Lab 9-10 – Nanoprocessor Design Competition

CS1050 Computer Organization and Digital Design
Dept. of Computer Science and Engineering
University of Moratuwa

Group members

- Name :- Sampath W.I.D
  Reg. No :- 230571F
- Name :- Dabarera G.D.M
  Reg. No :- 230111X
- Name :- Kannangara T.P
  Reg. No :- 230315C
- Name :- Priyadarshana D.M.R.N
  Reg. No :- 230501T

Lab Task

1. **Execute Simple Programs Using a 4-bit Processor**
   The primary goal is to run simple programs and obtain results using a custom-designed 4-bit processor capable of executing 8 basic instructions.
2. **Develop a 4-bit Arithmetic Unit**
   A core component of the processor is a 4-bit arithmetic unit designed to perform both addition and subtraction on signed integers efficiently.
3. **Implement Instruction Decoding**
   To correctly execute instructions, an Instruction Decoder is developed to activate the appropriate internal components of the processor based on the input operation.
4. **Design Multiplexers or Tri-State Buses**
   k-way, b-bit multiplexers and/or tri-state buses are designed to manage data flow and control signal selection throughout the processor architecture.
5. **Simulate and Test for Verification**
   Once design and development are completed, all components are thoroughly tested through simulations and validated on a development board to ensure proper functionality.
6. **Practice Effective Team Collaboration**
   Team members enhance their skills in communication, task sharing, coordination, and integration of separately developed modules to form a cohesive system.

VHDL Files

4-bit Add/Sub Unit

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

entity Add_Sub_4bit is
    Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
           B : in STD_LOGIC_VECTOR(3 downto 0);
           AddSubSelect : in STD_LOGIC;
           S : inout STD_LOGIC_VECTOR(3 downto 0);
           Zero : out STD_LOGIC;
           Overflow : out STD_LOGIC);
end Add_Sub_4bit;

architecture Behavioral of Add_Sub_4bit 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 : STD_LOGIC;
signal B0_xor, B1_xor, B2_xor, B3_xor : STD_LOGIC;

begin
    B0_xor <= B(0) xor AddSubSelect;
    B1_xor <= B(1) xor AddSubSelect;
    B2_xor <= B(2) xor AddSubSelect;
    B3_xor <= B(3) xor AddSubSelect;
    FA_0: FA
    port map(
        A => A(0),
        B => B0_xor,
        C_in => AddSubSelect,
        S => S(0),
        C_out => FA0_C);

    FA_1: FA
    port map(
```

```
        A => A(1),
        B => B1_xor,
        C_in => FA0_C,
        S => S(1),
        C_out => FA1_C);

    FA_2: FA
    port map(
        A => A(2),
        B => B2_xor,
        C_in => FA1_C,
        S => S(2),
        C_out => FA2_C);

    FA_3: FA
    port map(
        A => A(3),
        B => B3_xor,
        C_in => FA2_C,
        S => S(3),
        C_out => Overflow);

    Zero <= NOT(S(0) OR S(1) OR S(2) OR S(3));

end Behavioral;
```

3-bit Program Counter

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

entity PC_3bit is
    Port (
        Reset        : in  STD_LOGIC;
        Clk          : in  STD_LOGIC;
        MemorySelect : out STD_LOGIC_VECTOR(2 downto 0);
        JumpAddr     : in  STD_LOGIC_VECTOR(2 downto 0);
        JumpFlag     : in  STD_LOGIC
    );
end PC_3bit;

architecture Behavioral of PC_3bit is
    signal PC_reg : std_logic_vector(2 downto 0) := "000";
```

```
begin
    process(Clk, Reset)
    begin
        if Reset = '1' then
            PC_reg <= "000";
        elsif rising_edge(Clk) then
            if JumpFlag = '1' then
                PC_reg <= JumpAddr;
            else
                case PC_reg is
                    when "000" => PC_reg <= "001";
                    when "001" => PC_reg <= "010";
                    when "010" => PC_reg <= "011";
                    when "011" => PC_reg <= "100";
                    when "100" => PC_reg <= "101";
                    when "101" => PC_reg <= "110";
                    when "110" => PC_reg <= "111";
                    when others => PC_reg <= "000";
                end case;
            end if;
        end if;
    end process;

    MemorySelect <= PC_reg;
end Behavioral;
```

Program Rom

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

entity Program_ROM is
    Port ( MemSelect : in STD_LOGIC_VECTOR (2 downto 0);
           InstructBus : 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 Assembly_Code : rom_type := (
                    "101110000000", -- 0
                    "100010000001", -- 1
                    "100100000010", -- 2
```

```
                    "100110000011", -- 3
                    "001110010000", -- 4
                    "001110100000", -- 5
                    "001110110000", -- 6
                    "110000000111"  -- 7
        );
begin
    InstructBus <= Assembly_Code(to_integer(unsigned(MemSelect)));
end Behavioral;
```

Register Bank

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

entity Register_Bank is
    Port ( D_input : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
           Clock : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D_out0 : out STD_LOGIC_VECTOR (3 downto 0):= "0000";
           D_out1 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out2 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out3 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out4 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out5 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out6 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out7 : out STD_LOGIC_VECTOR (3 downto 0));
end Register_Bank;

architecture Behavioral of Register_Bank is

    component Decoder_3_to_8 is
        Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
               EN : in STD_LOGIC;
               Y : out STD_LOGIC_VECTOR (7 downto 0));
    end component;

    component Reg is
        Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
               En : in STD_LOGIC;
               Clk : in STD_LOGIC;
               Reset : in STD_LOGIC;
               Q : out STD_LOGIC_VECTOR (3 downto 0));
```

```
    end component;

signal y : STD_LOGIC_VECTOR (7 downto 0);

begin
    decoder : Decoder_3_to_8
    port map (
        I => Reg_En,
        En => '1',
        Y => y );

    Reg0 : Reg
    port map (
        D => D_input,
        En =>'0',
        Clk => Clock,
        Reset => Reset,
        Q => D_out0  );

    Reg1 : Reg
    port map (
        D => D_input,
        En => y(1),
        Clk => Clock,
        Reset => Reset,
        Q => D_out1 );

     Reg2 : Reg
     port map (
        D => D_input,
        En => y(2),
        Clk => Clock,
        Reset => Reset,
        Q => D_out2 );

     Reg3 : Reg
     port map (
        D => D_input,
        En => y(3),
        Clk => Clock,
        Reset => Reset,
        Q => D_out3 );

     Reg4 : Reg
     port map (
        D => D_input,
```

```
        En => y(4),
        Clk => Clock,
        Reset => Reset,
        Q => D_out4 );

    Reg5 : Reg
    port map (
        D => D_input,
        En => y(5),
        Clk => Clock,
        Reset => Reset,
        Q => D_out5 );

    Reg6 : Reg
    port map (
        D => D_input,
        En => y(6),
        Clk => Clock,
        Reset => Reset,
        Q => D_out6 );

    Reg7 : Reg
    port map (
        D => D_input,
        En => y(7),
        Clk => Clock,
        Reset => Reset,
        Q => D_out7 );

end Behavioral;
```

Register

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

entity Reg is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;   -- add reset pin for reset
output
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Reg;
```

```
architecture Behavioral of Reg is

begin
    process(Clk) begin
    if (Reset = '1') then
        Q <= (others => '0');
    elsif (rising_edge(Clk)) then
        if En = '1' then
            Q <= D;
        end if;
    end if;
    end process;

end Behavioral;
```

Counter

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

entity Counter is
    Port ( Dir : in STD_LOGIC;
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end Counter;

architecture Behavioral of Counter 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;

COMPONENT Slow_Clk
PORT (
    Clk_in : in STD_LOGIC;
    Clk_out : out STD_LOGIC );
END COMPONENT;
```

```
signal D0, D1, D2 : std_logic;
signal Q0, Q1, Q2 : std_logic;
signal Clk_slow : std_logic;

begin

    Slow_Clk0 : Slow_Clk
        port map (
            Clk_in => Clk,
            Clk_out => Clk_slow );

    D0 <= ((not Q2) and (not Dir)) or (Q1 and Dir);
    D1 <= (Q0 and (not Dir)) or (Q2 and Dir);
    D2 <= (Q1 and (not Dir)) or ((not Q0) and Dir);

    D_FF0 : D_FF
        port map (
            D => D0,
            Res => Res,
            Clk => Clk_Slow,
            Q => Q0 );

    D_FF1 : D_FF
        port map (
            D => D1,
            Res => Res,
            Clk => Clk_Slow,
            Q => Q1 );

    D_FF2 : D_FF
         port map (
             D => D2,
             Res => Res,
             Clk => Clk_Slow,
             Q => Q2 );

    Q(0) <= Q0;
    Q(1) <= Q1;
    Q(2) <= Q2;

end Behavioral;
```

Decoder 3 to 8

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

entity Decoder_3_to_8 is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Y : 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;
    EN: in STD_LOGIC;
    Y: out STD_LOGIC_VECTOR );
end component;
signal I0,I1 : STD_LOGIC_VECTOR(1 downto 0);
signal Y0,Y1 : STD_LOGIC_VECTOR (3 downto 0);
signal en0,en1, I2 : STD_LOGIC;

begin
     Decode_2_to_4_0 : Decoder_2_to_4
     port map(
        I => I0,
        EN => en0,
        Y => Y0 );

     Decode_2_to_4_1 : Decoder_2_to_4
     port map(
        I => I1,
        EN => en1,
        Y => Y1 );

     en0 <= NOT(I(2)) AND EN;
     en1 <= I(2) AND EN;
     I0 <= I(1 downto 0);
     I1 <= I(1 downto 0);
     I2 <= I(2);
     Y(3 downto 0) <= Y0;
     Y(7 downto 4) <= Y1;

end Behavioral;
```

LUT_16_7

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.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     0= on 1=off    g f e d c b a
        "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;
```

