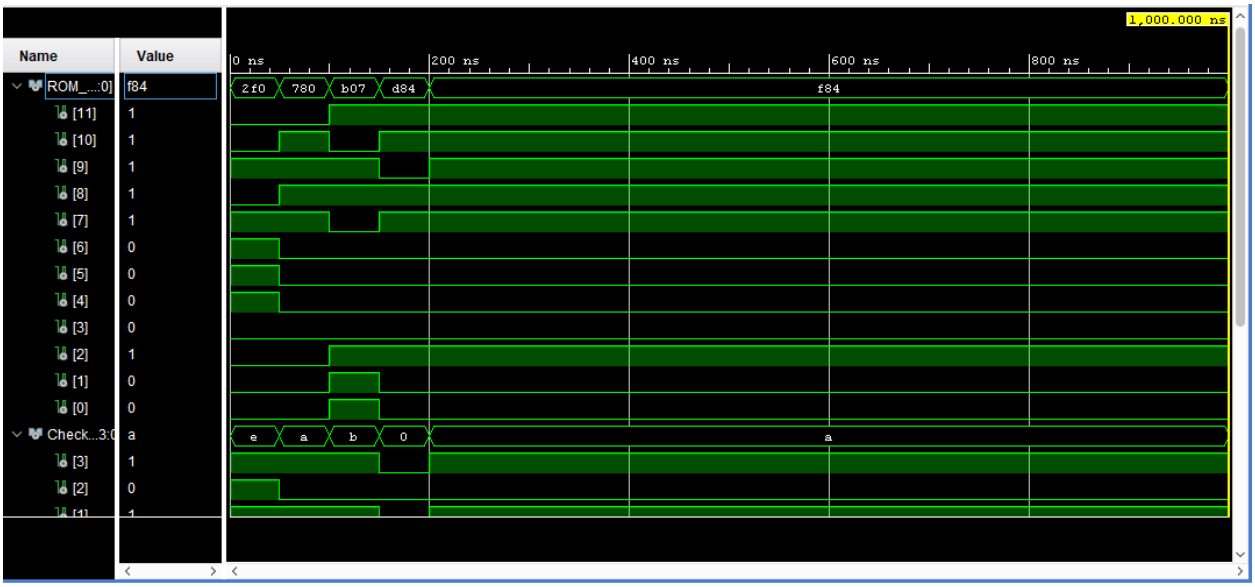
2. *Adder/ Subtractor*

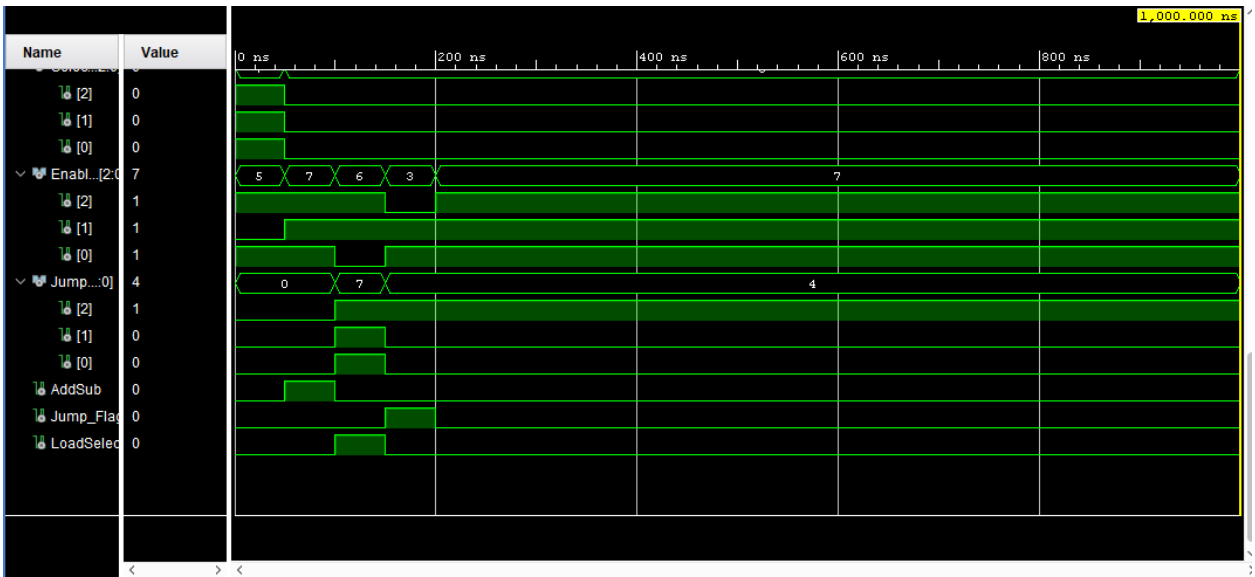


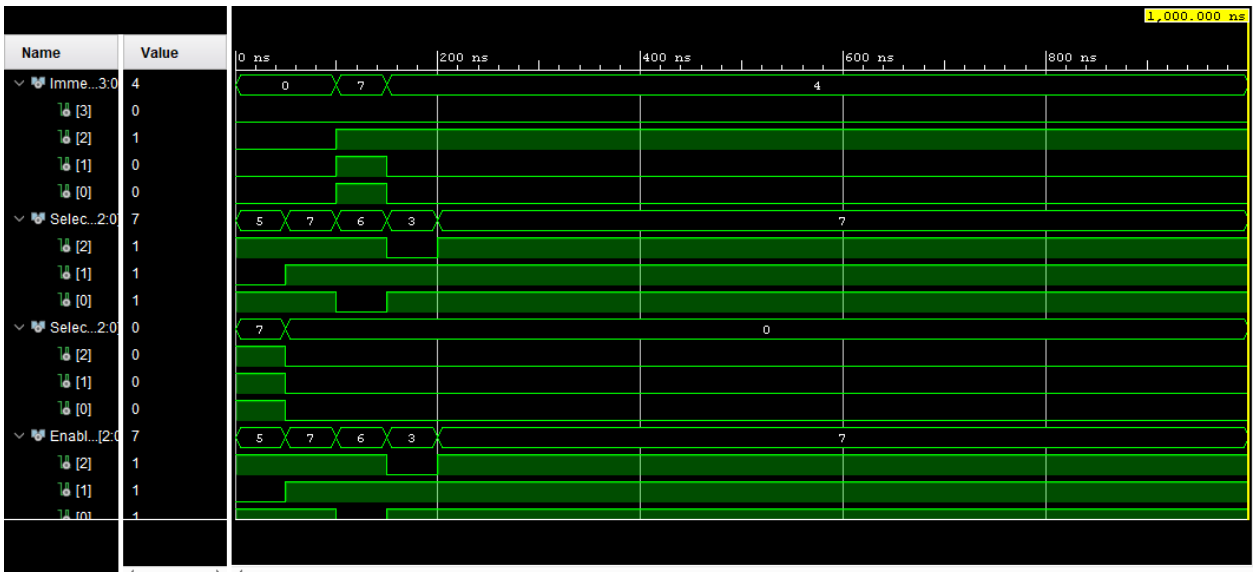
3. 3 bit Adder



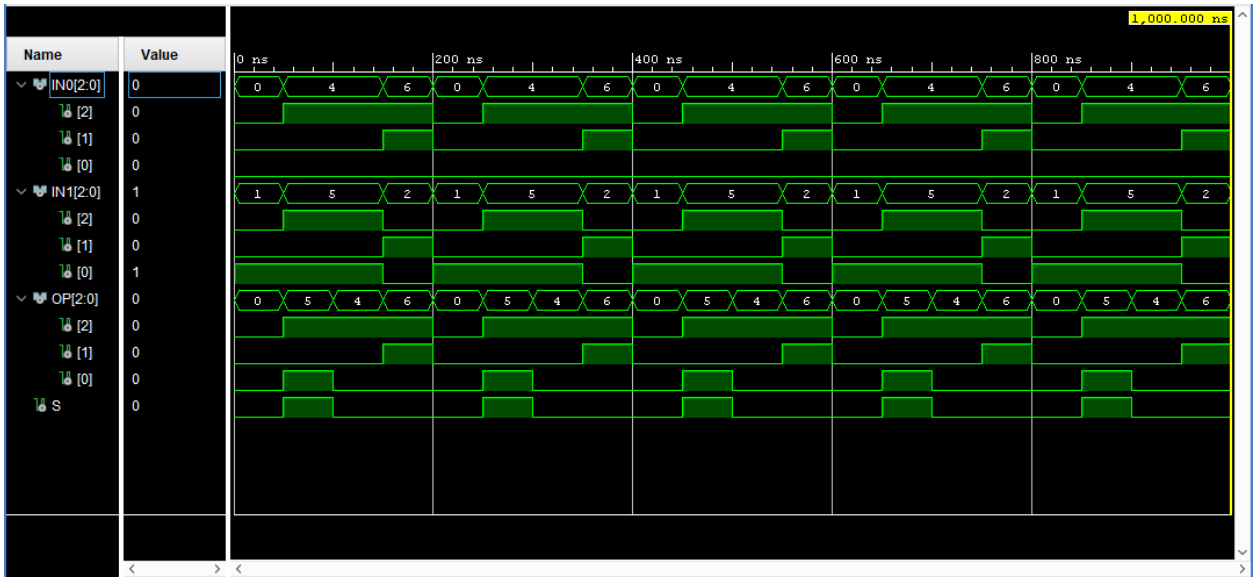
4. Instruction Decoder


