CS1050 - Computer Organization and Digital Design

Lab 9 & 10 Nano Processor Design Competition

- **Group 41**

  - RATHNAYAKE R.M.I.B. 220526N
  - JAYATHUNGA R.D.S.S. 220277B
  - DISSANAYAKE C.N. 220133G
  - RAVISHAN A.D.P. 220532E

Introduction

- Simply the lab task was to design a nano processor using the components we learned in our previous lab classes. The task was a team project with 4 members in each team.
- We designed 2 versions for the nano processor. First, we designed the nanoprocessor with the minimum requirements mentioned in the lab sheet. There we used 4-bit Add/Subtract unit, 3-bit adder, 3-bit Program Counter, 8-way 2-bit multiplexers, 2-way 3-bit multiplexers, 2-way 4-bit multiplexers, Register Bank, Program ROM and Instruction Decoder as our main components.
- Then we designed an extended version of our nanoprocessor using ALU and comparator as new components. In there we replaced 4-bit Add/Subtract unit with the ALU. We used 4-bit comparator, 4-bit Add/Subtract unit, 2-bit multiplier, and a logic operation unit inside the ALU.
- The nanoprocessor was used to execute up to 8 instructions for different tasks. Primarily we implemented the design to get the sum of all integers between 1 and 3. Then we used an ALU to implement summation, subtraction, and multiplication as arithmetic operations and logical operations. Such as bitwise OR, bitwise AND etc. in our extended version.

Table of Content

1) Design with minimum qualifications

• Instructions for operating the machine

1. Allocated reset button
    - The nano processor can be reset by pressing the btnC.
    - The clock speed of the internal clock of the machine was reduced from 100MHz to 0.5MHz using a slow clock to make the calculation process visible to the naked eye.
    - In order to reset the nanoprocessor, it is necessary to press and hold the reset button for at least 2 seconds.

2. LED signal
    - LED0-LED3 Output of R7 register in Register Bank
    - LED0-LED3 → Outputs a 4-bit number as a signed number in two's complement method

- LED12-LED15 Flags of 4-bit Add/Subtract Unit
    - LED12 → Sign Flag
    - LED13 → Overflow Flag
    - LED14 → Zero Flag
    - LED15 → Carry Flag

3. 7-Segment Display
    - The rightmost segment of the 7-Segment display is used to display the magnitude of the output from the R7 register in the Register Bank.

4. Expected behavior of the program
    - After the calculation is performed the process will be held because of the jump instruction implemented.
    - Therefore, it is necessary to manually reset the nanoprocessor to perform the instructions hardcoded in the ROM.

- Slice Logic and Design Primitives

- Slice Logic

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	74	0	20800	0.36
LUT as Logic	74	0	20800	0.36
LUT as Memory	0	0	9600	0.00
Slice Registers	49	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

- Design Primitives

7. Primitives

Ref Name	Used	Functional Category
FDRE	37	Flop & Latch
LUT2	32	LUT
LUT4	25	LUT
LUT3	23	LUT
OBUF	19	IO
FDCE	12	Flop & Latch
LUT5	11	LUT
CARRY4	8	CarryLogic
LUT6	7	LUT
LUT1	2	LUT
IBUF	2	IO
BUFG	1	Clock

Executable Instruction Set

Instruction	Description	Format(12-bit instruction)
MOVI R,d	Move(11 downto 10 ==10) immediate value d(3 downto 0) to register R (9 downto 7).	1 0 R R R 0 0 0 d d d d
ADD Ra,Rb	Add (11 downto 10 ==00) values in registers Ra(9 downto 7) and Rb(6 downto 4) and store the result in Ra.	0 0 Ra Ra Ra Ra Rb Rb Rb 0 0 0 0
NEG R	Get negation(11 downto 10 ==01) of the value that store in the register R(9 downto 7) and restore the created negation to R register.	0 1 R R R 0 0 0 0 0 0 0 0
JZR R,d	Jump(11 downto 10 ==11) to Pc to value d(2 downto 0) if value in register R(9 downto 7) is 0. i.e, If R==0; Pc←d; Else; Pc← Pc + 1;	1 1 R R R 0 0 0 0 d d d

Assembly Code For The Design

```
;       process => 3 + 2 + 1 = 6

MOVI R1,4       ; R1 <- 4
MOVI R2,1       ; R2 <- 1
NEG  R2         ; R2 <- -1
ADD  R1,R2      ; R1 <- R2 + R1    3
JZR  R1,7       ; Jump to line 8 if R1 = 0
ADD  R7,R1      ; R7 <- R7 + R1    5
JZR  R0,3       ; Jump to line 4 if R0 = 0
JZR  R0,7       ; Jump to line 8 if R0 = 0
```

- Design Components

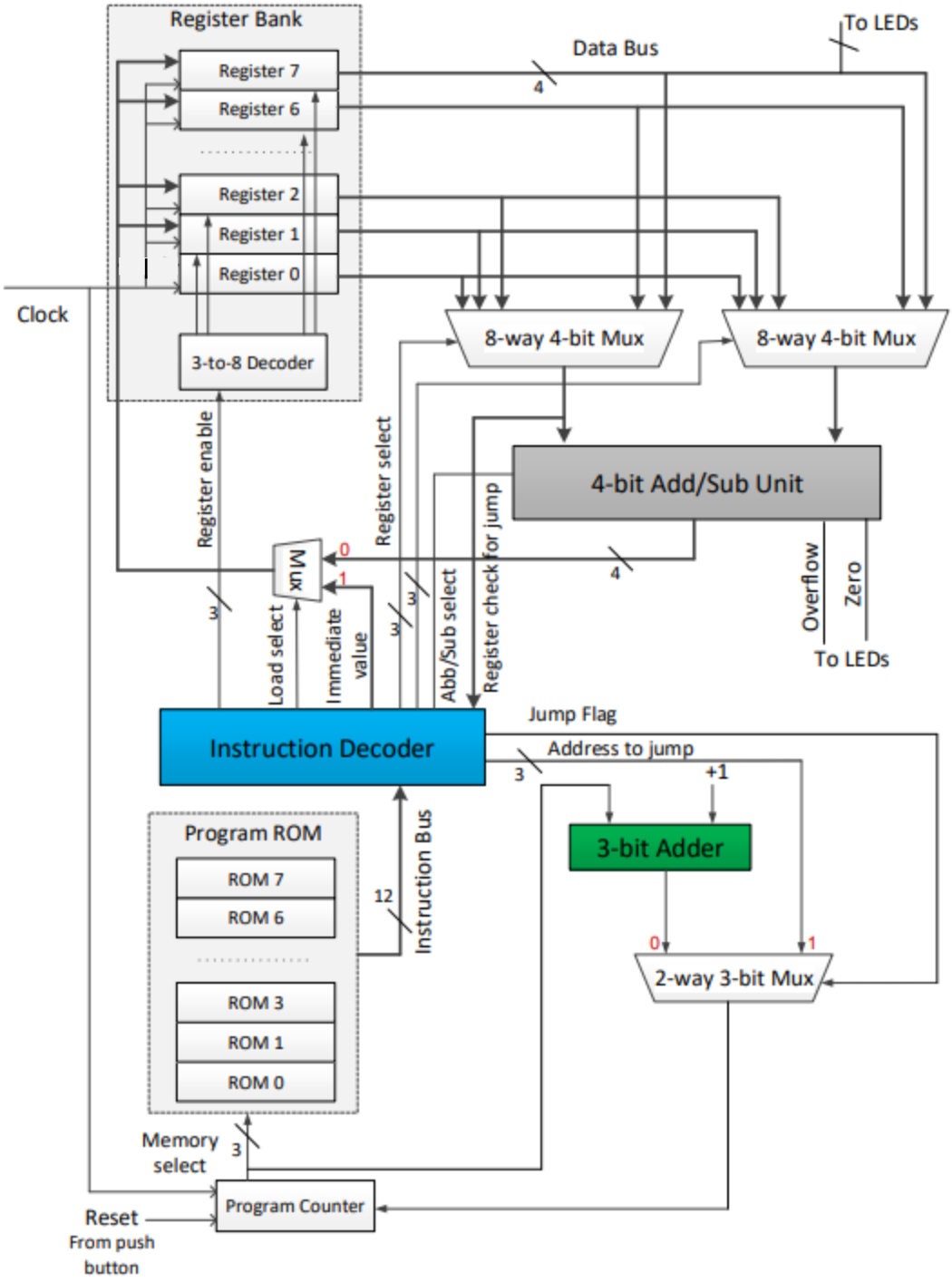


Figure 1 – High-level diagram of the nanoprocessor.

1. Nano Processor

- Design source VHDL code of Nano processor.

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

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

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

entity NANO_PROCESSOR is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0));
end NANO_PROCESSOR;

architecture Behavioral of NANO_PROCESSOR is

component REGISTER_BANK
    Port (Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          MUX_Out : in STD_LOGIC_VECTOR (3 downto 0);
          Reset : in STD_LOGIC;
          R0_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R1_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R2_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R3_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R4_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R5_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R6_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R7_Out : out STD_LOGIC_VECTOR (3 downto 0)
          );
end component;

component ADDER_SUBTRACTOR
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           M : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component ADDER_3bit
```

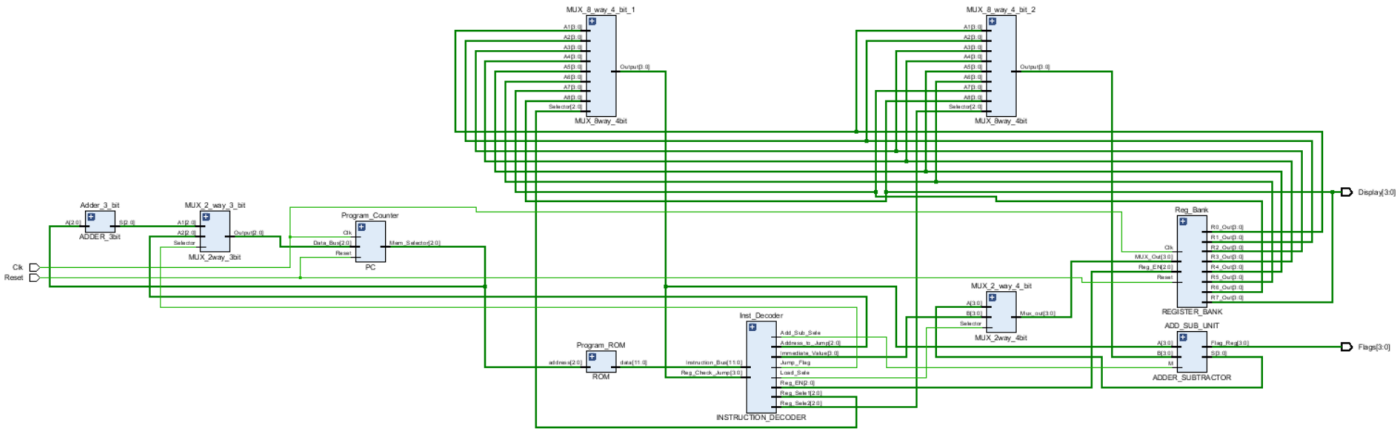
```
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC);
end component;

component PC
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Data_Bus: in STD_LOGIC_VECTOR (2 downto 0);
           Mem_Selector : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component INSTRUCTION_DECODER
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Add_Sub_Sele : out STD_LOGIC;
           Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Load_Sele : out STD_LOGIC;
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Jump_Flag : out STD_LOGIC;
           Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component ROM
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component MUX_2way_3bit
    Port ( A1 : in STD_LOGIC_VECTOR (2 downto 0);
           A2 : in STD_LOGIC_VECTOR (2 downto 0);
           Selector : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component MUX_2way_4bit
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC;
           Mux_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component MUX_8way_4bit
    Port ( A1 : in STD_LOGIC_VECTOR (3 downto 0);
           A2 : in STD_LOGIC_VECTOR (3 downto 0);
           A3 : in STD_LOGIC_VECTOR (3 downto 0);
           A4 : in STD_LOGIC_VECTOR (3 downto 0);
           A5 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            A6 : in STD_LOGIC_VECTOR (3 downto 0);
            A7 : in STD_LOGIC_VECTOR (3 downto 0);
            A8 : in STD_LOGIC_VECTOR (3 downto 0);
            Selector : in STD_LOGIC_VECTOR (2 downto 0);
            Output : out STD_LOGIC_VECTOR (3 downto 0));
end component;


signal ADD_SUB_sele : std_logic;
signal Load_sele, Jump_Flag : std_logic;

signal Mem_Selector, Output_2_3bit_MUX ,Address_to_Jump, Output_3bit_Adder :
std_logic_vector(2 downto 0);
signal Selector_8_4bit_MUX_1, Selector_8_4bit_MUX_2, Reg_EN: std_logic_vector(2 downto
0);

signal Instruction_Bus: std_logic_vector(11 downto 0);

signal Output_8_4bit_MUX_1,Output_8_4bit_MUX_2, Immediate_Value ,Output_2_4bit_MUX :
std_logic_vector(3 downto 0);
signal R0_Out,R1_Out,R2_Out,R3_Out,R4_Out,R5_Out,R6_Out,R7_Out : std_logic_vector (3
downto 0);
signal Output_Add_Sub, Flag_Reg : std_logic_vector (3 downto 0);
begin

    Program_Counter : PC
        Port map (
            Clk => Clk,
            Reset => Reset,
            Data_Bus => Output_2_3bit_MUX,
            Mem_Selector => Mem_Selector);

     Program_ROM : ROM
         Port map (
            address => Mem_Selector,
            data => Instruction_Bus);

     MUX_2_way_3_bit : MUX_2way_3bit
         Port map (
            A1 => Output_3bit_Adder,
            A2 => Address_to_Jump,
            Selector => Jump_Flag,
            Output => Output_2_3bit_MUX);

    Inst_Decoder : INSTRUCTION_DECODER
        Port map (
            Instruction_Bus  => Instruction_Bus,
            Reg_Check_Jump => Output_8_4bit_MUX_1,
            Add_Sub_Sele => ADD_SUB_sele,
            Reg_Sele1 => Selector_8_4bit_MUX_1,
```

```
        Reg_Sele2 => Selector_8_4bit_MUX_2,
        Immediate_Value => Immediate_Value,
        Load_Sele => Load_sele,
        Reg_EN => Reg_EN,
        Jump_Flag => Jump_Flag,
        Address_to_Jump => Address_to_Jump);

Reg_Bank : REGISTER_BANK
    Port map (
        Reg_EN => Reg_EN,
        Clk => Clk,
        MUX_Out => Output_2_4bit_MUX,
        Reset => Reset,
        R0_Out => R0_Out,
        R1_Out => R1_Out,
        R2_Out => R2_Out,
        R3_Out => R3_Out,
        R4_Out => R4_Out,
        R5_Out => R5_Out,
        R6_Out => R6_Out,
        R7_Out => R7_Out
        );

MUX_2_way_4_bit : MUX_2way_4bit
    Port map (
        A => Output_Add_Sub,
        B => Immediate_Value,
        Selector => Load_sele,
        Mux_out => Output_2_4bit_MUX );

 MUX_8_way_4_bit_1 : MUX_8way_4bit
    Port map (
        A1=>R0_Out,
        A2=>R1_Out,
        A3=>R2_Out,
        A4=>R3_Out,
        A5=>R4_Out,
        A6=>R5_Out,
        A7=>R6_Out,
        A8=>R7_Out,
        Selector => Selector_8_4bit_MUX_1,
        Output => Output_8_4bit_MUX_1 );

 MUX_8_way_4_bit_2 : MUX_8way_4bit
    Port map (
        A1=>R0_Out,
        A2=>R1_Out,
        A3=>R2_Out,
        A4=>R3_Out,
        A5=>R4_Out,
```

```
            A6=>R5_Out,
            A7=>R6_Out,
            A8=>R7_Out,
            Selector => Selector_8_4bit_MUX_2,
            Output => Output_8_4bit_MUX_2);


    ADD_SUB_UNIT : ADDER_SUBTRACTOR
       Port map (
            A => Output_8_4bit_MUX_1,
            B => Output_8_4bit_MUX_2,
            M => ADD_SUB_sele,
            S => Output_Add_Sub,
            Flag_Reg => Flag_Reg);

