LAB 9-10 : NANOPROCESSOR DESIGN

Lab Report

CS1050 - Computer Organization and Digital Design
Department of Computer Science and Engineering
University of Moratuwa

Group 14

- GURUSINGHE C.R. 230222M
- DAISHIKA K.S.H. 230112C
- AMANTHA H.D.K.N. 230031C
- DIVYANGI W.A.S. 230166T

Table of Contents

1. Introduction

2. Components Design

- **2.1 Program ROM**
- **2.2 Buses**
- **2.3 Instruction Decoder**
- **2.4 Program Counter**
- **2.5 Register Bank**
- **2.6 k-way b-bit Multiplexers**
- **2.7 4-bit Add/Subtract Unit**
- **2.8 3-bit Adder**

3. Structure of Instructions

4. High-Level Diagram of the Nano Processor

5. Improved Nano Processor

- **5.1 Design Source File – Improved Nano Processor**
- **5.2 Elaborated Design Schematic – Improved Nano Processor**
- **5.3 Implemented Design Schematic – Improved Nano Processor**
- **5.4 Simulation Source File – Improved Nano Processor**
- **5.5 Timing Diagram – Improved Nano Processor**

6. Program ROM

- **6.1 Design Source File – Program ROM**
- **6.2 Elaborated Design Schematic – Program ROM**
- **6.3 Simulation Source File – Program ROM**
- **6.4 Timing Diagram – Program ROM**

7. Instruction Decoder

- **7.1 Design Source File – Instruction Decoder**
- **7.2 Elaborated Design Schematic – Instruction Decoder**
- **7.3 Simulation Source File – Instruction Decoder**
- **7.4 Timing Diagram – Instruction Decoder**

8. Program Counter

- **8.1 Design Source File – Program Counter**
- **8.2 Elaborated Design Schematic – Program Counter**
- **8.3 Simulation Source File – Program Counter**
- **8.4 Timing Diagram – Program Counter**

9. 3 Bit Adder

- **9.1 Design Source File – 3 Bit Adder**
- **9.2 Elaborated Design Schematic – 3 Bit Adder**
- **9.3 Simulation Source File – 3 Bit Adder**
- **9.4 Timing Diagram – 3 Bit Adder**

10. Register Bank

- **10.1 Design Source File – Register Bank**
- **10.2 Elaborated Design Schematic – Register Bank**
- **10.3 Simulation Source File – Register Bank**
- **10.4 Timing Diagram – Register Bank**

11. 4 Bit Adder/Subtractor

- **11.1 Design Source File – 4 Bit Adder/Subtractor**

- **11.2 Elaborated Design Schematic – 4 Bit Adder/Subtractor**
- **11.3 Simulation Source File – 4 Bit Adder/Subtractor**
- **11.4 Timing Diagram – 4 Bit Adder/Subtractor**

12. 8-Way 4-Bit Multiplexer

- **12.1 Design Source File – 8-Way 4-Bit Multiplexer**
- **12.2 Elaborated Design Schematic – 8-Way 4-Bit Multiplexer**
- **12.3 Simulation Source File – 8-Way 4-Bit Multiplexer**
- **12.4 Timing Diagram – 8-Way 4-Bit Multiplexer**

13. 2-Way 3-Bit Multiplexer

- **13.1 Design Source File – 2-Way 3-Bit Multiplexer**
- **13.2 Elaborated Design Schematic – 2-Way 3-Bit Multiplexer**
- **13.3 Simulation Source File – 2-Way 3-Bit Multiplexer**
- **13.4 Timing Diagram – 2-Way 3-Bit Multiplexer**

14. 2-Way 4-Bit Multiplexer

- **14.1 Design Source File – 2-Way 4-Bit Multiplexer**
- **14.2 Elaborated Design Schematic – 2-Way 4-Bit Multiplexer**
- **14.3 Simulation Source File – 2-Way 4-Bit Multiplexer**
- **14.4 Timing Diagram – 2-Way 4-Bit Multiplexer**

15. Slow Clock

- **15.1 Design Source File – Slow Clock**
- **15.2 Elaborated Design Schematic – Slow Clock**
- **15.3 Simulation Source File – Slow Clock**
- **15.4 Timing Diagram – Slow Clock**

16. 7 Segment Display

- **16.1 Design Source File – 7 Segment Display**
- **16.2 Elaborated Design Schematic – 7 Segment Display**
- **16.3 Simulation Source File – 7 Segment Display**
- **16.4 Timing Diagram – 7 Segment Display**

17. Improved Components and Modules

- **17.1 Improved Program ROM**
- **17.2 Improved Instruction Decoder**
- **17.3 Improved Register Bank**

- **17.4 Improved Logical Unit**
- **17.5 Improved 4-bit Multiplier**
- **17.6 Improved 4-bit Comparator**
- **17.7 Bit Shifter**
- **17.8 5-way 4-bit Multiplexer**
- **Constraint File**

18. Simulation Results and Discussion

19. Conclusion

20. Contribution of Team Members

Introduction

The purpose of this lab is to build a 4-bit nano processor that can carry out some basic instructions. Key parts of the design and use of this processor were the arithmetic unit, program counter, register bank, multiplexers, and instruction decoder. All the modules were crucial for the proper work of the processor. The lab had two benefits: it helped us learn digital circuit design and taught us how to work as a team by conducting simulations and hardware testing on the BASYS 3 board.

Components Design

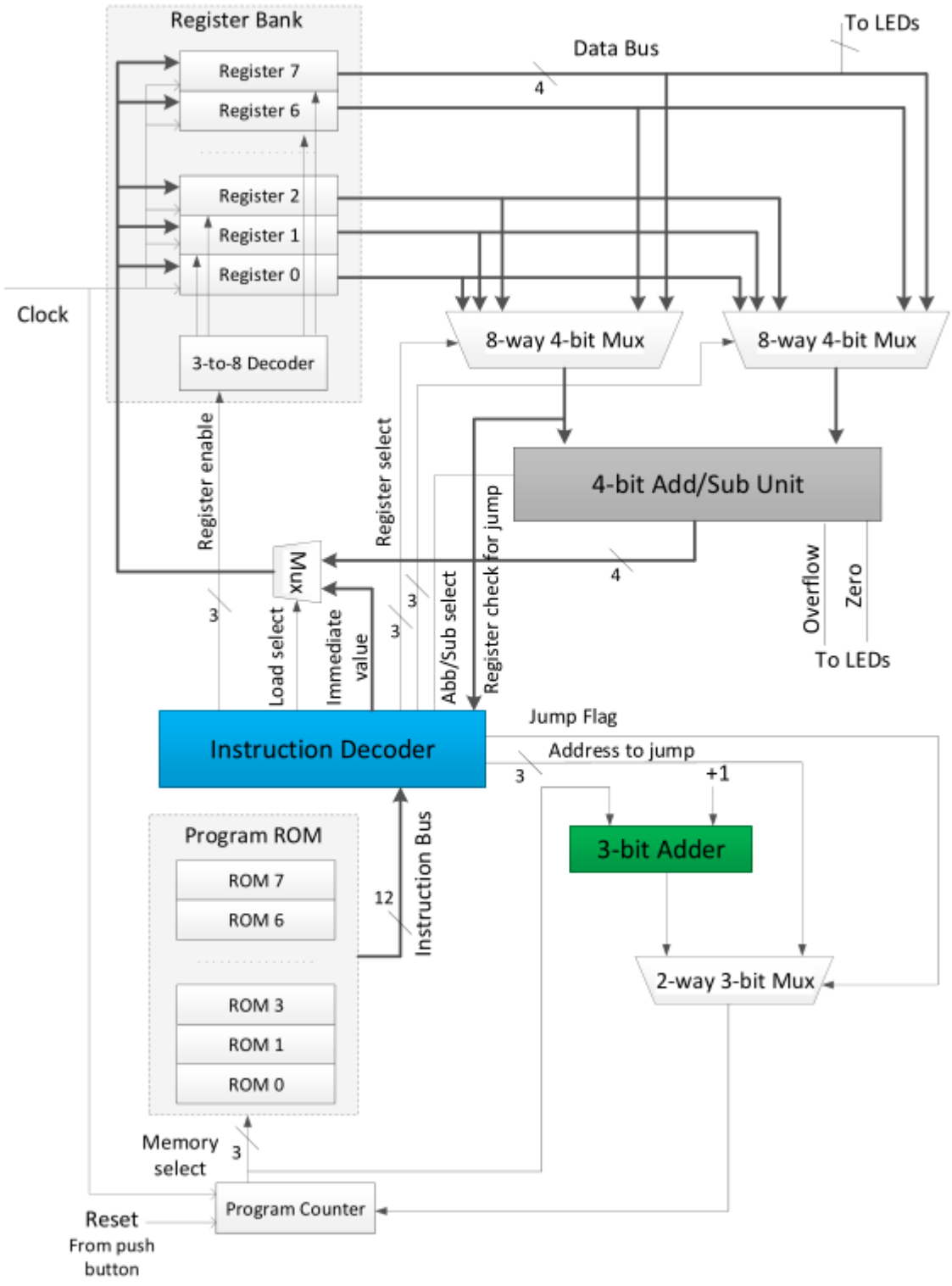
1. Program ROM: This module functions as the memory unit that stores the processor's assembly code. It holds the sequence of machine instructions that drive the processor's behavior during execution.
2. Buses: Buses serve as communication channels within the processor, enabling the transfer of instructions and data among different modules, such as the ROM, decoder, and register bank.
3. Instruction Decoder: The Instruction Decoder plays a crucial role in translating binary instructions retrieved from the ROM into specific control signals, ensuring that each processor component performs its designated task accurately.
4. Program Counter: Responsible for tracking the address of the current instruction, the Program Counter ensures orderly instruction flow by updating its value to point to the next instruction after each fetch cycle.
5. Register Bank: Acting as temporary memory, the Register Bank stores 4-bit data values used during processing. Its multiple registers allow efficient data access and quick read/write operations.

6. k-way b-bit Multiplexers: These components are essential for internal data routing. Controlled by select signals, they determine which data source is passed through, allowing flexible and efficient handling of operations.
7. 4-bit Add/Subtract Unit: Designed to perform core arithmetic functions, this unit supports both addition and subtraction using 2's complement representation. It works in conjunction with a multiplexer to select input registers and stores the result in the appropriate location.
8. 3-bit Adder: Used to update the Program Counter, the 3-bit Adder increments its value by one to facilitate step-by-step execution of instructions from memory.

Structure of Instructions

Instruction	Description	Format (12-bit instruction)
MOVI R, *d*	Move immediate value *d* to register R, i.e., $R \leftarrow d$ $R \in [0, 7]$, $d \in [0, 15]$	1 0 R R R 0 0 0 d d d d
ADD Ra, Rb	Add values in registers Ra and Rb and store the result in Ra, i.e., Ra ← Ra + Rb Ra, Rb ∈ [0, 7]	0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0
NEG R	2's complement of registers R, i.e., R ← – R R ∈ [0, 7]	0 1 R R R 0 0 0 0 0 0 0 0
JZR R, d	Jump if value in register R is 0, i.e., If R == 0 PC ← d; Else PC ← PC + 1; R ∈ [0, 7], *d* ∈ [0, 7]	1 1 R R R 0 0 0 0 d d d

High-Level Diagram of the Nano Processor



Nano Processor

Design Source File – Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/15/2025 12:37:57 AM
-- Design Name:
-- Module Name: Nano_Processor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

-- 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 (
  Reset : IN STD_LOGIC;
  Clk_in : IN STD_LOGIC;
  Zero : OUT STD_LOGIC;
  Overflow : OUT STD_LOGIC;
```

```
  LED : OUT STD_LOGIC_VECTOR(3 downto 0);
  Display : OUT STD_LOGIC_VECTOR(6 downto 0); --7 segment display
  Anode : OUT STD_LOGIC_VECTOR(3 downto 0)
   );
end Nano_Processor;

architecture Behavioral of Nano_Processor is

COMPONENT Program_ROM
    Port (
        Rom_IN  : IN  STD_LOGIC_VECTOR(2 downto 0);
        Rom_OUT : OUT STD_LOGIC_VECTOR(11 downto 0)
    );
END COMPONENT;

COMPONENT Ins_Decoder
  Port (
  Ins :          IN STD_LOGIC_VECTOR(11 downto 0); --Instructions
  Jump :         IN STD_LOGIC_VECTOR( 3 downto 0);
  Register_EN : OUT STD_LOGIC_VECTOR(2 downto 0); --Register
Enable
  Load_sel :     OUT STD_LOGIC;                    -- Select the
load
  I_value :      OUT STD_LOGIC_VECTOR(3 downto 0); --Immediate
value
  Reg_A :        OUT STD_LOGIC_VECTOR(2 downto 0);
  Reg_B :        OUT STD_LOGIC_VECTOR(2 downto 0);
  Add_sub :      OUT STD_LOGIC;
  Jump_flag :    OUT STD_LOGIC;
  Address_J :    OUT STD_LOGIC_VECTOR(2 downto 0)
    );
END COMPONENT;

COMPONENT p_counter
    Port ( Mux_Output : in STD_LOGIC_VECTOR (2 downto 0);
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
END COMPONENT;

COMPONENT T_bit_Adder --three bit adder
    Port (
        A      : in  STD_LOGIC_VECTOR(2 downto 0);
        B      : in  STD_LOGIC_VECTOR(2 downto 0);
```

```
        Sum    : out STD_LOGIC_VECTOR(2 downto 0)
    );
END COMPONENT;

COMPONENT Reg_Bank
  Port (
  reg_en : IN STD_LOGIC_VECTOR(2 downto 0); --select the register
  val_in : IN STD_LOGIC_VECTOR(3 downto 0); -- data for registers
  clk_in : IN STD_LOGIC; --input clock signal
  res    : IN STD_LOGIC; --reset the all registers in bank

  --out put of data fropm registers
  R_0 : out STD_LOGIC_VECTOR (3 downto 0);
  R_1 : out STD_LOGIC_VECTOR (3 downto 0);
  R_2 : out STD_LOGIC_VECTOR (3 downto 0);
  R_3 : out STD_LOGIC_VECTOR (3 downto 0);
  R_4 : out STD_LOGIC_VECTOR (3 downto 0);
  R_5 : out STD_LOGIC_VECTOR (3 downto 0);
  R_6 : out STD_LOGIC_VECTOR (3 downto 0);
  R_7 : out STD_LOGIC_VECTOR (3 downto 0)
   );
END COMPONENT;

COMPONENT Adder_Substractor
    PORT(
    A: IN STD_LOGIC_VECTOR(3 downto 0);
    B: IN STD_LOGIC_VECTOR(3 downto 0);
    EN:IN STD_LOGIC; --This enable system decide whether use
adder or substractor
    V: OUT STD_LOGIC;--overflow bit
    Ca_out:OUT STD_LOGIC;
    zero : OUT STD_LOGIC;
    D: OUT STD_LOGIC_VECTOR(3 downto 0));
END COMPONENT;

COMPONENT MUX_8way_4bit
    Port ( D0 : in STD_LOGIC_VECTOR (3 downto 0);
           D1 : in STD_LOGIC_VECTOR (3 downto 0);
           D2 : in STD_LOGIC_VECTOR (3 downto 0);
           D3 : in STD_LOGIC_VECTOR (3 downto 0);
           D4 : in STD_LOGIC_VECTOR (3 downto 0);
           D5 : in STD_LOGIC_VECTOR (3 downto 0);
           D6 : in STD_LOGIC_VECTOR (3 downto 0);
           D7 : in STD_LOGIC_VECTOR (3 downto 0);
```

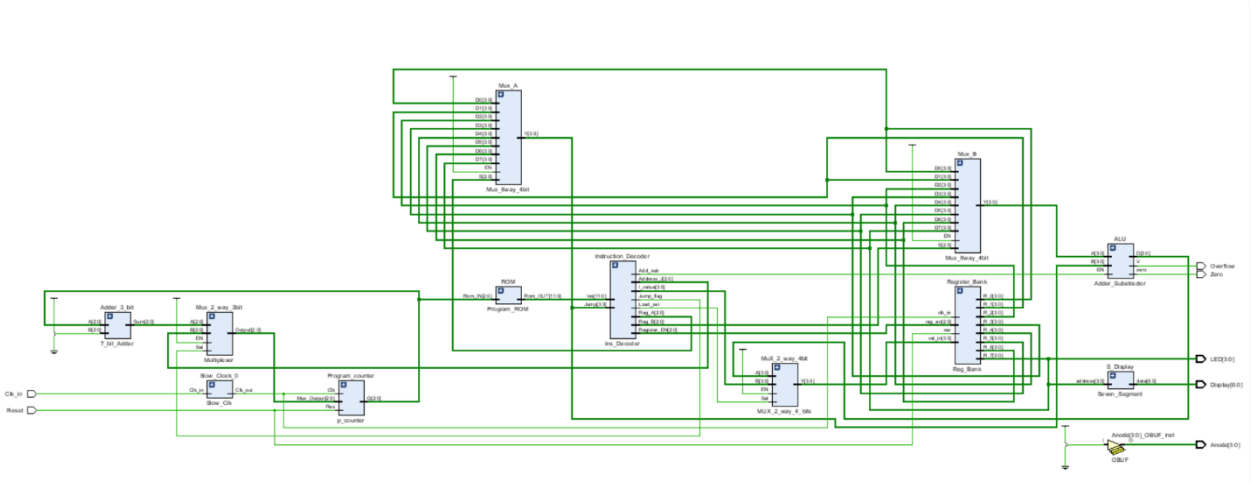
```
            EN : in STD_LOGIC;
            S : in STD_LOGIC_VECTOR (2 downto 0);
            Y : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;

COMPONENT MUX_2_way_4_bits
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Sel : in STD_LOGIC;
           EN : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;

COMPONENT Multiplexer
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Sel : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
END COMPONENT;

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

COMPONENT Seven_Segment
    PORT( address : in STD_LOGIC_VECTOR (3 downto 0);
    data : out STD_LOGIC_VECTOR (6 downto 0));
end COMPONENT;

SIGNAL Overflw,Z,slow_clock,carry : STD_LOGIC; --overflw for
Overflow Z for zero flag
SIGNAL Counter : STD_LOGIC_VECTOR( 2 downto 0); -- output of
program counter
SIGNAL Instructions : STD_LOGIC_VECTOR(11 downto 0); --Intruction
bus
SIGNAL Adder: STD_LOGIC_VECTOR(2 downto 0); --3 bit adder output
SIGNAL Multiplexer_out: STD_LOGIC_VECTOR(2 downto 0);  --2 way 3
bit multiplexer out
SIGNAL Load_Sel,Add_or_Sub,Jmp_Flag : STD_LOGIC;
SIGNAL Immediate_Value: STD_LOGIC_VECTOR (3 downto 0);
SIGNAL D_0,D_1,D_2,D_3,D_4,D_5,D_6,D_7,M_A,M_B,  M_0,R :
STD_LOGIC_VECTOR (3 downto 0);
```

```
SIGNAL Reg_Sel_MuxA,Reg_Enable,Reg_Sel_MuxB,Address_JMP :
STD_LOGIC_VECTOR (2 downto 0);


begin
Slow_Clock_0 : Slow_Clk
    PORT MAP ( Clk_in => Clk_in,
               Clk_out => slow_clock );

Program_counter: p_counter
    PORT MAP ( Mux_Output =>Multiplexer_out,
               Res =>Reset,
               Clk =>slow_clock,
               Q => counter);

ALU : Adder_Substractor
    PORT MAP(A =>M_B,
             B =>M_A,
             EN=>Add_or_Sub,
             V =>overflw,
             Ca_out=>carry,
             zero =>Z,
             D=>R);

Adder_3_bit:T_bit_Adder
    PORT MAP (
               A=>Counter ,
               B =>"001",
               Sum=>Adder
               );

Register_Bank: Reg_Bank
    PORT MAP(reg_en => Reg_Enable,
             val_in => M_0,
             clk_in => slow_clock,
             res    => Reset,
  --out put of data fropm registers
             R_0 =>D_0,
             R_1 =>D_1,
             R_2 =>D_2,
             R_3 =>D_3,
             R_4 =>D_4,
             R_5 =>D_5,
             R_6 =>D_6,
```

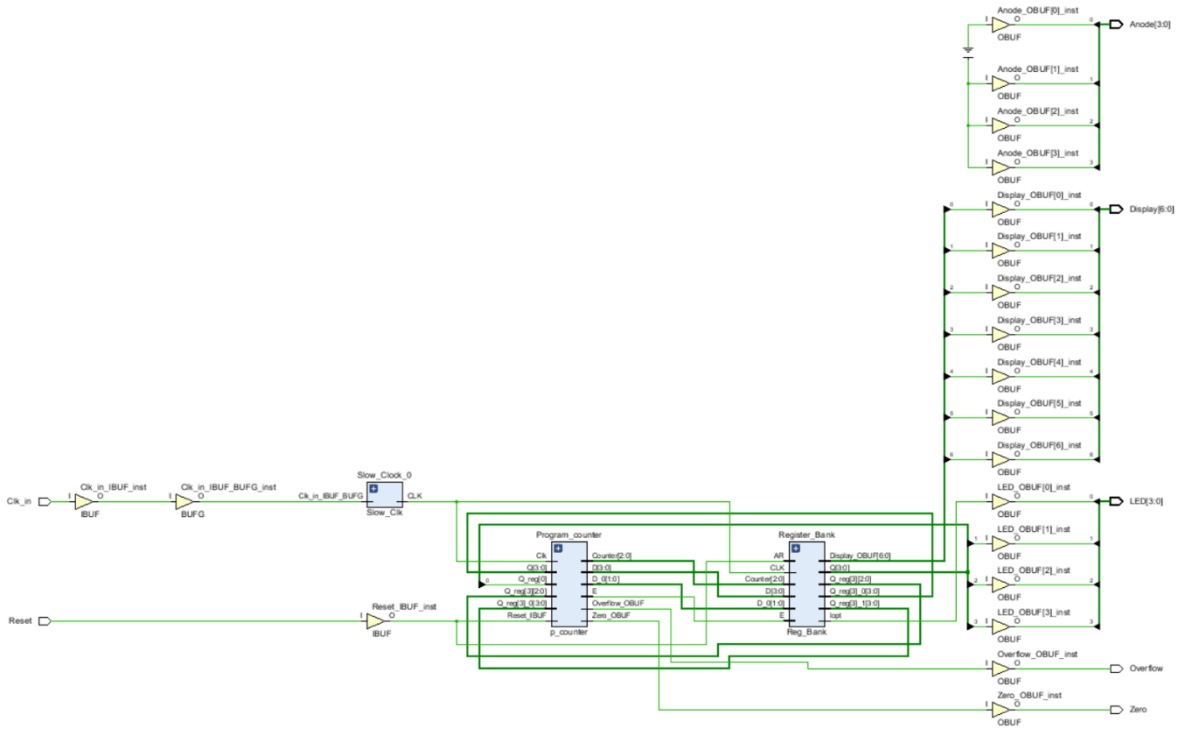
```
              R_7 =>D_7);

MuX_2_way_4bit : MUX_2_way_4_bits
    PORT MAP(A =>R,
             B =>Immediate_Value,
             Sel =>Load_Sel,
             EN =>'1',
             Y =>M_0);

Mux_2_way_3bit : Multiplexer
    PORT MAP(A=>Adder,
             B=>Address_JMP,
             EN=>'1',
             Sel=>Jmp_Flag,
             Output=>Multiplexer_out);

Mux_A : MUX_8way_4bit
    PORT MAP (D0=>D_0,
              D1=>D_1,
              D2=>D_2,
              D3=>D_3,
              D4=>D_4,
              D5=>D_5,
              D6=>D_6,
              D7=>D_7,
              EN=>'1',
              S =>Reg_Sel_MuxA,
              Y =>M_A);

Mux_B : MUX_8way_4bit
    PORT MAP (D0=>D_0,
              D1=>D_1,
              D2=>D_2,
              D3=>D_3,
              D4=>D_4,
              D5=>D_5,
              D6=>D_6,
              D7=>D_7,
              EN=>'1',
              S =>Reg_Sel_MuxB,
              Y =>M_B);

 ROM : Program_ROM
    PORT MAP (Rom_IN=>Counter ,
```

```
                Rom_OUT=>Instructions );

 S_Display : Seven_Segment
    PORT MAP(address=>D_7,
             data=>Display );

 Instruction_Decoder: Ins_Decoder
    PORT MAP(
      Ins =>Instructions,
      Jump =>M_A,
      Register_EN=>Reg_Enable,
      Load_sel=> Load_Sel,
      I_value=>Immediate_Value,
      Reg_A =>Reg_Sel_MuxA,
      Reg_B =>Reg_Sel_MuxB,
      Add_sub=>Add_or_Sub,
      Jump_flag=>Jmp_Flag,
      Address_J=>Address_JMP
      );