Slow Clock

```
library IEEE;
use IEEE.STD_LOGIC_1164.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 --50000000 for project (5 for test)
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

MUX 2way 4bit

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

entity Mux_2way_4bit is
    Port ( AddSub : in STD_LOGIC_VECTOR (3 downto 0);
           Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2way_4bit;

architecture Behavioral of Mux_2way_4bit is

COMPONENT Mux_2_to_1
PORT ( D : in STD_LOGIC_VECTOR (1 downto 0);
       S : in STD_LOGIC;
       Y : out STD_LOGIC);
END COMPONENT;

begin
    Mux_0: Mux_2_to_1 port map(
        D(1) => AddSub(0),
        D(0) => Immediate_Val(0),
        S => Load_Select,
```

```
        Y => output(0)
    );

    Mux_1: Mux_2_to_1 port map(
        D(1) => AddSub(1),
        D(0) => Immediate_Val(1),
        S => Load_Select,
        Y => output(1)
    );

    Mux_2: Mux_2_to_1 port map(
        D(1) => AddSub(2),
        D(0) => Immediate_Val(2),
        S => Load_Select,
        Y => output(2)
    );

    Mux_3: Mux_2_to_1 port map(
        D(1) => AddSub(3),
        D(0) => Immediate_Val(3),
        S => Load_Select,
        Y => output(3)
    );

end Behavioral;
```

MUX 2way 3bit

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

entity Mux_2way_3bit is
    Port ( jmp_adrs : in STD_LOGIC_VECTOR (2 downto 0);
           adder_3bit : in STD_LOGIC_VECTOR (2 downto 0);
           jmp_flag : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2way_3bit;

architecture Behavioral of Mux_2way_3bit is
    COMPONENT Mux_2_to_1
    PORT ( D : in STD_LOGIC_VECTOR (1 downto 0);
           S : in STD_LOGIC;
           Y : out STD_LOGIC);
    END COMPONENT;

begin
```

```
    Mux_0: Mux_2_to_1 port map(
        D(0) => jmp_adrs(0),
        D(1) => adder_3bit(0),
        S => jmp_flag,
        Y => output(0)
    );

    Mux_1: Mux_2_to_1 port map(
        D(0) => jmp_adrs(1),
        D(1) => adder_3bit(1),
        S => jmp_flag,
        Y => output(1)
    );

    Mux_2: Mux_2_to_1 port map(
        D(0) => jmp_adrs(2),
        D(1) => adder_3bit(2),
        S => jmp_flag,
        Y => output(2)
    );


end Behavioral;
```

MUX 8way 4bit

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

entity Mux_8way_4bit is
    Port (  reg0 : in STD_LOGIC_VECTOR (3 downto 0);
            reg1 : in STD_LOGIC_VECTOR (3 downto 0);
            reg2 : in STD_LOGIC_VECTOR (3 downto 0);
            reg3 : in STD_LOGIC_VECTOR (3 downto 0);
            reg4 : in STD_LOGIC_VECTOR (3 downto 0);
            reg5 : in STD_LOGIC_VECTOR (3 downto 0);
            reg6 : in STD_LOGIC_VECTOR (3 downto 0);
            reg7 : in STD_LOGIC_VECTOR (3 downto 0);
            Reg_select : in STD_LOGIC_VECTOR (2 downto 0);
            output : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8way_4bit;

architecture Behavioral of Mux_8way_4bit is

begin
```

```
    process(reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7,
Reg_select)
    begin
        case (Reg_select) is
            when "000" =>
                output <= reg0;
            when "001" =>
               output <= reg1;
            when "010" =>
               output <= reg2;
            when "011" =>
               output <= reg3;
            when "100" =>
               output <= reg4;
            when "101" =>
               output <= reg5;
            when "110" =>
               output <= reg6;
            when "111" =>
               output <= reg7;
            when others =>
               output <= "0000";  -- For handle any undefined
cases
        end case;
    end process;
end Behavioral;
```

Instruction Decoder

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

entity Instruction_Dec is
    Port ( Reg_En : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC;
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           MuxA_Select : out STD_LOGIC_VECTOR (2 downto 0);
           MuxB_Select : out STD_LOGIC_VECTOR (2 downto 0);
           AddSub_Select : out STD_LOGIC;
           JmpFlag : out STD_LOGIC;
           JmpAddr : out STD_LOGIC_VECTOR (2 downto 0);
           Instruct_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           JumpCheck : in STD_LOGIC_VECTOR (3 downto 0));
end Instruction_Dec;

architecture Behavioral of Instruction_Dec is
```

```
begin
   process(Instruct_Bus, JumpCheck)
   begin

       -- for ADD intruction
       if Instruct_Bus (11 downto 10) = "00" then
           JmpFlag <= '0';

           MuxA_Select <= Instruct_Bus (9 downto 7);
           MuxB_Select <= Instruct_Bus (6 downto 4);
           AddSub_Select <= '0';
           Load_Select <= '0';
           Reg_En <= Instruct_Bus (9 downto 7);  -- Store the
result on Register A

       -- for NEG intruction
       elsif Instruct_Bus (11 downto 10) = "01" then
           JmpFlag <= '0';

           MuxA_Select <= "000";  -- For get -B we have to set A
to zero(i.e. set to Register 0) because adder perform only A+B.
           MuxB_Select <= Instruct_Bus (9 downto 7);  -- For get a
negative value of a particular number, it can be only done from
Mux B
           AddSub_Select <= '1';  -- Here we perform only
subtraction.
           Load_Select <= '0';
           Reg_En <= Instruct_Bus (9 downto 7);  -- Store the
result on Register A

       -- for MOV intruction
       elsif Instruct_Bus (11 downto 10) = "10" then
           JmpFlag <= '0';
           MuxA_Select <= "000";
           MuxB_Select <= "000";

           Immediate_Val <= Instruct_Bus (3 downto 0);
           Reg_En <= Instruct_Bus (9 downto 7);
           Load_Select <= '1';  -- Select IMMEDIATE VALUE port via

       -- for JMP intruction
       else
           MuxB_Select <= "000";
           Immediate_Val <= "0000";
           Load_Select <= '1';
```

```
            Reg_En <= "000";

            MuxA_Select <= Instruct_Bus (9 downto 7);
            if JumpCheck = "0000" then
                JmpAddr <= Instruct_Bus (2 downto 0);
                JmpFlag <= '1';
            else
                JmpFlag <= '0';
            end if;
        end if;
    end process;