    Adder_3_bit : ADDER_3bit
       Port map (
            A => Mem_Selector,
            S => Output_3bit_Adder );

Display <= R7_Out;
Flags <= Flag_Reg;

end Behavioral;
```

- RTL Schematic diagram of the Nano processor.

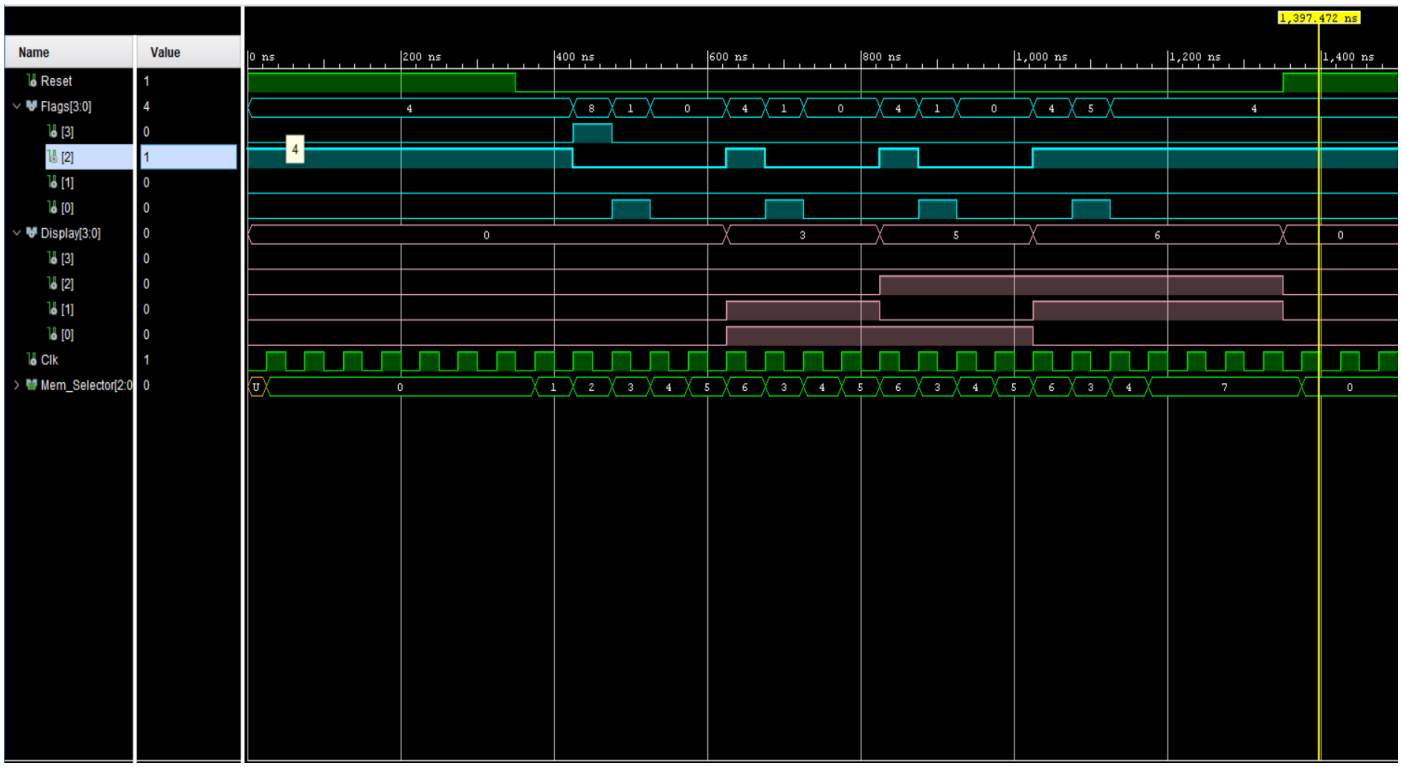


- Behavioral simulation Code for Nano processor

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

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

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

entity TB_NANO_PROCESSOR is
--  Port ( );
end TB_NANO_PROCESSOR;
architecture Behavioral of TB_NANO_PROCESSOR is

component NANO_PROCESSOR
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0));
end component;

signal Reset : std_logic;
signal Flags : std_logic_vector (3 downto 0);
signal Display : std_logic_vector (3 downto 0);
signal Clk : std_logic := '1';
begin
    UUT : NANO_PROCESSOR
        port map (
            Clk => Clk,
            Reset => Reset,
            Display => Display,
            Flags => Flags);

    process begin
        Clk <= not Clk;
        wait for 25 ns;
    end process;

    process begin
        Reset <= '1';
        wait for 350 ns;

        Reset <= '0';
        wait for 1000 ns;
```

```
        Reset <= '1';
        wait;

    end process;
end Behavioral;
```

- Timing Diagram for Nano processor:



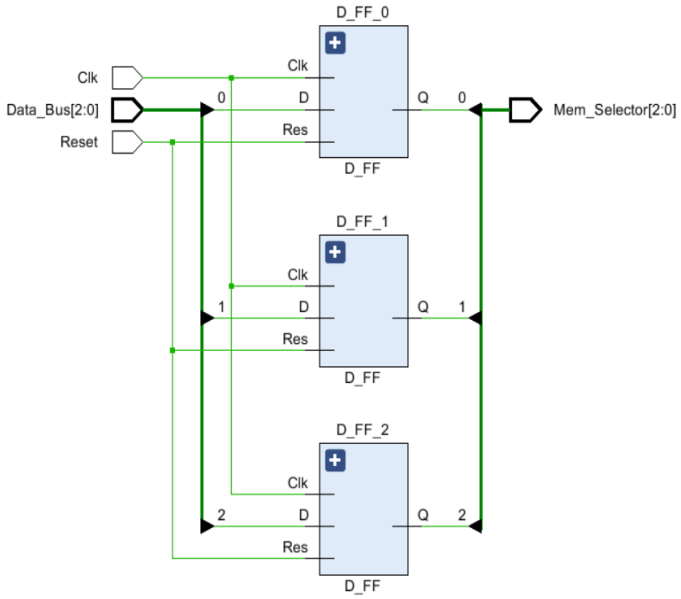
2. Program Counter

- Design source VHDL code of Program counter:

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

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

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

entity PC is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Data_Bus: in STD_LOGIC_VECTOR (2 downto 0);
           Mem_Selector : out STD_LOGIC_VECTOR (2 downto 0));
end PC;

architecture Behavioral of PC 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 => Data_Bus(0),
            Res => Reset,
            Clk => Clk,
            Q => Mem_Selector(0)
        );

    D_FF_1 : D_FF
        port map(
            D => Data_Bus(1),
            Res => Reset,
            Clk => Clk,
            Q => Mem_Selector(1)
        );
```

```
    D_FF_2 : D_FF
            port map(
            D => Data_Bus(2),
            Res => Reset,
            Clk => Clk,
            Q => Mem_Selector(2)
        );

end Behavioral;
```

- RTL Schematic diagram of the Program counter:



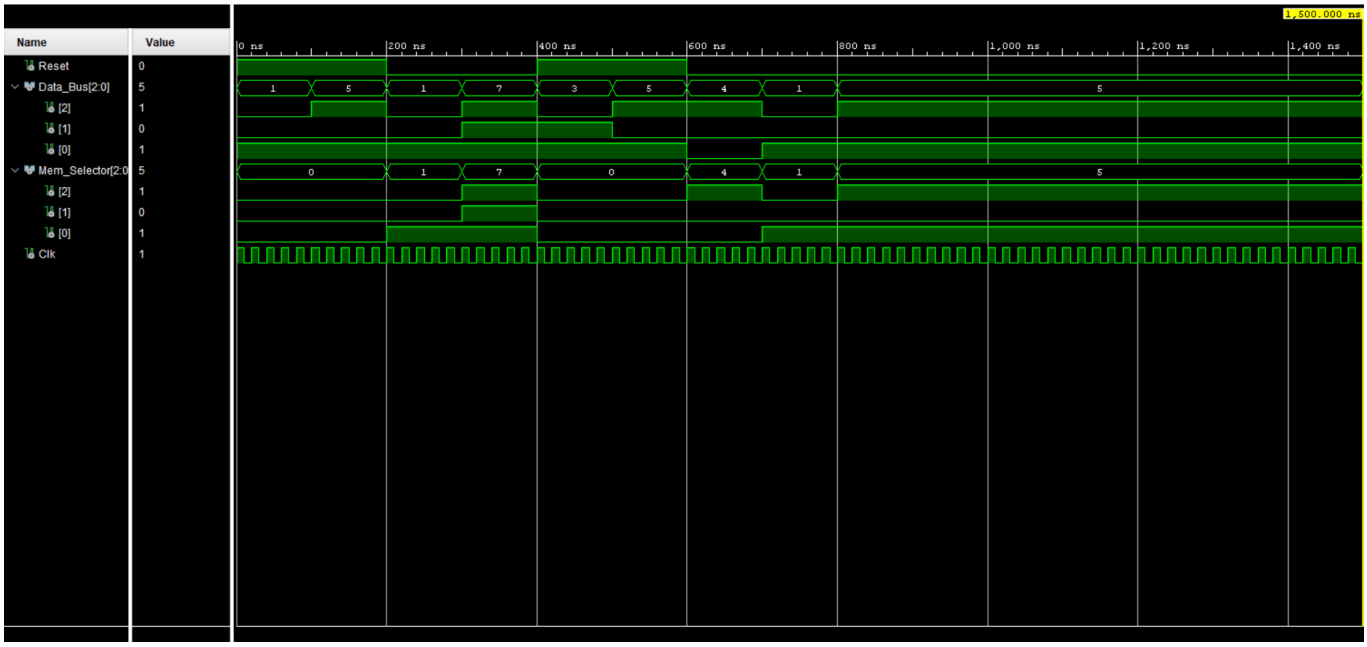
- Behavioral simulation Code for Program counter:

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

architecture Behavioral of TB_PC is

component PC
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Data_Bus: in STD_LOGIC_VECTOR (2 downto 0);
           Mem_Selector : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal  Reset : std_logic;
signal Data_Bus, Mem_Selector : std_logic_vector (2 downto 0);
signal Clk : std_logic := '0';
begin
UUT : PC port map (
        Clk => Clk,
        Reset => Reset,
        Data_Bus => Data_Bus,
        Mem_Selector => Mem_Selector
      );

process begin
    Clk <= not Clk;
    wait for 10 ns;
end process;

process begin
    Reset <= '1';

    Data_Bus <= "001";
    wait for 100 ns;

    Data_Bus <= "101";
    wait for 100 ns;
-----------------------------------------------------
    Reset <= '0';

    Data_Bus <= "001";
    wait for 100 ns;

    Data_Bus <= "111";
    wait for 100 ns;
-----------------------------------------------------
    Reset <= '1';

    Data_Bus <= "011";
    wait for 100 ns;
```

```
    Data_Bus <= "101";
    wait for 100 ns;
-----------------------------------------------------
    Reset <= '0';

    Data_Bus <= "100";
    wait for 100 ns;

    Data_Bus <= "001";
    wait for 100 ns;

    Data_Bus <= "101";
    wait ;
end process;
end Behavioral;
```

- Timing Diagram for Program counter

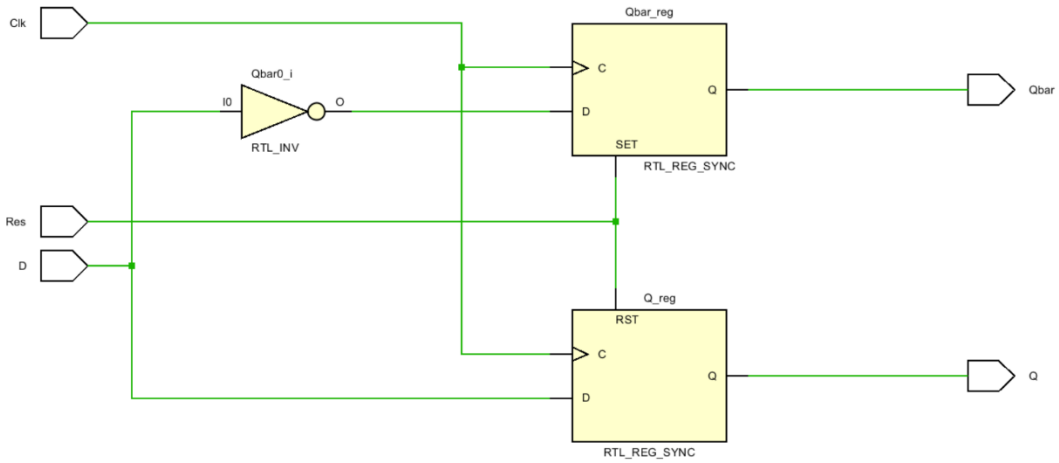


3. D- Flip Flop

- Design source VHDL code of D-flip flop.

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

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

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

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

architecture Behavioral of D_FF is

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

end Behavioral;
```

- RTL Schematic diagram of the D-flip flop.

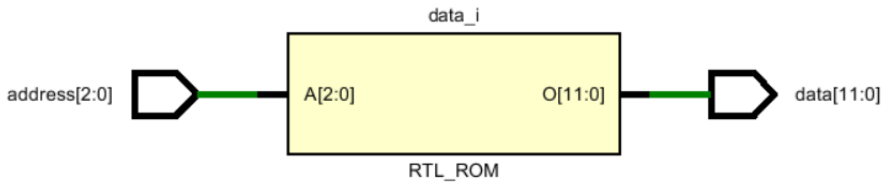


4. Program Rom

- Design source VHDL code of Program rom :

```
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 ROM is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end ROM;

architecture Behavioral of ROM is

type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);
signal prosseser_ROM : rom_type := (

    "100010000100", -- MOVI R1,4        R1 <- 4         0
    "100100000001", -- MOVI R2,1        R2 <- 1         1
    "010100000000", -- NEG  R2          R2 <- -1        2
    "000010100000", -- ADD  R1,R2       R1 <- R2 + R1   3
    "110010000111", -- JZR  R1,7                        4
    "001110010000", -- ADD  R7,R1       R7 <- R7 + R1   5
    "110000000011", -- JZR  R0,3                        6
    "110000000111" -- JZR  R0,7                         7

    );

begin

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

end Behavioral;
```

- RTL Schematic diagram of the Program rom.

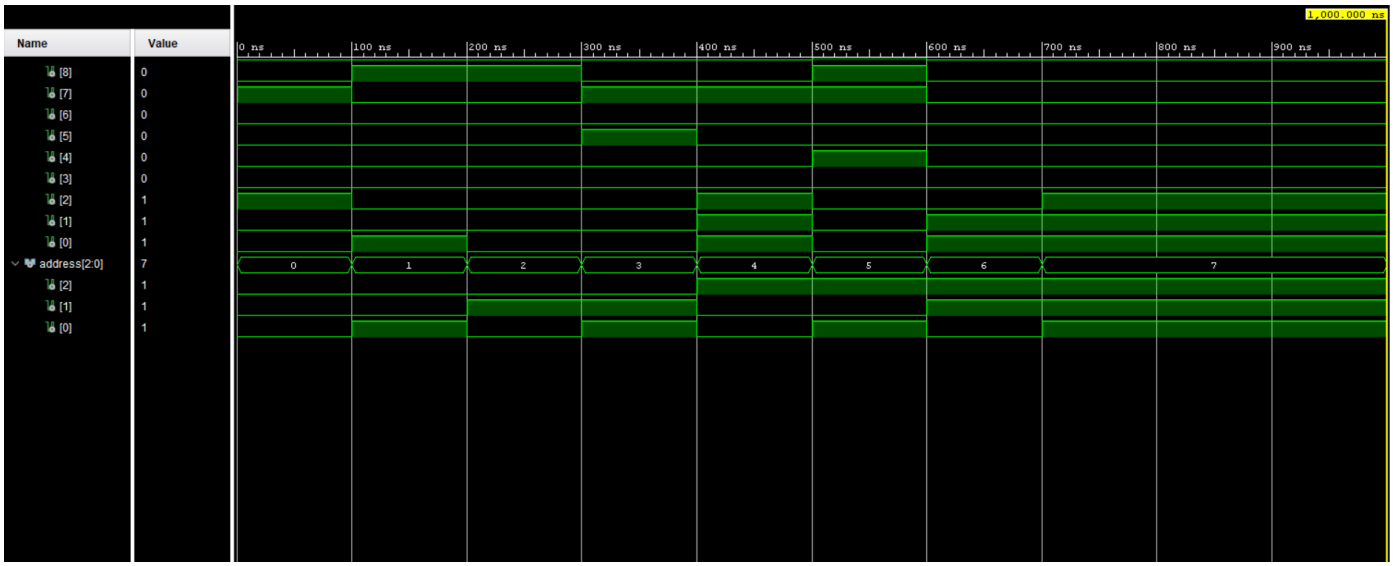


- Behavioral simulation Code for Program rom:

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

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

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

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

architecture Behavioral of TB_ROM is

component ROM
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end component;

signal data:std_logic_vector(11 downto 0);
signal address:std_logic_vector(2 downto 0);

begin
    UUT: ROM
        PORT MAP(
        address=>address,
        data=>data
    );
```

```
process begin

    address<="000";
    wait for 100ns;

    address<="001";
    wait for 100ns;

    address<="010";
    wait for 100ns;

    address<="011";
    wait for 100ns;

    address<="100";
    wait for 100ns;

    address<="101";
    wait for 100ns;

    address<="110";
    wait for 100ns;

    address<="111";
    wait ;

end process;
end Behavioral;
```