LED <= D_7;
Anode <= "1110";
Overflow <= Overflw;
Zero<=Z;
end Behavioral;
```

Elaborated Design Schematic – Nano Processor



Implemented Design Schematic – Nano Processor



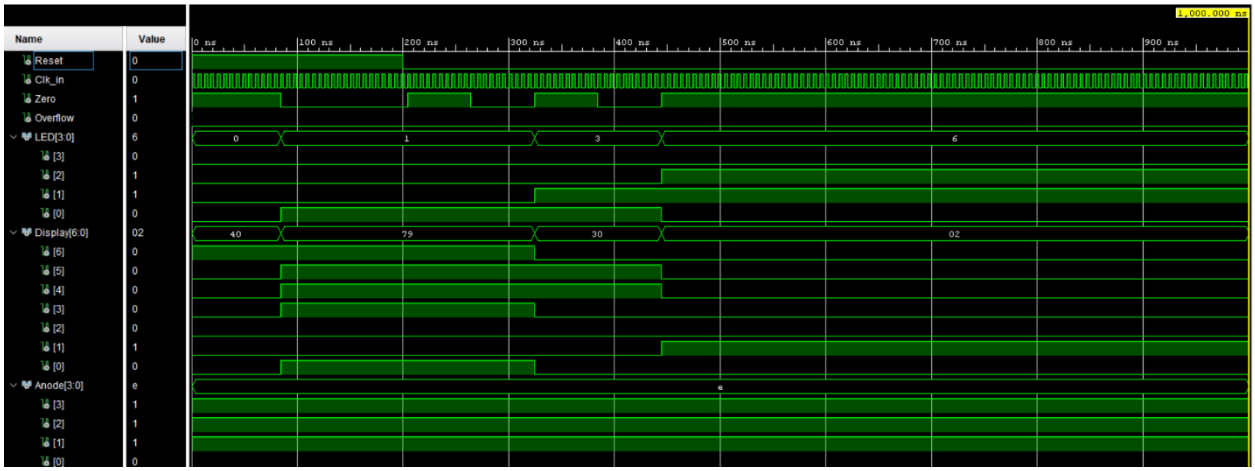
Simulation Source File – Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/15/2025 11:21:02 AM
-- Design Name:
-- Module Name: TB_Nanoprocessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

```
--------------------------------------------------------------------------------


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Nano_Processor_tb IS
END Nano_Processor_tb;

ARCHITECTURE behavior OF Nano_Processor_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT Nano_Processor
    PORT(
        Reset         : IN  std_logic;
        Clk_in        : IN  std_logic;
        Zero          : OUT std_logic;
        Overflow      : OUT std_logic;
        LED           : OUT std_logic_vector(3 downto 0);
        Display       : OUT std_logic_vector(6 downto 0);
        Anode         : OUT std_logic_vector(3 downto 0)
    );
    END COMPONENT;

    -- Testbench signals
    SIGNAL Reset        : std_logic := '0';
    SIGNAL Clk_in       : std_logic := '0';
    SIGNAL Zero         : std_logic;
    SIGNAL Overflow     : std_logic;
    SIGNAL LED          : std_logic_vector(3 downto 0);
    SIGNAL Display      : std_logic_vector(6 downto 0);
    SIGNAL Anode        : std_logic_vector(3 downto 0);

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: Nano_Processor
    PORT MAP (
        Reset        => Reset,
        Clk_in       => Clk_in,
        Zero         => Zero,
        Overflow     => Overflow,
        LED          => LED,
```

```
        Display      => Display,
        Anode        => Anode
    );

process
begin
    Clk_in <= NOT Clk_in;
    wait for 3ns;
    end process;

process
begin
    Reset <='1';
    wait for 200ns;
    Reset <='0';
    wait;

end process;

END behavior;
```

Timing Diagram – Nano Processor



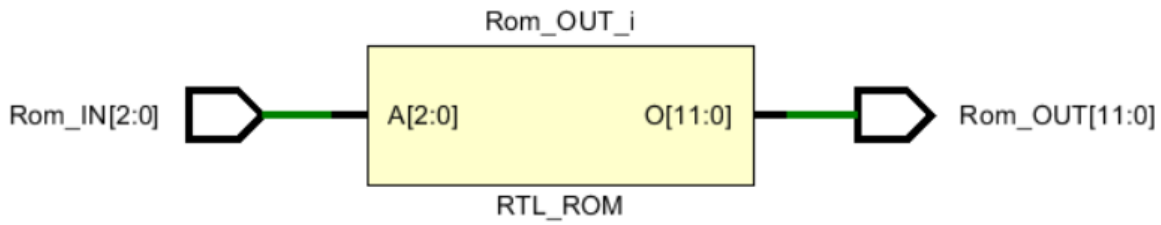
Program Rom

Design Source File – Program Rom

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;  -- Required for 'unsigned'
```

```
entity Program_ROM is
    Port (
        Rom_IN  : IN  STD_LOGIC_VECTOR(2 downto 0);
        Rom_OUT : 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 programRom : rom_type := (
        "101110000001", -- MOVI R7, 1
        "100100000010", -- MOVI R2, 2
        "001110100000", -- ADD R7, R2
        "100110000011", -- MOVI R3, 3
        "001110110000", -- ADD R7, R3
        "110000000111", -- JZR R0, 7
        "110000000111", -- JZR R0, 7
        "110000000111" -- JZR R0, 7
    );

begin
    Rom_OUT <= programRom(to_integer(unsigned(Rom_IN)));

end Behavioral;
```

Elaborated Design Schematic – Program Rom

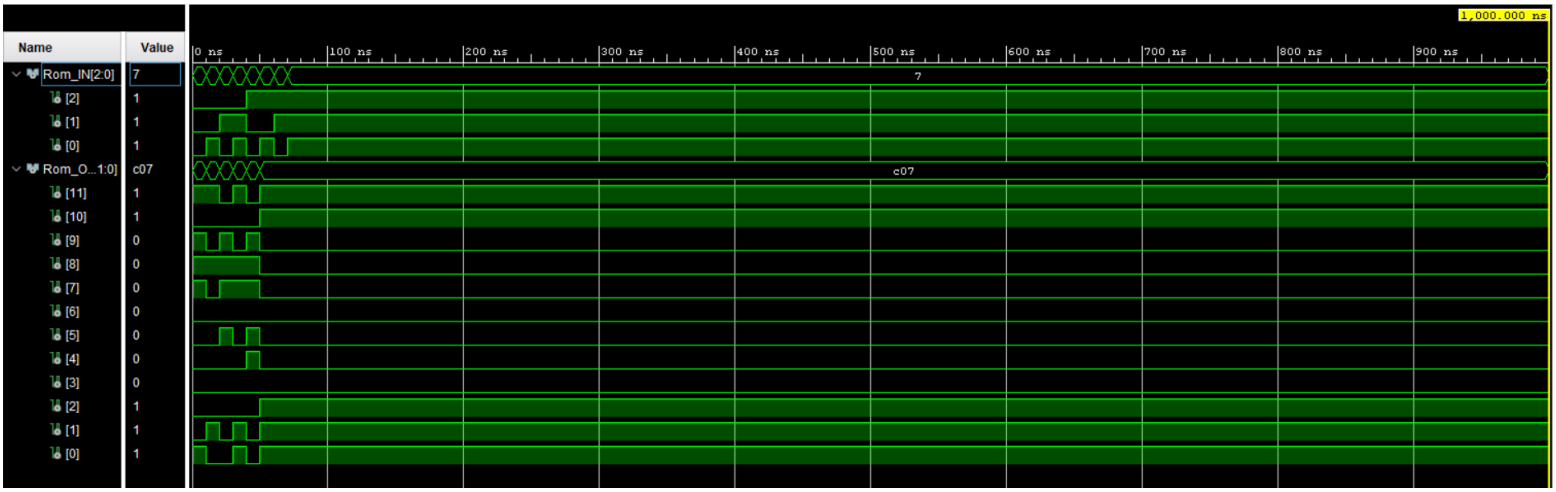


Simulation Source File - Program Rom

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

entity Program_ROM_tb is
```

```
-- Testbenches do not have ports
end Program_ROM_tb;

architecture Behavioral of Program_ROM_tb is

    -- Component declaration of the UUT
    component Program_ROM is
        Port (
            Rom_IN  : in  STD_LOGIC_VECTOR(2 downto 0);
            Rom_OUT : out STD_LOGIC_VECTOR(11 downto 0)
        );
    end component;

    -- Signals to connect to UUT
    signal Rom_IN  : STD_LOGIC_VECTOR(2 downto 0) := (others =>
'0');
    signal Rom_OUT : STD_LOGIC_VECTOR(11 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Program_ROM
        port map (
            Rom_IN  => Rom_IN,
            Rom_OUT => Rom_OUT
        );

    -- Stimulus process
    stim_proc: process
    begin
        for i in 0 to 7 loop
            Rom_IN <= std_logic_vector(to_unsigned(i, 3));  --
Apply address
            wait for 10 ns;                                 --
Wait for output to settle
        end loop;
        wait;  -- End of simulation
    end process;

end Behavioral;
```

Timing Diagram – Program Rom



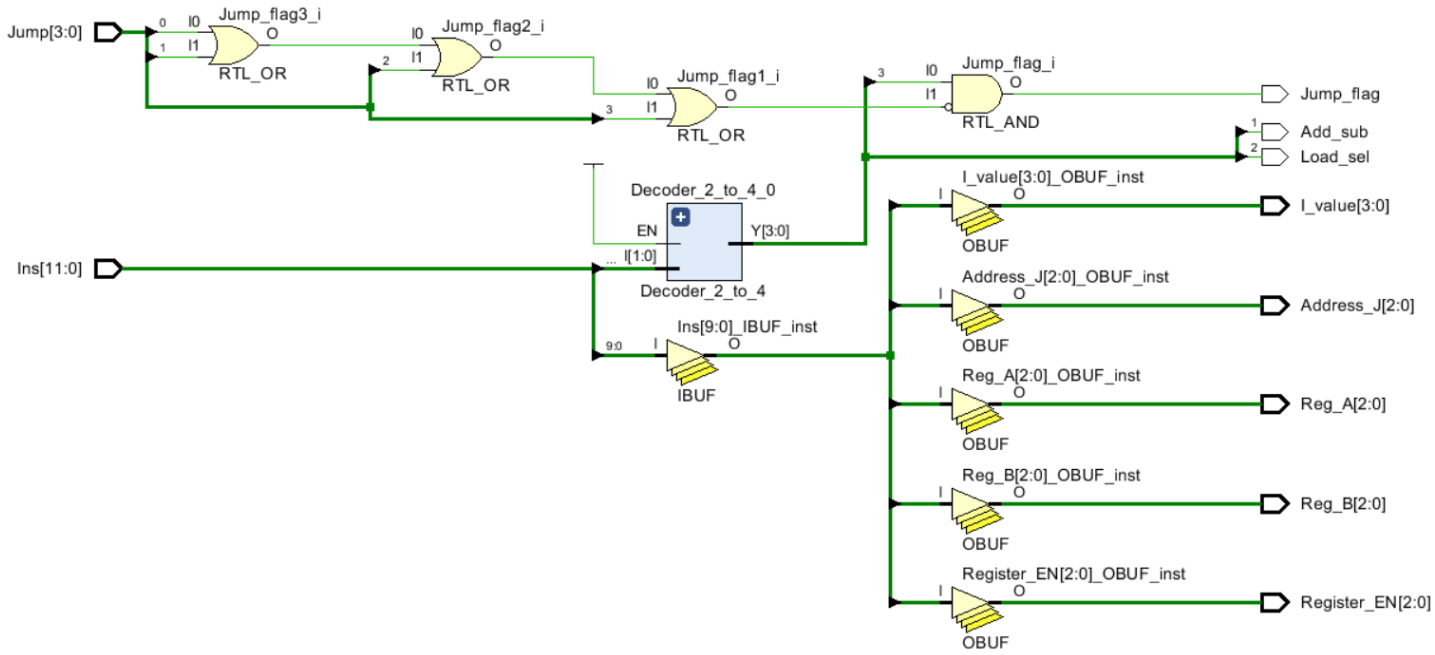
Instruction Decoder

Design Source File – 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 Ins_Decoder is
  Port (
  Ins :          IN STD_LOGIC_VECTOR(11 downto 0); --Instructions
  Jump :         IN STD_LOGIC_VECTOR( 3 downto 0);
  Register_EN : OUT STD_LOGIC_VECTOR(2 downto 0); --Register
Enable
  Load_sel :     OUT STD_LOGIC;                    -- Select the
load
  I_value :      OUT STD_LOGIC_VECTOR(3 downto 0); --Immediate
value
  Reg_A :        OUT STD_LOGIC_VECTOR(2 downto 0);
  Reg_B :        OUT STD_LOGIC_VECTOR(2 downto 0);
```

```
  Add_sub :     OUT STD_LOGIC;
  Jump_flag :   OUT STD_LOGIC;
  Address_J :   OUT STD_LOGIC_VECTOR(2 downto 0)
    );
end Ins_Decoder;

architecture Behavioral of Ins_Decoder is

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

SIGNAL ADD, NEG, MOV, JZR :STD_LOGIC;

begin

Decoder_2_to_4_0 : Decoder_2_to_4
    Port map ( I(0) => Ins(10),
    I(1) => Ins(11),
    EN => '1',
    Y(0) => ADD,
    Y(1) => NEG,
    Y(2) => MOV,
    Y(3) => JZR );

Reg_A <= Ins(9 downto 7);
Reg_B <= Ins(6 downto 4);
Register_EN <= Ins(9 downto 7);
Add_sub <= NEG;
I_value <= Ins(3 downto 0);
Jump_flag <= JZR AND ( NOT(Jump(0) OR Jump(1) OR Jump(2) OR
Jump(3)));
Address_J <= Ins(2 downto 0);
Load_Sel <= MOV;


end Behavioral;
```

Elaborated Design Schematic – Instruction Decoder



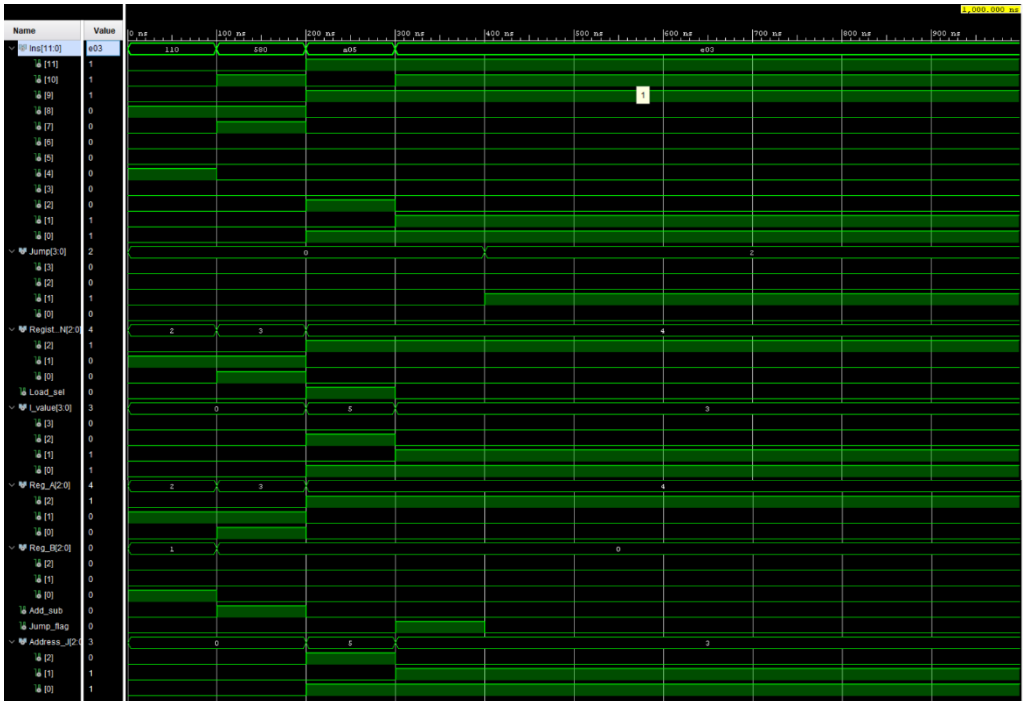
Simulation Source File – Instruction Decoder

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

entity Ins_Decoder_tb is
end Ins_Decoder_tb;

architecture Behavioral of Ins_Decoder_tb is

    -- Component declaration for the Unit Under Test (UUT)
    component Ins_Decoder
        Port (
            Ins         : in  STD_LOGIC_VECTOR(11 downto 0);
            Jump        : in  STD_LOGIC_VECTOR(3 downto 0);
            Register_EN : out STD_LOGIC_VECTOR(2 downto 0);
            Load_sel    : out STD_LOGIC;
            I_value     : out STD_LOGIC_VECTOR(3 downto 0);
            Reg_A       : out STD_LOGIC_VECTOR(2 downto 0);
            Reg_B       : out STD_LOGIC_VECTOR(2 downto 0);
            Add_sub     : out STD_LOGIC;
            Jump_flag   : out STD_LOGIC;
            Address_J   : out STD_LOGIC_VECTOR(2 downto 0)
        );
```

```
    end component;

    -- Signals for connecting to the UUT
    signal Ins         : STD_LOGIC_VECTOR(11 downto 0);
    signal Jump        : STD_LOGIC_VECTOR(3 downto 0);
    signal Register_EN : STD_LOGIC_VECTOR(2 downto 0);
    signal Load_sel    : STD_LOGIC;
    signal I_value     : STD_LOGIC_VECTOR(3 downto 0);
    signal Reg_A       : STD_LOGIC_VECTOR(2 downto 0);
    signal Reg_B       : STD_LOGIC_VECTOR(2 downto 0);
    signal Add_sub     : STD_LOGIC;
    signal Jump_flag   : STD_LOGIC;
    signal Address_J   : STD_LOGIC_VECTOR(2 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Ins_Decoder
        Port map (
            Ins         => Ins,
            Jump        => Jump,
            Register_EN => Register_EN,
            Load_sel    => Load_sel,
            I_value     => I_value,
            Reg_A       => Reg_A,
            Reg_B       => Reg_B,
            Add_sub     => Add_sub,
            Jump_flag   => Jump_flag,
            Address_J   => Address_J
        );

    -- Test Process
    process
    begin
        -- Test case 1: ADD instruction (opcode = "00")
        -- Ins = "00 AAABBB xxxx" e.g., Reg_A=010, Reg_B=001
        Ins <= "000100010000"; -- opcode=00, Reg_A=001,
Reg_B=001, rest=0000
        Jump <= "0000";  -- No jump condition
        wait for 100 ns;

        -- Test case 2: NEG instruction (opcode = "01")
        Ins <= "010110000000"; -- opcode=01, Reg_A=101
        Jump <= "0000";
```

```
        wait for 100 ns;

        -- Test case 3: MOV instruction (opcode = "10")
        Ins <= "101000000101"; -- opcode=10, Reg_A=100, Imme=0101
        Jump <= "0000";
        wait for 100 ns;

        -- Test case 4: JZR instruction with Jump = "0000"
(should trigger Jump_flag)
        Ins <= "111000000011"; -- opcode=11, Address_J = 0011
        Jump <= "0000";
        wait for 100 ns;

        -- Test case 5: JZR instruction with Jump != "0000"
(Jump_flag should be 0)
        Ins <= "111000000011"; -- same JZR instruction
        Jump <= "0010";
        wait for 100 ns;

        -- Finish simulation
        wait;
    end process;

end Behavioral;
```

Timing Diagram – Instruction Decoder



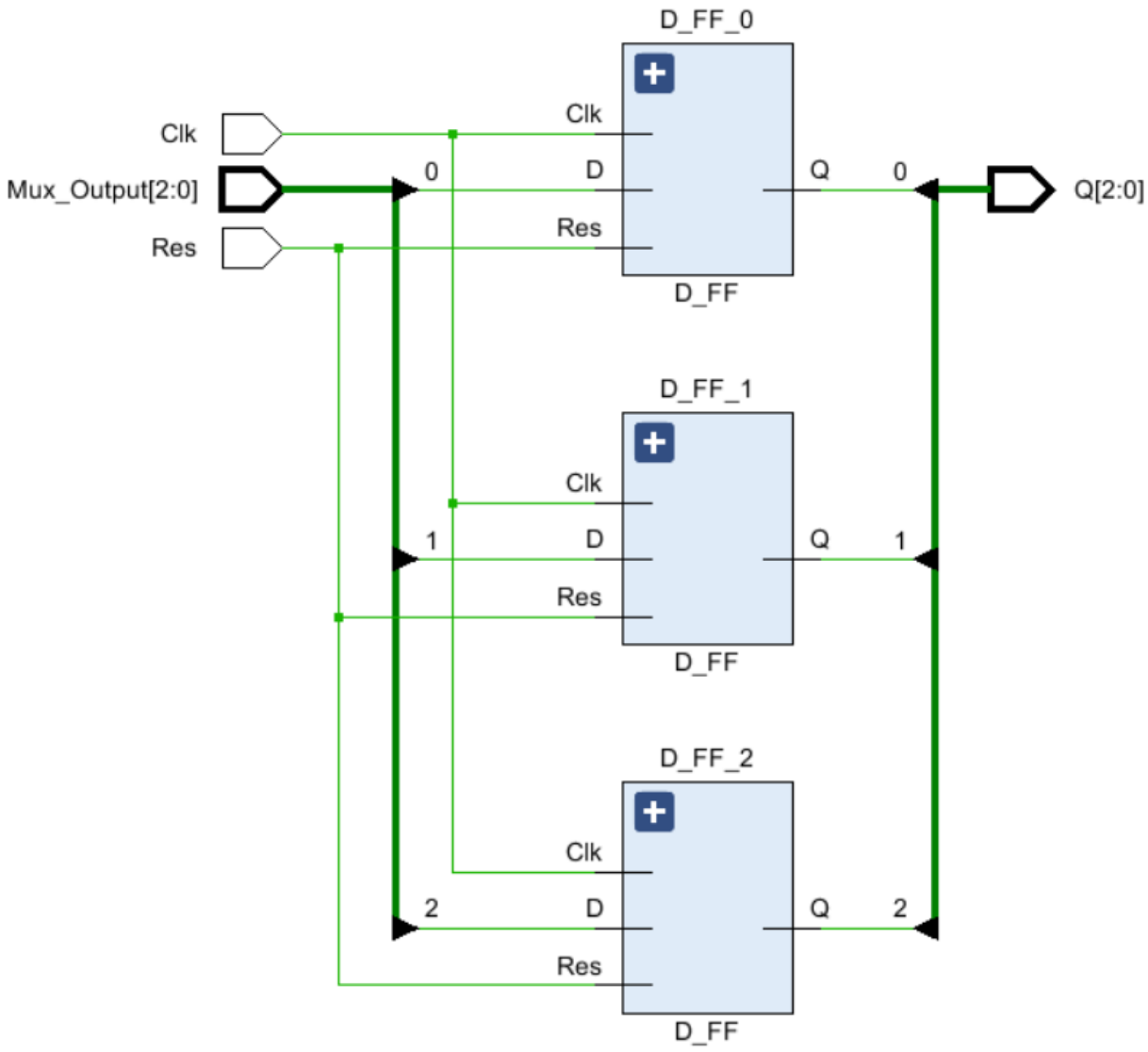
Program Counter

Design Source File – 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 p_counter is
    Port ( Mux_Output : in STD_LOGIC_VECTOR (2 downto 0):="000";
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end p_counter;

architecture Behavioral of p_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;

begin
    D_FF_0: D_FF
        port map ( D => Mux_Output(0),
                   Res => Res,
                   Clk => Clk,
                   Q => Q(0));

    D_FF_1: D_FF
        port map ( D => Mux_Output(1),
```