end Behavioral;
```

Nano Processor

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

entity Nanoprocessor is
    Port ( Reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           Zero_LED : out STD_LOGIC := '1';
           Overflow_LED : out STD_LOGIC;
           LED0 : out STD_LOGIC;
           LED1 : out STD_LOGIC;
           LED2 : out STD_LOGIC;
           LED3 : out STD_LOGIC;
           Anode : out STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0)
    );
end Nanoprocessor;

architecture Behavioral of Nanoprocessor is

component Register_Bank is
    Port ( D_input : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
           Clock : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D_out0 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out1 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out2 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out3 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out4 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out5 : out STD_LOGIC_VECTOR (3 downto 0);
```

```
           D_out6 : out STD_LOGIC_VECTOR (3 downto 0);
           D_out7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Mux_8way_4bit is
    Port (  reg0 : in STD_LOGIC_VECTOR (3 downto 0);
            reg1 : in STD_LOGIC_VECTOR (3 downto 0);
            reg2 : in STD_LOGIC_VECTOR (3 downto 0);
            reg3 : in STD_LOGIC_VECTOR (3 downto 0);
            reg4 : in STD_LOGIC_VECTOR (3 downto 0);
            reg5 : in STD_LOGIC_VECTOR (3 downto 0);
            reg6 : in STD_LOGIC_VECTOR (3 downto 0);
            reg7 : in STD_LOGIC_VECTOR (3 downto 0);
            Reg_select : in STD_LOGIC_VECTOR (2 downto 0);
            output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Add_Sub_4bit is
    Port (
        A : in STD_LOGIC_VECTOR(3 downto 0);
        B : in STD_LOGIC_VECTOR(3 downto 0);
        AddSubSelect : in STD_LOGIC;
        S : inout STD_LOGIC_VECTOR(3 downto 0);
        Zero : out STD_LOGIC;
        Overflow : out STD_LOGIC
    );
end component;

component Instruction_Dec is
    Port ( Reg_En : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC;
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           MuxA_Select : out STD_LOGIC_VECTOR (2 downto 0);
           MuxB_Select : out STD_LOGIC_VECTOR (2 downto 0);
           AddSub_Select : out STD_LOGIC;
           JmpFlag : out STD_LOGIC;
           JmpAddr : out STD_LOGIC_VECTOR (2 downto 0);
           Instruct_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           JumpCheck : in STD_LOGIC_VECTOR (3 downto 0));
end component;

component Program_ROM is
    Port ( MemSelect : in STD_LOGIC_VECTOR (2 downto 0);
           InstructBus : out STD_LOGIC_VECTOR (11 downto 0));
end component;
```

```
component PC_3bit is
    Port ( Reset : in STD_LOGIC;
           Clk : in STD_LOGIC;
           MemorySelect : inout STD_LOGIC_VECTOR (2 downto 0);
           JumpAddr : in STD_LOGIC_VECTOR (2 downto 0);
-- If there is a jump instruction, PC set MemorrySelect as the
           JumpFlag : in STD_LOGIC);
end component;

component Mux_2way_4bit is
    Port ( AddSub : in STD_LOGIC_VECTOR (3 downto 0);
           Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

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

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

signal Clk_slow : std_logic;
signal Memory_Select : std_logic_vector (2 downto 0);  -- from
program counter to program ROM
signal Jump_Address : std_logic_vector (2 downto 0);  -- from
instruction decoder to program counter
signal Jump_Flag : std_logic; -- from instruction decoder to
program counter
signal I : std_logic_vector (11 downto 0);  -- from program ROM to
instruction decoder
signal Register_Enable : std_logic_vector (2 downto 0);  -- from
instruction decoder to Register Bank
signal Load_Select : std_logic; -- from instruction decoder to
Mux_2way_4bit
signal Immediate_Value : std_logic_vector (3 downto 0);  -- from
instruction decoder to Mux_2way_4bit
signal Mux_A_Select : std_logic_vector (2 downto 0);  -- from
instruction decoder to Mux_8way_4bit(A)
signal Mux_B_Select : std_logic_vector (2 downto 0);  -- from
instruction decoder to Mux_8way_4bit(B)
```

```
signal ADD_SUB_Select : std_logic; -- from instruction decoder to
Add_Sub_4bit
signal Mux_A : std_logic_vector (3 downto 0);  -- from
Mux_8way_4bit(A) to Add_Sub_4bit / from instruction decoder to
Mux_8way_4bit(A) output
signal Mux_B : std_logic_vector (3 downto 0);  -- from
Mux_8way_4bit(B) to Add_Sub_4bit
signal Add_Sub_Result : std_logic_vector (3 downto 0);  -- from
Add_Sub_4bit to Mux_2way_4bit
signal Register_Input : std_logic_vector (3 downto 0);  -- from
Mux_2way_4bit to Register Bank
signal R0 : std_logic_vector (3 downto 0);  -- from R0 to
Mux_8way_4bit
signal R1 : std_logic_vector (3 downto 0);  -- from R1 to
Mux_8way_4bit
signal R2 : std_logic_vector (3 downto 0);  -- from R2 to
Mux_8way_4bit
signal R3 : std_logic_vector (3 downto 0);  -- from R3 to
Mux_8way_4bit
signal R4 : std_logic_vector (3 downto 0);  -- from R4 to
Mux_8way_4bit
signal R5 : std_logic_vector (3 downto 0);  -- from R5 to
Mux_8way_4bit
signal R6 : std_logic_vector (3 downto 0);  -- from R6 to
Mux_8way_4bit
signal R7 : std_logic_vector (3 downto 0);  -- from R7 to
Mux_8way_4bit

signal S_7s : STD_LOGIC_VECTOR (6 downto 0) := "0000000";
--signal clk_7seg : STD_LOGIC;


begin


Slow_Clock : Slow_Clk
port map (
    Clk_in => clk,
    Clk_out => Clk_slow
);

LUT_16_7_0 : LUT_16_7
port map (
    address => R7,
    data => S_7s
 );
```

```
Programming_Counter : PC_3bit
port map (
    Reset => Reset,
    Clk => Clk_slow,
    MemorySelect => Memory_Select,
    JumpAddr => Jump_Address,
    JumpFlag => Jump_Flag
);

Program_ROM0 : Program_ROM
port map (
    MemSelect => Memory_Select,
    InstructBus => I
);

Instruction_Decoder : Instruction_Dec
port map (
    Reg_En => Register_Enable,
    Load_Select => Load_Select,
    Immediate_Val => Immediate_Value,
    MuxA_Select => Mux_A_Select,
    MuxB_Select => Mux_B_Select,
    AddSub_Select => ADD_SUB_Select,
    JmpFlag => Jump_Flag,
    JmpAddr => Jump_Address,
    Instruct_Bus => I,
    JumpCheck => Mux_A
);

Mux_2way_4bit0 : Mux_2way_4bit
port map (
    AddSub =>  Add_Sub_Result,
    Immediate_Val => Immediate_Value,
    Load_Select => Load_Select,
    output => Register_Input
);

Add_Sub_Unit : Add_Sub_4bit
port map (
    A => Mux_A,
    B => Mux_B,
    AddSubSelect => ADD_SUB_Select,
    S => Add_Sub_Result,
    Zero => Zero_LED,
    Overflow => Overflow_LED
```

```
);

MuxA : Mux_8way_4bit
port map (
    reg0 => R0,
    reg1 => R1,
    reg2 => R2,
    reg3 => R3,
    reg4 => R4,
    reg5 => R5,
    reg6 => R6,
    reg7 => R7,
    Reg_select => Mux_A_Select,
    output => Mux_A
);

MuxB : Mux_8way_4bit
port map (
    reg0 => R0,
    reg1 => R1,
    reg2 => R2,
    reg3 => R3,
    reg4 => R4,
    reg5 => R5,
    reg6 => R6,
    reg7 => R7,
    Reg_select => Mux_B_Select,
    output => Mux_B
);

RegisterBank : Register_Bank
port map (
    D_input => Register_Input,
    Reg_En => Register_Enable,
    Clock => Clk_slow,
    Reset => Reset,
    D_out0 => R0,
    D_out1 => R1,
    D_out2 => R2,
    D_out3 => R3,
    D_out4 => R4,
    D_out5 => R5,
    D_out6 => R6,
    D_out7 => R7
);
```

```
LED0 <= R7(0);
LED1 <= R7(1);
LED2 <= R7(2);
LED3 <= R7(3);

Anode <= "1110";
S_7Seg <= S_7s;

end Behavioral;
```

Simulation Files

4-bit Add/Sub Unit

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

entity Add_Sub_4bit_Sim is
--  Port ( );
end Add_Sub_4bit_Sim;

architecture Behavioral of Add_Sub_4bit_Sim is

    -- Component Declaration for the Unit Under Test (UUT)
    component Add_Sub_4bit
        Port (
            A : in STD_LOGIC_VECTOR(3 downto 0);
            B : in STD_LOGIC_VECTOR(3 downto 0);
            AddSubSelect : in STD_LOGIC;
            S : inout STD_LOGIC_VECTOR(3 downto 0);
            Zero : out STD_LOGIC;
            Overflow : out STD_LOGIC
        );
    end component;

    -- Signals for testbench
    signal A : STD_LOGIC_VECTOR(3 downto 0);
    signal B : STD_LOGIC_VECTOR(3 downto 0);
    signal AddSubSelect : STD_LOGIC;
    signal S : STD_LOGIC_VECTOR(3 downto 0);
    signal Zero : STD_LOGIC;
    signal Overflow : STD_LOGIC;
```

```
begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Add_Sub_4bit
        Port Map (
            A => A,
            B => B,
            AddSubSelect => AddSubSelect,
            S => S,
            Zero => Zero,
            Overflow => Overflow
        );

    -- Stimulus Process
    stim_proc: process
    begin
        -- Test 1: for index 230111X
        A <= "1111";
        B <= "1101";
        AddSubSelect <= '0'; -- Addition
        wait for 100 ns;

        A <= "1111";
        B <= "1101";
        AddSubSelect <= '1'; -- Subtraction
        wait for 100 ns;

        -- Test 2: for index 230315C
        A <= "1011";
        B <= "1010";
        AddSubSelect <= '0'; -- Addition
        wait for 100 ns;

        A <= "1011";
        B <= "1010";
        AddSubSelect <= '1'; -- Subtraction
        wait for 100 ns;

        -- Test 3: for index 230501T
        A <= "0101";
        B <= "0110";
        AddSubSelect <= '0'; -- Addition
        wait for 100 ns;

        A <= "0101";
```

```
        B <= "0110";
        AddSubSelect <= '1'; -- Subtraction
        wait for 100 ns;

        -- Test 2: for index 230571F
        A <= "1011";
        B <= "1010";
        AddSubSelect <= '0'; -- Addition
        wait for 100 ns;

        A <= "1011";
        B <= "1010";
        AddSubSelect <= '1'; -- Subtraction
        wait for 100 ns;

        -- End simulation
        wait;
    end process;

end Behavioral;
```

3-bit Program Counter

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

entity PC_3bit_Sim is
end PC_3bit_Sim;

architecture Behavioral of PC_3bit_Sim is
    -- Component Declaration for the Unit Under Test (UUT)
    component PC_3bit
        Port (
            Reset         : in STD_LOGIC;
            Clk           : in STD_LOGIC;
            MemorySelect  : out STD_LOGIC_VECTOR (2 downto 0);
            JumpAddr      : in STD_LOGIC_VECTOR (2 downto 0);
            JumpFlag      : in STD_LOGIC
        );
    end component;

    -- Inputs
    signal Reset : STD_LOGIC := '0';
    signal Clk : STD_LOGIC := '0';
```

```
    signal JumpAddr : STD_LOGIC_VECTOR(2 downto 0) := "000";
    signal JumpFlag : STD_LOGIC := '0';

    -- Outputs
    signal MemorySelect : STD_LOGIC_VECTOR(2 downto 0);

    -- Clock period definitions
    constant Clk_period : time := 100 ns;


begin
    -- Instantiate the Unit Under Test (UUT)
    uut: PC_3bit port map (
        Reset => Reset,
        Clk => Clk,
        MemorySelect => MemorySelect,
        JumpAddr => JumpAddr,
        JumpFlag => JumpFlag
    );

    -- Clock process definitions
    Clk_process: process
    begin
        Clk <= '0';
        wait for Clk_period/2;
        Clk <= '1';
        wait for Clk_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
        begin
            -- Phase 1: Initial Reset
            Reset <= '1';
            wait for Clk_period*2;
            Reset <= '0';

            -- Phase 2: Count normally for 3 cycles (expect 001,
010, 011)
            wait for Clk_period*3;

            -- Phase 3: Jump to 5 ("101")
            JumpAddr <= "101";
            JumpFlag <= '1';
            wait for Clk_period;
            JumpFlag <= '0';
```

```
            Reset <= '0';
            -- Phase 4: Count normally for 1 cycle (expect 110)
            wait for Clk_period*3;

            -- End simulation
            wait;
        end process;
end Behavioral;
```

Program Rom

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

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

architecture Behavioral of Program_ROM_Sim is

    -- Component declaration of Program_ROM
    component Program_ROM
        Port (
            MemSelect : in STD_LOGIC_VECTOR(2 downto 0);
            InstructBus : out STD_LOGIC_VECTOR(11 downto 0)
        );
    end component;

    -- Testbench signals
    signal MemSelect    : STD_LOGIC_VECTOR(2 downto 0);
    signal InstructBus  : STD_LOGIC_VECTOR(11 downto 0);

begin

    -- Instantiate the Program_ROM
    uut: Program_ROM
        port map (
            MemSelect   => MemSelect,
            InstructBus => InstructBus
        );

    -- Test process to simulate ROM output
    stim_proc: process
    begin
        -- Loop through all addresses from 0 to 7
```

```
        for i in 0 to 7 loop
            MemSelect <= std_logic_vector(to_unsigned(i, 3));
            wait for 100 ns; -- Wait for propagation
        end loop;