- Timing Diagram for Program rom:



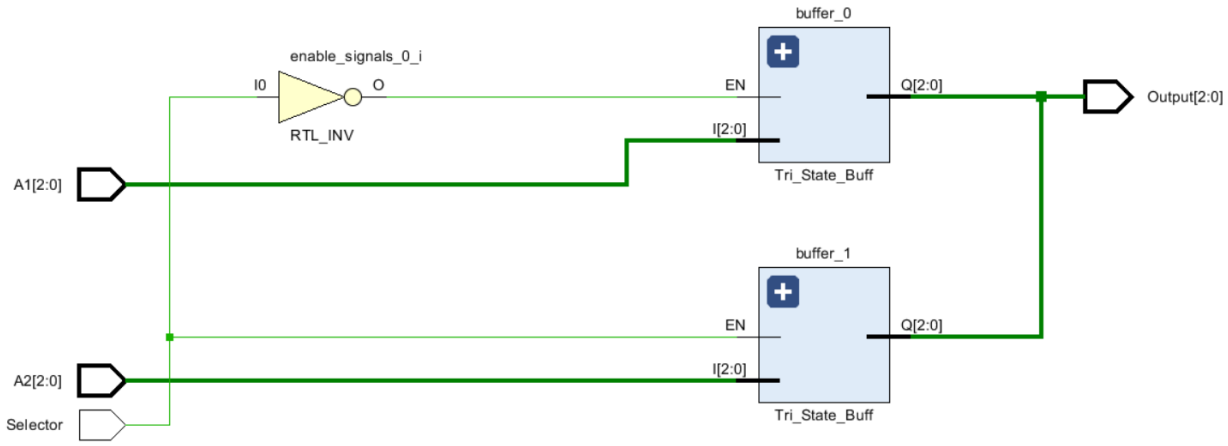
5. MUX_2way_3bit.

- Design source VHDL code of MUX_2way_3bit:

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

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

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

entity MUX_2way_3bit is
    Port ( A1 : in STD_LOGIC_VECTOR (2 downto 0);
           A2 : in STD_LOGIC_VECTOR (2 downto 0);
           Selector : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
end MUX_2way_3bit;

architecture Behavioral of MUX_2way_3bit is

component Tri_State_Buff is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal enable_signals : STD_LOGIC_VECTOR(1 downto 0) := "01" ;

begin

     enable_signals(0) <= NOT Selector;
     enable_signals(1) <= Selector;


     buffer_0: Tri_State_Buff
           port map (
               I => A1,
               EN => enable_signals(0),
               Q => Output
           );
     buffer_1: Tri_State_Buff
           port map (
               I => A2,
               EN => enable_signals(1),
               Q => Output
```

```
            );

end Behavioral;
```

- RTL Schematic diagram of the MUX_2way_3bit.



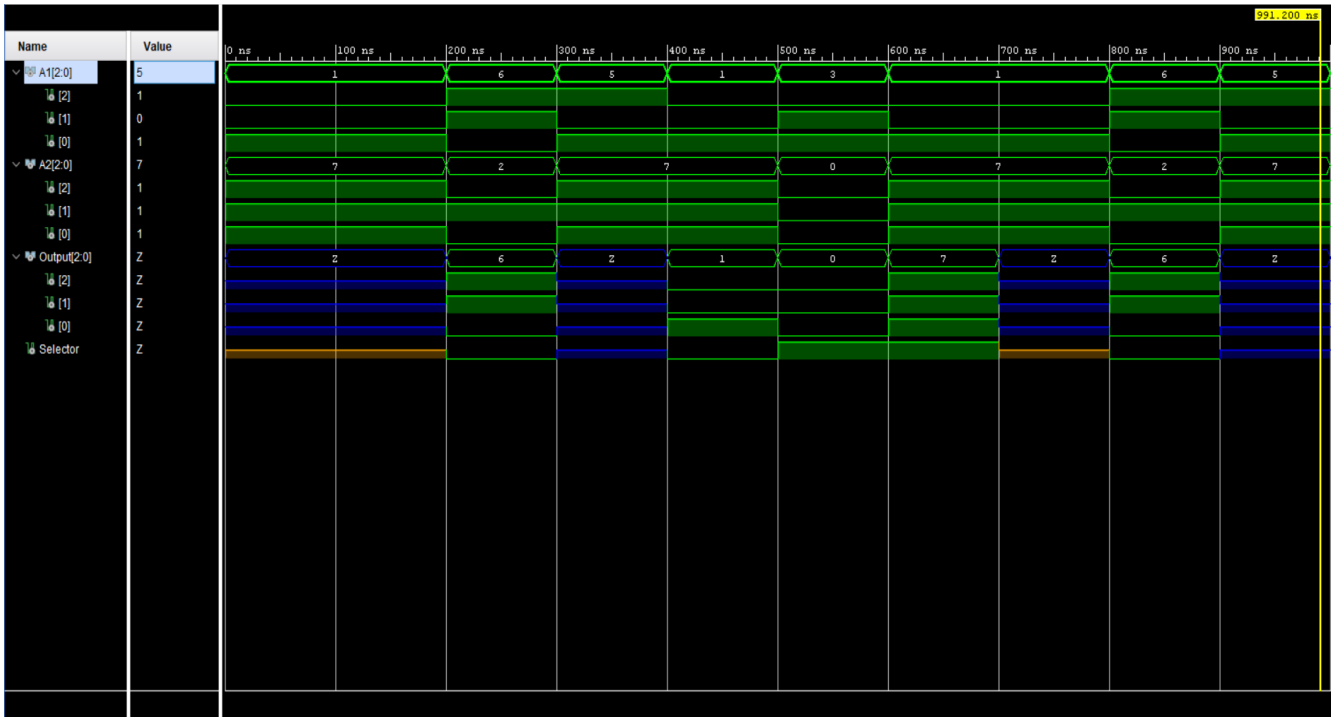
- Behavioral simulation Code for MUX_2way_3bit:

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

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

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

entity TB_MUX_2way_3bit is
--  Port ( );
end TB_MUX_2way_3bit;

architecture Behavioral of TB_MUX_2way_3bit is
```

```
component MUX_2way_3bit
    Port ( A1 : in STD_LOGIC_VECTOR (2 downto 0);
           A2 : in STD_LOGIC_VECTOR (2 downto 0);
           Selector : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal A1, A2, Output : std_logic_vector (2 downto 0);
signal Selector : std_logic;

begin
UUT : MUX_2way_3bit port map(
        A1 => A1,
        A2 => A2,
        Selector => Selector,
        Output => Output
     );

process begin
    A1 <= "001";
    A2 <= "111";
--    Selector <= '0';
    wait for 100 ns;

    A1 <= "001";
    A2 <= "111";
    Selector <= 'U';
    wait for 100 ns;

    A1 <= "110";
    A2 <= "010";
    Selector <= '0';
    wait for 100 ns;

    A1 <= "101";
    A2 <= "111";
    Selector <= 'Z';
    wait for 100 ns;

    A1 <= "001";
    A2 <= "111";
    Selector <= '0';
    wait for 100 ns;

    A1 <= "011";
    A2 <= "000";
    Selector <= '1';
    wait for 100 ns;

end process;
```

```
end Behavioral;
```

- Timing Diagram for MUX_2way_3bit.

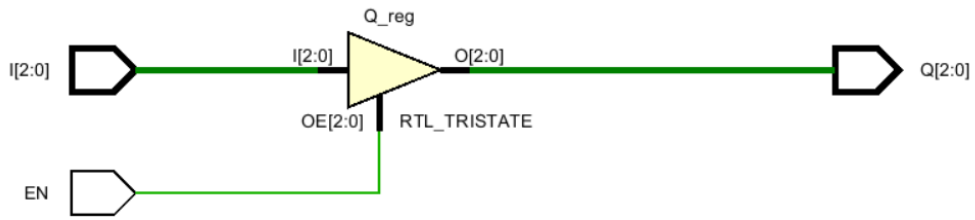


6. Tri_State_Buffer

- Design source VHDL code of Tri_State_Buffer.

```
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 Tri_State_Buff is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end Tri_State_Buff;

architecture Behavioral of Tri_State_Buff is

begin

      process (I,EN)
      begin
        if EN = '1' then
            Q <= I;
        else
            Q <= "ZZZ";
        end if;
      end process;

end Behavioral;
```

- RTL Schematic diagram of the Tri_State_Buffer:

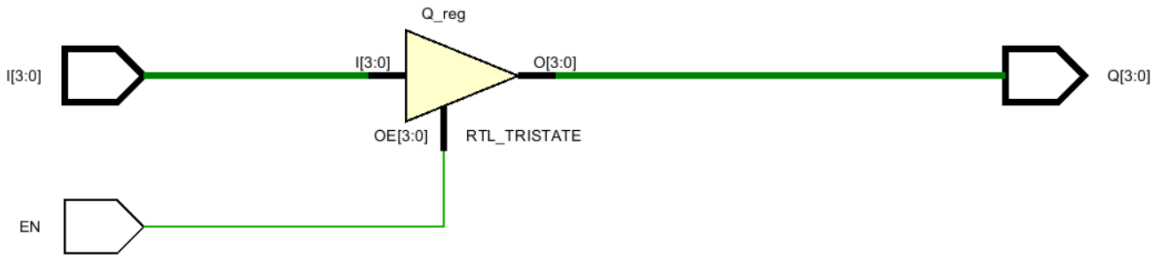


7. Tri_State_Buffer_4bit

- Design source VHDL code of Tri_State_Buffer_4bit:

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

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

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

entity Tri_State_Buffer_4bit is
    Port ( I : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Tri_State_Buffer_4bit;

architecture Behavioral of Tri_State_Buffer_4bit is

begin
  process (I,EN) begin
      if EN = '1' then
          Q <= I;
      else
          Q <= "ZZZZ";
      end if;
    end process;

end Behavioral;
```

- RTL Schematic diagram of the Tri_State_Buffer_4bit:



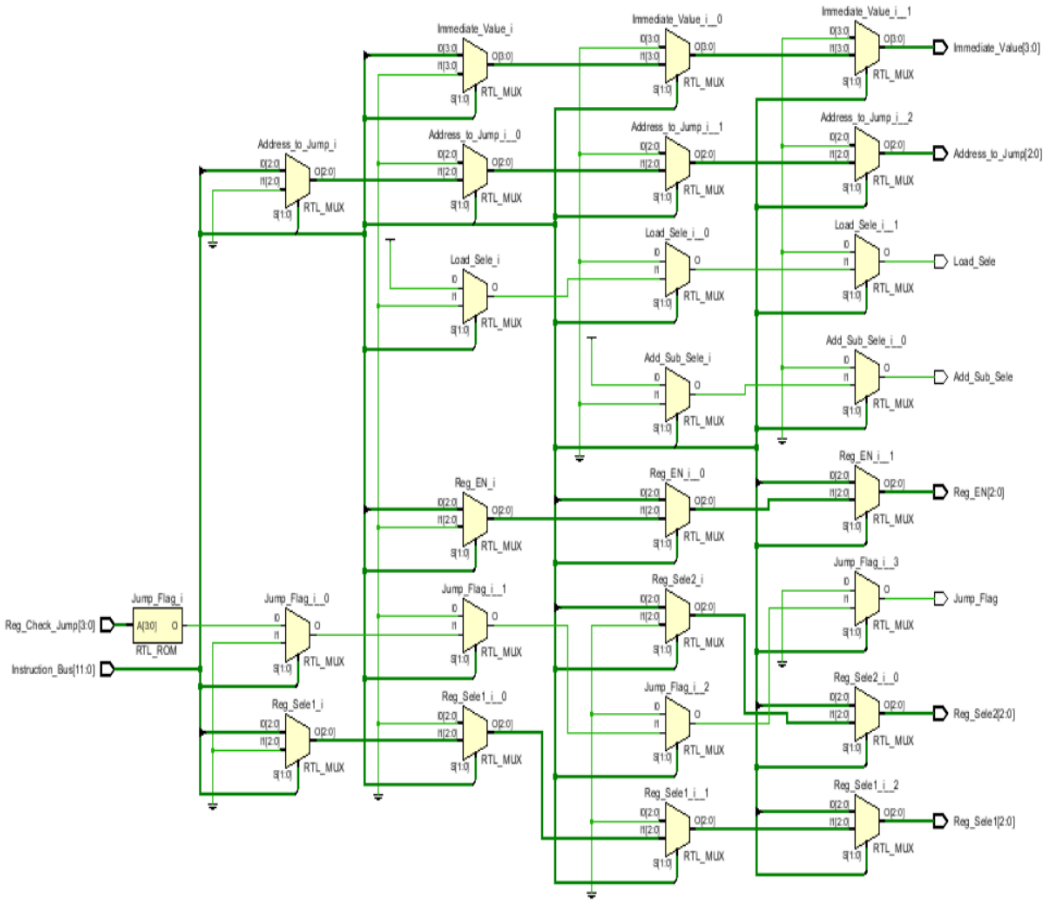
8. INSTRUCTION_DECODER

- Design source VHDL code of INSTRUCTION_DECODER:

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

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity INSTRUCTION_DECODER is
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Add_Sub_Sele : out STD_LOGIC;
           Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Load_Sele : out STD_LOGIC;
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Jump_Flag : out STD_LOGIC;
           Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end INSTRUCTION_DECODER;

architecture Behavioral of INSTRUCTION_DECODER is

signal Operator : std_logic_vector (1 downto 0);

begin
    Operator <= Instruction_Bus(11 downto 10);

    process (Operator, Instruction_Bus, Reg_Check_Jump) begin

        Add_Sub_Sele <= '0';
        Jump_Flag <= '0';
        Load_Sele <= '0';
        Reg_Sele1 <= "000";
        Reg_Sele2 <= "000";
        Immediate_Value <= "0000";
        Reg_EN <= "000";
        Address_to_Jump <= "000";

        if Operator = "00" then
            Reg_Sele1 <= Instruction_Bus(9 downto 7);
            Reg_Sele2 <= Instruction_Bus(6 downto 4);
            Reg_EN <= Instruction_Bus(9 downto 7);
```

```
        elsif Operator = "01" then

            Reg_EN <= Instruction_Bus(9 downto 7);
            Reg_Sele2 <= Instruction_Bus(9 downto 7);
            Add_Sub_Sele <= '1';

        elsif Operator = "10" then

            Immediate_Value <= Instruction_Bus(3 downto 0);
            Reg_EN <= Instruction_Bus(9 downto 7);
            Load_Sele <= '1';

        elsif Operator = "11" then
            Address_to_Jump <= Instruction_Bus(2 downto 0);
            Reg_Sele1 <= Instruction_Bus(9 downto 7);
            if Reg_Check_Jump = "0000" then
                Jump_Flag <= '1';
            end if;
        end if;
    end process;
end Behavioral;
```

- RTL Schematic diagram of the INSTRUCTION_DECODER.