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

    D_FF_2: D_FF
        port map ( D => Mux_Output(2),
                    Res => Res,
                    Clk => Clk,
                    Q => Q(2));

end Behavioral;
```

Elaborated Design Schematic – Program Counter

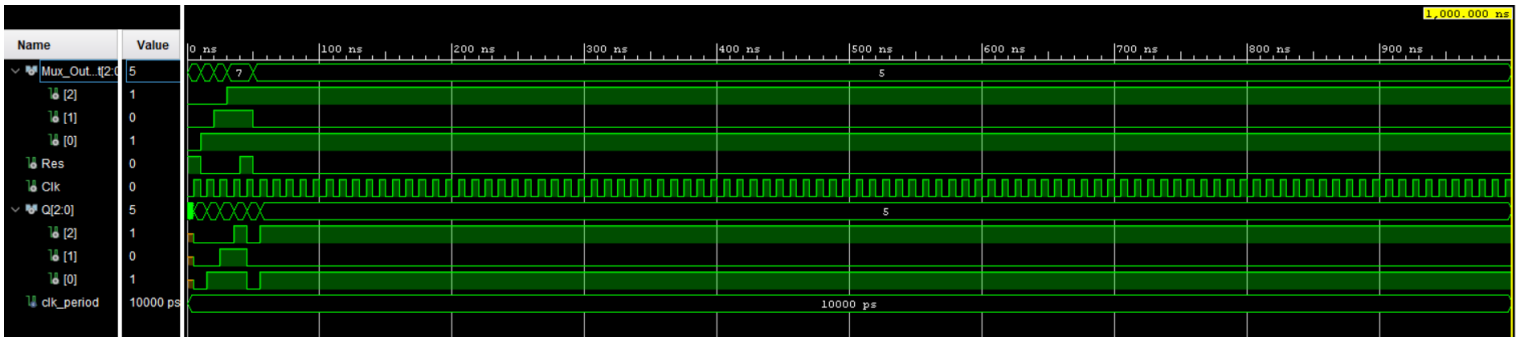


Simulation Source File – Program Counter

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

entity p_counter_tb is
end p_counter_tb;

architecture behavior of p_counter_tb is

    -- Component Declaration for the Unit Under Test (UUT)
    component p_counter
        Port (
            Mux_Output : in STD_LOGIC_VECTOR (2 downto 0);
            Res        : in STD_LOGIC;
            Clk        : in STD_LOGIC;
            Q          : out STD_LOGIC_VECTOR (2 downto 0)
        );
    end component;

    -- Signals to connect to UUT
    signal Mux_Output : STD_LOGIC_VECTOR (2 downto 0) := (others
=> '0');
    signal Res        : STD_LOGIC := '0';
    signal Clk        : STD_LOGIC := '0';
    signal Q          : STD_LOGIC_VECTOR (2 downto 0);

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

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: p_counter
        port map (
            Mux_Output => Mux_Output,
            Res        => Res,
            Clk        => Clk,
            Q          => Q
        );

    -- Clock process definition
    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
        -- Reset the counter
        Res <= '1';
        wait for clk_period;
        Res <= '0';

        -- Apply input and wait for rising clock edges
        Mux_Output <= "001";
        wait for clk_period;

        Mux_Output <= "011";
        wait for clk_period;

        Mux_Output <= "111";
        wait for clk_period;

        -- Activate reset again to observe behavior
        Res <= '1';
        wait for clk_period;
        Res <= '0';

        Mux_Output <= "101";
        wait for clk_period;

        wait;
    end process;

end behavioral;
```

Timing Diagram – Program Counter



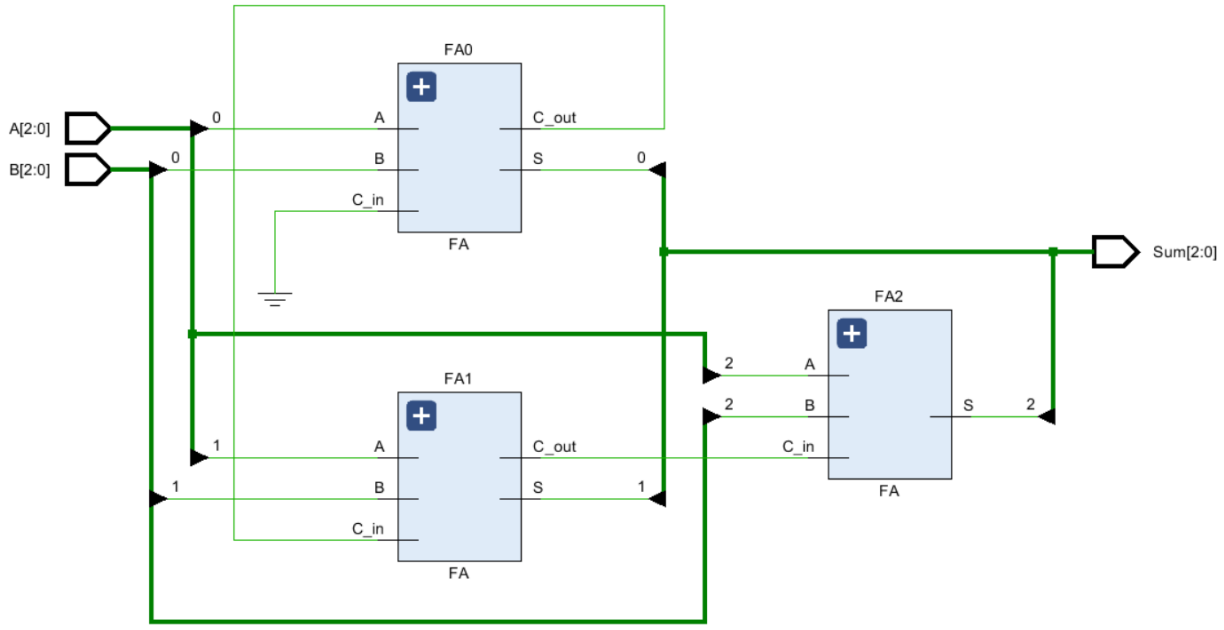
3 Bit Adder

Design Source File – 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 T_bit_Adder is
    Port (
        A       : in  STD_LOGIC_VECTOR(2 downto 0);
        B       : in  STD_LOGIC_VECTOR(2 downto 0);
        Sum     : out STD_LOGIC_VECTOR(2 downto 0)
    );
end T_bit_Adder;

architecture Behavioral of T_bit_Adder is

    component FA
        Port (
            A       : in  STD_LOGIC;
            B       : in  STD_LOGIC;
```

```
            C_in   : in  STD_LOGIC;
            C_out  : out STD_LOGIC;
            S      : out STD_LOGIC
        );
    end component;

    signal C0, C1,c2 : STD_LOGIC;

begin

    -- LSB full adder: no carry-in (set to '0')
    FA0: FA
        port map (
            A      => A(0),
            B      => B(0),
            C_in   => '0',
            S      => Sum(0),
            C_out  => C0
        );

    -- Middle bit
    FA1: FA
        port map (
            A      => A(1),
            B      => B(1),
            C_in   => C0,
            S      => Sum(1),
            C_out  => C1
        );

    -- MSB
    FA2: FA
        port map (
            A      => A(2),
            B      => B(2),
            C_in   => C1,
            S      => Sum(2),
            C_out  => c2
        );

end Behavioral;
```

Elaborated Design Schematic – 3 Bit Adder



Simulation Source File – 3 Bit Adder

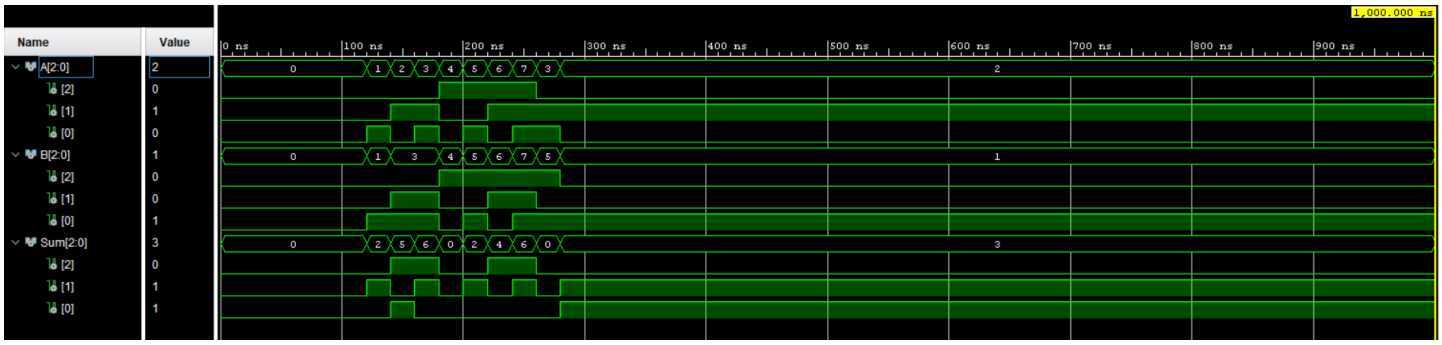
```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/14/2025
-- Design Name: Test Bench for Three-Bit Adder
-- Module Name: TB_T_bit_Adder - Simulation
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description: Test bench for verification of the Three-Bit
Adder
--
-- Dependencies: T_bit_Adder.vhd
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

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

entity TB_T_bit_Adder is
-- No ports for a test bench
end TB_T_bit_Adder;

architecture Simulation of TB_T_bit_Adder is
    -- Component declaration for the Unit Under Test (UUT)
    component T_bit_Adder
        Port (
            A      : in  STD_LOGIC_VECTOR(2 downto 0);
            B      : in  STD_LOGIC_VECTOR(2 downto 0);
            Sum    : out STD_LOGIC_VECTOR(2 downto 0);
            Carry  : out STD_LOGIC
        );
    end component;

    -- Inputs
    signal A      : STD_LOGIC_VECTOR(2 downto 0) := (others =>
'0');
    signal B      : STD_LOGIC_VECTOR(2 downto 0) := (others =>
'0');

    -- Outputs
    signal Sum    : STD_LOGIC_VECTOR(2 downto 0);
    signal Carry  : STD_LOGIC;

begin
    -- Instantiate the Unit Under Test (UUT)
    UUT: T_bit_Adder port map (
        A      => A,
        B      => B,
        Sum    => Sum,
        Carry  => Carry
    );

    -- Stimulus process
    stim_proc: process
    begin
        -- Hold reset state for 100 ns
```

```
wait for 100 ns;

-- Test various input combinations
-- Test case 1: A=0, B=0
A <= "000";
B <= "000";
wait for 20 ns;

-- Test case 2: A=1, B=1
A <= "001";
B <= "001";
wait for 20 ns;

-- Test case 3: A=2, B=3
A <= "010";
B <= "011";
wait for 20 ns;

-- Test case 4: A=3, B=3
A <= "011";
B <= "011";
wait for 20 ns;

-- Test case 5: A=4, B=4
A <= "100";
B <= "100";
wait for 20 ns;

-- Test case 6: A=5, B=5
A <= "101";
B <= "101";
wait for 20 ns;

-- Test case 7: A=6, B=6
A <= "110";
B <= "110";
wait for 20 ns;

-- Test case 8: A=7, B=7 (maximum values)
A <= "111";
B <= "111";
wait for 20 ns;

-- Test case 9: A=3, B=5
```

```
        A <= "011";
        B <= "101";
        wait for 20 ns;

        -- Test case 10: A=2, B=1
        A <= "010";
        B <= "001";
        wait for 20 ns;

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

Timing Diagram – 3 Bit Adder



Register Bank

Design Source File – Register Bank

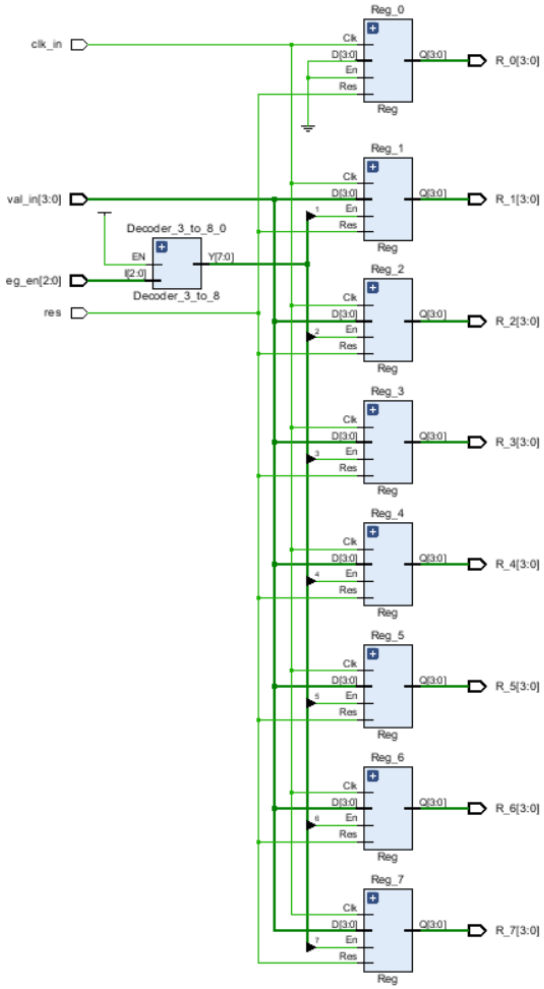
```
----------------------------------------------------------------------

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 Reg_Bank is
  Port (
  reg_en : IN STD_LOGIC_VECTOR(2 downto 0); --select the register
  val_in : IN STD_LOGIC_VECTOR(3 downto 0); -- data for registers
  clk_in : IN STD_LOGIC; --input clock signal
  res    : IN STD_LOGIC; --reset the all registers in bank

  --out put of data fropm registers
  R_0 : out STD_LOGIC_VECTOR (3 downto 0);
  R_1 : out STD_LOGIC_VECTOR (3 downto 0);
  R_2 : out STD_LOGIC_VECTOR (3 downto 0);
  R_3 : out STD_LOGIC_VECTOR (3 downto 0);
  R_4 : out STD_LOGIC_VECTOR (3 downto 0);
  R_5 : out STD_LOGIC_VECTOR (3 downto 0);
  R_6 : out STD_LOGIC_VECTOR (3 downto 0);
  R_7 : out STD_LOGIC_VECTOR (3 downto 0)
   );
end Reg_Bank;

architecture Behavioral of Reg_Bank is

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;

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

SIGNAL D : STD_LOGIC_VECTOR(7 downto 0);
SIGNAL E : STD_LOGIC := '1';

begin

Decoder_3_to_8_0 : Decoder_3_to_8
```

```
PORT MAP(
    I => reg_en,
    EN => E,
    Y => D
    );

Reg_0 : Reg
PORT MAP (
    EN => '0',
    D =>"0000" ,
    Res => res,
    Q => R_0,
    Clk => Clk_in
    );

Reg_1 : Reg
PORT MAP (
    EN => D(1),
    D =>val_in ,
    Res => res,
    Q => R_1,
    Clk => Clk_in
    );

Reg_2 : Reg
PORT MAP (
    EN => D(2),
    D =>val_in ,
    Res => res,
    Q => R_2,
    Clk => Clk_in
    );

Reg_3 : Reg
PORT MAP (
    EN => D(3),
    D =>val_in ,
    Res => res,
    Q => R_3,
    Clk => Clk_in
    );

Reg_4 : Reg
PORT MAP (
```

```
    EN => D(4),
    D =>val_in ,
    Res => res,
    Q => R_4,
    Clk => Clk_in
    );

Reg_5 : Reg
PORT MAP (
    EN => D(5),
    D =>val_in ,
    Res => res,
    Q => R_5,
    Clk => Clk_in
    );

Reg_6 : Reg
PORT MAP (
    EN => D(6),
    D =>val_in ,
    Res => res,
    Q => R_6,
    Clk => Clk_in
    );

Reg_7 : Reg
PORT MAP (
    EN => D(7),
    D =>val_in ,
    Res => res,
    Q => R_7,
    Clk => Clk_in
    );


end Behavioral;
```

Elaborated Design Schematic – Register Bank



Simulation Source File – Register Bank

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/14/2025 09:58:10 PM
-- Design Name:
-- Module Name: TB_Reg_Bank - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
```

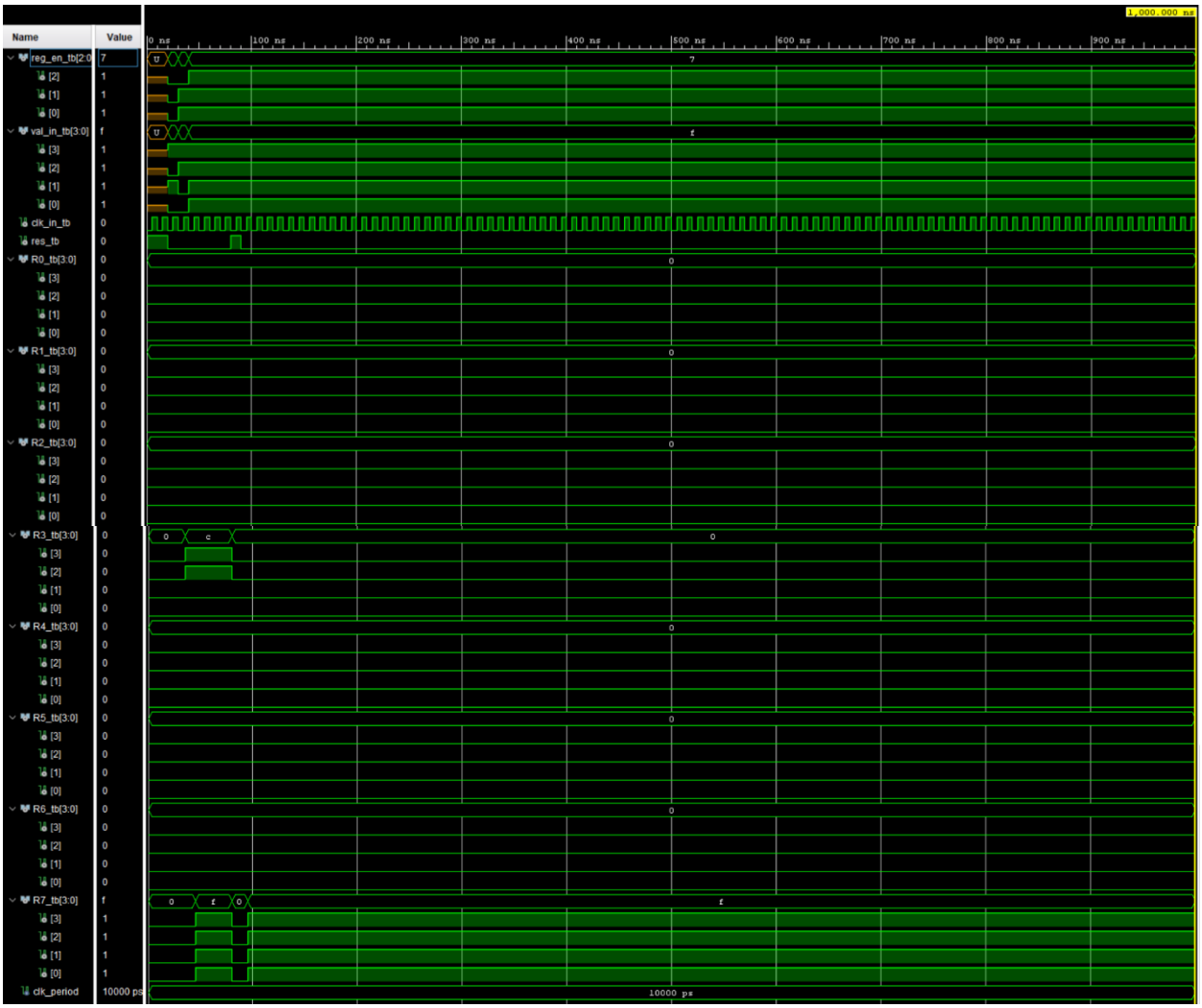
```
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-----------------


----------------------------------------------------------------------
-----------------
-- Company:
-- Engineer:
--
-- Create Date: 05/14/2025 04:59:54 PM
-- Design Name:
-- Module Name: Reg_Bank - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-----------------

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

entity Reg_Bank_tb is
-- Testbenches do not need ports
end Reg_Bank_tb;

architecture behavior of Reg_Bank_tb is

    -- Component Declaration
    component Reg_Bank
        Port (
```

```
            reg_en : IN STD_LOGIC_VECTOR(2 downto 0);
            val_in : IN STD_LOGIC_VECTOR(3 downto 0);
            clk_in : IN STD_LOGIC;
            res    : IN STD_LOGIC;

            R_0 : out STD_LOGIC_VECTOR (3 downto 0);
            R_1 : out STD_LOGIC_VECTOR (3 downto 0);
            R_2 : out STD_LOGIC_VECTOR (3 downto 0);
            R_3 : out STD_LOGIC_VECTOR (3 downto 0);
            R_4 : out STD_LOGIC_VECTOR (3 downto 0);
            R_5 : out STD_LOGIC_VECTOR (3 downto 0);
            R_6 : out STD_LOGIC_VECTOR (3 downto 0);
            R_7 : out STD_LOGIC_VECTOR (3 downto 0)
        );
    end component;

    -- Testbench signals
    signal reg_en_tb  : STD_LOGIC_VECTOR(2 downto 0);
    signal val_in_tb  : STD_LOGIC_VECTOR(3 downto 0);
    signal clk_in_tb  : STD_LOGIC := '0';
    signal res_tb     : STD_LOGIC := '1';

    signal R0_tb, R1_tb, R2_tb, R3_tb, R4_tb, R5_tb, R6_tb, R7_tb
: STD_LOGIC_VECTOR(3 downto 0);

    -- Clock generation process
    constant clk_period : time := 10 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Reg_Bank PORT MAP (
        reg_en => reg_en_tb,
        val_in => val_in_tb,
        clk_in => clk_in_tb,
        res    => res_tb,

        R_0 => R0_tb,
        R_1 => R1_tb,
        R_2 => R2_tb,
        R_3 => R3_tb,
        R_4 => R4_tb,
        R_5 => R5_tb,
        R_6 => R6_tb,
```

```
    R_7 => R7_tb
);

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

-- Stimulus process
stim_proc: process
begin
    -- Apply reset
    wait for 20 ns;
    res_tb <= '0';  -- Deassert reset

    -- Write "1010" to R0
    reg_en_tb <= "000";
    val_in_tb <= "1010";
    wait for clk_period;

    -- Write "1100" to R3
    reg_en_tb <= "011";
    val_in_tb <= "1100";
    wait for clk_period;

    -- Write "1111" to R7
    reg_en_tb <= "111";
    val_in_tb <= "1111";
    wait for clk_period;

    -- Wait and observe
    wait for 30 ns;

    -- Assert reset to clear all registers
    res_tb <= '1';
    wait for clk_period;
    res_tb <= '0';
```

```
        -- Final delay to observe cleared outputs
        wait for 30 ns;