        -- Finish simulation
        wait;
    end process;

end Behavioral;
```

Register Bank

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

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

architecture Behavioral of Register_Bank_Sim is

    component Register_Bank
        Port (
            D_input : in STD_LOGIC_VECTOR (3 downto 0);
            Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
            Clock : in STD_LOGIC;
            Reset : in STD_LOGIC;
            D_out0 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out1 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out2 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out3 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out4 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out5 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out6 : out STD_LOGIC_VECTOR (3 downto 0);
            D_out7 : out STD_LOGIC_VECTOR (3 downto 0)
        );
    end component;

    -- Signals
    signal D_input : STD_LOGIC_VECTOR(3 downto 0);
    signal Reg_En  : STD_LOGIC_VECTOR(2 downto 0);
    signal Clock   : STD_LOGIC := '0';
```

```
    signal Reset   : STD_LOGIC;

    signal D_out0, D_out1, D_out2, D_out3,
           D_out4, D_out5, D_out6, D_out7 : STD_LOGIC_VECTOR(3
downto 0);

    -- Clock process
    constant clk_period : time := 100 ns;
begin

    -- Clock generator
    clk_process : process
    begin
        while true loop
            Clock <= '0';
            wait for clk_period / 2;
            Clock <= '1';
            wait for clk_period / 2;
        end loop;
    end process;

    -- Instantiate the Unit Under Test (UUT)
    uut: Register_Bank
        port map (
            D_input => D_input,
            Reg_En => Reg_En,
            Clock => Clock,
            Reset => Reset,
            D_out0 => D_out0,
            D_out1 => D_out1,
            D_out2 => D_out2,
            D_out3 => D_out3,
            D_out4 => D_out4,
            D_out5 => D_out5,
            D_out6 => D_out6,
            D_out7 => D_out7
        );

    -- Stimulus
    stim_proc : process
    begin
        -- Initial values
        Reset <= '1';
        D_input <= "0000";
        Reg_En <= "000";
        wait for clk_period;
```

```
        -- Release reset
        Reset <= '0';
        wait for clk_period;
        -- Test 1: For index 230111X
        D_input <= "1111";
        Reg_En <= "001";
        wait for clk_period;

        D_input <= "1101";
        Reg_En <= "010";
        wait for clk_period;

        -- Test 2: For index 230315C and 230571F
        D_input <= "1011";
        Reg_En <= "011";
        wait for clk_period;

        D_input <= "1010";
        Reg_En <= "100";
        wait for clk_period;

        -- Test 3: For index 230501T
        D_input <= "0101";
        Reg_En <= "101";
        wait for clk_period;

        D_input <= "0110";
        Reg_En <= "110";
        wait for clk_period;
        -- Wait and then reset everything
        wait for 4 * clk_period;
        Reset <= '1';
        wait for clk_period;

        -- Release reset
        Reset <= '0';
        wait;

    end process;

end Behavioral;
```

Register

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;

entity TB_Reg is
end TB_Reg;

architecture Behavioral of TB_Reg is

    -- Component Declaration for the Unit Under Test (UUT)
    component Reg
        Port (
            D     : in  STD_LOGIC_VECTOR(3 downto 0);
            En    : in  STD_LOGIC;
            Clk   : in  STD_LOGIC;
            Reset : in  STD_LOGIC;
            Q     : out STD_LOGIC_VECTOR(3 downto 0)
        );
    end component;

    -- Testbench signals
    signal D_tb     : STD_LOGIC_VECTOR(3 downto 0) := (others =>
'0');
    signal En_tb    : STD_LOGIC := '0';
    signal Clk_tb   : STD_LOGIC := '0';
    signal Reset_tb : STD_LOGIC := '0';
    signal Q_tb     : STD_LOGIC_VECTOR(3 downto 0);

    -- Clock generation process (10 ns period)
    constant clk_period : time := 100 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Reg
        port map (
            D     => D_tb,
            En    => En_tb,
            Clk   => Clk_tb,
            Reset => Reset_tb,
            Q     => Q_tb
        );

    -- Clock process
    clk_process: process
    begin
        while true loop
            Clk_tb <= '0';
```

```
        wait for clk_period / 2;
        Clk_tb <= '1';
        wait for clk_period / 2;
    end loop;
end process;

-- Stimulus process
stim_proc: process
begin
    -- Reset the register
    Reset_tb <= '1';
    wait for clk_period;
    Reset_tb <= '0';

    -- Test 1: For index 230111X
    D_tb <= "1111";
    En_tb <= '1';
    wait for clk_period;

    -- Change D to 1101, but En=0 (Q should remain 1111)
    D_tb <= "1101";
    En_tb <= '0';
    wait for clk_period;

    -- Enable again, Q should update to 1101
    En_tb <= '1';
    wait for clk_period;

    -- Test 2: For index 230315C and 230571F
    D_tb <= "1011";
    En_tb <= '1';
    wait for clk_period;

    -- Change D to 1010, but En=0 (Q should remain 1011)
    D_tb <= "1010";
    En_tb <= '0';
    wait for clk_period;

    -- Enable again, Q should update to 1010
    En_tb <= '1';
    wait for clk_period;

    -- Test 3: For index 230501T
    D_tb <= "0101";
    En_tb <= '1';
    wait for clk_period;
```

```
        -- Change D to 0110, but En=0 (Q should remain 0101)
        D_tb <= "0110";
        En_tb <= '0';
        wait for clk_period;

        -- Enable again, Q should update to 0110
        En_tb <= '1';
        wait for clk_period;
        -- Reset again
        Reset_tb <= '1';
        wait for clk_period;
        Reset_tb <= '0';

        wait;
    end process;

end Behavioral;
```

Decoder 3 to 8

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

entity TB_Decoder_3_to_8 is
end TB_Decoder_3_to_8;

architecture Behavioral of TB_Decoder_3_to_8 is

    -- Component Declaration for Unit Under Test (UUT)
    component Decoder_3_to_8
        Port (
            I : in STD_LOGIC_VECTOR (2 downto 0);
            EN : in STD_LOGIC;
            Y : out STD_LOGIC_VECTOR (7 downto 0)
        );
    end component;

    -- Testbench signals
    signal I_tb  : STD_LOGIC_VECTOR(2 downto 0) := (others =>
'0');
    signal EN_tb : STD_LOGIC := '0';
    signal Y_tb  : STD_LOGIC_VECTOR(7 downto 0);
```

```
begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Decoder_3_to_8
        port map (
            I  => I_tb,
            EN => EN_tb,
            Y  => Y_tb
        );

    -- Stimulus process
    stim_proc: process
    begin
        -- Test all inputs with Enable = '0' (Y should be
00000000)
        EN_tb <= '0';
        for i in 0 to 7 loop
            I_tb <= std_logic_vector(to_unsigned(i, 3));
            wait for 50 ns;
        end loop;

        -- Enable the decoder and test all input combinations
        EN_tb <= '1';
        for i in 0 to 7 loop
            I_tb <= std_logic_vector(to_unsigned(i, 3));
            wait for 50 ns;
        end loop;

        wait;
    end process;

end Behavioral;
```

LUT_16_7

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.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     0= on 1=off    g f e d c b a
        "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;
```

Slow Clock

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

entity TB_Slow_Clk is
-- Testbench has no ports
end TB_Slow_Clk;

architecture Behavioral of TB_Slow_Clk is

    -- Component declaration of the Unit Under Test (UUT)
    component Slow_Clk
        Port (
            Clk_in : in STD_LOGIC;
            Clk_out : out STD_LOGIC
        );
    end component;
```

```
    -- Testbench signals
    signal Clk_in_tb : STD_LOGIC := '0';
    signal Clk_out_tb : STD_LOGIC;

    constant clk_period : time := 10 ns; -- Simulated clock period

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Slow_Clk
        Port map (
            Clk_in => Clk_in_tb,
            Clk_out => Clk_out_tb
        );

    -- Clock generation process
    clk_process : process
    begin
        while now < 500 ns loop  -- run for limited simulation
time
            Clk_in_tb <= '0';
            wait for clk_period / 2;
            Clk_in_tb <= '1';
            wait for clk_period / 2;
        end loop;
        wait; -- stop simulation
    end process;

end Behavioral;
```

MUX 2way 4bit

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

entity TB_Mux_2way_4bit is
-- no ports for testbench
end TB_Mux_2way_4bit;

architecture Behavioral of TB_Mux_2way_4bit is

    -- Component declaration
    component Mux_2way_4bit
        Port ( AddSub : in STD_LOGIC_VECTOR (3 downto 0);
               Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
               Load_Select : in STD_LOGIC;
```

```
                output : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    -- Test signals
    signal AddSub_tb : STD_LOGIC_VECTOR(3 downto 0) := (others =>
'0');
    signal Immediate_Val_tb : STD_LOGIC_VECTOR(3 downto 0) :=
(others => '0');
    signal Load_Select_tb : STD_LOGIC := '0';
    signal output_tb : STD_LOGIC_VECTOR(3 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Mux_2way_4bit port map(
        AddSub => AddSub_tb,
        Immediate_Val => Immediate_Val_tb,
        Load_Select => Load_Select_tb,
        output => output_tb
    );

    -- Stimulus process
    stim_proc: process
    begin
        -- Test case 1: for index 230111X Load_Select = 0, select
Immediate_Val
        Immediate_Val_tb <= "1111";
        AddSub_tb <= "1101";
        Load_Select_tb <= '0';
        wait for 100 ns;

        Load_Select_tb <= '1';
        wait for 100 ns;

        -- Test case 2: for index 230315C Load_Select = 0, select
Immediate_Val
        Immediate_Val_tb <= "1011";
        AddSub_tb <= "1010";
        Load_Select_tb <= '0';
        wait for 100 ns;

        Load_Select_tb <= '1';
        wait for 100 ns;

        -- Test case 3: for index 230501T Load_Select = 0, select
Immediate_Val
```

```
        Immediate_Val_tb <= "0101";
        AddSub_tb <= "0110";
        Load_Select_tb <= '0';
        wait for 100 ns;

        Load_Select_tb <= '1';
        wait for 100 ns;

        -- Test case 4: for index 230571F Load_Select = 0, select
Immediate_Val
        Immediate_Val_tb <= "1011";
        AddSub_tb <= "1010";
        Load_Select_tb <= '0';
        wait for 100 ns;