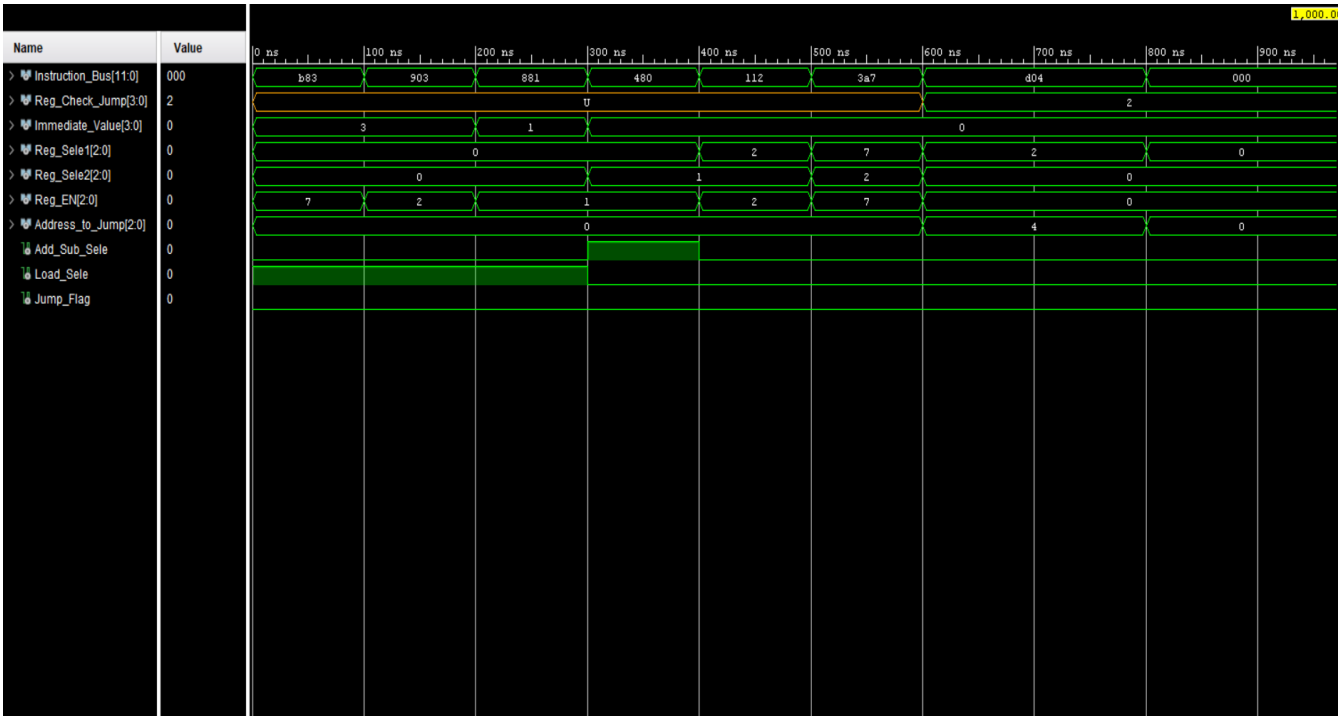


- Behavioral simulation Code for INSTRUCTION_DECODER:

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

architecture Behavioral of TB_INSTRUCTION_DECODER is

component INSTRUCTION_DECODER
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Add_Sub_Sele : out STD_LOGIC;
           Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Load_Sele : out STD_LOGIC;
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Jump_Flag : out STD_LOGIC;
           Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal Instruction_Bus : STD_LOGIC_VECTOR (11 downto 0);
signal Reg_Check_Jump, Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Reg_Sele1, Reg_Sele2, Reg_EN, Address_to_Jump : STD_LOGIC_VECTOR (2 downto 0);
signal Add_Sub_Sele, Load_Sele, Jump_Flag : STD_LOGIC;

begin
UUT : INSTRUCTION_DECODER port map(
        Instruction_Bus => Instruction_Bus,
        Reg_Check_Jump => Reg_Check_Jump,
        Add_Sub_Sele => Add_Sub_Sele,
        Reg_Sele1 => Reg_Sele1,
        Reg_Sele2 => Reg_Sele2,
        Immediate_Value => Immediate_Value,
        Load_Sele => Load_Sele,
        Reg_EN => Reg_EN,
        Jump_Flag => Jump_Flag,
        Address_to_Jump => Address_to_Jump
```

```
    );


stimulus_process: process
    begin

    Instruction_Bus <= "101110000011";
    wait for 100 ns;

    Instruction_Bus <= "100100000011";
    wait for 100 ns;

    Instruction_Bus <= "100010000001";
    wait for 100 ns;

    Instruction_Bus <= "010010000000";
    wait for 100 ns;

    Instruction_Bus <= "000100010010";
    wait for 100 ns;

    Instruction_Bus <= "001110100111";
    wait for 100 ns;

    Instruction_Bus <= "110100000100";
    Reg_Check_Jump <= "0010";
    wait for 100 ns;

    Reg_Check_Jump <= "0010";
    wait for 100 ns;

    Instruction_Bus <= "000000000000";
    wait;

    end process stimulus_process;

    end Behavioral;
```

- Timing Diagram for INSTRUCTION_DECODER.



9. Register Bank.

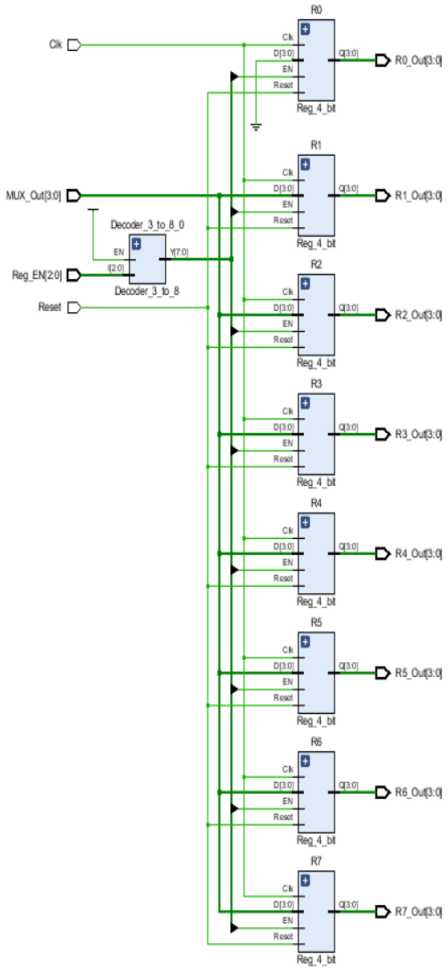
- Design source VHDL code of Register Bank.

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

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

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

entity REGISTER_BANK is
    Port (Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          MUX_Out : in STD_LOGIC_VECTOR (3 downto 0);
          Reset : in STD_LOGIC;
          R0_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R1_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R2_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R3_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R4_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R5_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R6_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R7_Out : out STD_LOGIC_VECTOR (3 downto 0)
          );
end REGISTER_BANK;

architecture Behavioral of Register_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_4_bit is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL Select_Reg : STD_LOGIC_VECTOR (7 downto 0);
```

```
begin

    Decoder_3_to_8_0 : Decoder_3_to_8
          Port map( I=>Reg_EN,
                    EN=>'1',
                    Y=>Select_Reg );

    R0 : Reg_4_bit
          Port map ( D=>"0000",
                EN=>Select_Reg(0),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R0_Out );

    R1 : Reg_4_bit
         Port map ( D=>MUX_Out,
                EN=>Select_Reg(1),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R1_Out );

    R2 : Reg_4_bit
         Port map ( D=>MUX_Out,
                EN=>Select_Reg(2),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R2_Out );

    R3 : Reg_4_bit
         Port map ( D=>MUX_Out,
                EN=>Select_Reg(3),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R3_Out );

    R4 : Reg_4_bit
         Port map ( D=>MUX_Out,
                EN=>Select_Reg(4),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R4_Out );

    R5 : Reg_4_bit
         Port map ( D=>MUX_Out,
                EN=>Select_Reg(5),
                Clk=>Clk,
                Reset=>Reset,
                Q=> R5_Out );

    R6 : Reg_4_bit
```

```
        Port map ( D=>MUX_Out,
                   EN=>Select_Reg(6),
                   Clk=>Clk,
                   Reset=>Reset,
                   Q=> R6_Out );

    R7 : Reg_4_bit
        Port map ( D=>MUX_Out,
                   EN=>Select_Reg(7),
                   Clk=>Clk,
                   Reset=>Reset,
                   Q=> R7_Out );

end Behavioral;
```

- RTL Schematic diagram of the Register Bank:

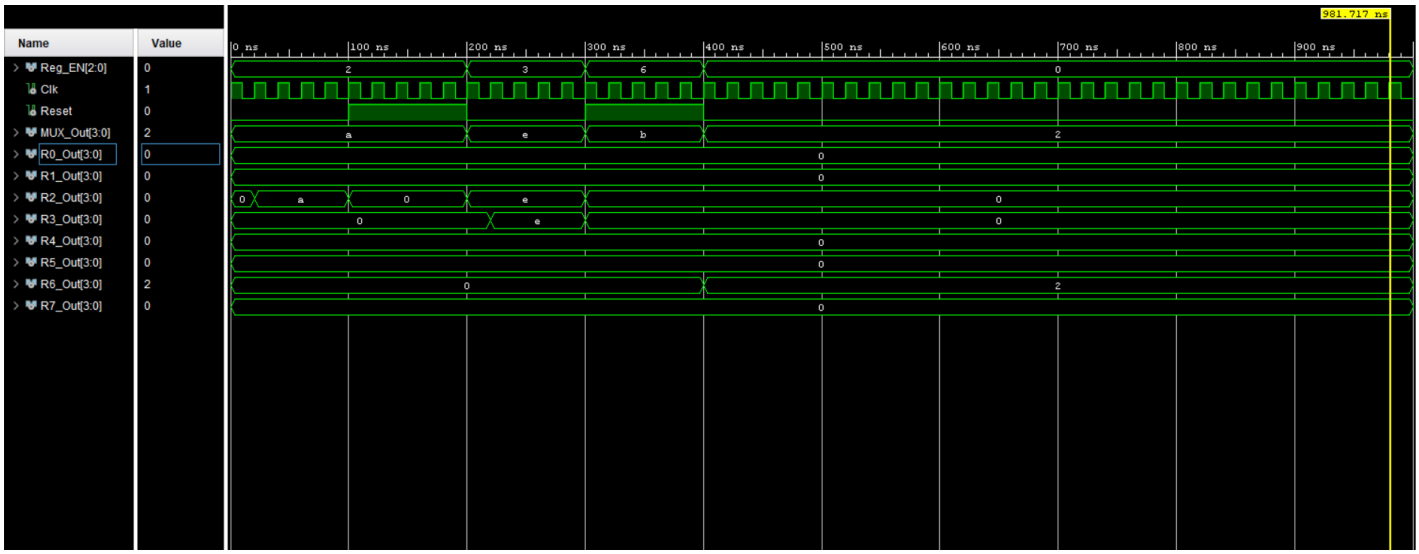


- Behavioral simulation Code for Register Bank:

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

architecture Behavioral of TB_REGISTER_BANK is

component REGISTER_BANK is
    Port (Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          MUX_Out : in STD_LOGIC_VECTOR (3 downto 0);
          Reset : in STD_LOGIC;
          R0_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R1_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R2_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R3_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R4_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R5_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R6_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R7_Out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL Reg_EN : STD_LOGIC_VECTOR (2 downto 0);
SIGNAL Clk,Reset :  STD_LOGIC;
SIGNAL MUX_Out,R0_Out,R1_Out,R2_Out,R3_Out,R4_Out,R5_Out,R6_Out,R7_Out :
STD_LOGIC_VECTOR (3 downto 0);

begin

UUT : REGISTER_BANK
    Port map(Reg_EN => Reg_EN,
          Clk => Clk,
          MUX_Out => MUX_Out,
          Reset => Reset,
          R0_Out => R0_Out,
          R1_Out => R1_Out,
```

```
            R2_Out => R2_Out,
            R3_Out => R3_Out,
            R4_Out => R4_Out,
            R5_Out => R5_Out,
            R6_Out => R6_Out,
            R7_Out => R7_Out);

process begin
    Clk <= '1';
    wait for 10 ns;
    Clk <='0';
    wait for 10 ns;
end process;

process begin
    Reg_EN <= "010";
    Reset <= '0';
    MUX_Out <= "1010";
    wait for 100ns;

    Reg_EN <= "010";
    Reset <= '1';
    MUX_Out <= "1010";
    wait for 100ns;

    Reg_EN <= "011";
    Reset <= '0';
    MUX_Out <= "1110";
    wait for 100ns;

    Reg_EN <= "110";
    Reset <= '1';
    MUX_Out <= "1011";
    wait for 100ns;

    Reg_EN <= "000";
    Reset <= '0';
    MUX_Out <= "0010";
    wait for 100ns;

    wait;
end process;

end Behavioral;
```

- Timing Diagram for Register Bank:



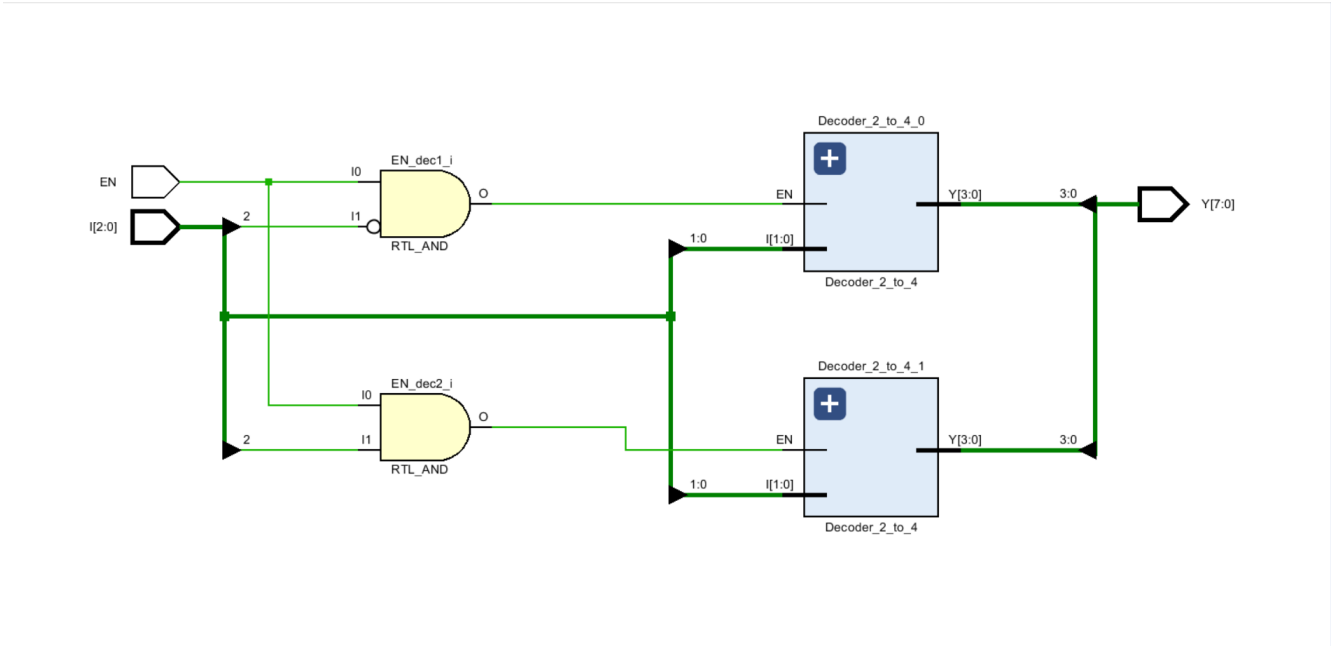
10. Decoder_3_to_8

- Design source VHDL code of Decoder_3_to_8 :

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

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

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

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

architecture Behavioral of Decoder_3_to_8 is
component Decoder_2_to_4
   Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
          EN : in STD_LOGIC;
          Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL input_vec : std_logic_vector (1 downto 0);
SIGNAL out_vec_1,out_vec_2 : std_logic_vector (3 downto 0);
SIGNAL EN_dec1, EN_dec2 : std_logic;

begin
    EN_dec1 <= EN and not(I(2));
    EN_dec2 <= EN and I(2);
    input_vec<= I(1 downto 0);



    Decoder_2_to_4_0: Decoder_2_to_4
        port map(I=>input_vec,
                 EN=>EN_dec1,
                 Y => out_vec_1);
    Decoder_2_to_4_1: Decoder_2_to_4
        port map(I=>input_vec,
                 EN=>EN_dec2,
                 Y => out_vec_2);

    Y(3 downto 0)<= out_vec_1;
```

```
    Y(7 downto 4)<= out_vec_2;

end Behavioral;
```

- RTL Schematic diagram of the Decoder_3_to_8.