        -- End simulation
        wait;
    end process;

end behavior;
```

Timing Diagram – Register Bank



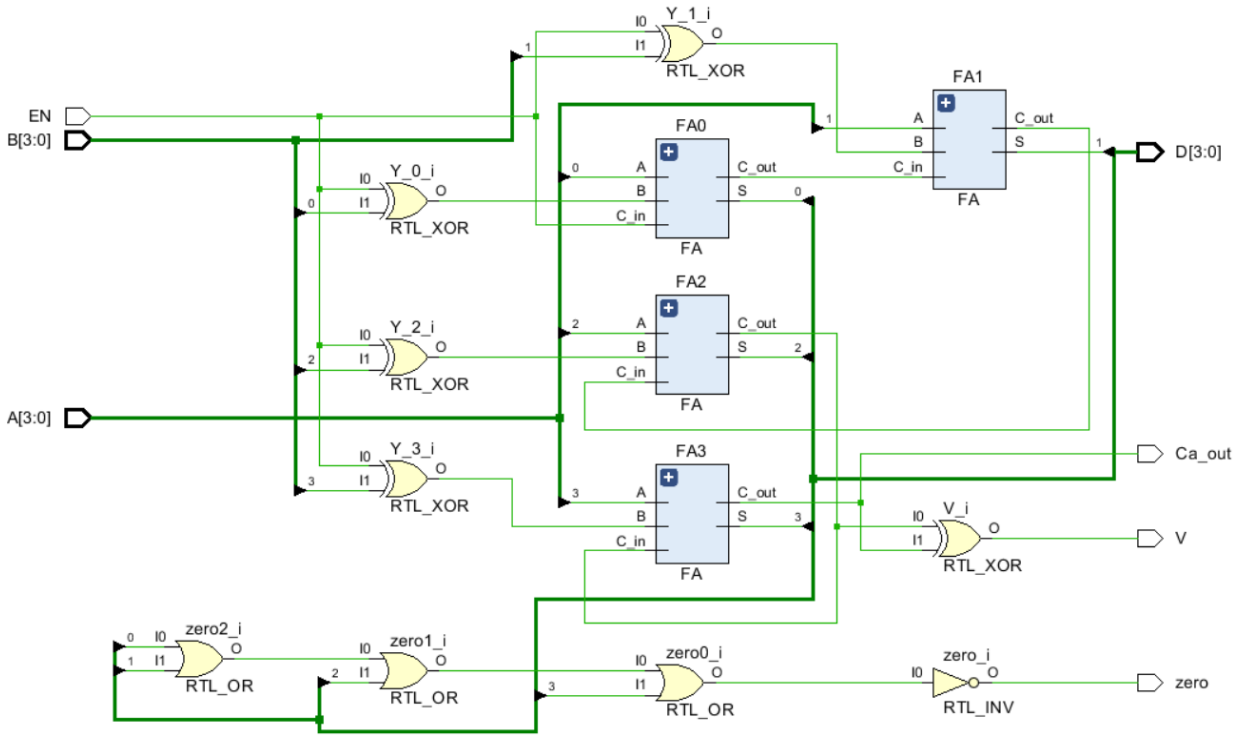
4 Bit Adder/Subtractor

Design Source File – 4 Bit 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 Adder_Substractor is
PORT(
A: IN STD_LOGIC_VECTOR(3 downto 0);
B: IN STD_LOGIC_VECTOR(3 downto 0);
EN:IN STD_LOGIC; --This enable system decide whether use adder or
substractor
V: OUT STD_LOGIC;--overflow bit
Ca_out:OUT STD_LOGIC;
zero : OUT STD_LOGIC;
D: OUT STD_LOGIC_VECTOR(3 downto 0));

end Adder_Substractor;

architecture Behavioral of Adder_Substractor is

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

SIGNAL S_OUT: STD_LOGIC_VECTOR(3 downto 0);
SIGNAL Y: STD_LOGIC_VECTOR(3 downto 0);
SIGNAL C0,C1,C2,C3: STD_LOGIC;

begin
FA0:FA
```

```
port map(
A=>A(0),
B=>Y(0),
C_in=>EN,
S=>S_OUT(0),
C_out=>C0);

FA1:FA
port map(
A=>A(1),
B=>Y(1),
C_in=>C0,
S=>S_OUT(1),
C_out=>C1);

FA2:FA
port map(
A=>A(2),
B=>Y(2),
C_in=>C1,
S=>S_OUT(2),
C_out=>C2);

FA3:FA
port map(
A=>A(3),
B=>Y(3),
C_in=>C2,
S=>S_OUT(3),
C_out=>C3);

Ca_out<=C3;

Y(0)<=EN XOR B(0);
Y(1)<=EN XOR B(1);
Y(2)<=EN XOR B(2);
Y(3)<=EN XOR B(3);
V<=C2 XOR C3;
D<=S_OUT;
zero<=NOT(S_out(0) or S_out(1) or S_out(2) or S_out(3));



end Behavioral;
```

Elaborated Design File – 4 Bit Adder/Subtractor



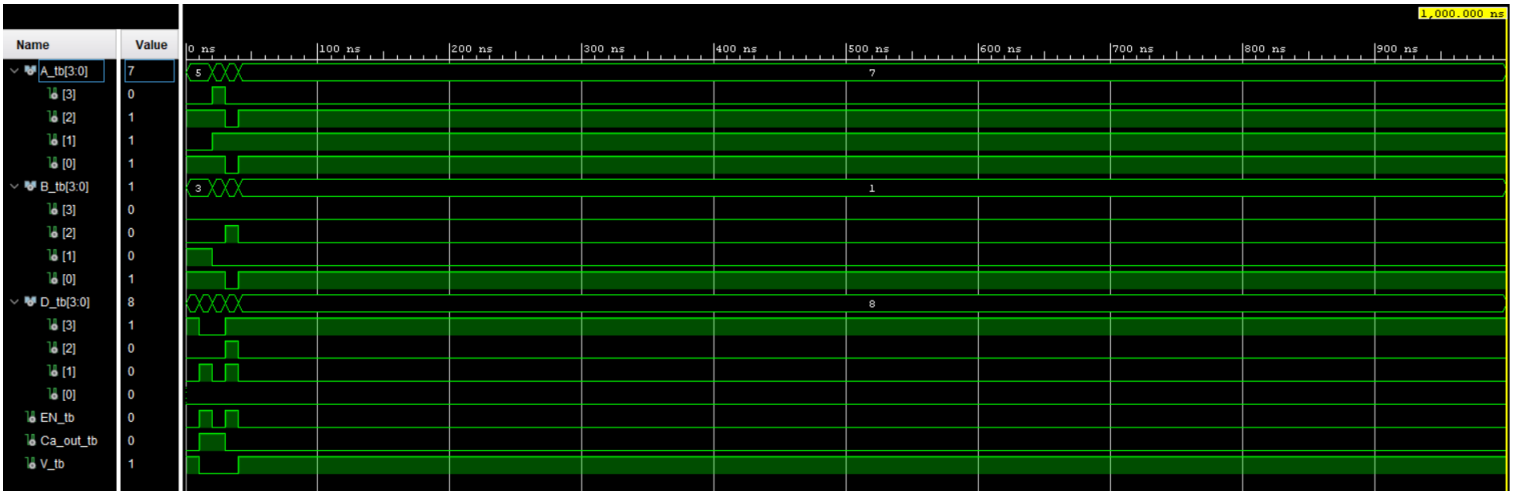
Simulation Source File – 4 Bit Adder/Subtractor

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

architecture Behavioral of Adder_Substractor_tb is
```

```
component Adder_Substractor
        PORT(
            A: IN STD_LOGIC_VECTOR(3 downto 0);
            B: IN STD_LOGIC_VECTOR(3 downto 0);
            EN:IN STD_LOGIC;
            Ca_out:OUT STD_LOGIC;
            V: OUT STD_LOGIC;
            zero : OUT STD_LOGIC;
            D: OUT STD_LOGIC_VECTOR(3 downto 0)
        );
end component;

signal A_tb, B_tb, D_tb : STD_LOGIC_VECTOR(3 downto 0);
signal EN_tb, Ca_out_tb,V_tb : STD_LOGIC;

begin
    uut: Adder_Substractor port map (
        A => A_tb,
        B => B_tb,
        EN => EN_tb,
        Ca_out => Ca_out_tb,
        D => D_tb,
        V=>V_tb
    );


    stim_proc: process
    begin
        -- Test 1: Addition  0101 + 0011 = 1000
        A_tb <= "0101";
        B_tb <= "0011";
        EN_tb <= '0'; -- Addition
        wait for 10 ns;

        -- Test 2: Subtraction 0101 - 0011 = 0010
        A_tb <= "0101";
        B_tb <= "0011";
        EN_tb <= '1'; -- Subtraction
        wait for 10 ns;

        -- Test 3: 1111 + 0001 = 0000 with carry out
        A_tb <= "1111";
        B_tb <= "0001";
        EN_tb <= '0'; -- Addition
```

```
        wait for 10 ns;

        -- Test 4: 0010 - 0100 = 1110 (underflow)
        A_tb <= "0010";
        B_tb <= "0100";
        EN_tb <= '1'; -- Subtraction
        wait for 10 ns;

        -- Test 5: 0111 - 0001 = 1111 (overflow)
        A_tb <= "0111";
        B_tb <= "0001";
        EN_tb <= '0'; -- Addition
        wait for 10 ns;


        -- Finish simulation
        wait;
    end process;

end Behavioral;
```

Timing Diagram – 4 Bit Adder/Subtractor



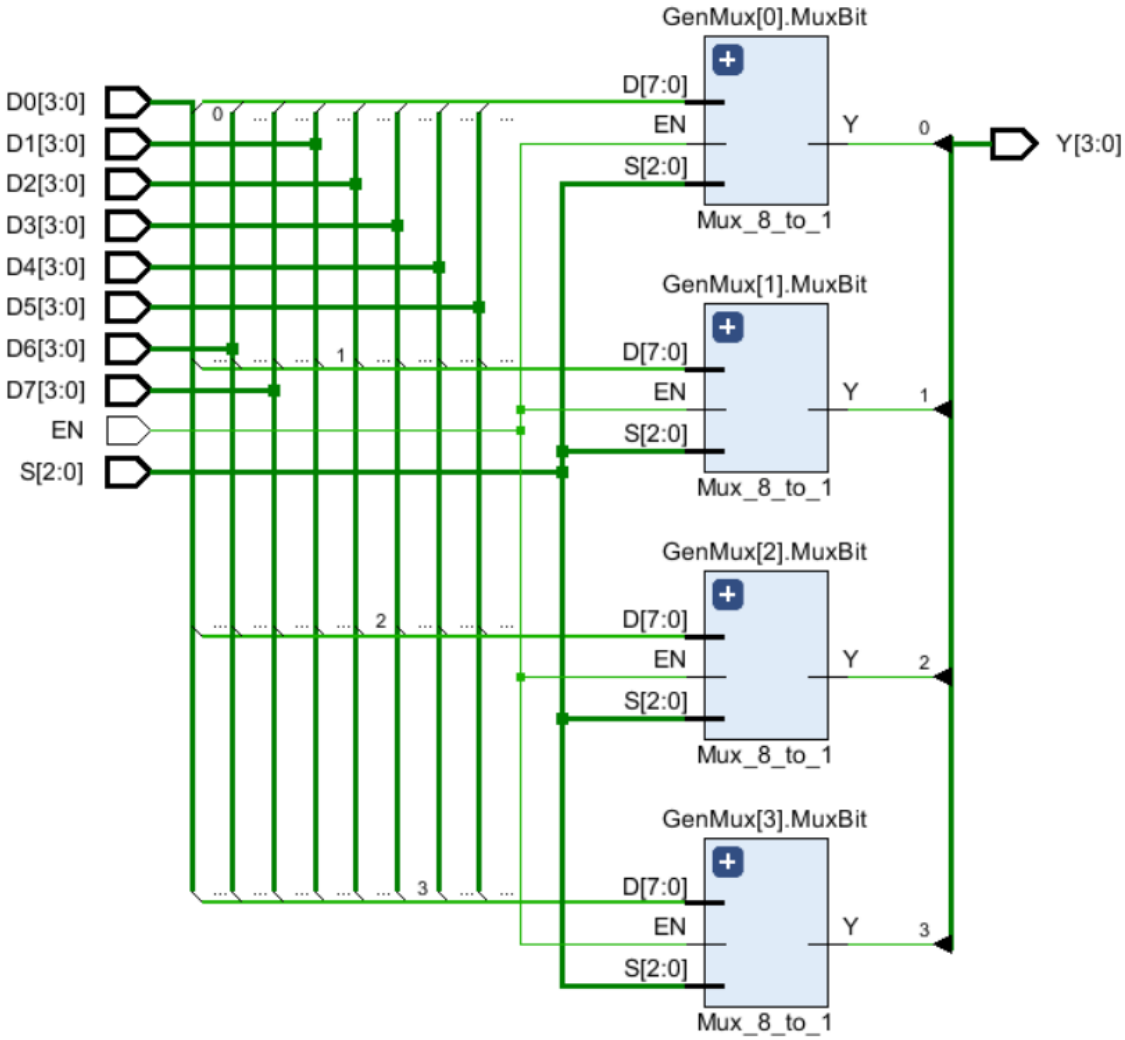
8-Way 4-Bit Multiplexer

Design Source File – 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_8way_4bit is
    Port ( D0 : in STD_LOGIC_VECTOR (3 downto 0);
           D1 : in STD_LOGIC_VECTOR (3 downto 0);
           D2 : in STD_LOGIC_VECTOR (3 downto 0);
           D3 : in STD_LOGIC_VECTOR (3 downto 0);
           D4 : in STD_LOGIC_VECTOR (3 downto 0);
           D5 : in STD_LOGIC_VECTOR (3 downto 0);
           D6 : in STD_LOGIC_VECTOR (3 downto 0);
           D7 : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8way_4bit;

architecture Structural of Mux_8way_4bit is
    component Mux_8_to_1
        Port (
            D : in STD_LOGIC_VECTOR (7 downto 0);
            EN : in STD_LOGIC;
            S : in STD_LOGIC_VECTOR (2 downto 0);
            Y : out STD_LOGIC
        );
    end component;

begin
    GenMux: for i in 0 to 3 generate
        MuxBit: Mux_8_to_1
            port map (
                D(7) => D7(i),
```

```
                D(6) => D6(i),
                D(5) => D5(i),
                D(4) => D4(i),
                D(3) => D3(i),
                D(2) => D2(i),
                D(1) => D1(i),
                D(0) => D0(i),
                EN => EN,
                S => S,
                Y => Y(i)
            );
    end generate GenMux;

end Structural;
```

Elaborated Design Schematic – 8-Way 4-Bit Multiplexer



Simulation Source File – 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 TB_Mux_8way_4bit is
--  Port ( );
end TB_Mux_8way_4bit;

architecture Behavioral of TB_Mux_8way_4bit is
component Mux_8way_4bit
    Port ( D0 : in STD_LOGIC_VECTOR (3 downto 0);
           D1 : in STD_LOGIC_VECTOR (3 downto 0);
           D2 : in STD_LOGIC_VECTOR (3 downto 0);
           D3 : in STD_LOGIC_VECTOR (3 downto 0);
           D4 : in STD_LOGIC_VECTOR (3 downto 0);
           D5 : in STD_LOGIC_VECTOR (3 downto 0);
           D6 : in STD_LOGIC_VECTOR (3 downto 0);
           D7 : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal D0 : STD_LOGIC_VECTOR (3 downto 0);
signal D1 : STD_LOGIC_VECTOR (3 downto 0);
signal D2 : STD_LOGIC_VECTOR (3 downto 0);
signal D3 : STD_LOGIC_VECTOR (3 downto 0);
signal D4 : STD_LOGIC_VECTOR (3 downto 0);
signal D5 : STD_LOGIC_VECTOR (3 downto 0);
signal D6 : STD_LOGIC_VECTOR (3 downto 0);
signal D7 : STD_LOGIC_VECTOR (3 downto 0);
signal EN : STD_LOGIC;
signal S : STD_LOGIC_VECTOR (2 downto 0);
signal Y : STD_LOGIC_VECTOR (3 downto 0);
```

```
begin
UUT : Mux_8way_4bit
    port map (
        D0 => D0,
        D1 => D1,
        D2 => D2,
        D3 => D3,
        D4 => D4,
        D5 => D5,
        D6 => D6,
        D7 => D7,
        EN => EN,
        S => S,
        Y => Y
    );

process
begin
    EN <= '1';
    D0 <= "0000";
    D1 <= "0001";
    D2 <= "0010";
    D3 <= "0011";
    D4 <= "0100";
    D5 <= "0101";
    D6 <= "0110";
    D7 <= "0111";

    S <= "000";
    wait for 20ns;

    S <= "001";
    wait for 20ns;

    S <= "010";
    wait for 20ns;

    S <= "011";
    wait for 20ns;

    S <= "100";
    wait for 20ns;

    S <= "101";
```

```
wait for 20ns;

S <= "110";
wait for 20ns;

EN <= '0';
D0 <= "0010";
D1 <= "0001";
D2 <= "0110";
D3 <= "1111";
D4 <= "0100";
D5 <= "0111";
D6 <= "0000";
D7 <= "0111";

S <= "000";
wait for 20ns;

S <= "001";
wait for 20ns;

S <= "010";
wait for 20ns;

S <= "011";
wait for 20ns;

S <= "100";
wait for 20ns;

S <= "101";
wait for 20ns;

S <= "110";
wait for 20ns;

EN <= '1';
D0 <= "0111";
D1 <= "0001";
D2 <= "0110";
D3 <= "1111";
D4 <= "0100";
D5 <= "0101";
D6 <= "0000";
```

```
D7 <= "0010";

S <= "000";
wait for 20ns;

S <= "001";
wait for 20ns;

S <= "010";
wait for 20ns;

S <= "011";
wait for 20ns;

S <= "100";
wait for 20ns;

S <= "101";
wait for 20ns;

S <= "110";
wait for 20ns;

EN <= '0';
D0 <= "0101";
D1 <= "0001";
D2 <= "0110";
D3 <= "1111";
D4 <= "0100";
D5 <= "0010";
D6 <= "0000";
D7 <= "1110";

S <= "000";
wait for 20ns;

S <= "001";
wait for 20ns;

S <= "010";
wait for 20ns;

S <= "011";
wait for 20ns;
```

```
    S <= "100";
    wait for 20ns;

    S <= "101";
    wait for 20ns;

    S <= "110";
    wait for 20ns;

    EN <= '1';
    D0 <= "1111";
    D1 <= "0001";
    D2 <= "0101";
    D3 <= "1110";
    D4 <= "0100";
    D5 <= "0010";
    D6 <= "0000";
    D7 <= "1110";

    S <= "000";
    wait for 20ns;

    S <= "001";
    wait for 20ns;

    S <= "010";
    wait for 20ns;

    S <= "011";
    wait for 20ns;

    S <= "100";
    wait for 20ns;

    S <= "101";
    wait for 20ns;

    S <= "110";
    wait for 20ns;
    wait;
    end process;
end Behavioral;
```

Timing Diagram – 8-Way 4-Bit Multiplexer

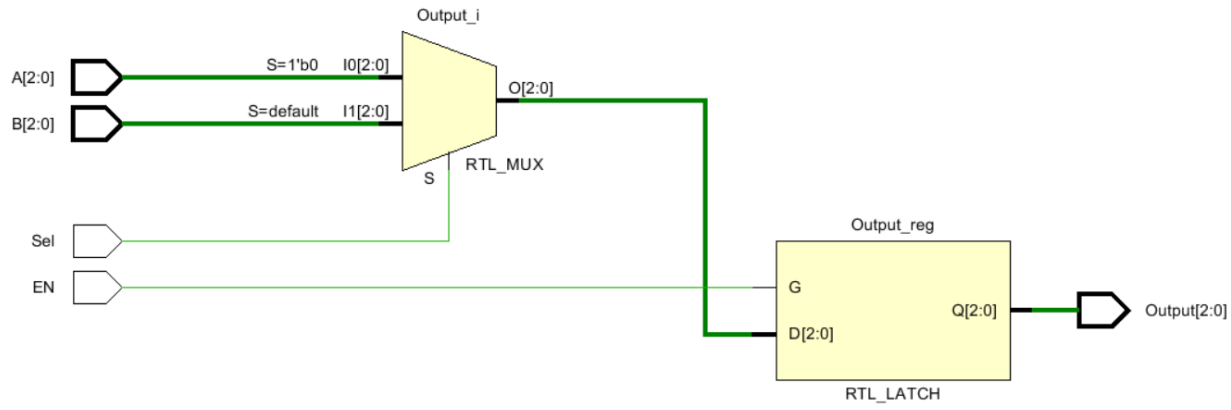
2-Way 3-Bit Multiplexer

Design Source File – 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 Multiplexer is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Sel : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
end Multiplexer;

architecture Behavioral of Multiplexer is

begin
    process (A, B, Sel)
        begin
            if EN = '1' then
                if Sel = '0' then
                    Output <= A;
                else
                    Output <= B;
                end if;
            end if;
    end process;
end Behavioral;
```

Elaborated Design Schematic – 2-Way 3-Bit Multiplexer

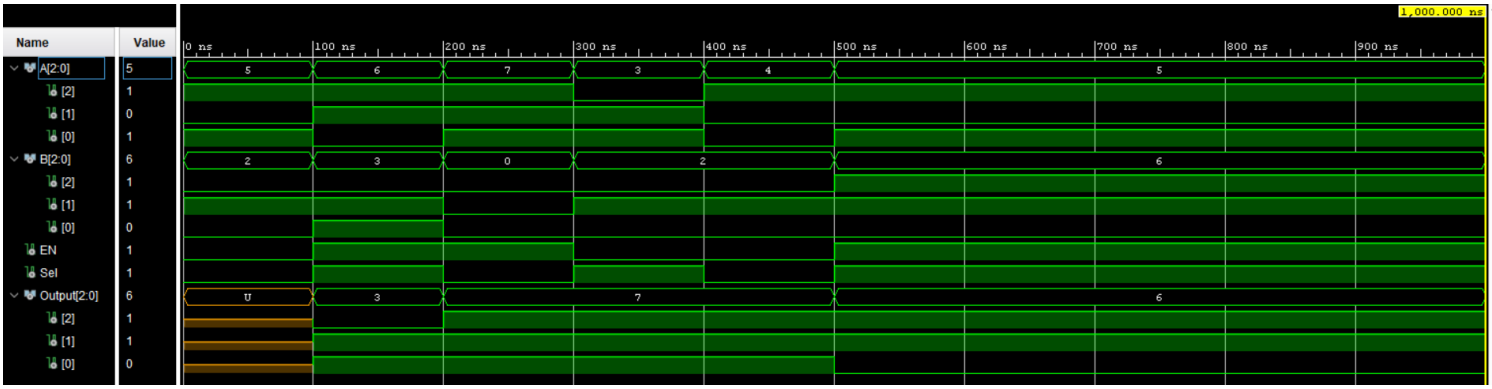


Simulation Source File – 2-Way 3-Bit Multiplexer

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

architecture Behavioral of Multiplexer_TB is
    component Multiplexer
        Port (
            A   : in  STD_LOGIC_VECTOR(2 downto 0);
            B   : in  STD_LOGIC_VECTOR(2 downto 0);
            EN  : in  STD_LOGIC;
            Sel : in  STD_LOGIC;
            Output   : out STD_LOGIC_VECTOR(2 downto 0)
        );
    end component;

    signal A   : STD_LOGIC_VECTOR(2 downto 0) := "000";
    signal B   : STD_LOGIC_VECTOR(2 downto 0) := "000";
    signal EN  : STD_LOGIC := '0';
    signal Sel : STD_LOGIC := '0';
    signal Output   : STD_LOGIC_VECTOR(2 downto 0);

begin
    UUT: Multiplexer port map (
        A => A,
        B => B,
        EN => EN,
        Sel => Sel,
        Output => Output
    );
    process
        begin
```

```
            A <= "101";
            B <= "010";
            EN <= '0';
            Sel <= '0';
            wait for 100 ns;

            A <= "110";
            B <= "011";
            EN <= '1';
            Sel <= '1';
            wait for 100 ns;

            A <= "111";
            B <= "000";
            EN <= '1';
            Sel <= '0';
            wait for 100 ns;

            A <= "011";
            B <= "010";
            EN <= '0';
            Sel <= '1';
            wait for 100 ns;

            A <= "100";
            B <= "010";
            EN <= '0';
            Sel <= '0';
            wait for 100 ns;

            A <= "101";
            B <= "110";
            EN <= '1';
            Sel <= '1';
            wait for 100 ns;

        wait;
    end process;

end Behavioral;
```

Timing Diagram – 2-Way 3-Bit Multiplexer



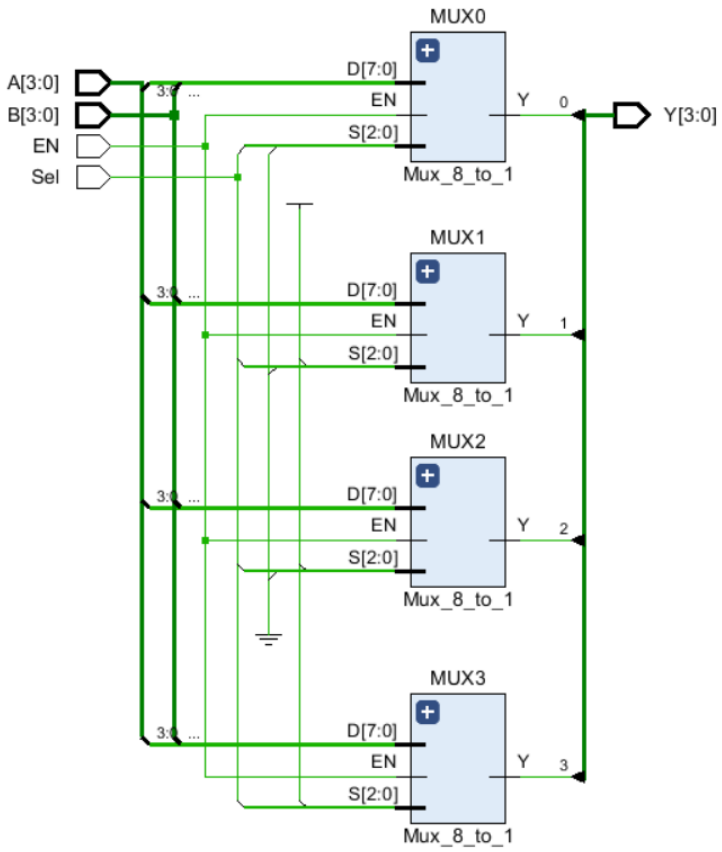
2-Way 4-Bit Multiplexer

Design Source File – 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_bits is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Sel : in STD_LOGIC;
           EN : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end MUX_2_way_4_bits;

architecture Behavioral of MUX_2_way_4_bits is
component Mux_8_to_1
port(
    D : in STD_LOGIC_VECTOR (7 downto 0);
    EN : in STD_LOGIC;
    S : in STD_LOGIC_VECTOR (2 downto 0);
    Y : out STD_LOGIC
```

```
);
end component;

signal D : std_logic_vector(7 downto 0);
signal S0,S1,S2,S3 :std_logic_vector(2 downto 0);

begin
--Mapping A and B into D[0] to D[7]
   D(0) <= A(0);
   D(1) <= A(1);
   D(2) <= A(2);
   D(3) <= A(3);
   D(4) <= B(0);
   D(5) <= B(1);
   D(6) <= B(2);
   D(7) <= B(3);

   S0 <= Sel & "00"; -- selects A(0) or B(0)
   S1 <= Sel & "01"; -- selects A(1) or B(1)
   S2 <= Sel & "10"; -- selects A(2) or B(2)
   S3 <= Sel & "11"; -- selects A(3) or B(3)

   MUX0: Mux_8_to_1
   port map(
       D => D,
       EN => EN,
       S => S0,--Declare S0 and map it into S(3 bit) selector
       Y => Y(0)--Mapping MUX0's output Y into 2 way 4 bit mux's
Y(0) output
   );
   MUX1: Mux_8_to_1 port map(D => D, EN => EN, S => S1, Y =>
Y(1));
   MUX2: Mux_8_to_1 port map(D => D, EN => EN, S => S2, Y =>
Y(2));
   MUX3: Mux_8_to_1 port map(D => D, EN => EN, S => S3, Y =>
Y(3));

end Behavioral;
```