        Load_Select_tb <= '1';
        wait for 100 ns;
        -- Finish simulation
        wait;
    end process;

end Behavioral;
```

MUX 8way 4bit

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

entity TB_Mux_8way_4bit is
end TB_Mux_8way_4bit;

architecture Behavioral of TB_Mux_8way_4bit is

    component Mux_8way_4bit
        Port (  reg0 : in STD_LOGIC_VECTOR (3 downto 0);
                reg1 : in STD_LOGIC_VECTOR (3 downto 0);
                reg2 : in STD_LOGIC_VECTOR (3 downto 0);
                reg3 : in STD_LOGIC_VECTOR (3 downto 0);
                reg4 : in STD_LOGIC_VECTOR (3 downto 0);
                reg5 : in STD_LOGIC_VECTOR (3 downto 0);
                reg6 : in STD_LOGIC_VECTOR (3 downto 0);
                reg7 : in STD_LOGIC_VECTOR (3 downto 0);
                Reg_select : in STD_LOGIC_VECTOR (2 downto 0);
                output : out STD_LOGIC_VECTOR (3 downto 0));
    end component;
```

```
    signal reg0_tb, reg1_tb, reg2_tb, reg3_tb, reg4_tb, reg5_tb,
reg6_tb, reg7_tb : STD_LOGIC_VECTOR(3 downto 0);
    signal Reg_select_tb : STD_LOGIC_VECTOR(2 downto 0);
    signal output_tb : STD_LOGIC_VECTOR(3 downto 0);

begin

    UUT: Mux_8way_4bit
        port map(
            reg0 => reg0_tb,
            reg1 => reg1_tb,
            reg2 => reg2_tb,
            reg3 => reg3_tb,
            reg4 => reg4_tb,
            reg5 => reg5_tb,
            reg6 => reg6_tb,
            reg7 => reg7_tb,
            Reg_select => Reg_select_tb,
            output => output_tb
        );

    stim_proc: process
    begin
        -- Initialize all inputs
        -- index 230111X
        reg0_tb <= "1111";
        reg1_tb <= "1101";
        -- index 230315C
        reg2_tb <= "1011";
        reg3_tb <= "1010";
        -- index 230501T
        reg4_tb <= "0101";
        reg5_tb <= "0110";
        -- index 230571F
        reg6_tb <= "1011";
        reg7_tb <= "1010";

        -- Test all select lines
        for i in 0 to 7 loop
            Reg_select_tb <= std_logic_vector(to_unsigned(i, 3));
            wait for 100 ns;
        end loop;

        wait;
    end process;
```

```
end Behavioral;
```

Instruction Decoder

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

entity TB_Instruction_Decoder is
end TB_Instruction_Decoder;

architecture Behavioral of TB_Instruction_Decoder is

    -- Component Declaration
    component Instruction_Dec
        Port (
            Reg_En        : out STD_LOGIC_VECTOR (2 downto 0);
            Load_Select   : out STD_LOGIC;
            Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
            MuxA_Select   : out STD_LOGIC_VECTOR (2 downto 0);
            MuxB_Select   : out STD_LOGIC_VECTOR (2 downto 0);
            AddSub_Select : out STD_LOGIC;
            JmpFlag       : out STD_LOGIC;
            JmpAddr       : out STD_LOGIC_VECTOR (2 downto 0);
            Instruct_Bus  : in STD_LOGIC_VECTOR (11 downto 0);
            JumpCheck     : in STD_LOGIC_VECTOR (3 downto 0)
        );
    end component;

    -- Signals
    signal Reg_En        : STD_LOGIC_VECTOR (2 downto 0);
    signal Load_Select   : STD_LOGIC;
    signal Immediate_Val : STD_LOGIC_VECTOR (3 downto 0);
    signal MuxA_Select   : STD_LOGIC_VECTOR (2 downto 0);
    signal MuxB_Select   : STD_LOGIC_VECTOR (2 downto 0);
    signal AddSub_Select : STD_LOGIC;
    signal JmpFlag       : STD_LOGIC;
    signal JmpAddr       : STD_LOGIC_VECTOR (2 downto 0);
    signal Instruct_Bus  : STD_LOGIC_VECTOR (11 downto 0);
    signal JumpCheck     : STD_LOGIC_VECTOR (3 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Instruction_Dec
        Port Map (
```

```
            Reg_En        => Reg_En,
            Load_Select   => Load_Select,
            Immediate_Val => Immediate_Val,
            MuxA_Select   => MuxA_Select,
            MuxB_Select   => MuxB_Select,
            AddSub_Select => AddSub_Select,
            JmpFlag       => JmpFlag,
            JmpAddr       => JmpAddr,
            Instruct_Bus  => Instruct_Bus,
            JumpCheck     => JumpCheck
        );

    -- Stimulus Process
    stim_proc: process
    begin
        -- Test ADD (opcode "00"), A=001, B=010
        Instruct_Bus <= "00" & "001" & "010" & "0000";  -- 2 + 3 +
3 + 4 = 12 bits
        JumpCheck <= "0000";
        wait for 100 ns;

        -- Test NEG (opcode "01"), B=011
        Instruct_Bus <= "01" & "011" & "000" & "0000";  -- A =
unused here
        wait for 100 ns;

        -- Test MOV (opcode "10"), Reg=100, Imm=1100
        Instruct_Bus <= "10" & "100" & "000" & "1100";
        wait for 100 ns;

        -- Test JMP with JumpCheck = 0000 (should jump), A=101,
Addr=011
        Instruct_Bus <= "11" & "101" & "000" & "0011";
        JumpCheck <= "0000";
        wait for 100 ns;

        -- Test JMP with JumpCheck ? 0000 (no jump), A=110,
Addr=101
        Instruct_Bus <= "11" & "110" & "000" & "0101";
        JumpCheck <= "1111";
        wait for 100 ns;

        wait;
    end process;

end Behavioral;
```

Nano Processor

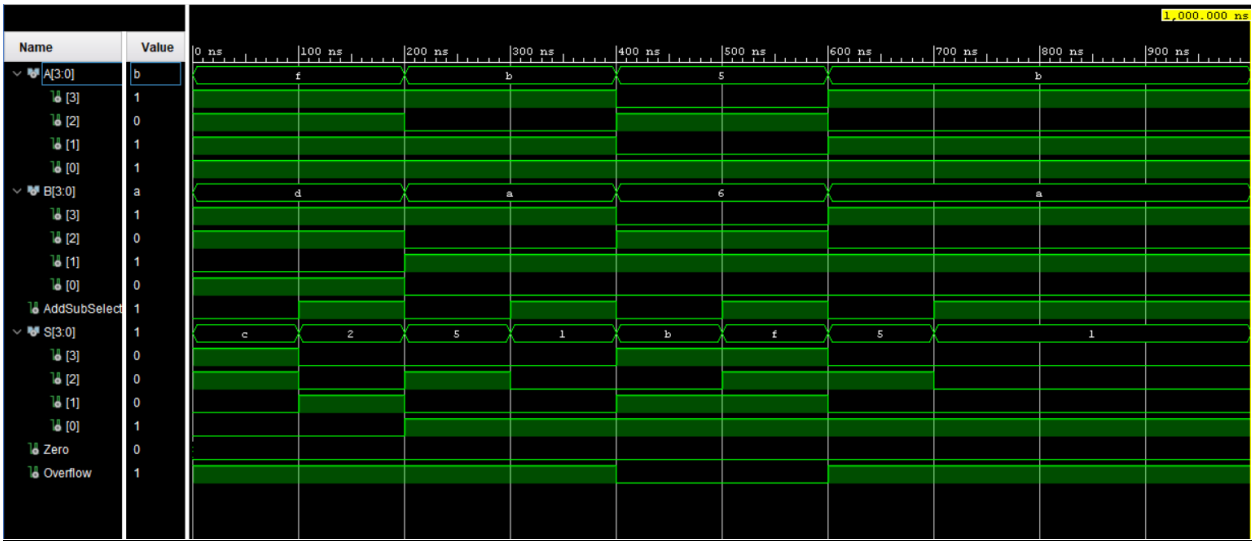
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_TEXTIO.all; -- For hwrite
use STD.TEXTIO.all;           -- For file operations

entity TB_Nanoprocessor is
end TB_Nanoprocessor;

architecture Behavioral of TB_Nanoprocessor is

    component Nanoprocessor is
        Port (
            Reset : in STD_LOGIC;
            clk : in STD_LOGIC;
            Zero_LED : out STD_LOGIC;
            Overflow_LED : out STD_LOGIC;
            LED0 : out STD_LOGIC;
            LED1 : out STD_LOGIC;
            LED2 : out STD_LOGIC;
            LED3 : out STD_LOGIC;
            Anode : out STD_LOGIC_VECTOR (3 downto 0);
            S_7Seg : out STD_LOGIC_VECTOR (6 downto 0)
        );
    end component;

    -- Signals
    signal clk          : STD_LOGIC := '0';
    signal reset        : STD_LOGIC := '1';
    signal zero_led     : STD_LOGIC;
    signal overflow_led : STD_LOGIC;
    signal led0, led1, led2, led3 : STD_LOGIC;
    signal anode        : STD_LOGIC_VECTOR(3 downto 0);
    signal s_7seg       : STD_LOGIC_VECTOR(6 downto 0);

    -- Clock period
    constant clk_period : time := 10 ns;

    -- Simulation control
    signal sim_finished : boolean := false;

    -- Function to convert std_logic_vector to string
    function vec2str(vec: std_logic_vector) return string is
```

```
        variable result: string(1 to vec'length);
    begin
        for i in vec'range loop
            result(i+1) := std_logic'image(vec(i))(2);
        end loop;
        return result;
    end function;

begin

    -- Instantiate Unit Under Test
    uut: Nanoprocessor
    port map (
        Reset => reset,
        clk => clk,
        Zero_LED => zero_led,
        Overflow_LED => overflow_led,
        LED0 => led0,
        LED1 => led1,
        LED2 => led2,
        LED3 => led3,
        Anode => anode,
        S_7Seg => s_7seg
    );

    -- Clock generation
    clk_process: process
    begin
        while not sim_finished loop
            clk <= '0';
            wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
        end loop;
        wait;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
        -- Initial reset
        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        -- Allow enough time for all 8 instructions to execute
```