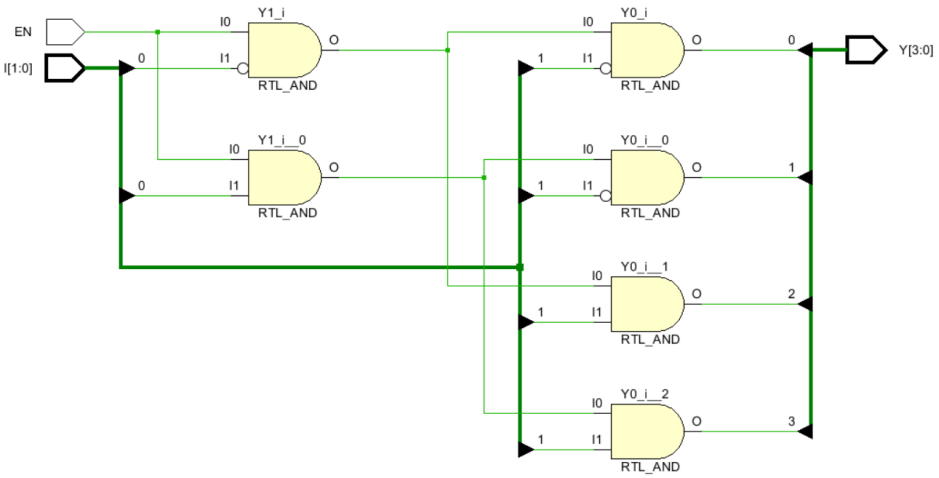


11. Decoder_2_to_4

- Design source VHDL code of Decoder_2_to_4:

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

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

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

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

architecture Behavioral of Decoder_2_to_4 is

begin
    Y(0) <= EN AND (NOT I(0)) AND (NOT I(1));
    Y(1) <= EN AND I(0) AND (NOT I(1));
    Y(2) <= EN AND (NOT I(0)) AND  I(1);
    Y(3) <= EN AND I(0) AND I(1);

end Behavioral;
```

- RTL Schematic diagram of the Decoder_2_to_4.

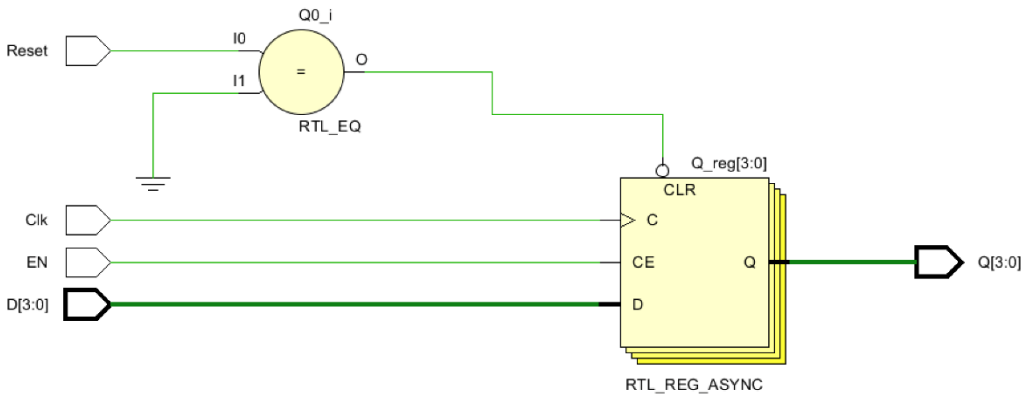


12. Reg_4_bit.

- Design source VHDL code of Reg_4_bit:

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

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

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

entity Reg_4_bit is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Reg_4_bit;

architecture Behavioral of Reg_4_bit is

begin

    process (Clk,EN,Reset) begin
        if Reset = '0' then
            if (rising_edge(Clk)) then
                if EN = '1' then
                   Q <= D;
                end if;
            end if;
        else
            Q <= "0000";
        end if;
    end process;

end Behavioral;
```

- RTL Schematic diagram of the Reg_4_bit.



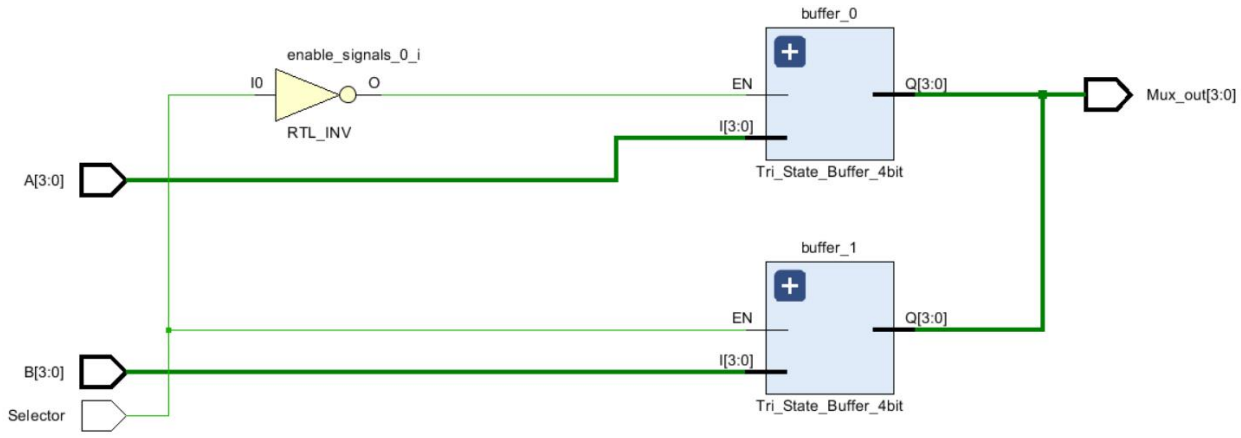
13. MUX_2way_4bit.

- Design source VHDL code of MUX_2way_4bit.

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

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

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

entity MUX_2way_4bit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC;
           Mux_out : out STD_LOGIC_VECTOR (3 downto 0));
end MUX_2way_4bit;

architecture Behavioral of MUX_2way_4bit is

component Tri_State_Buffer_4bit is
    Port ( I : in STD_LOGIC_VECTOR (3 downto 0);
       EN : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal enable_signals : STD_LOGIC_VECTOR(1 downto 0) := "01" ;
begin

     enable_signals(0) <= NOT Selector;
     enable_signals(1) <= Selector;

     buffer_0: Tri_State_Buffer_4bit
           port map (
               I => A,
               EN => enable_signals(0),
               Q => Mux_out
           );

     buffer_1: Tri_State_Buffer_4bit
           port map (
               I => B,
               EN => enable_signals(1),
               Q => Mux_out
           );
```

```
end Behavioral;
```

- RTL Schematic diagram of the MUX_2way_4bit:

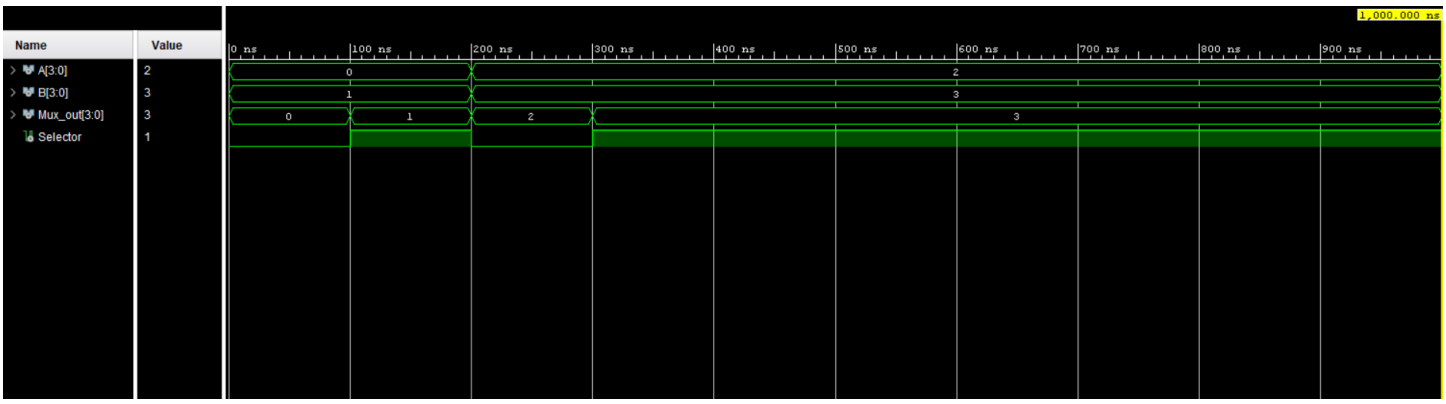


- Behavioral simulation Code for MUX_2way_4bit:

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

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

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

entity TB_MUX_2way_4bit is
--  Port ( );
end TB_MUX_2way_4bit;

architecture Behavioral of TB_MUX_2way_4bit is

component Mux_2way_4bit
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC;
           Mux_out : out STD_LOGIC_VECTOR (3 downto 0));
```

```
end component;

signal A,B,Mux_out:std_logic_vector(3 downto 0);
signal Selector: std_logic;

begin
    UUT:Mux_2way_4bit
        PORT MAP(
        A=>A,
        B=>B,
        Mux_out=>Mux_out,
        Selector=>Selector
    );

process begin

    A<="0000";
    B<="0001";
    Selector<='0';
    wait for 100ns;

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

    A<="0010";
    B<="0011";
    Selector<='0';
    wait for 100ns;

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

    wait;
end process;
end Behavioral;
```

- Timing Diagram for MUX_2way_4bit



14. MUX_8way_4bit.

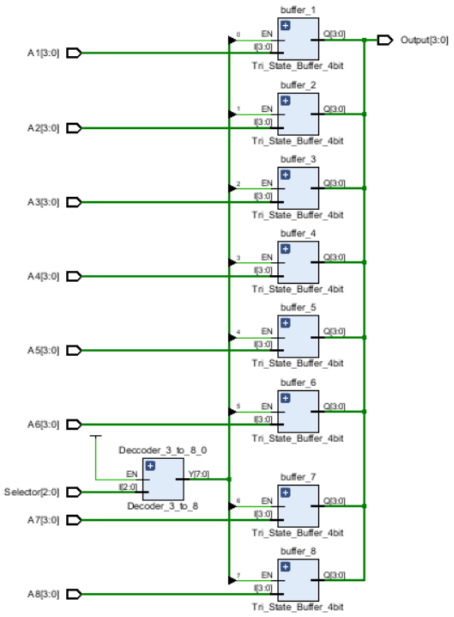
- Design source VHDL code of MUX_8way_4bit:

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

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

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

entity MUX_8way_4bit is
    Port (
        A1 : in STD_LOGIC_VECTOR (3 downto 0);
        A2 : in STD_LOGIC_VECTOR (3 downto 0);
        A3 : in STD_LOGIC_VECTOR (3 downto 0);
        A4 : in STD_LOGIC_VECTOR (3 downto 0);
        A5 : in STD_LOGIC_VECTOR (3 downto 0);
        A6 : in STD_LOGIC_VECTOR (3 downto 0);
        A7 : in STD_LOGIC_VECTOR (3 downto 0);
        A8 : in STD_LOGIC_VECTOR (3 downto 0);
        Selector : in STD_LOGIC_VECTOR (2 downto 0);
        Output : out STD_LOGIC_VECTOR (3 downto 0)
        );
end MUX_8way_4bit;

architecture Behavioral of MUX_8way_4bit is

component Tri_State_Buffer_4bit is
    Port ( I : in STD_LOGIC_VECTOR (3 downto 0);
       EN : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;

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 enable_signal : STD_LOGIC_VECTOR (7 downto 0);

begin

    Deccoder_3_to_8_0 : Decoder_3_to_8
```

```
    port map(
        I => Selector,
        EN => '1',
        Y => enable_signal
);

buffer_1 : Tri_State_Buffer_4bit
    port map(
        I => A1,
        EN => enable_signal(0),
        Q  => Output
);

buffer_2 : Tri_State_Buffer_4bit
    port map(
        I => A2,
        EN => enable_signal(1),
        Q  => Output
);

buffer_3 : Tri_State_Buffer_4bit
    port map(
        I => A3,
        EN => enable_signal(2),
        Q  => Output
);

buffer_4 : Tri_State_Buffer_4bit
    port map(
        I => A4,
        EN => enable_signal(3),
        Q  => Output
);

buffer_5 : Tri_State_Buffer_4bit
    port map(
        I => A5,
        EN => enable_signal(4),
        Q  => Output
);

buffer_6 : Tri_State_Buffer_4bit
    port map(
        I => A6,
        EN => enable_signal(5),
        Q  => Output
);

buffer_7 : Tri_State_Buffer_4bit
    port map(
```

```
            I => A7,
            EN => enable_signal(6),
            Q  => Output
    );

    buffer_8 : Tri_State_Buffer_4bit
        port map(
            I => A8,
            EN => enable_signal(7),
            Q  => Output
    );

end Behavioral;
```

- RTL Schematic diagram of the MUX_8way_4bit:



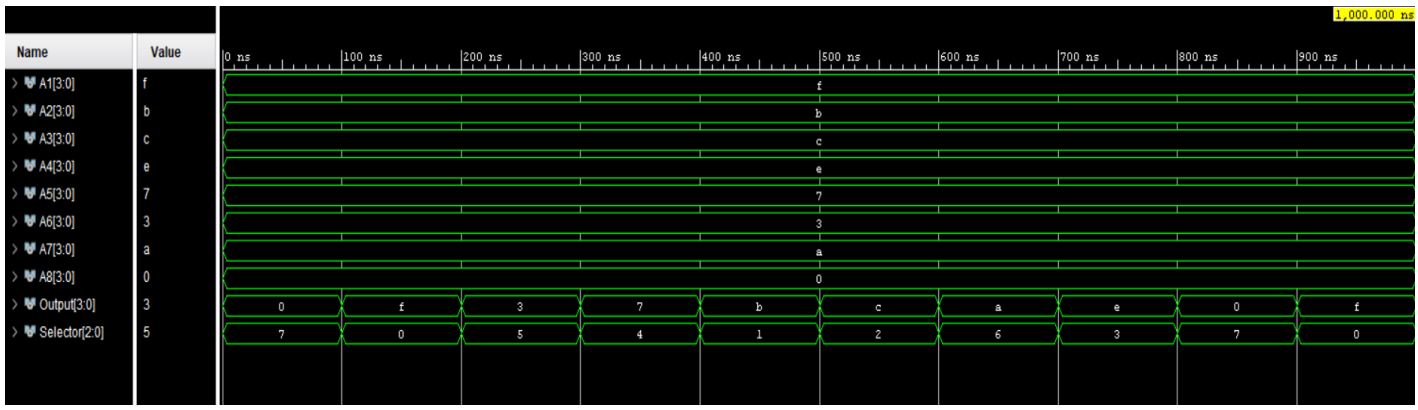
- Behavioral simulation Code for MUX_8way_4bit:

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

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

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

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

entity TB_MUX_8way_4bit is
--  Port ( );
end TB_MUX_8way_4bit;

architecture Behavioral of TB_MUX_8way_4bit is

component MUX_8way_4bit
    Port (
        A1 : in STD_LOGIC_VECTOR (3 downto 0);
        A2 : in STD_LOGIC_VECTOR (3 downto 0);
        A3 : in STD_LOGIC_VECTOR (3 downto 0);
        A4 : in STD_LOGIC_VECTOR (3 downto 0);
        A5 : in STD_LOGIC_VECTOR (3 downto 0);
        A6 : in STD_LOGIC_VECTOR (3 downto 0);
        A7 : in STD_LOGIC_VECTOR (3 downto 0);
        A8 : in STD_LOGIC_VECTOR (3 downto 0);
        Selector : in STD_LOGIC_VECTOR (2 downto 0);
        Output : out STD_LOGIC_VECTOR (3 downto 0)
        );
end component;

signal A1,A2,A3,A4,A5,A6,A7,A8,Output : std_logic_vector (3 downto 0);
signal Selector : std_logic_vector (2 downto 0);

begin
UUT : MUX_8way_4bit port map(
        A1 => A1,
        A2 => A2,
        A3 => A3,
        A4 => A4,
        A5 => A5,
        A6 => A6,
        A7 => A7,
        A8 => A8,
        Selector => Selector,
        Output => Output
        );

process begin
    A1 <= "1111";
    A2 <= "1011";
    A3 <= "1100";
    A4 <= "1110";
    A5 <= "0111";
    A6 <= "0011";
    A7 <= "1010";
    A8 <= "0000";
```

```
    Selector <= "111";
    wait for 100 ns;

    Selector <= "000";
    wait for 100 ns;

    Selector <= "101";
    wait for 100 ns;

    Selector <= "100";
    wait for 100 ns;

    Selector <= "001";
    wait for 100 ns;

    Selector <= "010";
    wait for 100 ns;

    Selector <= "110";
    wait for 100 ns;

    Selector <= "011";
    wait for 100 ns;

end process;
end Behavioral;
```