Elaborated Design Schematic – 2-Way 4-Bit Multiplexer



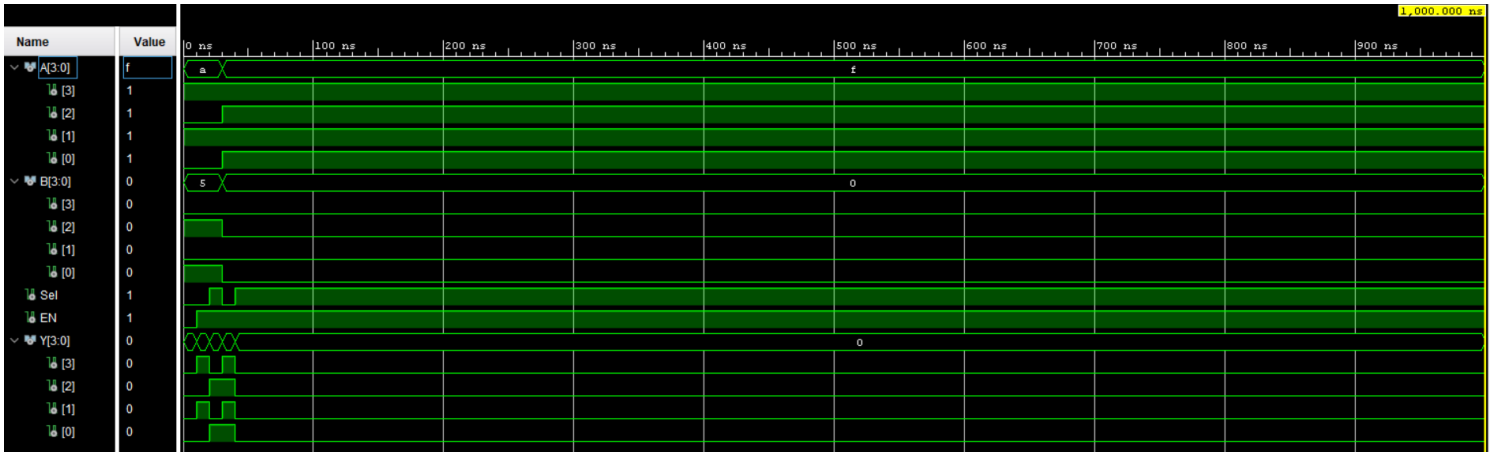
Simulation Source File – 2-Way 4-Bit Multiplexer

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

entity tb_MUX_2_way_4_bits is
-- Testbench has no ports
end tb_MUX_2_way_4_bits;

architecture behavior of tb_MUX_2_way_4_bits is

    -- Component Declaration for the Unit Under Test (UUT)
    component MUX_2_way_4_bits
        Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
               B : in STD_LOGIC_VECTOR (3 downto 0);
               Sel : in STD_LOGIC;
               EN : in STD_LOGIC;
               Y : out STD_LOGIC_VECTOR (3 downto 0));
    end component;
```

```
    -- Signals for connecting to UUT
    signal A     : STD_LOGIC_VECTOR (3 downto 0) := (others =>
'0');
    signal B     : STD_LOGIC_VECTOR (3 downto 0) := (others =>
'0');
    signal Sel   : STD_LOGIC := '0';
    signal EN    : STD_LOGIC := '0';
    signal Y     : STD_LOGIC_VECTOR (3 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: MUX_2_way_4_bits Port Map (
          A => A,
          B => B,
          Sel => Sel,
          EN => EN,
          Y => Y
        );

    -- Stimulus Process
    stim_proc: process
    begin
        -- Test case 1: Enable LOW, should output zeros
        A <= "1010";
        B <= "0101";
        Sel <= '0';
        EN <= '0';
        wait for 10 ns;

        -- Test case 2: Enable HIGH, Sel = 0 => output A
        EN <= '1';
        Sel <= '0';
        wait for 10 ns;

        -- Test case 3: Enable HIGH, Sel = 1 => output B
        Sel <= '1';
        wait for 10 ns;

        -- Test case 4: Change A and B again
        A <= "1111";
        B <= "0000";
        Sel <= '0';
```

```
        wait for 10 ns;

        Sel <= '1';
        wait for 10 ns;

        -- End of test
        wait;
    end process;

end behavior;
```

Timing Diagram – 2-Way 4-Bit Multiplexer



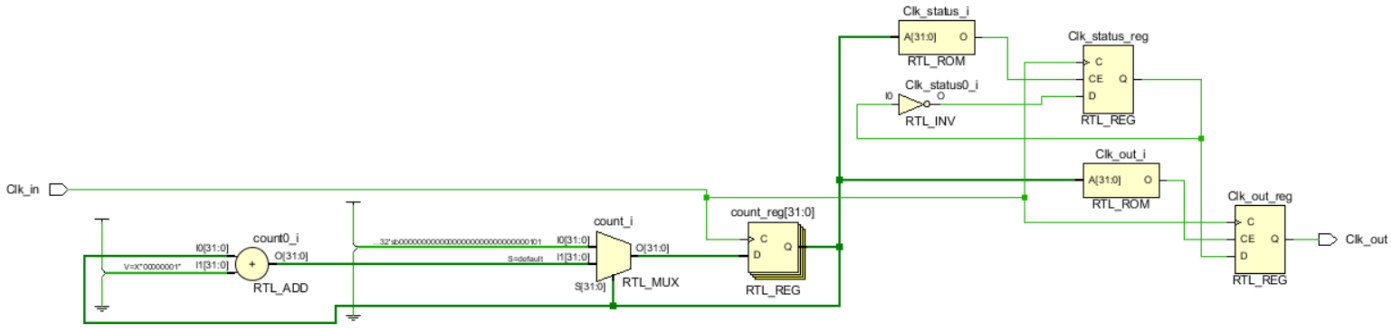
Slow Clock

Design Source File – 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 :='1';
```

```
begin
process (Clk_in) begin
    if (rising_edge(Clk_in)) then
        count <= count +1 ;
        if (count = 5) then --count 50000000 for basys board this
is for simulation only
            Clk_status <= NOT (Clk_status);
            Clk_out <= Clk_status ;
            count <= 1;
        end if;
    end if;

end process;

end Behavioral;
```

Elaborated Design Schematic – Slow Clock

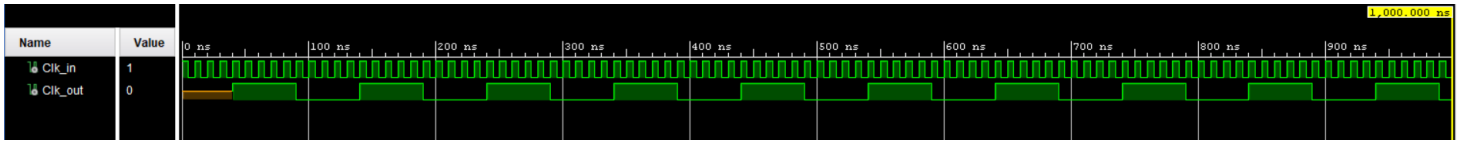


Simulation Source File – Slow Clock

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

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

architecture Behavioral of Slow_Clk_Sim is

COMPONENT Slow_Clk
    PORT (Clk_in :IN STD_LOGIC := '0';
    Clk_out :OUT STD_LOGIC);
end COMPONENT;
```

```
signal Clk_in : STD_LOGIC := '0';
signal Clk_out : STD_LOGIC;

begin
clock_process: process
    begin
        Clk_in <= NOT(Clk_in);
        wait for 5ns;
    end process;

UUT: Slow_Clk PORT MAP(
Clk_in => Clk_in,
Clk_out => Clk_out );

end Behavioral;
```

Timing Diagram – Slow Clock



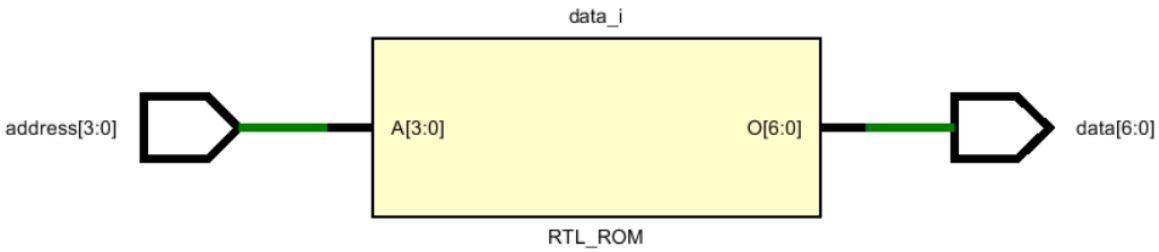
7 Segment Display

Design Source File – 7 Segment Display

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

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

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

entity Seven_Segment is
    Port (
        address : in STD_LOGIC_VECTOR (3 downto 0);
```

```
        data : out STD_LOGIC_VECTOR (6 downto 0)
    );
end Seven_Segment;

architecture Behavioral of Seven_Segment is


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

    -- ROM initialization
    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 data based on the address
    data <= sevenSegment_ROM(to_integer(unsigned(address)));

end Behavioral;
```

Elaborated Design Schematic – 7 Segment Display

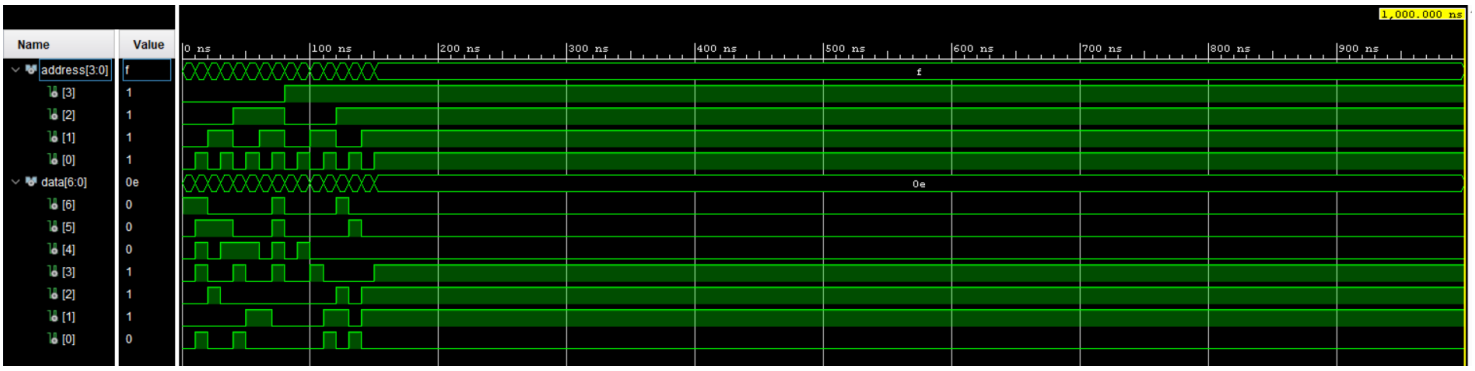


Simulation Source File – 7 Segment Display

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

entity Seven_Segment_tb is
end Seven_Segment_tb;

architecture behavior of Seven_Segment_tb is

    -- Component Declaration for the Unit Under Test (UUT)
    component Seven_Segment
        Port (
            address : in STD_LOGIC_VECTOR (3 downto 0);
            data : out STD_LOGIC_VECTOR (6 downto 0)
        );
    end component;

    -- Signals to connect to UUT
    signal address : STD_LOGIC_VECTOR(3 downto 0) := (others =>
'0');
    signal data : STD_LOGIC_VECTOR(6 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Seven_Segment
        Port map (
            address => address,
            data => data
        );

    -- Stimulus Process
```

```
    stim_proc: process
    begin
        -- Iterate through all 16 input values from 0 to F
        for i in 0 to 15 loop
            address <= std_logic_vector(to_unsigned(i, 4));
            wait for 10 ns;
        end loop;

        wait;
    end process;

end behavior;
```

Timing Diagram – 7 Segment Display



Constraint File

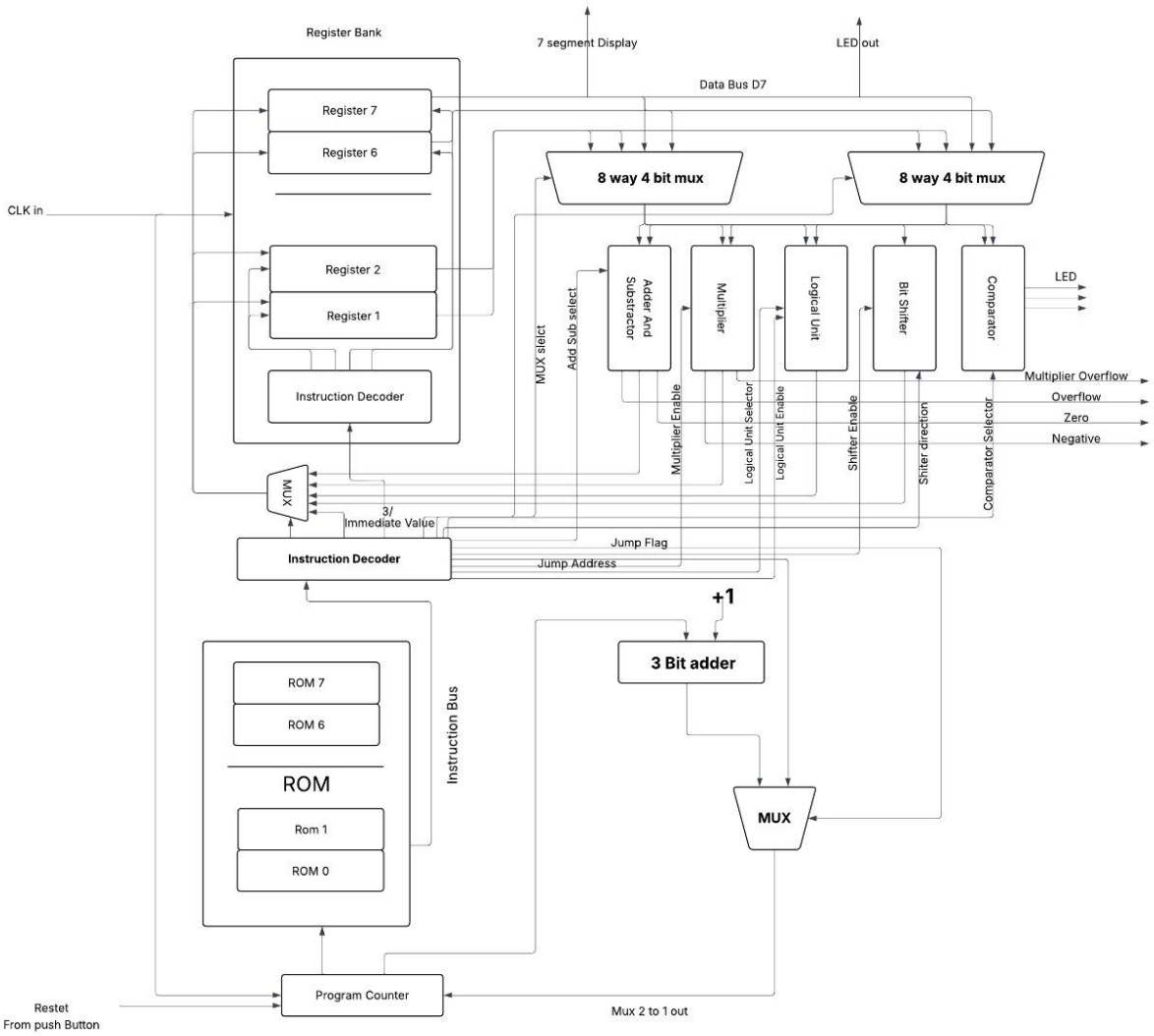
```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports Clk_in]
     set_property IOSTANDARD LVCMOS33 [get_ports Clk_in]
     create_clock -add -name sys_clk_pin -period 10.00 -waveform
{0 5} [get_ports {Clk_in}]




## LEDs

##Register 7 value
set_property PACKAGE_PIN U16 [get_ports {LED[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
```

```
set_property PACKAGE_PIN E19 [get_ports {LED[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {LED[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {LED[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]


##Adder Subtractor
set_property PACKAGE_PIN P1 [get_ports {Zero}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]
set_property PACKAGE_PIN L1 [get_ports {Overflow}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Overflow}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {Display[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[0]}]
set_property PACKAGE_PIN W6 [get_ports {Display[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[1]}]
set_property PACKAGE_PIN U8 [get_ports {Display[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[2]}]
set_property PACKAGE_PIN V8 [get_ports {Display[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[3]}]
set_property PACKAGE_PIN U5 [get_ports {Display[4]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[4]}]
set_property PACKAGE_PIN V5 [get_ports {Display[5]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[5]}]
set_property PACKAGE_PIN U7 [get_ports {Display[6]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[6]}]


set_property PACKAGE_PIN U2 [get_ports {Anode[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {Anode[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {Anode[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[3]}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports Reset]
     set_property IOSTANDARD LVCMOS33 [get_ports Reset]
```

Conclusion

- We successfully designed and implemented a 4-bit nanoprocessor capable of executing four basic instructions: MOVI, ADD, NEG, and JZR.
- The project involved building key components including:
    - 4-bit arithmetic unit for addition and subtraction
    - 3-bit adder and program counter with reset functionality
    - 3-to-8 decoder, register bank, and instruction decoder
    - Various multiplexers and tri-state buffers
- We tested and verified each module through simulation before integrating them into the complete processor design.
- The processor was implemented on the BASYS 3 board, with outputs visualized via LEDs and 7-segment display.
- This project enhanced our practical knowledge of processor architecture, digital design, and VHDL implementation.
- Working collaboratively as a team, we efficiently divided tasks and successfully integrated each component to create a fully functional nanoprocessor.
- The lab provided a solid foundation in processor design and deepened our understanding of how individual components work together to execute instructions.

NANO PROCESSOR (IMPROVED VERSION)



Design Source File - Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/15/2025 12:37:57 AM
-- Design Name:
-- Module Name: Nano_Processor - Behavioral
-- Project Name:
-- Target Devices:
```

```
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-----------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

-- 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 (
  Reset : IN STD_LOGIC;
  Clk_in : IN STD_LOGIC;
  Zero : OUT STD_LOGIC;
  Overflow : OUT STD_LOGIC;
  LED : OUT STD_LOGIC_VECTOR(3 downto 0);
  Display : OUT STD_LOGIC_VECTOR(6 downto 0); --7 segment display
  Anode : OUT STD_LOGIC_VECTOR(3 downto 0);
  A_greater_than_B : out STD_LOGIC;
  A_equal_B : out STD_LOGIC;
  A_less_than_B : out STD_LOGIC;
  Overflow_Mul : out STD_LOGIC
   );
end Nano_Processor;

architecture Behavioral of Nano_Processor is

COMPONENT Program_ROM
```

```
    Port (
        Rom_IN  : IN  STD_LOGIC_VECTOR(2 downto 0);
        Rom_OUT : OUT STD_LOGIC_VECTOR(12 downto 0)
    );
END COMPONENT;

COMPONENT Ins_Decoder
  Port (
  Ins :          IN STD_LOGIC_VECTOR(12 downto 0); --Instructions
  Jump :         IN STD_LOGIC_VECTOR( 3 downto 0);
  Register_EN : OUT STD_LOGIC_VECTOR(2 downto 0); --Register
Enable
  Load_sel :    OUT STD_LOGIC_VECTOR(2 downto 0);
-- Select the load
  I_value :     OUT STD_LOGIC_VECTOR(3 downto 0); --Immediate
value
  Reg_A :       OUT STD_LOGIC_VECTOR(2 downto 0);
  Reg_B :       OUT STD_LOGIC_VECTOR(2 downto 0);
  Add_sub :     OUT STD_LOGIC;
  Jump_flag :   OUT STD_LOGIC;
  Address_J :   OUT STD_LOGIC_VECTOR(2 downto 0);
  Cmp_EN :      OUT STD_LOGIC;
  Mul_EN :      OUT STD_LOGIC;
  Sft_EN :      OUT STD_LOGIC;
  Sft_Dir :     OUT STD_LOGIC;
  LU_EN :       OUT STD_LOGIC;
  LU_Op_Select :OUT STD_LOGIC_VECTOR(1 downto 0)
    );
END COMPONENT;

COMPONENT p_counter
    Port ( Mux_Output : in STD_LOGIC_VECTOR (2 downto 0);
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
END COMPONENT;

COMPONENT T_bit_Adder --three bit adder
    Port (
        A      : in  STD_LOGIC_VECTOR(2 downto 0);
        B      : in  STD_LOGIC_VECTOR(2 downto 0);
        Sum    : out STD_LOGIC_VECTOR(2 downto 0)
    );
END COMPONENT;
```

```
COMPONENT Reg_Bank
  Port (
  reg_en : IN STD_LOGIC_VECTOR(2 downto 0); --select the register
  val_in : IN STD_LOGIC_VECTOR(3 downto 0); -- data for registers
  clk_in : IN STD_LOGIC; --input clock signal

  --out put of data fropm registers
  R_0 : out STD_LOGIC_VECTOR (3 downto 0);
  R_1 : out STD_LOGIC_VECTOR (3 downto 0);
  R_2 : out STD_LOGIC_VECTOR (3 downto 0);
  R_3 : out STD_LOGIC_VECTOR (3 downto 0);
  R_4 : out STD_LOGIC_VECTOR (3 downto 0);
  R_5 : out STD_LOGIC_VECTOR (3 downto 0);
  R_6 : out STD_LOGIC_VECTOR (3 downto 0);
  R_7 : out STD_LOGIC_VECTOR (3 downto 0)
   );
END COMPONENT;

COMPONENT Adder_Substractor
    PORT(
    A: IN STD_LOGIC_VECTOR(3 downto 0);
    B: IN STD_LOGIC_VECTOR(3 downto 0);
    EN:IN STD_LOGIC; --This enable system decide whether use
adder or substractor
    V: OUT STD_LOGIC;--overflow bit
    Ca_out:OUT STD_LOGIC;
    zero : OUT STD_LOGIC;
    D: OUT STD_LOGIC_VECTOR(3 downto 0));
END COMPONENT;

COMPONENT MUX_8way_4bit
    Port ( D0 : in STD_LOGIC_VECTOR (3 downto 0);
           D1 : in STD_LOGIC_VECTOR (3 downto 0);
           D2 : in STD_LOGIC_VECTOR (3 downto 0);
           D3 : in STD_LOGIC_VECTOR (3 downto 0);
           D4 : in STD_LOGIC_VECTOR (3 downto 0);
           D5 : in STD_LOGIC_VECTOR (3 downto 0);
           D6 : in STD_LOGIC_VECTOR (3 downto 0);
           D7 : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;
```

```
COMPONENT Mux_5W_4B
    Port ( Immediate_Value : in STD_LOGIC_VECTOR (3 downto 0);
           R : in STD_LOGIC_VECTOR (3 downto 0);
           M_1 : in STD_LOGIC_VECTOR (3 downto 0);
           M_2 : in STD_LOGIC_VECTOR (3 downto 0);
           M_3 : in STD_LOGIC_VECTOR (3 downto 0);
           Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Mux_Out : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;

COMPONENT Multiplexer
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Sel : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
END COMPONENT;

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

COMPONENT Seven_Segment
    PORT( address : in STD_LOGIC_VECTOR (3 downto 0);
    data : out STD_LOGIC_VECTOR (6 downto 0));
end COMPONENT;

COMPONENT comparator
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           A_equal_B : out STD_LOGIC;
           A_greater_B : out STD_LOGIC;
           A_lesser_B : out STD_LOGIC;
           Cmp_EN : in STD_LOGIC);
end COMPONENT;

COMPONENT Multiplier_4
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Mul_EN : in STD_LOGIC );
```

```
end COMPONENT;

COMPONENT shifter_4B
    Port ( Data_In : in STD_LOGIC_VECTOR (3 downto 0);
           Shift_Dir : in STD_LOGIC;
           Data_Out : out STD_LOGIC_VECTOR (3 downto 0);
           Sft_EN : in STD_LOGIC);
end COMPONENT;

COMPONENT Logical_Unit_4B
    Port ( A         : in  STD_LOGIC_VECTOR (3 downto 0);
           B         : in  STD_LOGIC_VECTOR (3 downto 0);
           LU_EN     : in  STD_LOGIC;
           LU_Op_Select : in  STD_LOGIC_VECTOR (1 downto 0);
           Out_Result : out STD_LOGIC_VECTOR (3 downto 0));
end COMPONENT;

SIGNAL Overflw, Z, slow_clock, carry : STD_LOGIC; --overflw for
Overflow Z for zero flag
SIGNAL Counter : STD_LOGIC_VECTOR( 2 downto 0); -- output of
program counter
SIGNAL Instructions : STD_LOGIC_VECTOR(12 downto 0); --Intruction
bus
SIGNAL Adder: STD_LOGIC_VECTOR(2 downto 0); --3 bit adder output
SIGNAL Multiplexer_out: STD_LOGIC_VECTOR(2 downto 0);  --2 way 3
bit multiplexer out
SIGNAL Add_or_Sub, Jmp_Flag : STD_LOGIC;
SIGNAL Load_Sel : STD_LOGIC_VECTOR(2 downto 0);
SIGNAL Immediate_Value: STD_LOGIC_VECTOR (3 downto 0);
SIGNAL D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7, M_A, M_B,  M_0, R :
STD_LOGIC_VECTOR (3 downto 0);
SIGNAL Reg_Sel_MuxA, Reg_Enable, Reg_Sel_MuxB, Address_JMP :
STD_LOGIC_VECTOR (2 downto 0);
SIGNAL Cmp_EN, Mul_EN, Sft_EN, LU_EN : STD_LOGIC;
SIGNAL M_3, M_2, M_1 : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL Sft_Dir : STD_LOGIC;
SIGNAL LU_Op_Select : STD_LOGIC_VECTOR (1 downto 0);


begin
Slow_Clock_0 : Slow_Clk
    PORT MAP ( Clk_in => Clk_in,
               Clk_out => slow_clock );
```