Name
Value
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
ROM_...:0]
f84
2f0
780
b07
d84
f84
[11] 1
[10] 1
[9] 1
[8] 1
[7] 1
[6] 0
[5] 0
[4] 0
[3] 0
[2] 1
[1] 0
[0] 0
Check...3:0
a
e
a
b
0
a
[3] 1
[2] 0

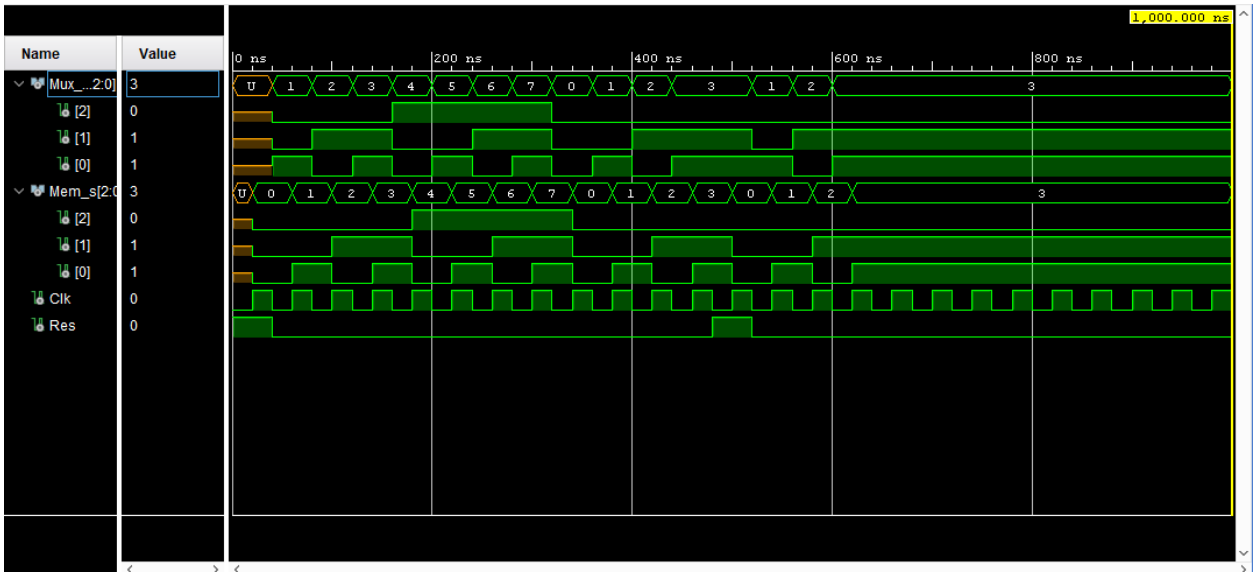

Name
Value
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
[2] 0
[1] 0
[0] 0
Enabl...[2:0]
7
5
7
6
3
7
[2] 1
[1] 1
[0] 1
Jump...:0]
4
0
7
4
[2] 1
[1] 0
[0] 0
AddSub 0
Jump_Flag 0
LoadSelect 0