- Timing Diagram for MUX_8way_4bit:



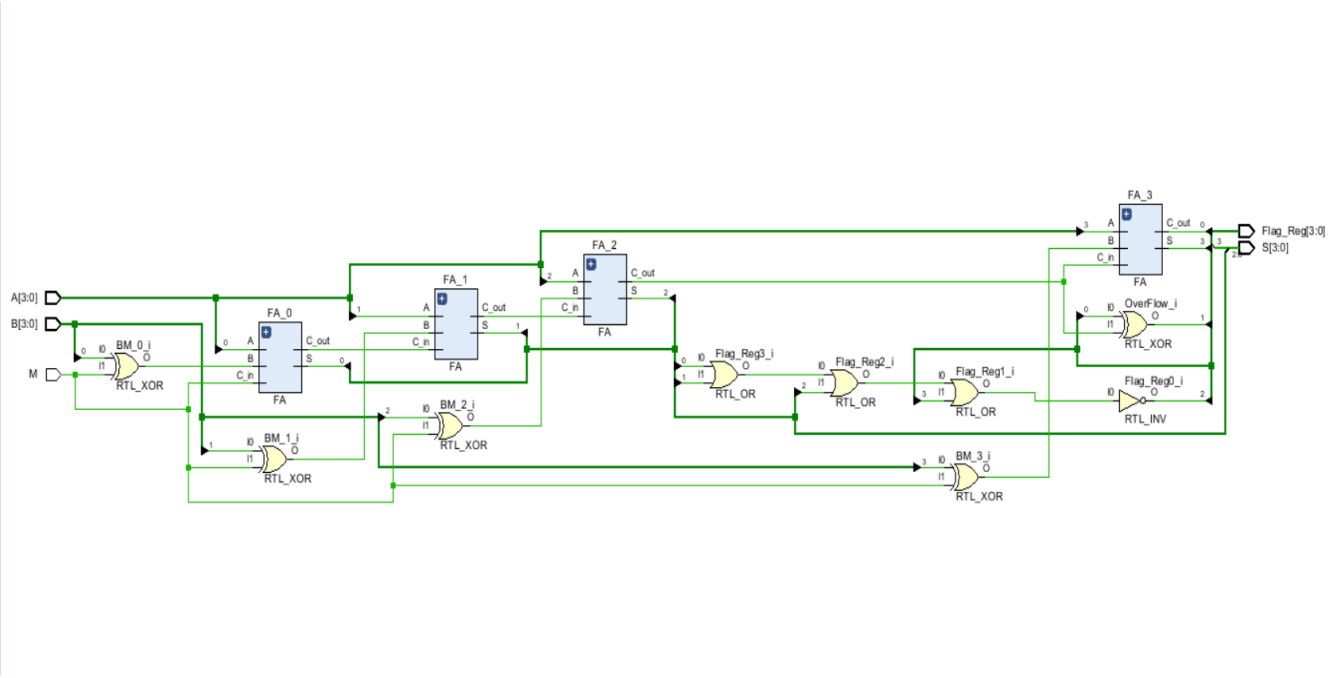
15. ADDER_SUBTRACTOR.

- Design source VHDL code of 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_SUBTRACTOR is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           M : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0));

end ADDER_SUBTRACTOR;

architecture Behavioral of ADDER_SUBTRACTOR is

component FA
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S : out STD_LOGIC;
           C_out : out STD_LOGIC);
end component;
signal BM, OutSum : std_logic_vector(3 downto 0);
signal C1,C2,C3,C4, OverFlow : std_logic;


begin

    BM(0) <= B(0) XOR M;
    BM(1) <= B(1) XOR M;
    BM(2) <= B(2) XOR M;
    BM(3) <= B(3) XOR M;

    FA_0 : FA
        port map(
            A => A(0),
            B => BM(0),
            C_in => M,
```

```
            S => OutSum(0),
            C_out => C1
        );

    FA_1 : FA
        port map(
            A => A(1),
            B => BM(1),
            C_in => C1,
            S => OutSum(1),
            C_out => C2
        );

    FA_2 : FA
        port map(
            A => A(2),
            B => BM(2),
            C_in => C2,
            S => OutSum(2),
            C_out => C3
        );

    FA_3 : FA
        port map(
            A => A(3),
            B => BM(3),
            C_in => C3,
            S => OutSum(3),
            C_out => C4
        );

        S <= OutSum ;
        OverFlow <= C4 XOR C3;


        Flag_Reg(0) <= C4;                                                    --C
        Flag_Reg(1) <= OverFlow;                                              --O
        Flag_Reg(2) <= NOT (OutSum(0) OR OutSum(1) OR OutSum(2) OR OutSum(3)); --Z
        Flag_Reg(3) <= OutSum(3);                                             --S

end Behavioral;
```

- RTL Schematic diagram of the ADDER_SUBTRACTOR:



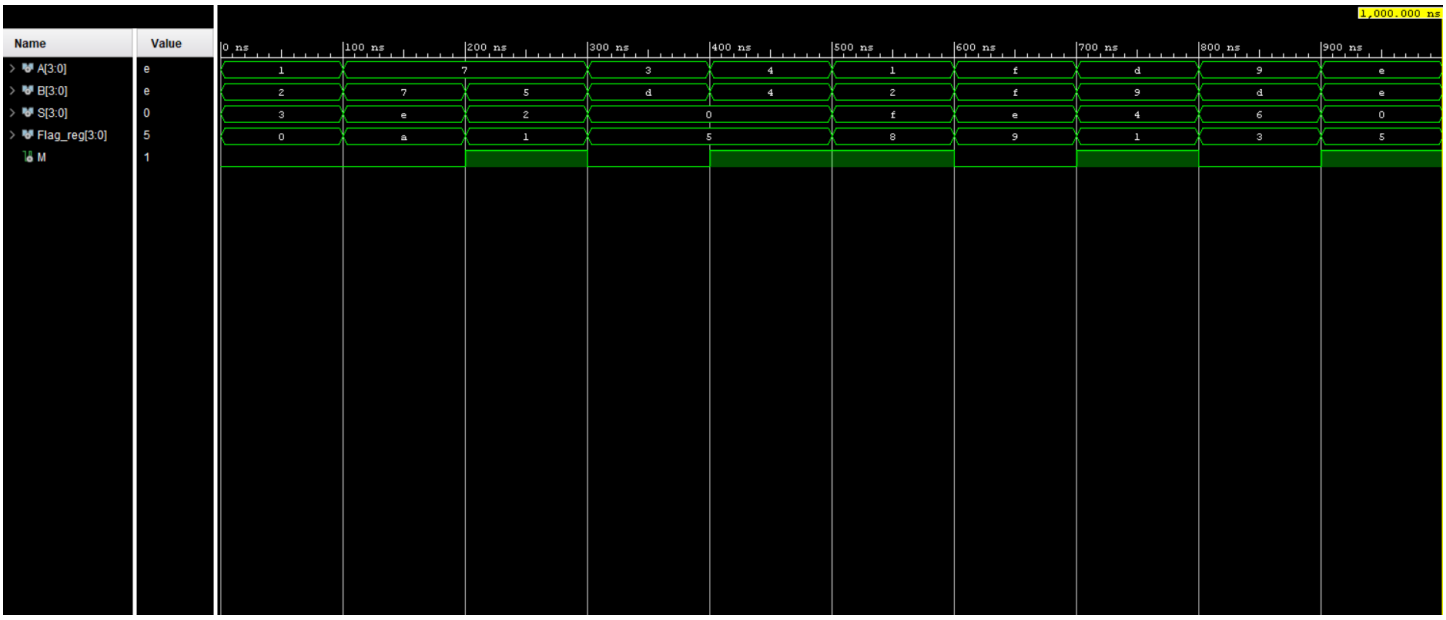
- Behavioral simulation Code for ADDER_SUBTRACTOR:

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

architecture Behavioral of TB_ADDER_SUBTRACTOR is

component ADDER_SUBTRACTOR
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           M : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
```

```
--            Overflow_Flag : out STD_LOGIC;
--            C_out : out STD_LOGIC;
--            Zero_Flag : out STD_LOGIC;
--            Sign_Flag : out STD_LOGIC
            Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0));
end component;

--signal A,B,S : std_logic_vector (3 downto 0);
--signal M, Overflow_Flag, C_out, Zero_Flag, Sign_Flag: std_logic;
signal A,B,S, Flag_reg : std_logic_vector (3 downto 0);
signal M: std_logic;

begin
    UUT: ADDER_SUBTRACTOR port map(
        A => A,
        B => B,
        M => M,
        S => S,
        Flag_Reg => Flag_Reg
        );

process begin

    -- 1 + 2 --
    A <= "0001";
    B <= "0010";
    M <= '0';
    wait for 100 ns;

    -- 7 + 7 --
    A <= "0111";
    B <= "0111";
    M <= '0';
    wait for 100 ns;

    -- 7 - 5 --
    A <= "0111";
    B <= "0101";
    M <= '1';
    wait for 100 ns;

    -- 3 + (-3) --
    A <= "0011";
    B <= "1101";
    M <= '0';
    wait for 100 ns;

    -- 4 - 4 --
    A <= "0100";
    B <= "0100";
```

```
    M <= '1';
    wait for 100 ns;

    -- 1 - 2 --
    A <= "0001";
    B <= "0010";
    M <= '1';
    wait for 100 ns;

    -- (-1) + (-1) --
    A <= "1111";
    B <= "1111";
    M <= '0';
    wait for 100 ns;

    -- (-3) - (-7) --
    A <= "1101";
    B <= "1001";
    M <= '1';
    wait for 100 ns;

    -- (-7) + (-3) --
    A <= "1001";
    B <= "1101";
    M <= '0';
    wait for 100 ns;

    -- (-2) - (-2) --
    A <= "1110";
    B <= "1110";
    M <= '1';
    wait ;

end process;
end Behavioral;
```

- Timing Diagram for ADDER_SUBTRACTOR:



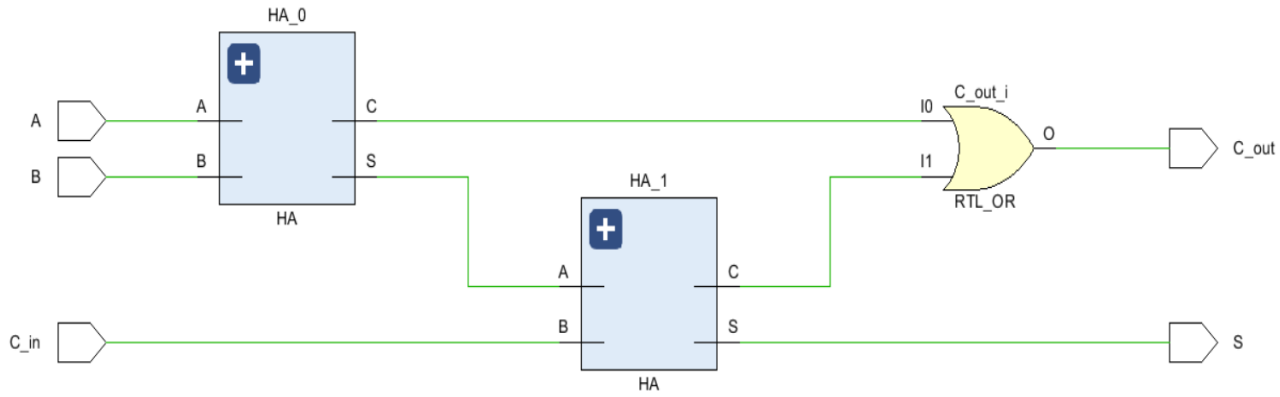
16. Full Adder.

- Design source VHDL code of Full adder:

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

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

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

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

architecture Behavioral of FA is
component HA
     port (
     A: in std_logic;
     B: in std_logic;
     S: out std_logic;
     C: out std_logic);
end component;

SIGNAL HA0_S, HA0_C, HA1_S, HA1_C : std_logic;

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

     HA_1 : HA
         port map (
         A => HA0_S,
         B => C_in,
         S => HA1_S,
         C => HA1_C);
```

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

end Behavioral;
```

- RTL Schematic diagram of the Full adder.

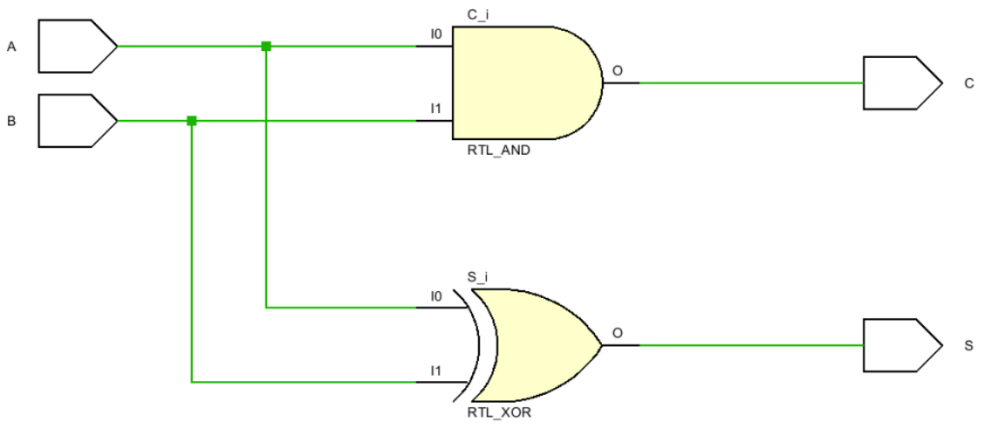


17. Half Adder.

- Design source VHDL code of Half adder:

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

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

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

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

architecture Behavioral of HA is

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

end Behavioral;
```

- RTL Schematic diagram of the Half adder.



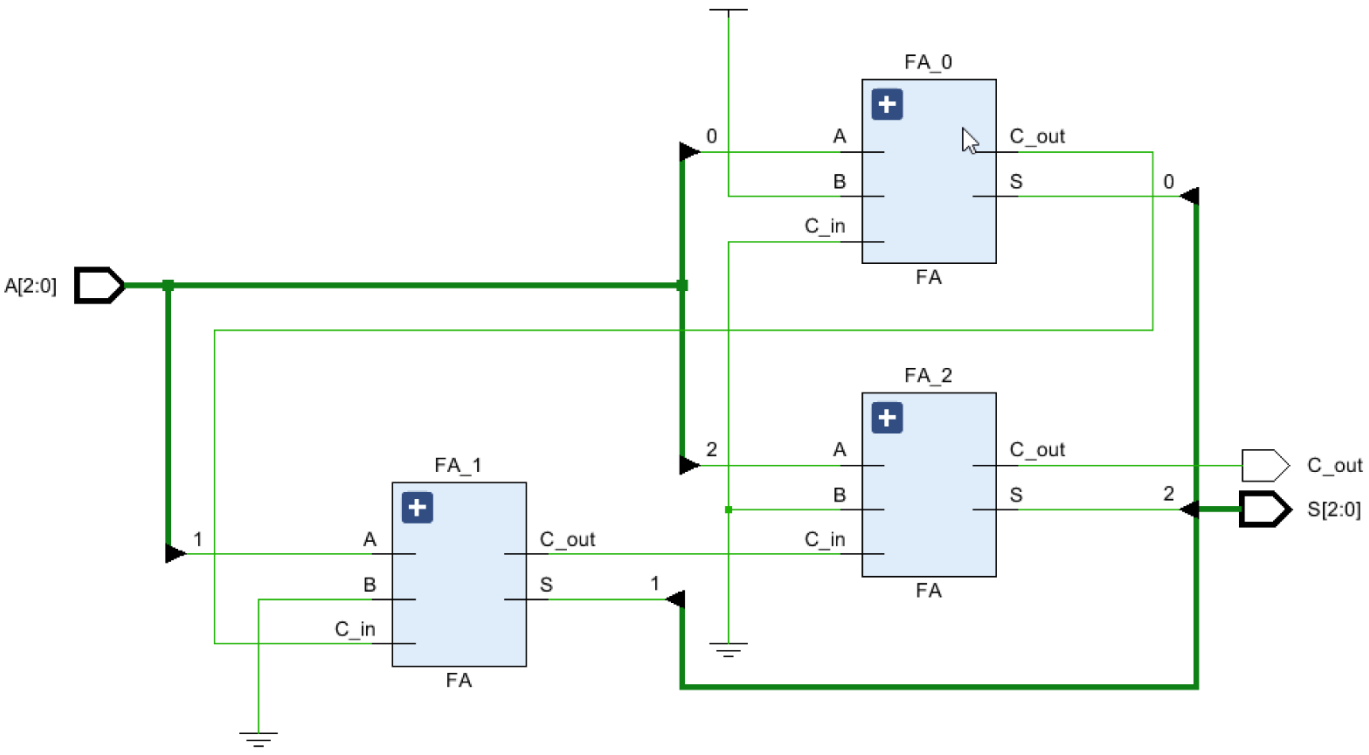
18. ADDER_3bit.

- Design source VHDL code of ADDER_3bit.

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

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

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

entity ADDER_3bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC);
end ADDER_3bit;

architecture Behavioral of ADDER_3bit is

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

signal C1,C2,C3 : std_logic;
signal OutSum : std_logic_vector (2 downto 0);
signal B : std_logic := '1';

begin
    FA_0 : FA
        port map(
            A => A(0),
            B => B,
            C_in => '0',
            S => OutSum(0),
            C_out => C1
        );

    FA_1 : FA
        port map(
            A => A(1),
            B => '0',
```

```
            C_in => C1,
            S => OutSum(1),
            C_out => C2
        );

    FA_2 : FA
        port map(
            A => A(2),
            B => '0',
            C_in => C2,
            S => OutSum(2),
            C_out => C3
        );

 S <= OutSum;
 C_out <= C3;


end Behavioral;
```

- RTL Schematic diagram of the ADDER_3bit:

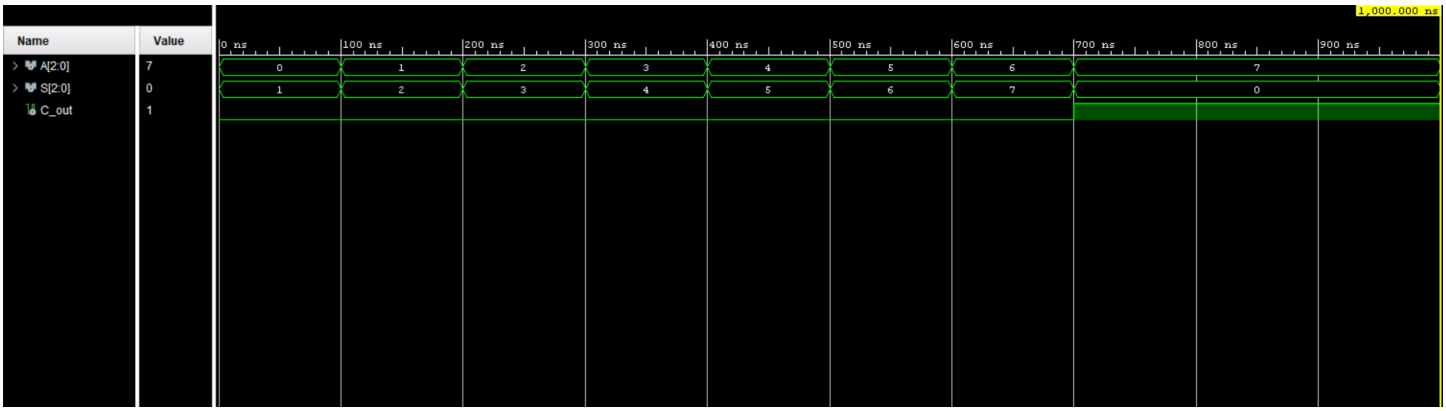


- Behavioral simulation Code for ADDER_3bit:

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

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

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

entity TB_ADDER_3bit is
--  Port ( );
end TB_ADDER_3bit;

architecture Behavioral of TB_ADDER_3bit is

component ADDER_3bit
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC);
end component;

signal A,S : std_logic_vector (2 downto 0);
signal C_out : std_logic;

begin
UUT :ADDER_3bit port map(
        A => A,
        S => S,
        C_out => C_out
    );

process begin
   A <= "000";
   wait for 100 ns;

   A <= "001";
   wait for 100 ns;

   A <= "010";
   wait for 100 ns;

   A <= "011";
   wait for 100 ns;

   A <= "100";
```

```
    wait for 100 ns;

    A <= "101";
    wait for 100 ns;

    A <= "110";
    wait for 100 ns;

    A <= "111";
    wait;

end process;

end Behavioral;
```

- Timing Diagram for ADDER_3bit:

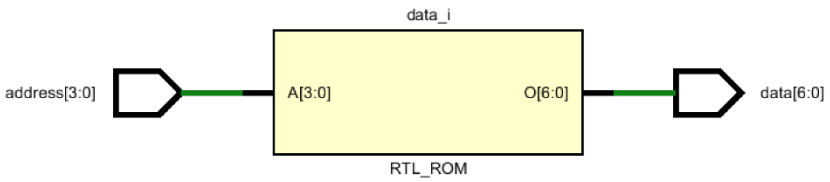


19. LUT_16_7.

- Design source VHDL code of LUT_16_7.

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

architecture Behavioral of LUT_16_7 is

type rom_type is array (0 to 15) of std_logic_vector (6 downto 0);
    signal sevenSegment_ROM : rom_type := (
                              "1000000", --0
                              "1111001", --1
                              "0100100", --2
                              "0110000", --3
                              "0011001", --4
                              "0010010", --5
                              "0000010", --6
                              "1111000", --7
                              "0000000", --8
                              "0010000", --9
                              "0001000", --a
                              "0000011", --b
                              "1000110", --c
                              "0100001", --d
                              "0000110", --e
                              "0001110"  --f
          );

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


end Behavioral;
```

- RTL Schematic diagram of the LUT_16_7.

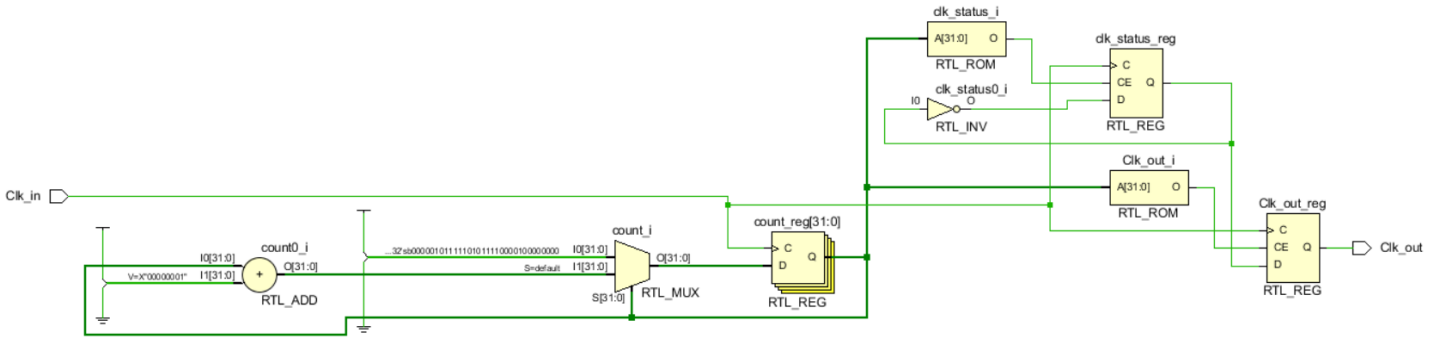


20. Slow Clock

- Design source VHDL code of Nano processor.

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

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

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

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

architecture Behavioral of Slow_Clk is

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

begin
    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1;
            if(count = 100000000) then       --For simulation purposes, reduce 100000000
to 5
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;
end Behavioral;
```

- RTL Schematic diagram of the Nano processor.



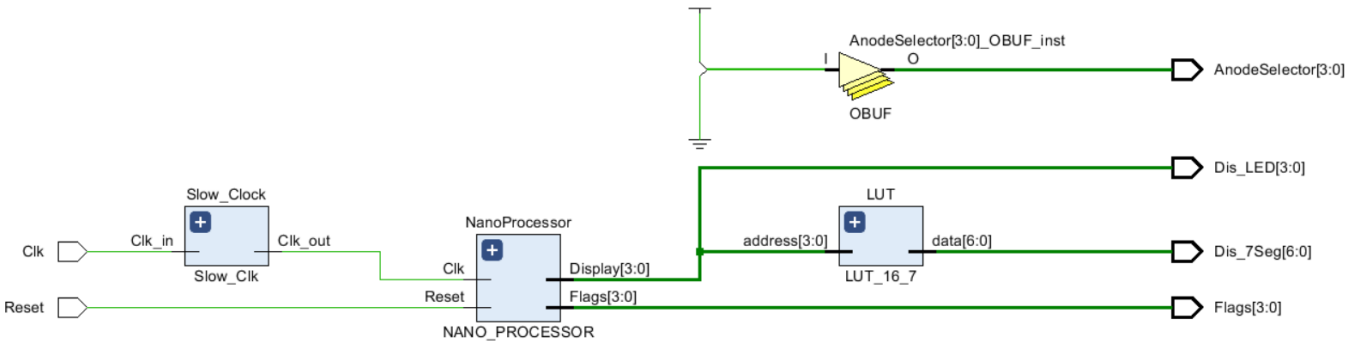
21. MAIN

- Design source VHDL code of Main:

```
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 MAIN is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Dis_LED : out STD_LOGIC_VECTOR (3 downto 0);
           Dis_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           Flags : out STD_LOGIC_VECTOR (3 downto 0);
           AnodeSelector : out STD_LOGIC_VECTOR (3 downto 0)
          );
end MAIN;

architecture Behavioral of MAIN is

component NANO_PROCESSOR is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0));
end component;

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

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

signal Display_out : STD_LOGIC_VECTOR (3 downto 0);
signal SlowClk_out : std_logic;

begin
    Slow_Clock : Slow_Clk
```

```
        Port map(
            Clk_in => Clk,
            Clk_out => SlowClk_out
         );

    NanoProcessor : NANO_PROCESSOR
         PORT MAP(
             Clk => SlowClk_out,
             Reset => Reset,
             Display => Display_out,
             Flags => Flags
          );

    LUT : LUT_16_7
        port map(
            address => Display_out,
            data => Dis_7Seg
         );

   Dis_LED <= Display_out;
   AnodeSelector <= "1110";

end Behavioral;
```

- RTL Schematic diagram of the Main:

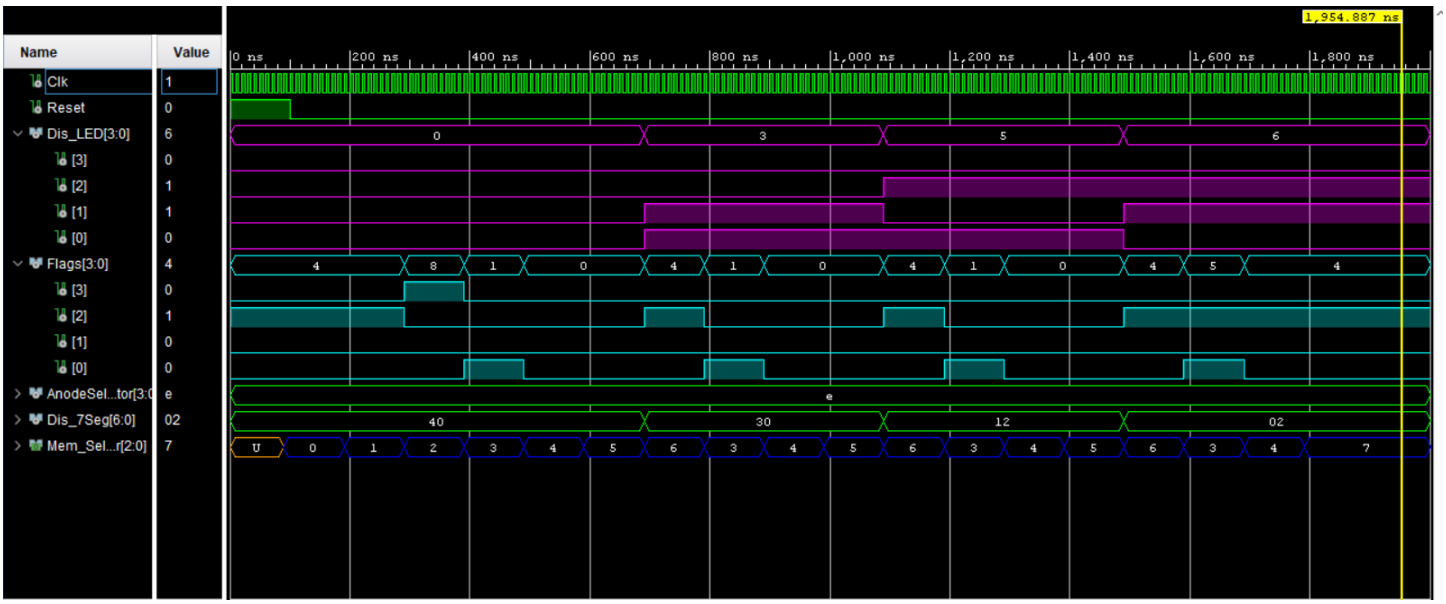


- Behavioral simulation Code for Main:

```
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_MAIN is
--  Port ( );
end TB_MAIN;

architecture Behavioral of TB_MAIN is

component MAIN
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Dis_LED : out STD_LOGIC_VECTOR (3 downto 0);
           Dis_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           Flags : out STD_LOGIC_VECTOR (3 downto 0);
           AnodeSelector : out STD_LOGIC_VECTOR (3 downto 0)
           );
end component;

signal Clk : std_logic := '0';
signal Reset : std_logic;
signal Dis_LED, Flags, AnodeSelector : std_logic_vector (3 downto 0);
signal Dis_7Seg : std_logic_vector (6 downto 0);

begin
    UUT : MAIN
        port map (
            Clk => Clk,
            Reset => Reset,
            Dis_LED => Dis_LED,
            Dis_7Seg => Dis_7Seg,
            Flags => Flags,
            AnodeSelector => AnodeSelector
         );

process begin
    Clk <= not Clk;
    wait for 5 ns;
end process;
```

```
process begin
    Reset <= '1';
    wait for 100 ns;

    Reset <= '0';
    wait;
end process;

end Behavioral;
```

- Timing Diagram for Main:



- Constraint File

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


## LEDs
set_property PACKAGE_PIN U16 [get_ports {Dis_LED[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {Dis_LED[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {Dis_LED[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {Dis_LED[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[3]}]

set_property PACKAGE_PIN P3 [get_ports {Flags[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[3]}]
set_property PACKAGE_PIN N3 [get_ports {Flags[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[1]}]
set_property PACKAGE_PIN P1 [get_ports {Flags[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[2]}]
set_property PACKAGE_PIN L1 [get_ports {Flags[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[0]}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {Dis_7Seg[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {Dis_7Seg[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {Dis_7Seg[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {Dis_7Seg[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {Dis_7Seg[4]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {Dis_7Seg[5]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {Dis_7Seg[6]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[6]}]


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

```
      set_property IOSTANDARD LVCMOS33 [get_ports {AnodeSelector[2]}]
set_property PACKAGE_PIN W4 [get_ports {AnodeSelector[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {AnodeSelector[3]}]


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

2) Design with extended qualifications

- Instructions for operating the machine

1. Allocated reset button
    - The nano processor can be reset by pressing the btnC.
    - The clock speed of the internal clock of the machine was reduced from 100MHz to 0.5MHz using a slow clock to make the calculation process visible to the naked eye.
    - In order to reset the nano processor,it is necessary to press and hold the reset button for at least 2 seconds.