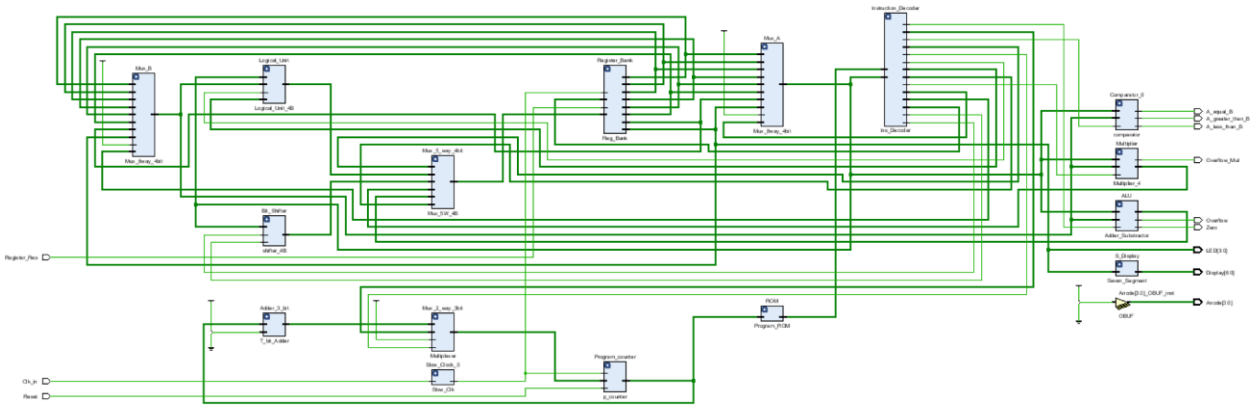
```
Program_counter: p_counter
    PORT MAP ( Mux_Output =>Multiplexer_out,
               Res =>Reset,
               Clk =>slow_clock,
               Q => counter);

ALU : Adder_Substractor
    PORT MAP(A =>M_A,
             B =>M_B,
             EN=>Add_or_Sub,
             V =>overflw,
             Ca_out=>carry,
             zero =>Z,
             D =>R);

Adder_3_bit:T_bit_Adder
    PORT MAP (
               A=>Counter ,
               B =>"001",
               Sum=>Adder
               );

Register_Bank: Reg_Bank
    PORT MAP(reg_en => Reg_Enable,
             val_in => M_0,
             clk_in => slow_clock,
  --out put of data fropm registers
             R_0 =>D_0,
             R_1 =>D_1,
             R_2 =>D_2,
             R_3 =>D_3,
             R_4 =>D_4,
             R_5 =>D_5,
             R_6 =>D_6,
             R_7 =>D_7);

Mux_5_way_4bit : Mux_5W_4B
    PORT MAP ( Immediate_Value => Immediate_Value,
               R => R,
               M_1 => M_1,
               M_2 => M_2,
               M_3 => M_3,
               Sel => Load_Sel,
               Mux_Out => M_0);
```

```
Mux_2_way_3bit : Multiplexer
    PORT MAP(A=>Adder,
             B=>Address_JMP,
             EN=>'1',
             Sel=>Jmp_Flag,
             Output=>Multiplexer_out);

Mux_A : MUX_8way_4bit
    PORT MAP (D0=>D_0,
              D1=>D_1,
              D2=>D_2,
              D3=>D_3,
              D4=>D_4,
              D5=>D_5,
              D6=>D_6,
              D7=>D_7,
              EN=>'1',
              S =>Reg_Sel_MuxA,
              Y =>M_A);

Mux_B : MUX_8way_4bit
    PORT MAP (D0=>D_0,
              D1=>D_1,
              D2=>D_2,
              D3=>D_3,
              D4=>D_4,
              D5=>D_5,
              D6=>D_6,
              D7=>D_7,
              EN=>'1',
              S =>Reg_Sel_MuxB,
              Y =>M_B);

 ROM : Program_ROM
    PORT MAP (Rom_IN=>Counter ,
              Rom_OUT=>Instructions );

 S_Display : Seven_Segment
    PORT MAP(address=>D_7,
             data=>Display );

 Instruction_Decoder: Ins_Decoder
    PORT MAP(
```

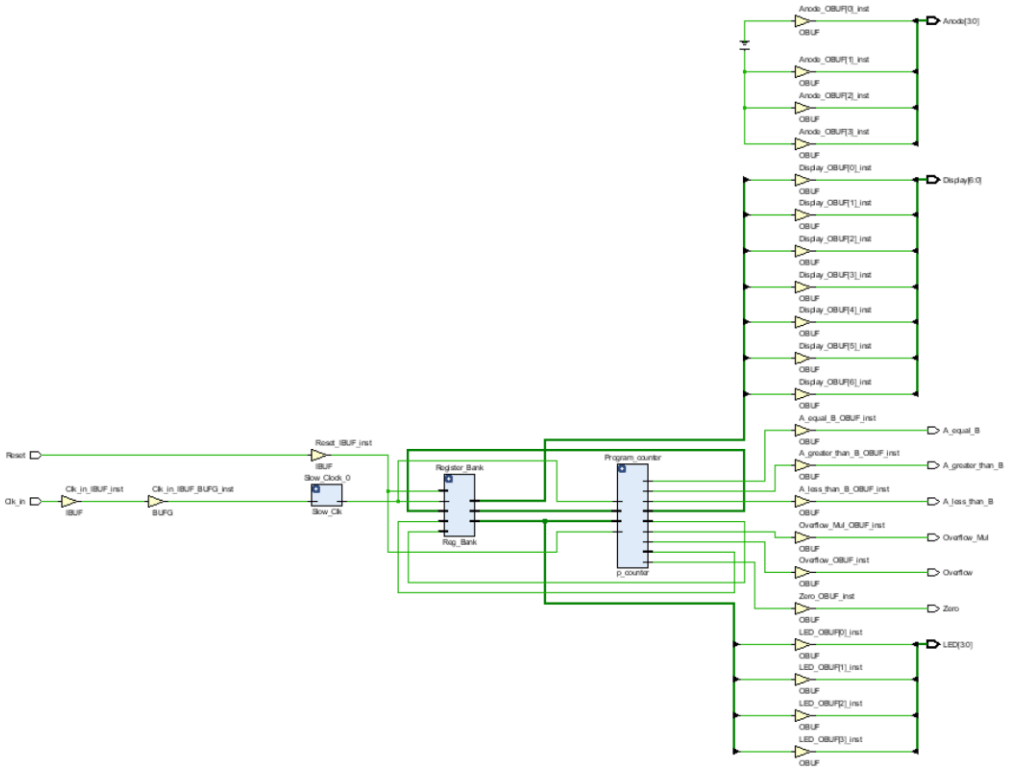
```
      Ins =>Instructions,
      Jump =>M_A,
      Register_EN=>Reg_Enable,
      Load_sel=> Load_Sel,
      I_value=>Immediate_Value,
      Reg_A =>Reg_Sel_MuxA,
      Reg_B =>Reg_Sel_MuxB,
      Add_sub=>Add_or_Sub,
      Jump_flag=>Jmp_Flag,
      Address_J=>Address_JMP ,
      Cmp_EN => Cmp_EN,
      Mul_EN => Mul_EN,
      Sft_EN => Sft_EN,
      Sft_Dir => Sft_Dir,
      LU_EN => LU_EN,
      LU_Op_Select => LU_Op_Select
      );

Comparator_0: comparator
    PORT MAP( A => M_A,
           B => M_B,
           A_equal_B => A_equal_B,
           A_greater_B => A_greater_than_B,
           A_lesser_B => A_less_than_B,
           Cmp_EN => Cmp_EN
           );

Multiplier: Multiplier_4
    PORT MAP( A => M_A,
              B => M_B,
              Y => M_3,
              Overflow => Overflow_Mul,
              Mul_EN => Mul_EN
              );

Bit_Shifter: shifter_4B
    PORT MAP( Data_In => M_A,
              Shift_Dir => Sft_Dir,
              Data_Out => M_2,
              Sft_EN => Sft_EN
              );

Logical_Unit: Logical_Unit_4B
    PORT MAP( A => M_A,
```

```
                B => M_B,
                Out_Result => M_1,
                LU_EN => LU_EN,
                LU_Op_Select => LU_Op_Select
                );

LED <= D_7;
Anode <= "1110";
Overflow <= Overflw;
Zero<=Z;

end Behavioral;
```

Elaborated Design Schematic - Nano Processor



Implemented Design Schematic - Nano Processor



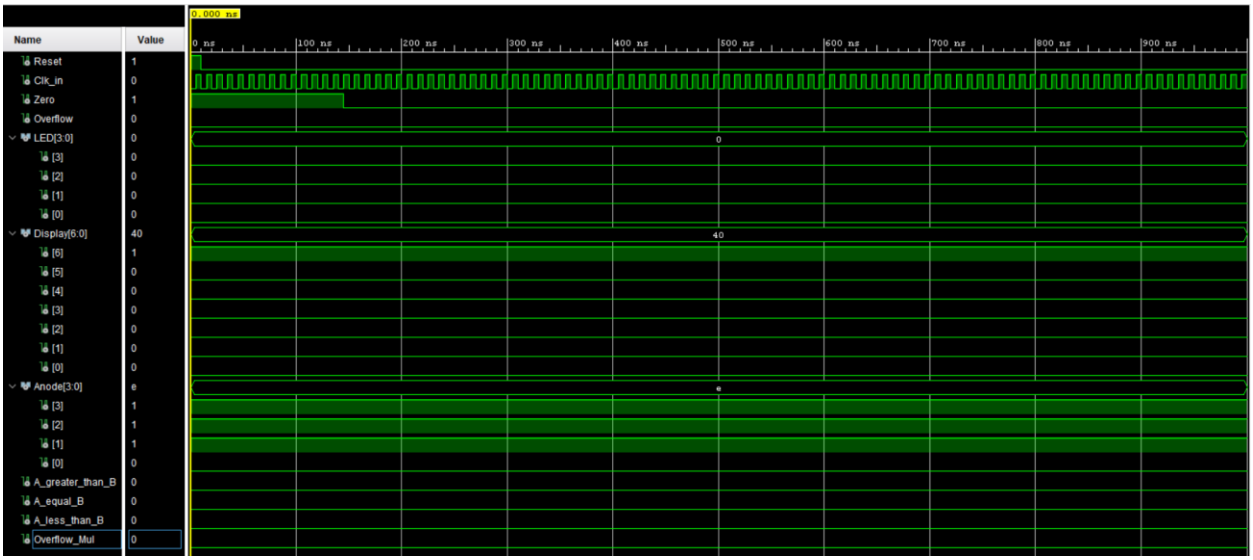
Simulation Source File - Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 05/15/2025 11:21:02 AM
-- Design Name:
-- Module Name: TB_Nanoprocessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
```

```
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------
-----------------


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY Nano_Processor_tb IS
END Nano_Processor_tb;

ARCHITECTURE behavior OF Nano_Processor_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT Nano_Processor
    PORT(
        Reset         : IN  std_logic;
        Clk_in        : IN  std_logic;
        Zero          : OUT std_logic;
        Overflow      : OUT std_logic;
        LED           : OUT std_logic_vector(3 downto 0);
        Display       : OUT std_logic_vector(6 downto 0);
        Anode         : OUT std_logic_vector(3 downto 0);
        A_greater_than_B : out STD_LOGIC;
        A_equal_B : out STD_LOGIC;
        A_less_than_B : out STD_LOGIC;
        Overflow_Mul : out STD_LOGIC
    );
    END COMPONENT;

    -- Testbench signals
    SIGNAL Reset        : std_logic := '0';
    SIGNAL Clk_in       : std_logic := '0';
    SIGNAL Zero         : std_logic;
    SIGNAL Overflow     : std_logic;
    SIGNAL LED          : std_logic_vector(3 downto 0);
    SIGNAL Display      : std_logic_vector(6 downto 0);
    SIGNAL Anode        : std_logic_vector(3 downto 0);
    SIGNAL A_greater_than_B : std_logic;
    SIGNAL A_equal_B : std_logic;
    SIGNAL A_less_than_B : std_logic;
```

```
    SIGNAL Overflow_Mul : std_logic;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: Nano_Processor
    PORT MAP (
        Reset        => Reset,
        Clk_in       => Clk_in,
        Zero         => Zero,
        Overflow     => Overflow,
        LED          => LED,
        Display      => Display,
        Anode        => Anode,
        A_greater_than_B => A_greater_than_B,
        A_equal_B => A_equal_B,
        A_less_than_B => A_less_than_B,
        Overflow_Mul => Overflow_Mul
    );

    -- Clock generation process
    clock_process: process
    begin
        while true loop
            Clk_in <= '0';
            wait for 5 ns;
            Clk_in <= '1';
            wait for 5 ns;
        end loop;
    end process;

    -- Reset and Register Reset process
    stimulus_process: process
    begin
        Reset <= '1';
        wait for 10 ns;
        Reset <= '0';
        wait for 100 ns;
        wait;
    end process;

END behavior;
```

Timing Diagram - Nano Processor



INSTRUCTION DECODER (IMPROVED VERSION)

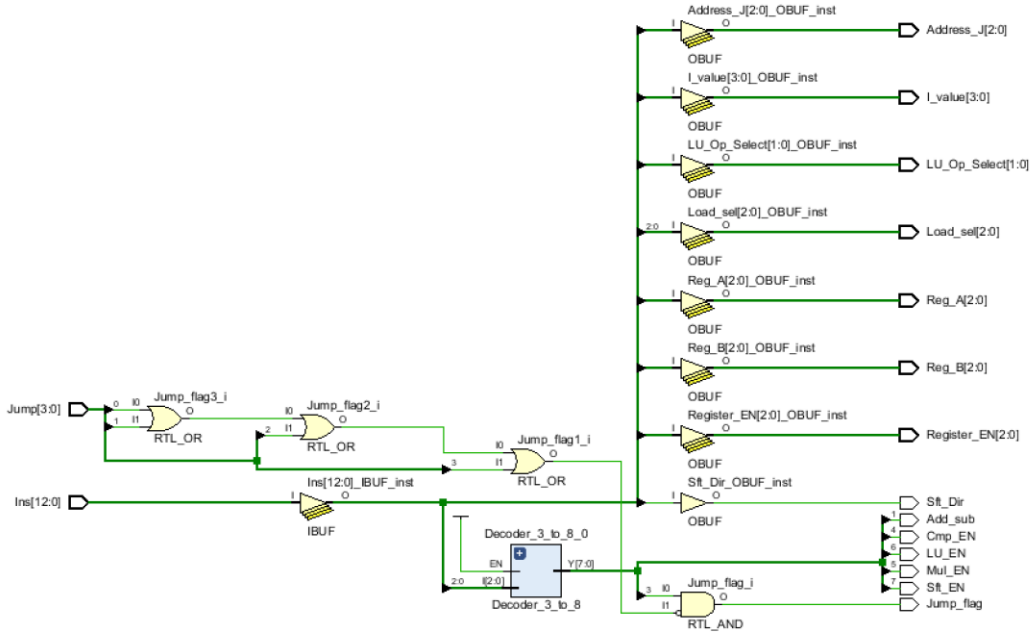
Design Source File - Instruction Decoder

```
----------------------------------------------------------------------
-----

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 Ins_Decoder is
  Port (
  Ins :          IN STD_LOGIC_VECTOR(12 downto 0); --Instructions
  Jump :         IN STD_LOGIC_VECTOR( 3 downto 0);
  Register_EN : OUT STD_LOGIC_VECTOR(2 downto 0); --Register
Enable
  Load_sel :     OUT STD_LOGIC_VECTOR(2 downto 0); -- Select the
load
```

```
  I_value :     OUT STD_LOGIC_VECTOR(3 downto 0); --Immediate
value
  Reg_A :       OUT STD_LOGIC_VECTOR(2 downto 0);
  Reg_B :       OUT STD_LOGIC_VECTOR(2 downto 0);
  Add_sub :     OUT STD_LOGIC;
  Jump_flag :   OUT STD_LOGIC;
  Address_J :   OUT STD_LOGIC_VECTOR(2 downto 0);
  Cmp_EN :      OUT STD_LOGIC;  -- Enable signal for Comparator
  Mul_EN :      OUT STD_LOGIC; -- Enable signal for Multiplier
  Sft_EN :      OUT STD_LOGIC; -- Enable signal for Bit Shifter
  Sft_Dir :     OUT STD_LOGIC; -- Shift Direction
  LU_EN :       OUT STD_LOGIC; -- Enable signal for Logical Unit
  LU_Op_Select :OUT STD_LOGIC_VECTOR(1 downto 0)-- Operation
select for Logical Unit
    );
end Ins_Decoder;

architecture Behavioral of Ins_Decoder 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;

SIGNAL ADD, NEG, MOV, JZR, CMP, MUL, SFT, LOG :STD_LOGIC;

begin

Decoder_3_to_8_0 : Decoder_3_to_8
    Port map (
    I(0) => Ins(10),
    I(1) => Ins(11),
    I(2) => Ins(12),
    EN => '1',
    Y(0) => ADD,
    Y(1) => NEG,
    Y(2) => MOV,
    Y(3) => JZR,
    Y(4) => CMP,
    Y(5) => MUL,
    Y(6) => LOG,
    Y(7) => SFT );
```

```
Reg_A <= Ins(9 downto 7);
Reg_B <= Ins(6 downto 4);
Register_EN <= Ins(9 downto 7);
Add_sub <= NEG;
I_value <= Ins(3 downto 0);
Jump_flag <= JZR AND ( NOT(Jump(0) OR Jump(1) OR Jump(2) OR
Jump(3)));
Address_J <= Ins(2 downto 0);
Load_Sel <= Ins(12 downto 10);
Cmp_EN <= CMP;
Mul_EN <= MUL;
Sft_EN <= SFT;
Sft_Dir <= Ins(2);
LU_EN <= LOG;
LU_Op_Select <= Ins(1 downto 0);

end Behavioral;
```

Elaborated Design Schematic - Instruction Decoder



Simulation Source File - Instruction Decoder

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

```
entity Ins_Decoder_tb is
end Ins_Decoder_tb;

architecture Behavioral of Ins_Decoder_tb is

    -- Component declaration for the Unit Under Test (UUT)
    component Ins_Decoder
        Port (
            Ins         : in  STD_LOGIC_VECTOR(12 downto 0);
            Jump        : in  STD_LOGIC_VECTOR(3 downto 0);
            Register_EN : out STD_LOGIC_VECTOR(2 downto 0);
            Load_sel    : out STD_LOGIC_VECTOR(2 downto 0);
            I_value     : out STD_LOGIC_VECTOR(3 downto 0);
            Reg_A       : out STD_LOGIC_VECTOR(2 downto 0);
            Reg_B       : out STD_LOGIC_VECTOR(2 downto 0);
            Add_sub     : out STD_LOGIC;
            Jump_flag   : out STD_LOGIC;
            Address_J   : out STD_LOGIC_VECTOR(2 downto 0);
            Cmp_EN      : out STD_LOGIC;
            Mul_EN      : out STD_LOGIC;
            Sft_EN      : out STD_LOGIC;
            Sft_Dir     : out STD_LOGIC;
            LU_EN       : out STD_LOGIC;
            LU_Op_Select: out STD_LOGIC_VECTOR(1 downto 0)
        );
    end component;

    -- Signals for connecting to the UUT
    signal Ins         : STD_LOGIC_VECTOR(12 downto 0);
    signal Jump        : STD_LOGIC_VECTOR(3 downto 0);
    signal Register_EN : STD_LOGIC_VECTOR(2 downto 0);
    signal Load_sel    : STD_LOGIC_VECTOR(2 downto 0);
    signal I_value     : STD_LOGIC_VECTOR(3 downto 0);
    signal Reg_A       : STD_LOGIC_VECTOR(2 downto 0);
    signal Reg_B       : STD_LOGIC_VECTOR(2 downto 0);
    signal Add_sub     : STD_LOGIC;
    signal Jump_flag   : STD_LOGIC;
    signal Address_J   : STD_LOGIC_VECTOR(2 downto 0);
    signal Cmp_EN      : STD_LOGIC;
    signal Mul_EN      : STD_LOGIC;
    signal Sft_EN      : STD_LOGIC;
    signal Sft_Dir     : STD_LOGIC;
    signal LU_EN       : STD_LOGIC;
    signal LU_Op_Select: STD_LOGIC_VECTOR(1 downto 0);
```

```
begin

    -- Instantiate the Unit Under Test (UUT)
    UUT: Ins_Decoder
        Port map (
            Ins         => Ins,
            Jump        => Jump,
            Register_EN => Register_EN,
            Load_sel    => Load_sel,
            I_value     => I_value,
            Reg_A       => Reg_A,
            Reg_B       => Reg_B,
            Add_sub     => Add_sub,
            Jump_flag   => Jump_flag,
            Address_J   => Address_J,
            Cmp_EN      => Cmp_EN,
            Mul_EN      => Mul_EN,
            Sft_EN      => Sft_EN,
            Sft_Dir     => Sft_Dir,
            LU_EN       => LU_EN,
            LU_Op_Select=> LU_Op_Select
        );

    -- Test Process
    process
    begin
        -- Test case 1: MOV instruction (opcode = "010")
        Ins <= "0100010000101"; -- MOV R1, 5
        Jump <= "0000";
        wait for 100 ns;

        Ins <= "0001110010000"; -- ADD R7, R2
        Jump <= "0000";
        wait for 100 ns;

        -- Test case 2: CMP instruction (opcode = "100")
        Ins <= "1000100110000"; -- CMP R2, R3
        Jump <= "0000";
        wait for 100 ns;
        -- Test case 3: MUL instruction (opcode = "101")
        Ins <= "1010100110000"; -- MUL R2, R3
        Jump <= "0000";
        wait for 100 ns;
```