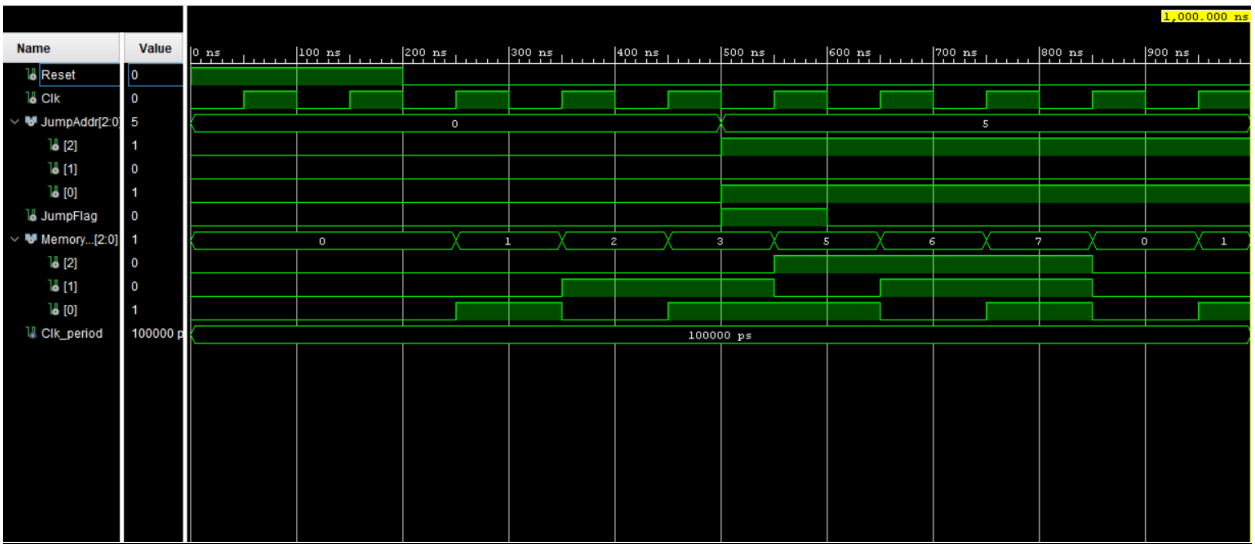
```
        wait for 80 * clk_period;

        wait for 10 ns;

        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        wait for 80 * clk_period;
        -- Check final outputs
        wait for 10 ns;

        wait;
    end process;

end Behavioral;
```