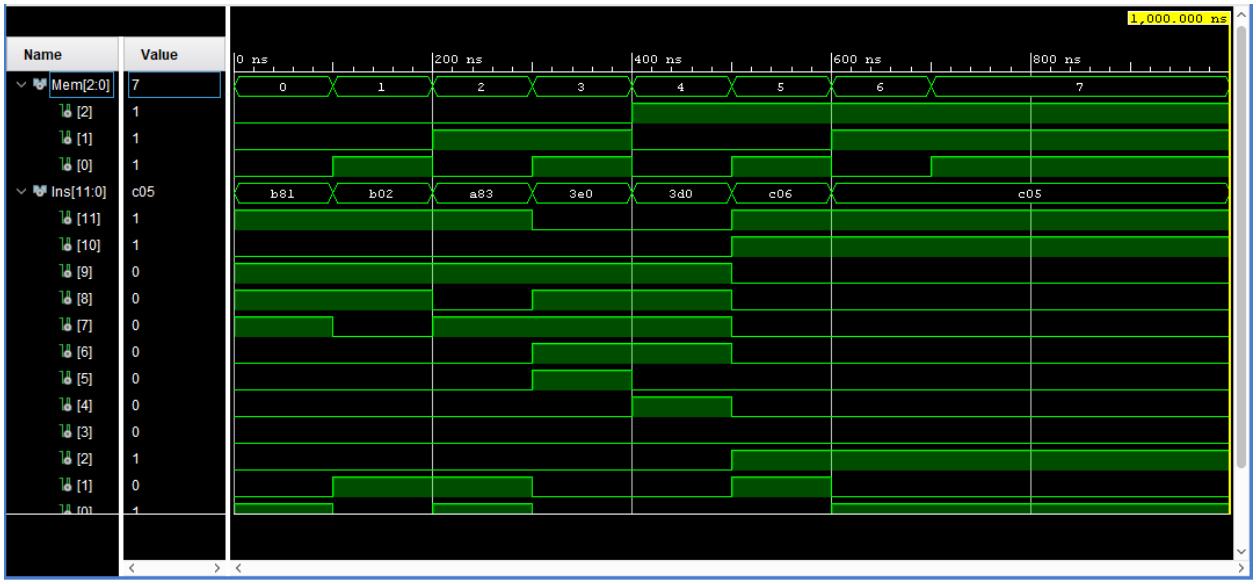
5. 2 way 3 Bit Multiplexer



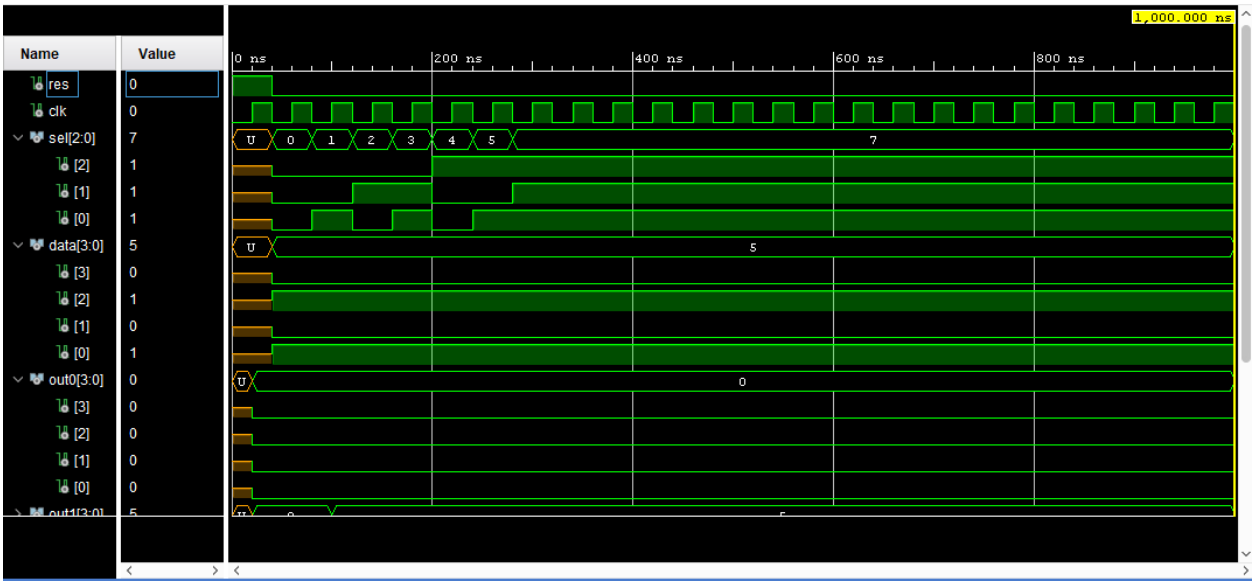
6. Program Counter

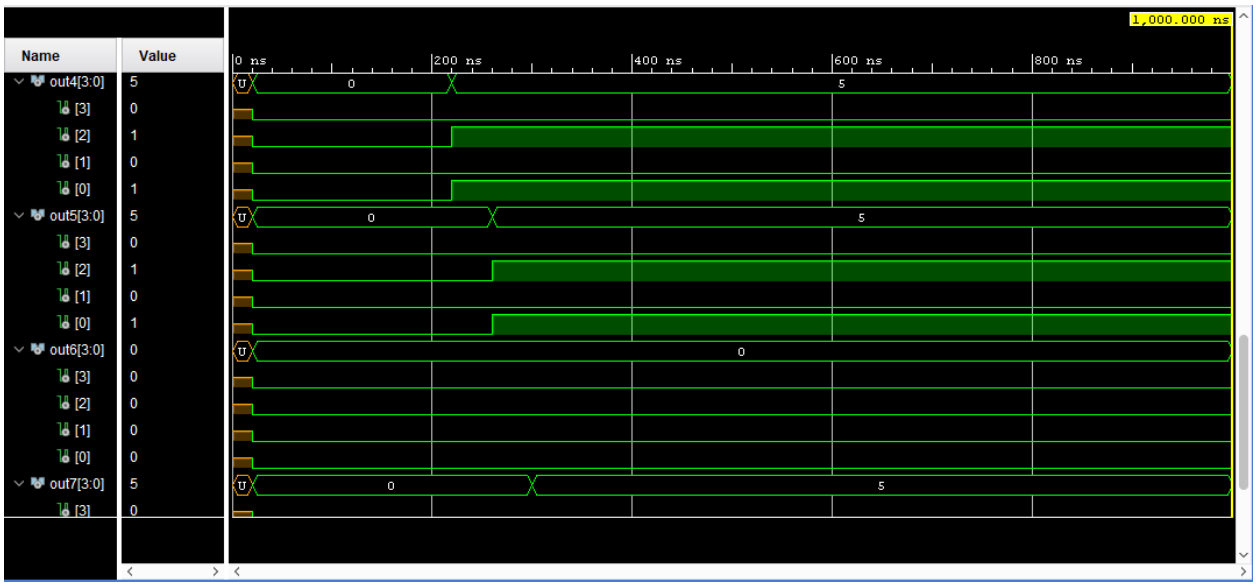


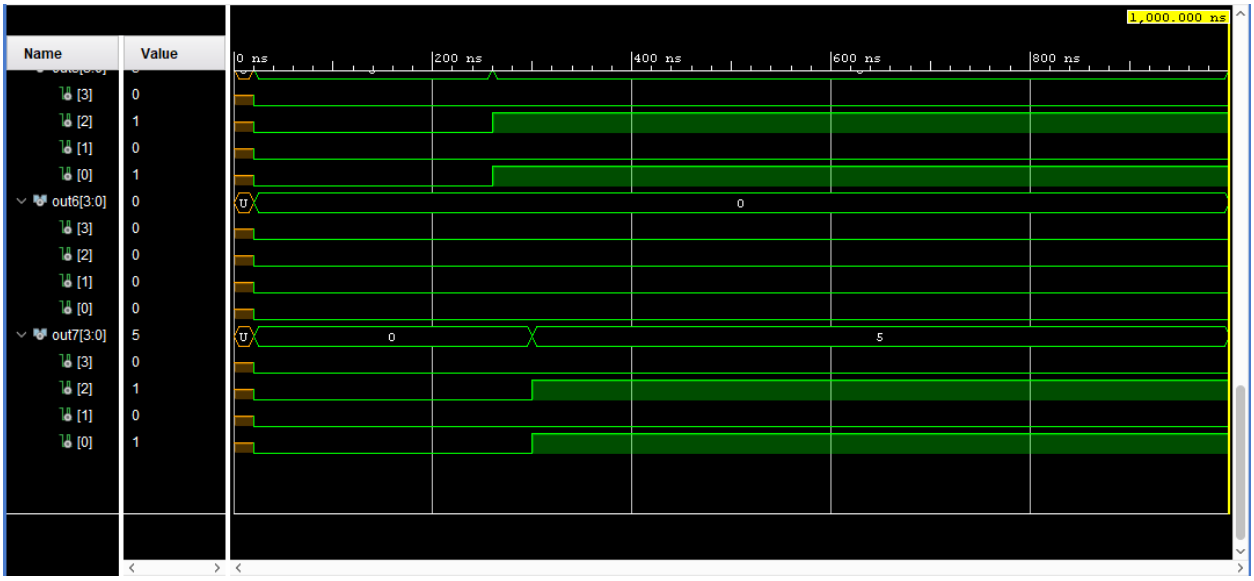
7. Program ROM



8. ***Register Bank***







Conclusion from this lab

Designing a 4-bit processor, comprehending arithmetic operations, instruction decoding, and creating multiplexers are now skills we can put into practice. We tested our processor design physically and virtually on a BASYS 3 development board in order to confirm its operation.

By completing this lab successfully, we have improved our comprehension of processor architecture and acquired practical experience in creating a straightforward 4-bit processor. These abilities are important for further research and study in the area of digital system design and computer architecture.

Contribution from partners

Name	Contribution	Hours spent
Subavarshana	Nano processor 3-bit adder *k*-way *b*-bit multiplexers Register Bank Program ROM Instruction Decoder Simulation codes Lab report design Test on board	10 h
Madhushika	4-bit Add/Subtract unit 3-bit Program Counter (PC) lab report content Test on board	8h