```
        -- Test case 4: SFT instruction (opcode = "111") - Shift
Right
        Ins <= "1110110000100"; -- SFT R3, RIGHT
        Jump <= "0000";
        wait for 100 ns;

        -- Test case 5: JZR instruction (opcode = "011")
        Ins <= "0110000000111"; -- JZR R0, 7
        Jump <= "0000";
        wait for 100 ns;

        -- End Simulation
        wait;
    end process;

end Behavioral;
```

Timing Diagram - Instruction Decoder

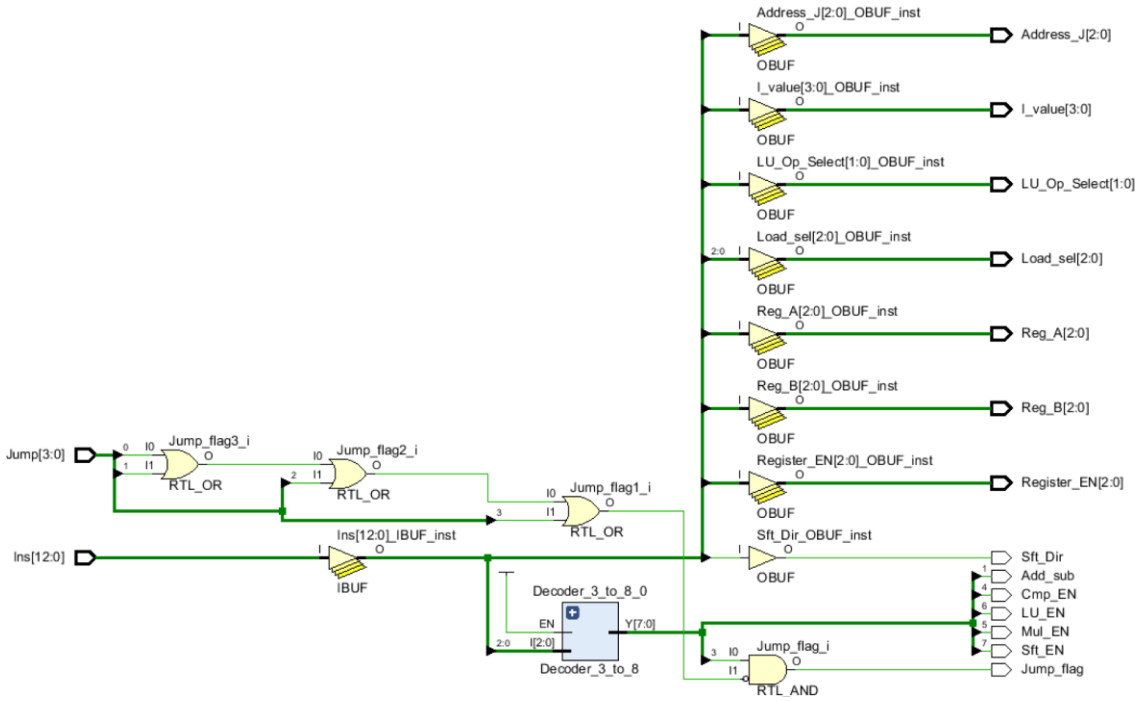
Logical Unit

Design Source File - Logical Unit

```
----------------------------------------------------------------------
-----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Logical_Unit_4B is
    Port (
        A          : in  STD_LOGIC_VECTOR (3 downto 0);
        B          : in  STD_LOGIC_VECTOR (3 downto 0);
        LU_EN      : in  STD_LOGIC;
        LU_Op_Select : in  STD_LOGIC_VECTOR (1 downto 0);
        Out_Result : out STD_LOGIC_VECTOR (3 downto 0)
    );
end Logical_Unit_4B;

architecture Behavioral of Logical_Unit_4B is
    signal AND_Result : STD_LOGIC_VECTOR(3 downto 0);
    signal OR_Result  : STD_LOGIC_VECTOR(3 downto 0);
    signal XOR_Result : STD_LOGIC_VECTOR(3 downto 0);
    signal Result     : STD_LOGIC_VECTOR(3 downto 0);

begin
    -- Logical Operations
    AND_Result <= A and B;
    OR_Result  <= A or B;
    XOR_Result <= A xor B;

    -- Select the operation based on LU_Op_Select
    process(LU_Op_Select, AND_Result, OR_Result, XOR_Result)
    begin
        case LU_Op_Select is
            when "00" => Result <= AND_Result;
            when "01" => Result <= OR_Result;
            when "10" => Result <= XOR_Result;
            when others => Result <= (others => '0');
        end case;
    end process;

    -- Apply Enable Signal
    Out_Result <= Result when LU_EN = '1' else (others => '0');
```

```
end Behavioral;
```

Elaborated Design Schematic - Logical Unit



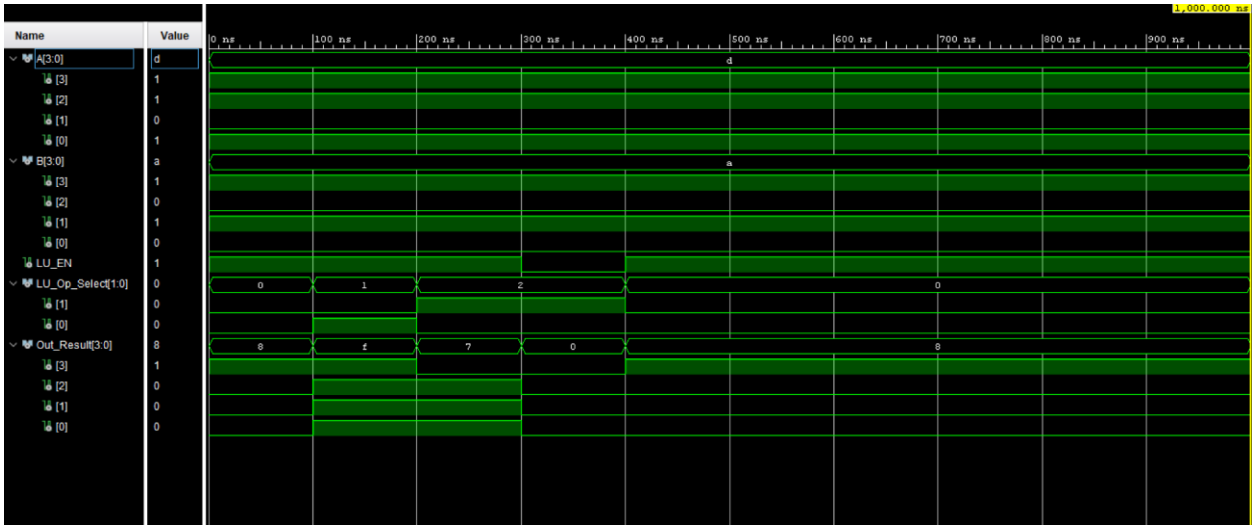
Simulation Source File - Logical Unit

```
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB_Logical_Unit is
end TB_Logical_Unit;

architecture Behavioral of TB_Logical_Unit is

    -- Component declaration for the DUT (Design Under Test)
    component Logical_Unit_4B is
        Port (
            A          : in  STD_LOGIC_VECTOR (3 downto 0);
            B          : in  STD_LOGIC_VECTOR (3 downto 0);
            LU_EN      : in  STD_LOGIC;
            LU_Op_Select : in  STD_LOGIC_VECTOR (1 downto 0);
            Out_Result : out STD_LOGIC_VECTOR (3 downto 0)
```

```
        );
    end component;

    -- Signals for the testbench
    signal A        : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal B        : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal LU_EN    : STD_LOGIC := '0';
    signal LU_Op_Select : STD_LOGIC_VECTOR(1 downto 0) := "00";
    signal Out_Result : STD_LOGIC_VECTOR(3 downto 0);

begin

    -- Instantiate the DUT
    DUT: Logical_Unit_4B
        port map(
            A => A,
            B => B,
            LU_EN => LU_EN,
            LU_Op_Select => LU_Op_Select,
            Out_Result => Out_Result
        );

    -- Stimulus process
    stimulus_proc: process
    begin
        -- Test case 1: AND Operation with Enable = 1
        A <= "1101"; B <= "1010"; LU_EN <= '1'; LU_Op_Select <=
"00";
        wait for 100 ns;

        -- Test case 2: OR Operation
        LU_Op_Select <= "01";
        wait for 100 ns;

        -- Test case 3: XOR Operation
        LU_Op_Select <= "10";
        wait for 100 ns;

        -- Test case 4: Disable Output
        LU_EN <= '0';
        wait for 100 ns;

        -- Test case 5: Enable and AND Operation again
        LU_EN <= '1'; LU_Op_Select <= "00";
```

```
        wait for 100 ns;

        wait;
    end process;
```

Timing Diagram - Logical Unit



Multiplier (4 bit)

Design Source File - Multiplier

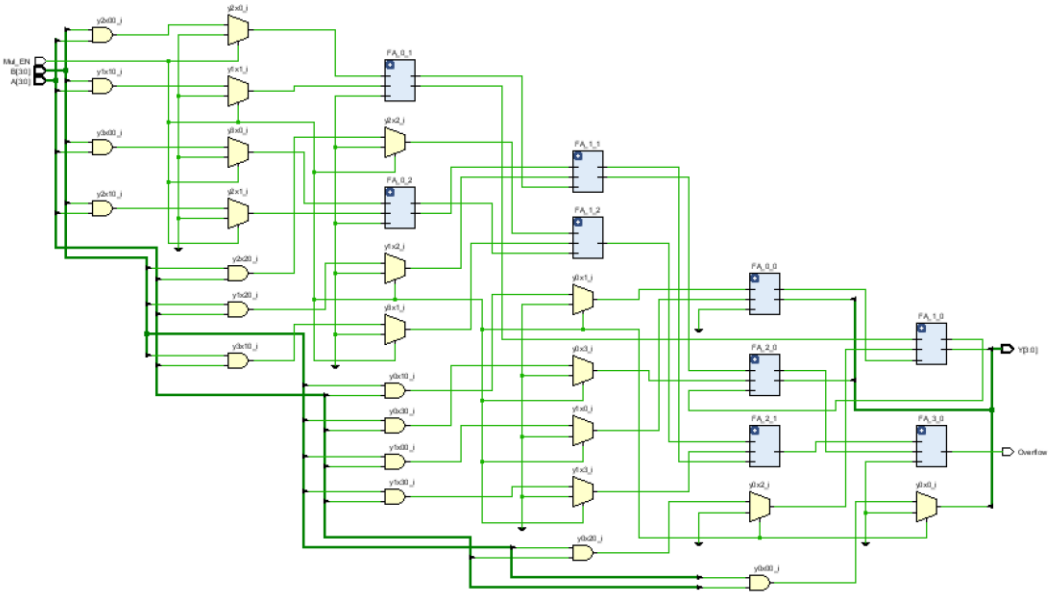
```
----------------------------------------------------------------------------
-----


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 Multiplier_4 is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Mul_EN : in STD_LOGIC);
end Multiplier_4;

architecture Behavioral of Multiplier_4 is

    COMPONENT 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 COMPONENT;

    SIGNAL y0x0, y0x1, y0x2, y0x3 : std_logic;
    SIGNAL y1x0, y1x1, y1x2, y1x3 : std_logic;
    SIGNAL y2x0, y2x1, y2x2       : std_logic;
    SIGNAL y3x0, y3x1             : std_logic;

    SIGNAL s_0_1, s_0_2, s_0_3, s_0_4       : std_logic;
    SIGNAL s_1_0, s_1_1, s_1_2, s_1_3, s_1_4: std_logic;
    SIGNAL c_0_0, c_0_1, c_0_2, c_0_3       : std_logic;
    SIGNAL c_1_0, c_1_1, c_1_2, c_1_3, c_1_4: std_logic;

begin

    FA_0_0 : FA port map(
        A     => y0x1,
        B     => y1x0,
        C_in  => '0',
        S     => s_0_1,
        C_out => c_0_0
    );

    FA_0_1 : FA port map(
        A     => y2x0,
        B     => y1x1,
        C_in  => '0',
```

```
    S     => s_1_0,
    C_out => c_1_0
);

FA_1_0 : FA port map(
    A     => s_1_0,
    B     => y0x2,
    C_in  => c_0_0,
    S     => s_0_2,
    C_out => c_0_1
);

FA_0_2 : FA port map(
    A     => y3x0,
    B     => y2x1,
    C_in  => '0',
    S     => s_1_1,
    C_out => c_1_1
);

FA_1_1 : FA port map(
    A     => s_1_1,
    B     => y1x2,
    C_in  => c_1_0,
    S     => s_1_2,
    C_out => c_1_2
);

FA_2_0 : FA port map(
    A     => s_1_2,
    B     => y0x3,
    C_in  => c_0_1,
    S     => s_0_3,
    C_out => c_0_2
);

FA_1_2 : FA port map(
    A     => y2x2,
    B     => y3x1,
    C_in  => c_1_1,
    S     => s_1_3,
    C_out => c_1_3
);
```

```
    FA_2_1 : FA port map(
        A     => s_1_3,
        B     => y1x3,
        C_in  => c_1_2,
        S     => s_1_4,
        C_out => c_1_4
    );

    FA_3_0 : FA port map(
        A     => s_1_4,
        B     => c_0_2,
        C_in  => '0',
        S     => s_0_4,
        C_out => c_0_3
    );

    -- Partial Products
    y0x0 <= B(0) AND A(0) when Mul_EN = '1' else '0';
    y0x1 <= B(0) AND A(1) when Mul_EN = '1' else '0';
    y0x2 <= B(0) AND A(2) when Mul_EN = '1' else '0';
    y0x3 <= B(0) AND A(3) when Mul_EN = '1' else '0';

    y1x0 <= B(1) AND A(0) when Mul_EN = '1' else '0';
    y1x1 <= B(1) AND A(1) when Mul_EN = '1' else '0';
    y1x2 <= B(1) AND A(2) when Mul_EN = '1' else '0';
    y1x3 <= B(1) AND A(3) when Mul_EN = '1' else '0';

    y2x0 <= B(2) AND A(0) when Mul_EN = '1' else '0';
    y2x1 <= B(2) AND A(1) when Mul_EN = '1' else '0';
    y2x2 <= B(2) AND A(2) when Mul_EN = '1' else '0';

    y3x0 <= B(3) AND A(0) when Mul_EN = '1' else '0';
    y3x1 <= B(3) AND A(1) when Mul_EN = '1' else '0';

    -- Output Assignment
    Y(0)      <= y0x0;
    Y(1)      <= s_0_1;
    Y(2)      <= s_0_2;
    Y(3)      <= s_0_3;
    Overflow  <= s_0_4;  -- Changed to output only 5 bits

end Behavioral;
```

Elaborated Design Schematic - Multiplier



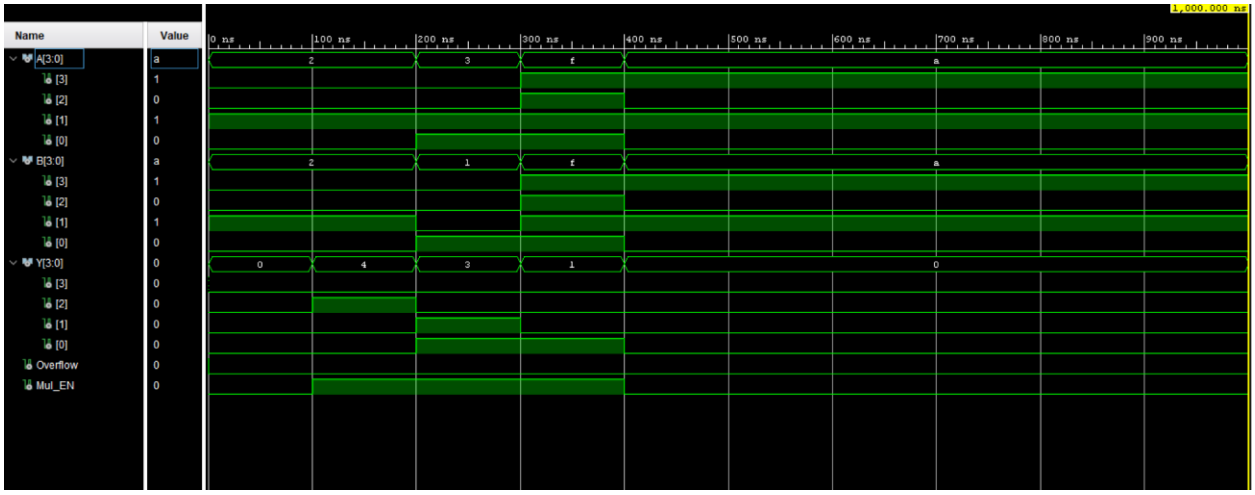
Simulation Source File - Multiplier

```
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB_Multiplier_4 is
end TB_Multiplier_4;

architecture Behavioral of TB_Multiplier_4 is
    -- Component declaration for the DUT (Design Under Test)
    component Multiplier_4 is
        Port (
            A : in STD_LOGIC_VECTOR (3 downto 0);
            B : in STD_LOGIC_VECTOR (3 downto 0);
            Y : out STD_LOGIC_VECTOR (3 downto 0);
            Overflow : out std_logic;
            Mul_EN : in STD_LOGIC
        );
    end component;

    -- Signals for stimulus generation
```

```
    signal A       : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal B       : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal Y       : STD_LOGIC_VECTOR(3 downto 0);
    signal Overflow: std_logic;
    signal Mul_EN  : std_logic := '0';

begin
    -- Instantiate the DUT
    DUT: Multiplier_4
        port map (
            A => A,
            B => B,
            Y => Y,
            Overflow => Overflow,
            Mul_EN => Mul_EN
        );

    -- Stimulus process
    stimulus_proc: process
    begin
        -- Test case 1: Multiplication Disabled
        A <= "0010";
        B <= "0010";
        Mul_EN <= '0';
        wait for 100 ns;

        -- Test case 2: Multiplication Enabled
        Mul_EN <= '1';
        wait for 100 ns;

        -- Test case 3: Another Multiplication
        A <= "0011";
        B <= "0001";
        wait for 100 ns;

        -- Test case 4: Overflow Scenario
        A <= "1111";
        B <= "1111";
        wait for 100 ns;

        -- Test case 5: Multiplication Disabled
        Mul_EN <= '0';
        A <= "1010";
        B <= "1010";
```

```
        wait for 100 ns;

        -- End of test cases
        wait;
    end process stimulus_proc;

end Behavioral;
```

Timing Diagram - Multiplier



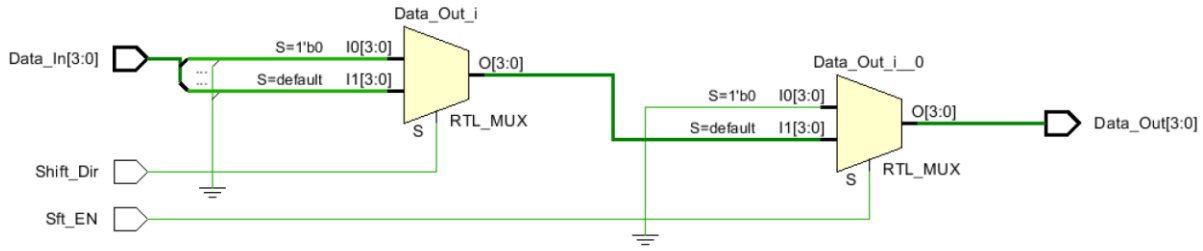
Bit shifter (4 bit)

Design Source File - Bit Shifter

```
----------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shifter_4B is
    Port (
        Data_In : in STD_LOGIC_VECTOR (3 downto 0);
        Shift_Dir : in STD_LOGIC;
        Sft_EN : in STD_LOGIC;
        Data_Out : out STD_LOGIC_VECTOR (3 downto 0)
    );
```

```
end shifter_4B;

architecture Behavioral of shifter_4B is
begin
    process(Data_In, Shift_Dir, Sft_EN)
    begin
        if Sft_EN = '0' then
            Data_Out <= "0000";
        else
            if Shift_Dir = '0' then
                Data_Out(0) <= '0';
                Data_Out(1) <= Data_In(0);
                Data_Out(2) <= Data_In(1);
                Data_Out(3) <= Data_In(2);
            else
                Data_Out(0) <= Data_In(1);
                Data_Out(1) <= Data_In(2);
                Data_Out(2) <= Data_In(3);
                Data_Out(3) <= '0';
           end if;
        end if;
    end process;

end Behavioral;
```

Elaborated Design Schematic - Bit Shifter



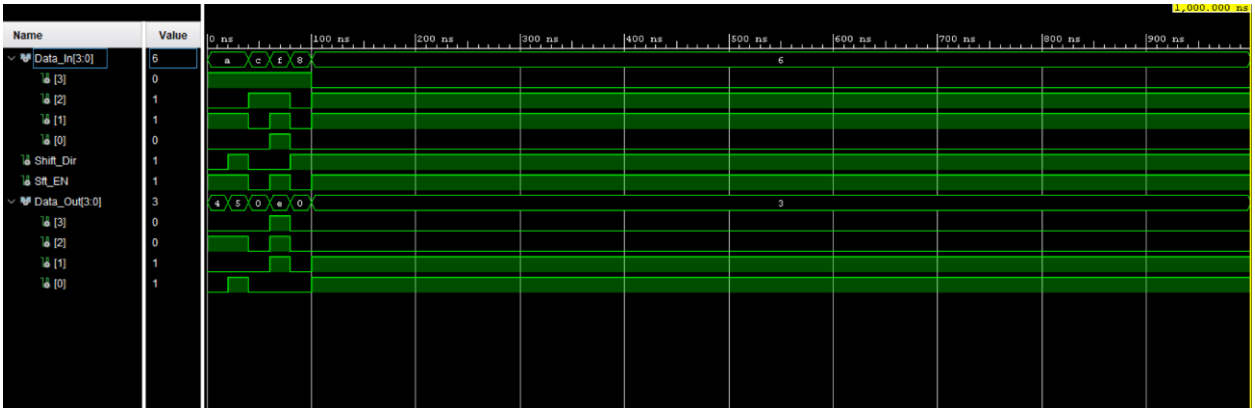
Simulation Source File - Bit Shifter

```
----------------------------------------------------------------------
-----

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

```
entity TB_shifter_4B is
end TB_shifter_4B;

architecture Behavioral of TB_shifter_4B is

    component shifter_4B
        Port (
            Data_In : in STD_LOGIC_VECTOR (3 downto 0);
            Shift_Dir : in STD_LOGIC;
            Sft_EN : in STD_LOGIC;
            Data_Out : out STD_LOGIC_VECTOR (3 downto 0)
        );
    end component;

    signal Data_In : STD_LOGIC_VECTOR (3 downto 0);
    signal Shift_Dir : STD_LOGIC;
    signal Sft_EN : STD_LOGIC;
    signal Data_Out : STD_LOGIC_VECTOR (3 downto 0);

begin

    UUT : shifter_4B
    port map(
        Data_In => Data_In,
        Shift_Dir => Shift_Dir,
        Sft_EN => Sft_EN,
        Data_Out => Data_Out
    );

    process
    begin
        -- Enable shifting and shift left
        Sft_EN <= '1';
        Data_In <= "1010";
        Shift_Dir <= '0';
        wait for 20 ns;

        -- Enable shifting and shift right
        Sft_EN <= '1';
        Data_In <= "1010";
        Shift_Dir <= '1';
        wait for 20 ns;
```

```
        -- Disable shifting, hold the previous state
        Sft_EN <= '0';
        Data_In <= "1100";
        Shift_Dir <= '0';
        wait for 20 ns;

        -- Enable shifting and shift left
        Sft_EN <= '1';
        Data_In <= "1111";
        Shift_Dir <= '0';
        wait for 20 ns;

        -- Disable shifting, hold the previous state
        Sft_EN <= '0';
        Data_In <= "1000";
        Shift_Dir <= '1';
        wait for 20 ns;

        -- Enable shifting and shift right
        Sft_EN <= '1';
        Data_In <= "0110";
        Shift_Dir <= '1';
        wait for 20 ns;

        wait;
    end process;

end Behavioral;
```

Timing Diagram - Bit Shifter

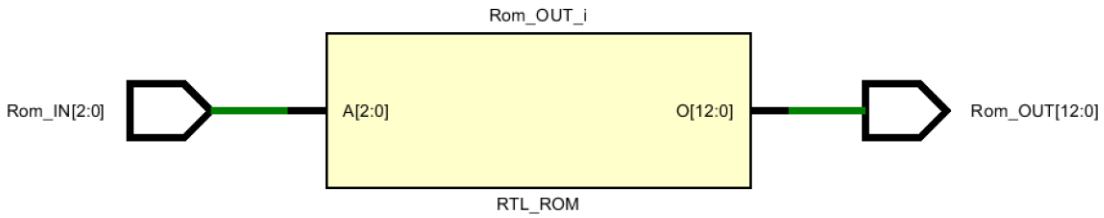


Program ROM (Improved)