Timing Diagrams

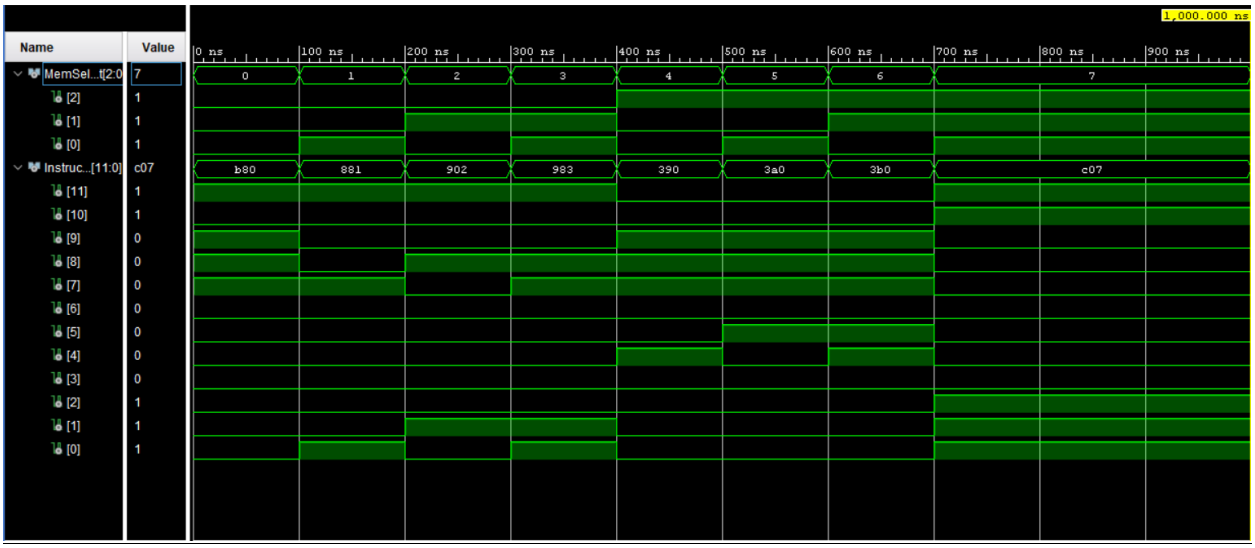
4-bit Add/Sub Unit



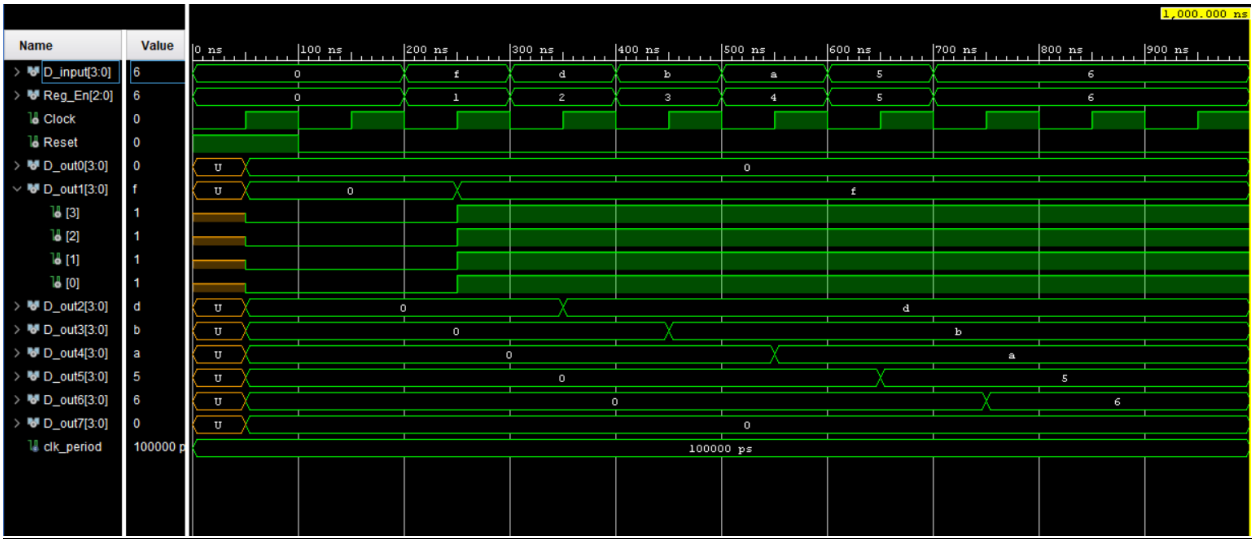
3-bit Program Counter



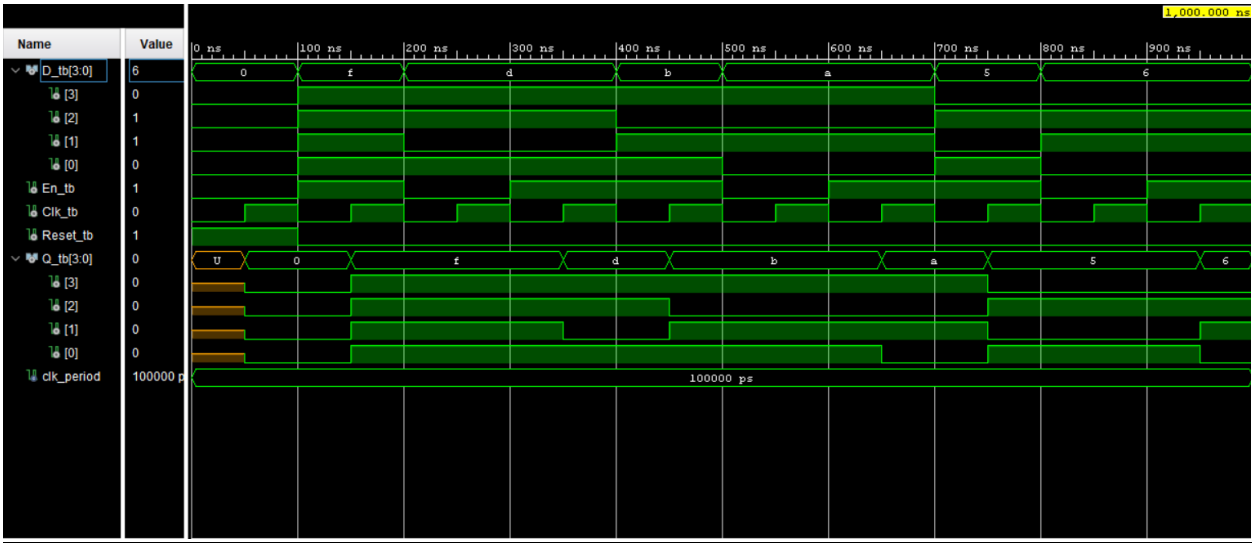
Program Rom



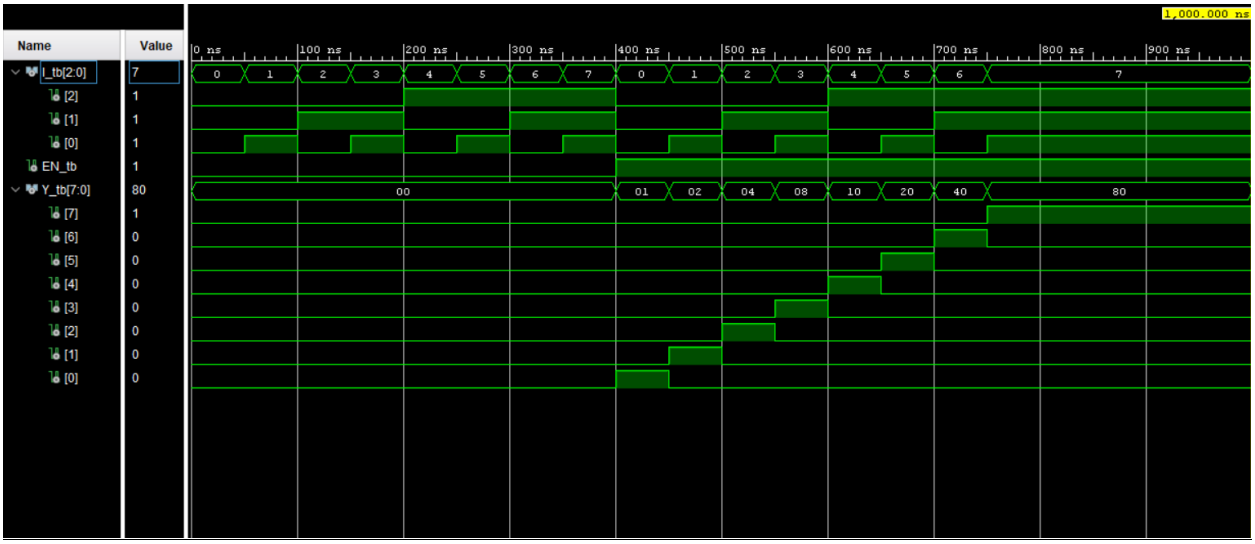
Register Bank



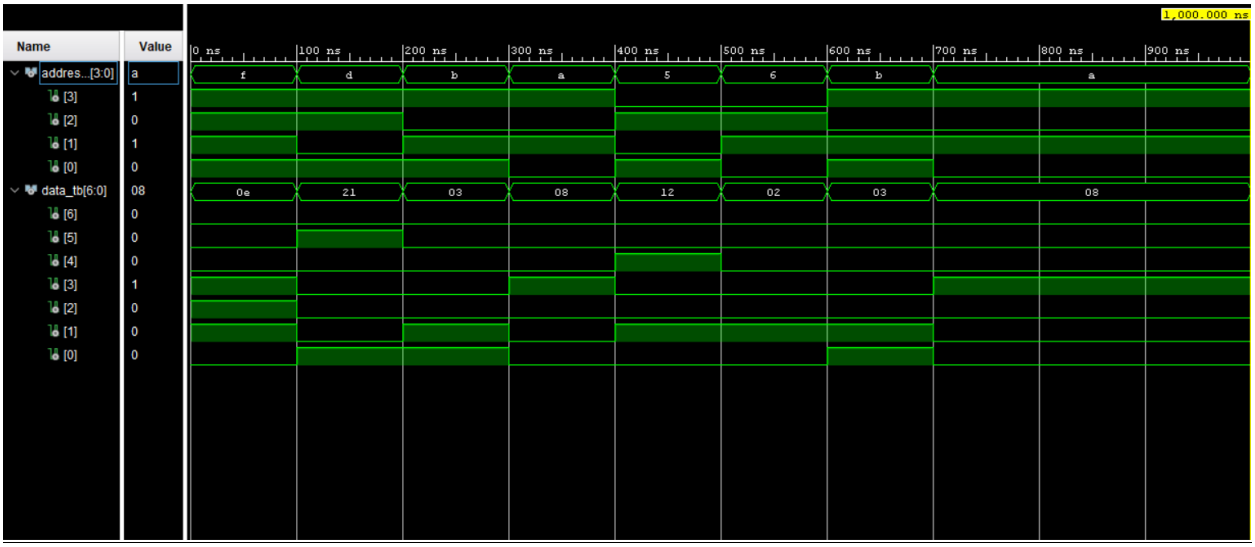
Register



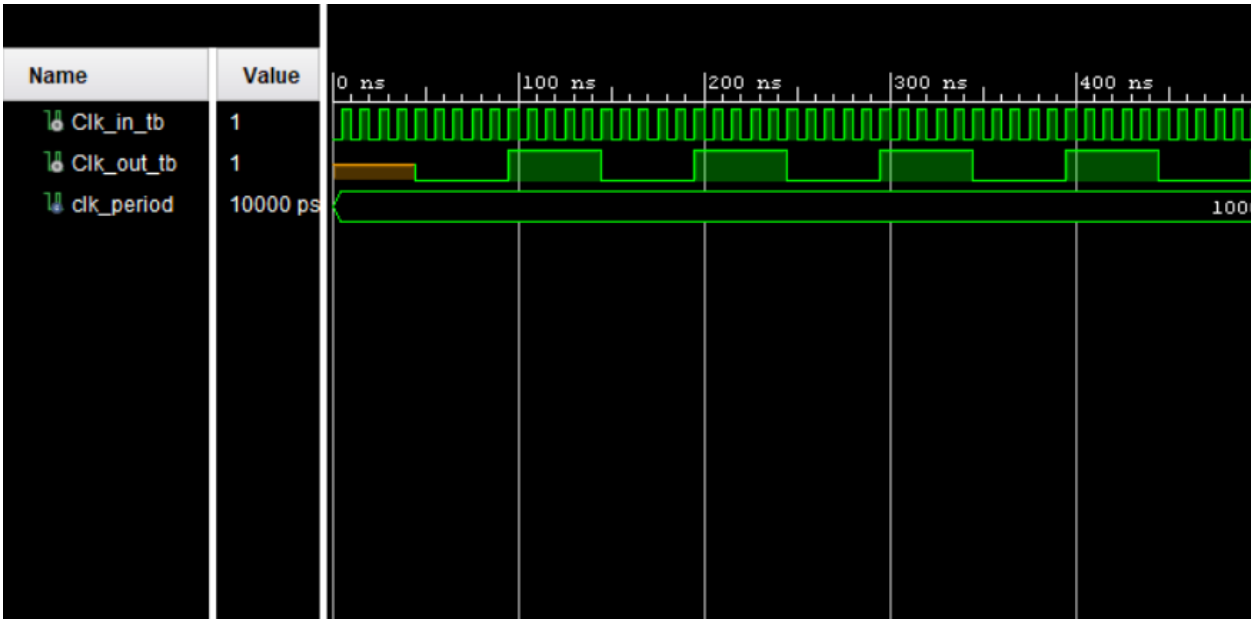
Decoder 3 to 8



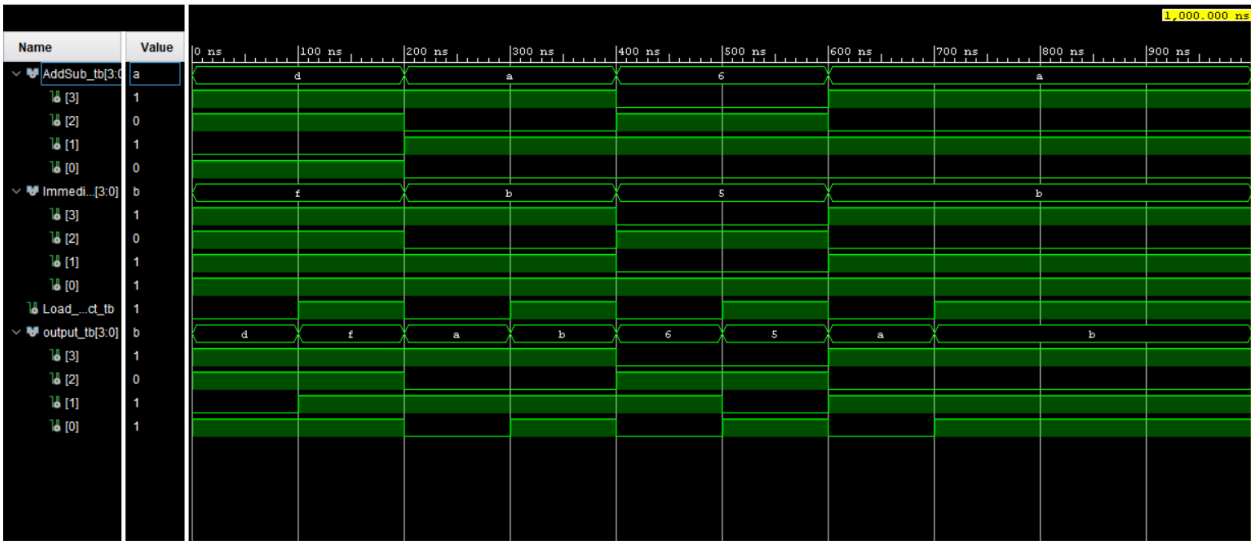
LUT_16_7



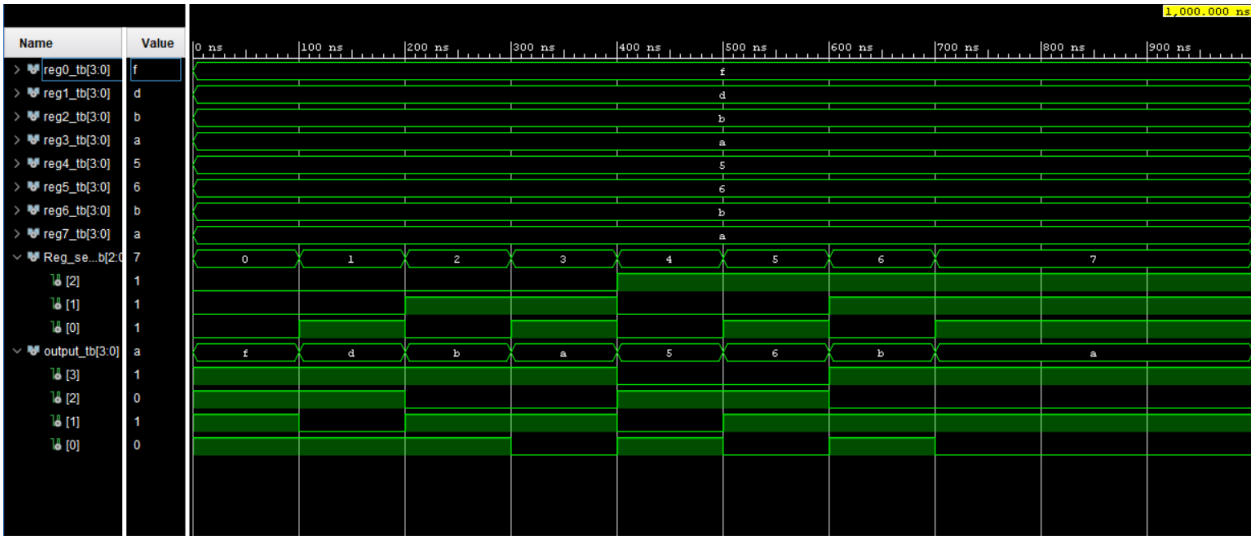
Slow Clock



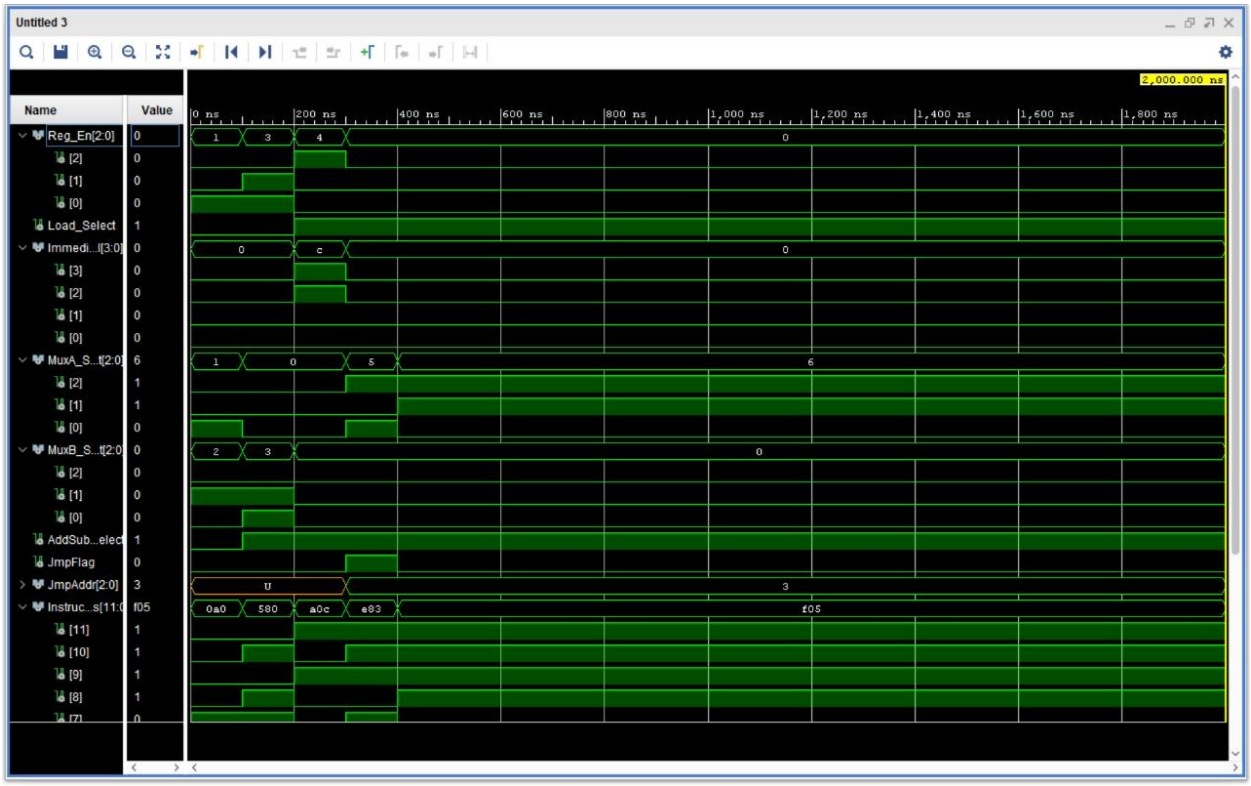
MUX 2way 4bit



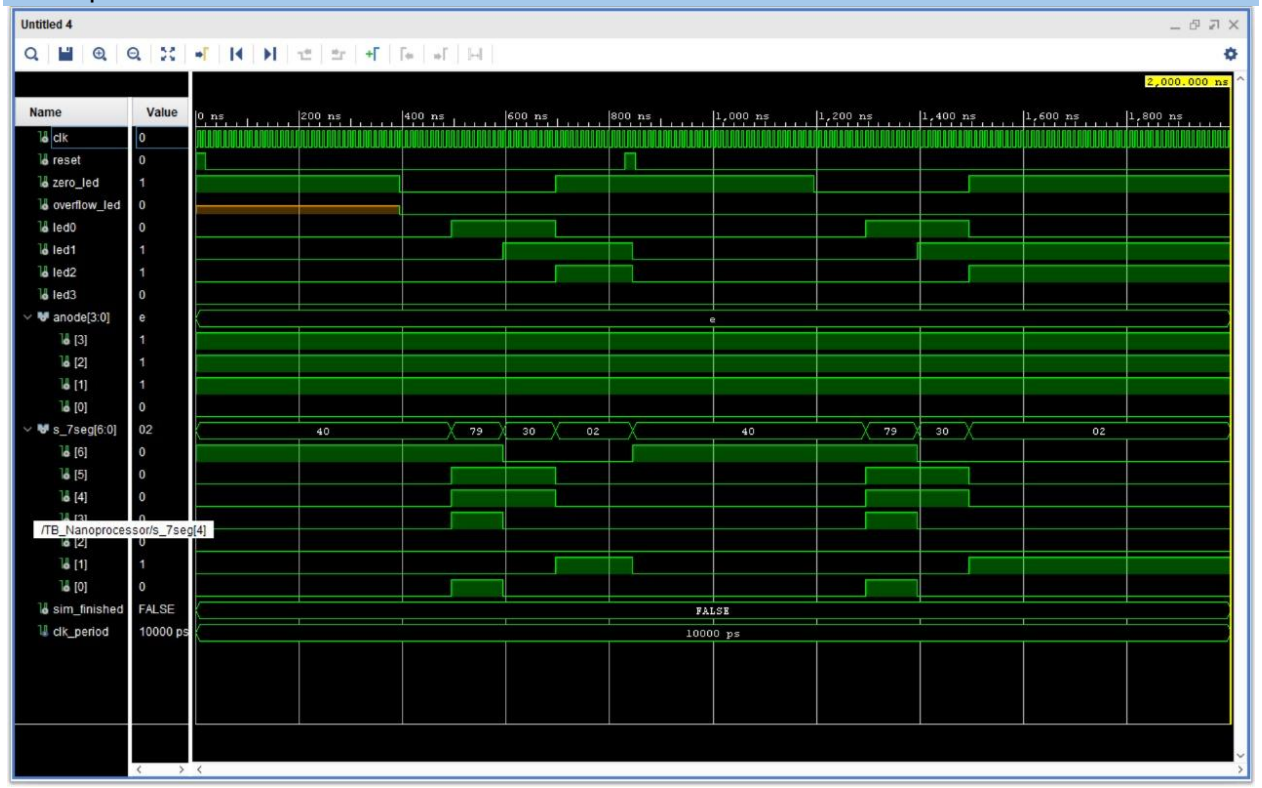
MUX 8way 4bit



Instruction decoder



Nano processor



Resource Utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	LUT Flip Flop Pairs (20800)	Bonded IOB (106)	BUFGCTRL (32)
Nanoprocessor	29	28	11	29	9	19	1
Instruction_Decoder (I...	0	5	2	0	0	0	0
Programming_Count...	19	3	6	19	3	0	0
RegisterBank (Registe...	9	16	6	9	0	0	0
Reg1 (Reg)	0	4	1	0	0	0	0
Reg2 (Reg_0)	0	4	1	0	0	0	0
Reg3 (Reg_1)	5	4	3	5	0	0	0
Reg7 (Reg_2)	4	4	3	4	0	0	0
Slow_Clock (Slow_Clk)	2	4	1	2	2	0	0

Issues and Solutions

Program Counter (PC_3bit) Synchronization Problem [Fixed]

Issue Identified:

- The original implementation using separate components (Mux_2way_3bit + RCA_3 adder) caused timing violations when The JumpFlag signal arrived later than the clock edge.

Possible Root Causes:

- Component Propagation Delays from the 3-bit ripple-carry adder (RCA_3) and the multiplexer.
- The JumpFlag control signal could change too close to the clock edge causing non-deterministic behavior.

Solution Implemented:

- Removed the 3-bit ripple-carry adder (RCA_3) and the multiplexer which were causing the delays and replaced with case based incrementor and inline jump logic.

Zero Flag and LED Synchronization Issue [Fix not implemented]

Issue Identified:

- There is a consistent 50ns delay between the zero-flag activation and the corresponding LED outputs turning off.

Possible Root Causes:

- The zero flag was generated combinatorially from ALU outputs. This maybe creating a critical path that must be resolved before LEDs update.

Proposed Solution:

- Adding a synchronized flag register that updates with the ALU result and using the registered zero flag (not combinational) to drive the zero LED.

Conclusions

- Through this lab, we successfully engineered a functional 4-bit nanoprocessor capable of executing a basic instruction set. Key accomplishments included the development of a 4-bit arithmetic unit for signed addition and subtraction, the design of a 3-bit adder for incrementing the Program Counter (PC), and the implementation of a PC using D Flip-Flops with reset capability. Additionally, we constructed k-way b-bit multiplexers for effective control and data routing, and built an Instruction Decoder to ensure precise control of processor operations by selectively activating relevant components.

- The project emphasized the importance of collaboration, with team members contributing to different subsystems in parallel. Effective communication and coordination were essential to integrate each component into a cohesive processor design.

- This lab experience offered valuable practical insight into digital circuit design and microprocessor architecture, strengthening our understanding of control logic, data path design, and collaborative engineering practices.

Contribution of the members

Dabarera G.D.M

- MUX 8way 4bit (Time spent: 1 day)
- MUX 2way 4bit (Time spent: 1 day)
- Lab report (Time spent: 1 day)

Kannangara T.P

- Nano processor (Time spent: 2 days)
- Instruction decoder (Time spent: 1 day)
- Slow Clock (Time spent: 1 day)

Priyadarshana D.M.R.N

- 4-bit Add/Sub Unit (Time spent: 1 day)
- Program Rom (Time spent: 1 day)
- 3-bit Adder (Time spent: 1 day)

Sampath W.I.D

- Decoder 3 to 8 (Time spent: 1 day)
- Register Bank (Time spent: 1 day)
- 3-bit Program Counter (Time spent: 1 day)

Total time spent: about 14 days