Design Source File - Program ROM

```
----------------------------------------------------------------------
-----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;  -- Required for 'unsigned'

entity Program_ROM is
    Port (
        Rom_IN  : IN  STD_LOGIC_VECTOR(2 downto 0);
        Rom_OUT : OUT STD_LOGIC_VECTOR(12 downto 0)
    );
end Program_ROM;

architecture Behavioral of Program_ROM is

    type rom_type is array (0 to 7) of STD_LOGIC_VECTOR(12 downto
0);
    signal programRom : rom_type := (
        "0101110000001", -- MOVI R7, 1
        "0101110000010", -- MOVI R7, 2
        "0100100000010", --MOVI R2,2
        "0101110000001", -- MOVI R7, 3
--"1101110100010", --XOR R7
        "1111110000000", -- MOVI R3, 3
        "1000100110000", -- CMP R2, R3
        "1011110100000", -- MUL R2, R3
        "0001110100000" -- Add R7,R2

    );

begin
    Rom_OUT <= programRom(to_integer(unsigned(Rom_IN)));

end Behavioral;
```

Elaborated Design Schematic - Program ROM

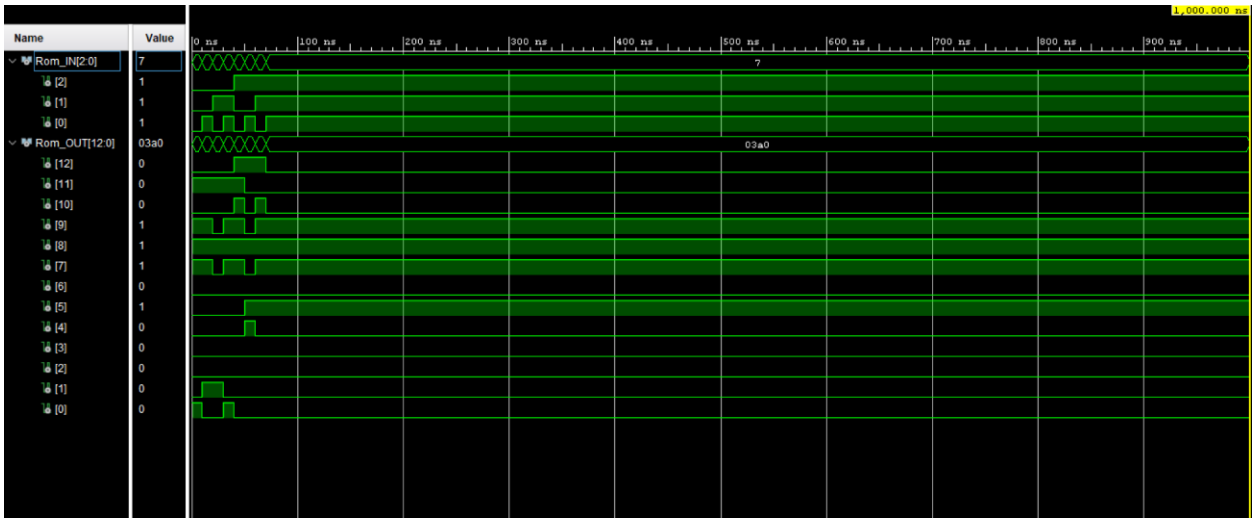


Simulation Source File - Program ROM

```
--------------------------------------------------------------------------------

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

entity Program_ROM_tb is
end Program_ROM_tb;

architecture Behavioral of Program_ROM_tb is

    -- Component declaration of the UUT
    component Program_ROM is
        Port (
            Rom_IN  : in  STD_LOGIC_VECTOR(2 downto 0);
            Rom_OUT : out STD_LOGIC_VECTOR(12 downto 0)
        );
    end component;

    -- Signals to connect to UUT
    signal Rom_IN  : STD_LOGIC_VECTOR(2 downto 0) := (others =>
'0');
    signal Rom_OUT : STD_LOGIC_VECTOR(12 downto 0);

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Program_ROM
        port map (
            Rom_IN  => Rom_IN,
            Rom_OUT => Rom_OUT
```

```
        );

    -- Stimulus process
    stim_proc: process
    begin
        -- Iterate through all possible addresses
        for i in 0 to 7 loop
            Rom_IN <= std_logic_vector(to_unsigned(i, 3));  --
Apply address
            wait for 10 ns;                                 --
Wait for output to settle
        end loop;
        wait;  -- End of simulation
    end process;

end Behavioral;
```

Timing Diagram - Program ROM



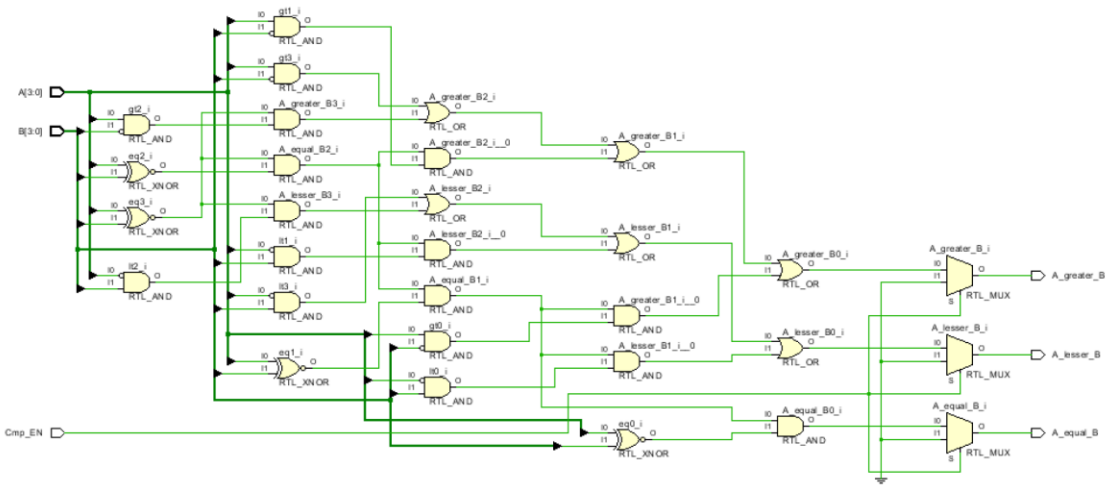
Comparator (4 bit)

Design Source File - Comparator

```
----------------------------------------------------------------------

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 comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           A_equal_B : out STD_LOGIC;
           A_greater_B : out STD_LOGIC;
           A_lesser_B : out STD_LOGIC;
           Cmp_EN : in STD_LOGIC);
end comparator;

architecture Behavioral of comparator is

    signal eq0, eq1, eq2, eq3 : STD_LOGIC;
    signal gt0, gt1, gt2, gt3 : STD_LOGIC;
    signal lt0, lt1, lt2, lt3 : STD_LOGIC;

begin

    eq0 <= A(0) xnor B(0);
    eq1 <= A(1) xnor B(1);
    eq2 <= A(2) xnor B(2);
    eq3 <= A(3) xnor B(3);

    gt3 <= A(3) and not B(3);
    gt2 <= A(2) and not B(2);
    gt1 <= A(1) and not B(1);
    gt0 <= A(0) and not B(0);
```

```
    lt3 <= not A(3) and B(3);
    lt2 <= not A(2) and B(2);
    lt1 <= not A(1) and B(1);
    lt0 <= not A(0) and B(0);

    -- Conditional Output based on Cmp_EN
    A_equal_B <= (eq3 and eq2 and eq1 and eq0) when Cmp_EN = '1'
else '0';
    A_greater_B <= (gt3 or (eq3 and gt2) or (eq3 and eq2 and gt1)
or (eq3 and eq2 and eq1 and gt0)) when Cmp_EN = '1' else '0';
    A_lesser_B <= (lt3 or (eq3 and lt2) or (eq3 and eq2 and lt1)
or (eq3 and eq2 and eq1 and lt0)) when Cmp_EN = '1' else '0';

end Behavioral;
```

Elaborated Design Schematic - Comparator



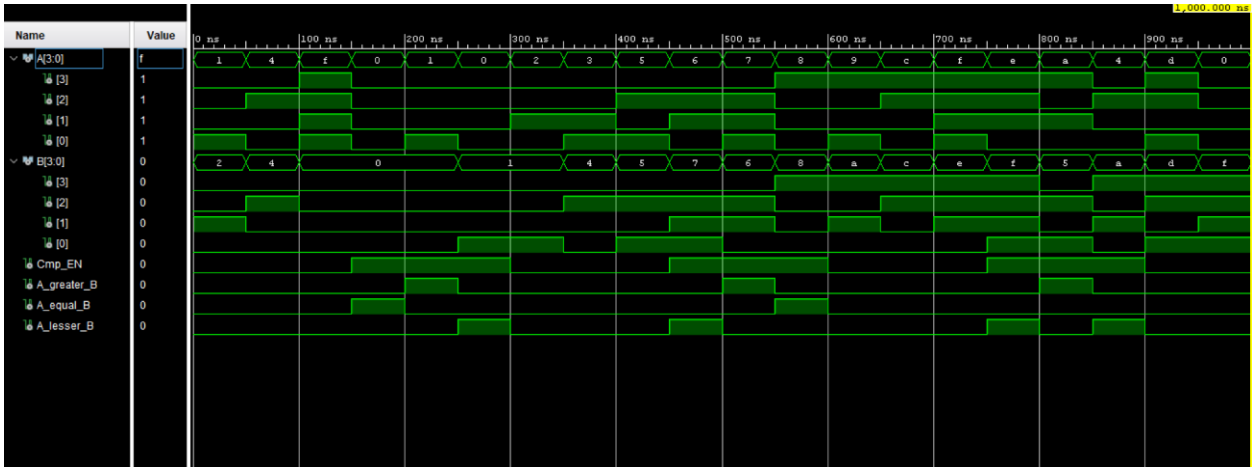
Simulation Source File - Comparator

```
--------------------------------------------------------------------
-----

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Comparator_TB is
end Comparator_TB;
```

```
architecture Behavioral of Comparator_TB is

    component comparator
        Port (
            A         : in  STD_LOGIC_VECTOR (3 downto 0);
            B         : in  STD_LOGIC_VECTOR (3 downto 0);
            Cmp_EN    : in  STD_LOGIC;
            A_greater_B : out STD_LOGIC;
            A_equal_B   : out STD_LOGIC;
            A_lesser_B  : out STD_LOGIC
        );
    end component;

    signal A, B       : STD_LOGIC_VECTOR (3 downto 0);
    signal Cmp_EN     : STD_LOGIC;
    signal A_greater_B : STD_LOGIC;
    signal A_equal_B   : STD_LOGIC;
    signal A_lesser_B  : STD_LOGIC;

begin

    uut: comparator
        Port Map (
            A => A,
            B => B,
            Cmp_EN => Cmp_EN,
            A_greater_B => A_greater_B,
            A_equal_B => A_equal_B,
            A_lesser_B => A_lesser_B
        );

    -- Test Process
    process
    begin
        -- Test when Cmp_EN is '0'
        Cmp_EN <= '0';
        A <= "0001"; B <= "0010"; wait for 50 ns;
        A <= "0100"; B <= "0100"; wait for 50 ns;
        A <= "1111"; B <= "0000"; wait for 50 ns;

        -- Test when Cmp_EN is '1'
        Cmp_EN <= '1';
        A <= "0000"; B <= "0000"; wait for 50 ns;
```

```
        A <= "0001"; B <= "0000"; wait for 50 ns;
        A <= "0000"; B <= "0001"; wait for 50 ns;
        Cmp_EN <= '0';
        A <= "0010"; B <= "0001"; wait for 50 ns;
        A <= "0011"; B <= "0100"; wait for 50 ns;
        A <= "0101"; B <= "0101"; wait for 50 ns;
        Cmp_EN <= '1';
        A <= "0110"; B <= "0111"; wait for 50 ns;
        A <= "0111"; B <= "0110"; wait for 50 ns;
        A <= "1000"; B <= "1000"; wait for 50 ns;
        Cmp_EN <= '0';
        A <= "1001"; B <= "1010"; wait for 50 ns;
        A <= "1100"; B <= "1100"; wait for 50 ns;
        A <= "1111"; B <= "1110"; wait for 50 ns;
        Cmp_EN <= '1';
        A <= "1110"; B <= "1111"; wait for 50 ns;
        A <= "1010"; B <= "0101"; wait for 50 ns;
        A <= "0100"; B <= "1010"; wait for 50 ns;
        Cmp_EN <= '0';
        A <= "1101"; B <= "1101"; wait for 50 ns;
        A <= "0000"; B <= "1111"; wait for 50 ns;
        A <= "1111"; B <= "0000"; wait for 50 ns;
        Cmp_EN <= '1';
        A <= "0110"; B <= "0110"; wait for 50 ns;
        A <= "1001"; B <= "1001"; wait for 50 ns;

        wait;
    end process;

end Behavioral;
```

Timing Diagram - Comparator



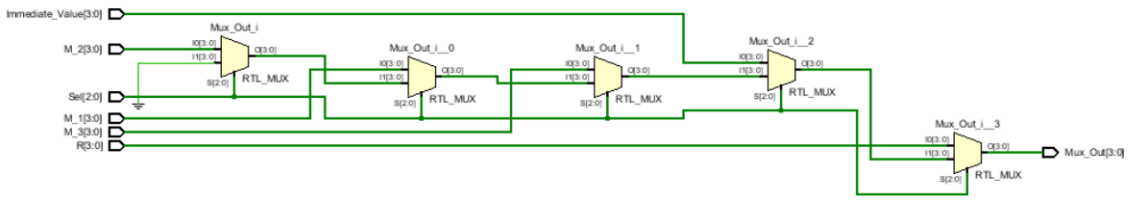
5-way 4-bit Multiplexer

Design Source File - 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_5W_4B is
    Port ( Immediate_Value : in STD_LOGIC_VECTOR (3 downto 0);
           R : in STD_LOGIC_VECTOR (3 downto 0);
           M_1 : in STD_LOGIC_VECTOR (3 downto 0);
           M_2 : in STD_LOGIC_VECTOR (3 downto 0);
           M_3 : in STD_LOGIC_VECTOR (3 downto 0);
           Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Mux_Out : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_5W_4B;

architecture Behavioral of Mux_5W_4B is

begin
    process(Immediate_Value,R,M_1,M_2,M_3,Sel)
    begin
        if Sel = "000" then
            Mux_Out <= R;
        elsif Sel = "010" then
            Mux_Out <= Immediate_Value;
        elsif Sel = "101" then
            Mux_Out <= M_3;
        elsif Sel = "110" then
            Mux_Out <= M_1;
        elsif Sel = "111" then
            Mux_Out <= M_2;
        else
```

```
            Mux_Out <= "0000";

    end if;
    end process;
end Behavioral;
```

Elaborated Design Schematic - Multiplexer

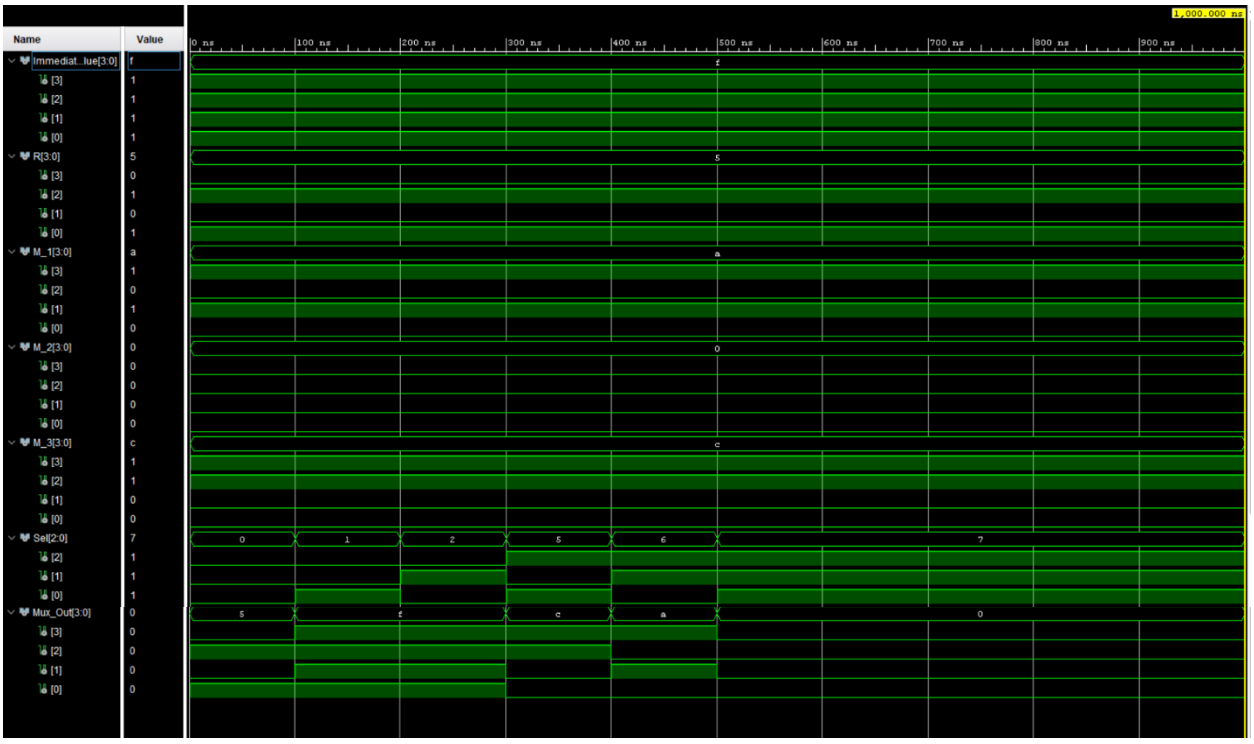


Simulation Source File - Multiplexer

```
--------------------------------------------------------------------------------


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 TB_Mux_5W_4B is
--  Port ( );
end TB_Mux_5W_4B;

architecture Behavioral of TB_Mux_5W_4B is
component Mux_5W_4B
port( Immediate_Value : in STD_LOGIC_VECTOR (3 downto 0);
           R : in STD_LOGIC_VECTOR (3 downto 0);
           M_1 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            M_2 : in STD_LOGIC_VECTOR (3 downto 0);
            M_3 : in STD_LOGIC_VECTOR (3 downto 0);
            Sel : in STD_LOGIC_VECTOR (2 downto 0);
            Mux_Out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal Immediate_Value, R, M_1, M_2, M_3 : STD_LOGIC_VECTOR(3
downto 0);
signal Sel : STD_LOGIC_VECTOR(2 downto 0);
signal Mux_Out : STD_LOGIC_VECTOR(3 downto 0);

begin
UUT : Mux_5W_4B
port map(
    Immediate_Value => Immediate_Value,
        R => R,
        M_1 => M_1,
        M_2 => M_2,
        M_3 => M_3,
        Sel => Sel,
        Mux_Out => Mux_Out
);
process
begin
    R <= "0101"; -- Sel = "000"
    Immediate_Value <= "1111"; -- Sel = "010"
    M_1 <= "1010"; -- Sel = "110"
    M_2 <= "0000"; -- Sel = "111"
    M_3 <= "1100"; -- Sel = "101"

    Sel <= "000"; wait for 100 ns; -- Output: R
        Sel <= "001"; wait for 100 ns; -- Output: 0000 (default)
        Sel <= "010"; wait for 100 ns; -- Output: Immediate_Value
        Sel <= "101"; wait for 100 ns; -- Output: M_3
        Sel <= "110"; wait for 100 ns; -- Output: M_1
        Sel <= "111"; wait for 100 ns; -- Output: M_2

    wait;
 end process;

end Behavioral;
```

Timing Diagram - Multiplexer



Constraint File

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports Clk_in]
     set_property IOSTANDARD LVCMOS33 [get_ports Clk_in]
     create_clock -add -name sys_clk_pin -period 10.00 -waveform
{0 5} [get_ports {Clk_in}]
## LEDs

##Register 7 value
set_property PACKAGE_PIN U16 [get_ports {LED[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {LED[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {LED[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {LED[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]

##Comparator output
set_property PACKAGE_PIN U14 [get_ports {A_less_than_B}]
```

```
     set_property IOSTANDARD LVCMOS33 [get_ports {A_less_than_B}]
set_property PACKAGE_PIN V14 [get_ports {A_equal_B}]
     set_property IOSTANDARD LVCMOS33 [get_ports {A_equal_B}]
set_property PACKAGE_PIN V13 [get_ports {A_greater_than_B}]
     set_property IOSTANDARD LVCMOS33 [get_ports
{A_greater_than_B}]

##Multiplier Overflow
set_property PACKAGE_PIN U3 [get_ports {Overflow_Mul}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Overflow_Mul}]

##Adder Subtractor
set_property PACKAGE_PIN P1 [get_ports {Zero}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]
set_property PACKAGE_PIN L1 [get_ports {Overflow}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Overflow}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {Display[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[0]}]
set_property PACKAGE_PIN W6 [get_ports {Display[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[1]}]
set_property PACKAGE_PIN U8 [get_ports {Display[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[2]}]
set_property PACKAGE_PIN V8 [get_ports {Display[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[3]}]
set_property PACKAGE_PIN U5 [get_ports {Display[4]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[4]}]
set_property PACKAGE_PIN V5 [get_ports {Display[5]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[5]}]
set_property PACKAGE_PIN U7 [get_ports {Display[6]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Display[6]}]


set_property PACKAGE_PIN U2 [get_ports {Anode[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {Anode[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {Anode[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[3]}]
```

```
##Buttons
set_property PACKAGE_PIN U18 [get_ports Reset]
     set_property IOSTANDARD LVCMOS33 [get_ports Reset]
```

Optimization Details

Nanoprocessor

Name	Slice LUTs (20800)	Bonded IOB (106)	BUFGCTRL (32)	Slice Registers (41600)	Slice (8 50)	LUT as Logic (20800)
Nano_Processor	29	19	1	42	17	29
Slow_Clock_0 (Slow_Clk)	1	0	0		7	1
Register_Bank (Reg_Bank)	7	0	0		8	7
Reg_7 (Reg_1)	7	0	0		6	7
Reg_2 (Reg_0)	0	0	0		1	0
Reg_1 (Reg)	0	0	0		2	0
Program_counter (p_counter)	21	0	0		8	21
D_FF_2 (D_FF_3)	15	0	0		7	15
D_FF_1 (D_FF_2)	5	0	0		4	5
D_FF_0 (D_FF)	1	0	0		2	1

No of LUTs = 29

Improved Nanoprocessor

Name 1	Slice LUTs (20800)	Bonded IOB (106)	BUFGCTRL (32)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)
Nano_Processor	49	23	1	37	25	49
Program_counter (p_counter)	44	0	0		15	44
D_FF_0 (D_FF)	9	0	0		7	9
D_FF_1 (D_FF_1)	1	0	0		2	1
D_FF_2 (D_FF_2)	34	0	0		14	34
Register_Bank (Reg_Bank)	4	0	0		5	4
Reg_2 (Reg)	0	0	0		1	0
Reg_7 (Reg_0)	4	0	0		4	4
Slow_Clock_0 (Slow_Clk)	1	0	0		7	1

No of LUTs = 53

CONTRIBUTION OF THE TEAM MEMBERS

Name	Work done
DAISHIKA K.S.H.	k-way b-bit multiplexers (2 hours) Instruction Decoder (1 hour) Final Nano Processor (2 hours) Instruction Decoder (1 hour) 4-bit Add/Subtract unit (1 hour) Constrain file (1 hour)
AMANTHA H.D.K.N.	3-bit adder (1 hour) 4-bit multiplier (improved version) (2 hours) Logical Unit (improved version) (2 hours) 4-bit Comparator (improved version) (1 hour) Simulation files (Multiplexers, Instruction Decoder) (1 hour) Final Lab Report (1 hour)
GURUSINGHE C.R.	3-bit Program Counter (PC) (1 hour) Final Nano Processor (improved version) (2 hours) Register Bank (1 hour) Program ROM (improved version) (1 hour) Instruction Decoder (improved version) (2 hours) Final Lab Report (1 hour)
DIVYANGI W.A.S.	Bit shifter (improved version) (2 hours) 5-way *4*-bit multiplexer (improved version) (2 hours) Register Bank (1 hour) Program ROM (1 hour) Simulation files (Bit shifter, 5-way *4*-bit multiplexer) (1 hour)