2. LED signal
    - LED0-LED3 Output of R7 register in Register Bank
    - LED0-LED3 → Outputs a 4-bit number as a signed number in two's complement method

- LED6-LED7 Output of the Comparator
    - LED6 → Equal
    - LED7 → Greater
    - LED8 → Lesser

- LED12-LED15 Flags of 4-bit Add/Subtract Unit
    - LED12 → Sign Flag
    - LED13 → Overflow Flag
    - LED14 → Zero Flag
    - LED15 → Carry Flag

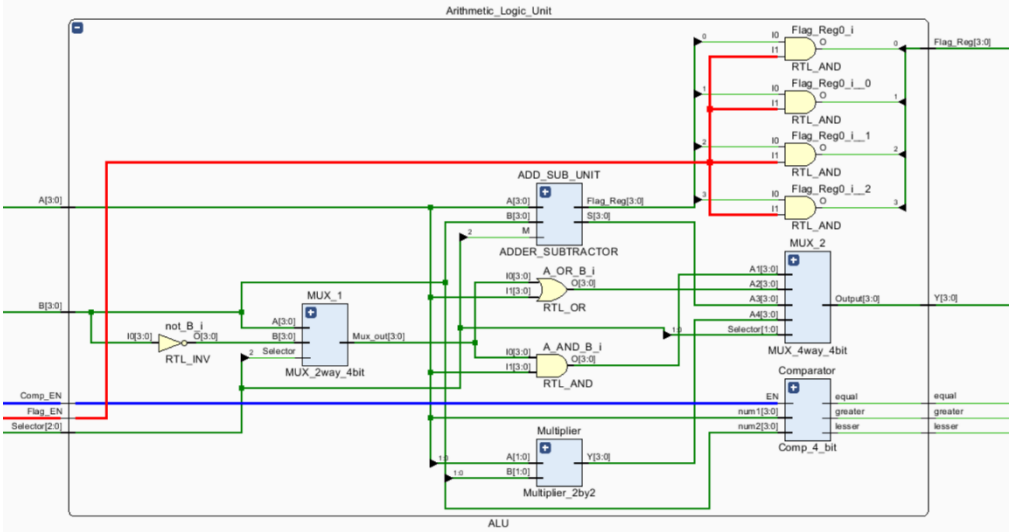
3. 7 Segment Display
    - The rightmost segment of the 7-Segment display is used to display the magnitude of the output from the R7 register in the Register Bank.

4. Expected behavior of the program
    - After the calculation is performed the process will be held because of the jump instruction implemented.
    - Therefore, it is necessary to manually reset the nanoprocessor to perform the instructions hardcoded in the ROM.

• Discussion

- In this extended design we designed extra components to do various tasks other than the primary task. There we used ALU and comparators. we replaced the ADD/SUBSTRACTOR unit with the ALU that we designed to do multiple tasks. There we also added a Comparator Enable and a Flag Enable to this extended design as extra elements.

- Comparator works with the same 2 inputs which inputs to the ALU. We used this Comparator Enable to stop getting outputs from the comparator at unwanted times. (i.e. when the jump instruction executes). Also, the Flag enable was used to be turned on only when the ALU executes the ADD and SUB instructions in this design.



- In this design 1-bit comparators were designed to compare two 1-bit numbers and using 4 of those 1-bit comparators we designed 4-bit comparator. These comparators work as sign-magnitude comparators.

- Optimized Slice Logic and Design Primitives

- Slice Logic

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	110	0	20800	0.53
LUT as Logic	110	0	20800	0.53
LUT as Memory	0	0	9600	0.00
Slice Registers	49	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

- Design Primitives

7. Primitives

Ref Name	Used	Functional Category
LUT2	60	LUT
FDRE	37	Flop & Latch
LUT4	36	LUT
LUT3	26	LUT
OBUF	22	IO
LUT5	17	LUT
FDCE	12	Flop & Latch
LUT6	11	LUT
CARRY4	8	CarryLogic
LUT1	6	LUT
IBUF	2	IO
BUFG	1	Clock

Executable Instruction Set

- We extended the instruction set by the following ways,
- In the previous design's instruction set, the **ADD** instruction had 3 unused LSBs. We used that 3 bits in this extended version to implement more instructions.

Instruction	Description	Format(12-bit instruction)
MOVI R,d	Move(11 downto 10 ==10) immediate value d(3 downto 0) to register R (9 downto 7).	1 0 R R R 0 0 0 d d d d
Operate(11 downto 10 == 00) d(2 downto 0) operation to the values in registers Ra(9 downto 7) and Rb(6 downto 4) and store the result in Ra.		0 0 Ra Ra Ra Ra Rb Rb Rb 0 d d d
AND Ra,Rb	Bitwise AND operation (d==000)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 0 0 0
OR Ra,Rb	Bitwise OR operation (d==001)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 0 0 1
ADD Ra,Rb	Addition operation (d==010)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 0 1 0
Un-used instruction		0 0 Ra Ra Ra Ra Rb Rb Rb 0 0 1 1
ANDN Ra,Rb	Bitwise AND with NOT operation (`A AND (NOT B)`) (d==100)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 1 0 0
ORN Ra,Rb	Bitwise OR with NOT operation (`A OR (NOT B)`) (d==101)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 1 0 1
SUB Ra,Rb	Subtraction operation (d==110)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 1 1 0
MUL Ra,Rb	Multiplication operation (d==111)	0 0 Ra Ra Ra Ra Rb Rb Rb 0 1 1 1

NEG R	Get negation (11 downto 10 == 01) of the value that store in the register R (9 downto 7) and restore the created negation to R register.	0 1 R R R 0 0 0 0 0 0 0
JZR R,d	Jump(11 downto 10 ==11) to Pc to value d(2 downto 0) if value in register R(9 downto 7) is 0. i.e, If R==0; Pc←d; Else; Pc← Pc + 1;	1 1 R R R 0 0 0 0 d d d

- Design Components:

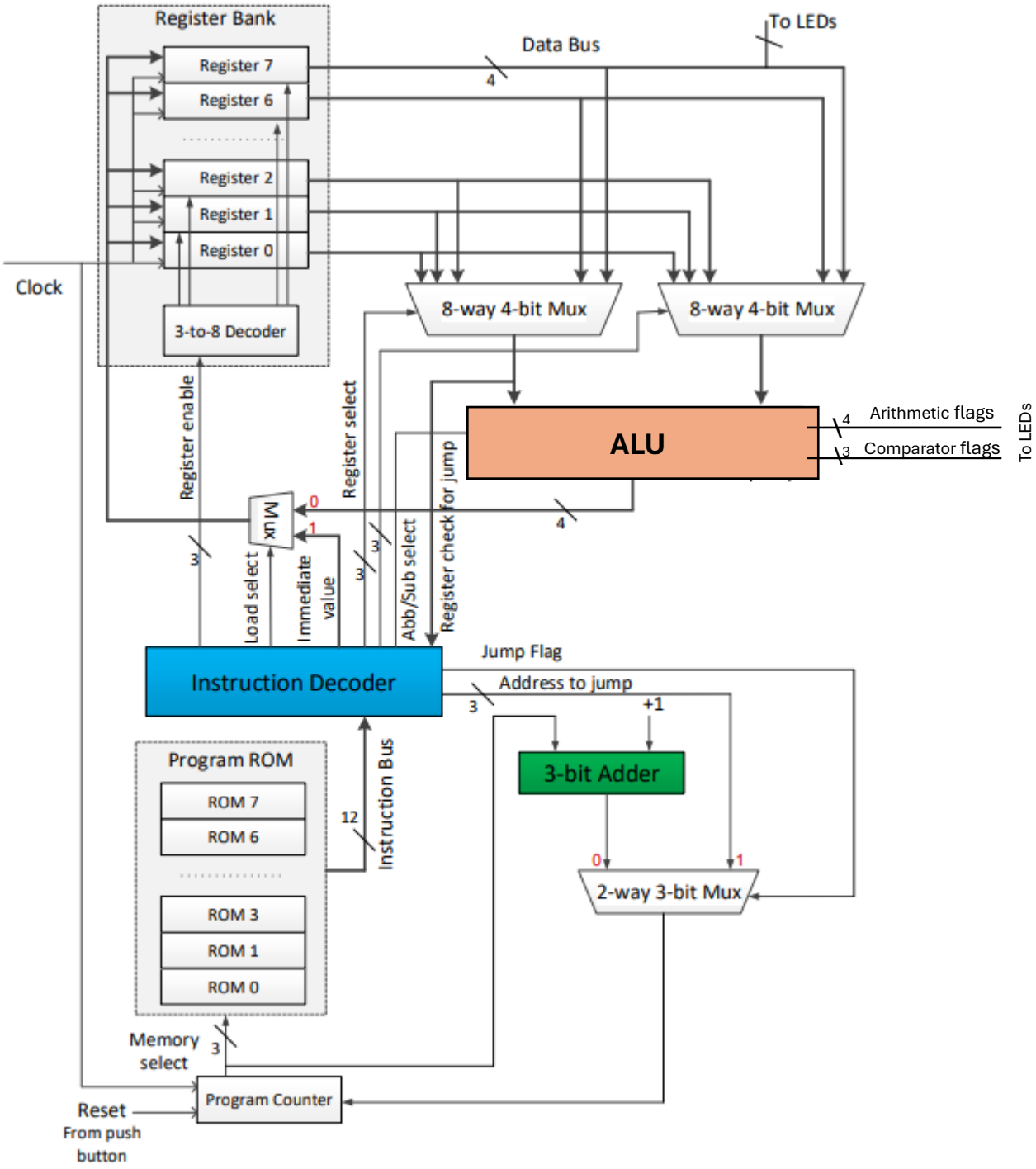


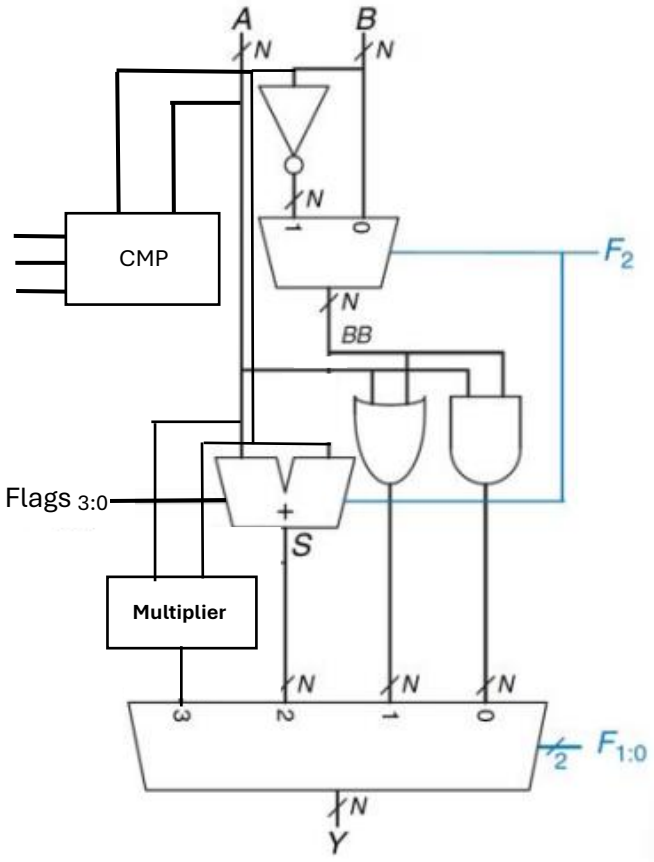
Figure 1 – High-level diagram of the nanoprocessor.

1. ALU

- High level implementation of the ALU

Table 5.1 ALU operations

$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A – B
111	A MUL B



- Design source VHDL code of ALU :

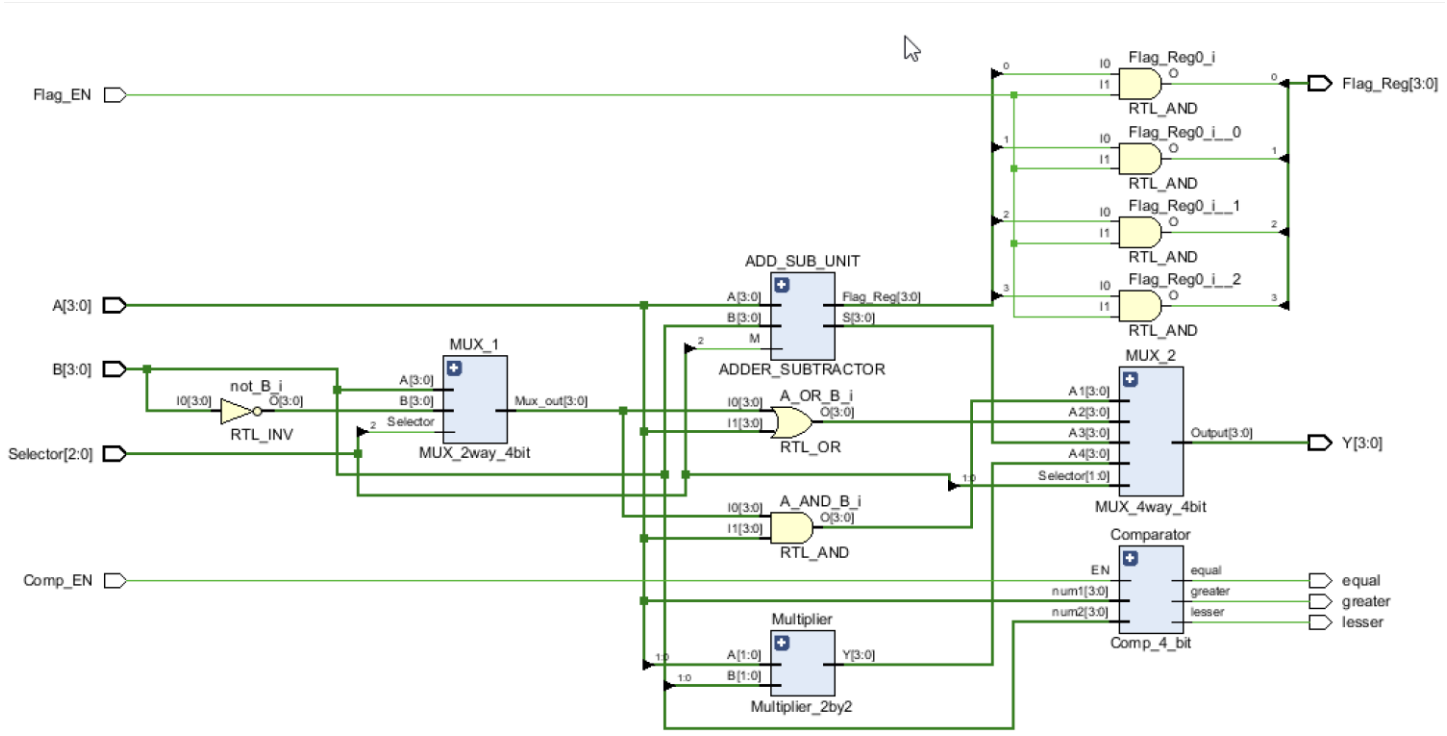
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- 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 ALU is
    port (
          A : in std_logic_vector (3 downto 0);
          B : in std_logic_vector (3 downto 0);
          Flag_EN : in std_logic;
          Comp_EN : in std_logic;
          Selector : in std_logic_vector (2 downto 0);
```

```
          Y : out std_logic_vector (3 downto 0);
          Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0);
          equal : out STD_LOGIC;
          greater : out STD_LOGIC;
          lesser : out STD_LOGIC
          );
end ALU;

architecture Behavioral of ALU is

component Comp_4_bit is
    Port ( num1 : in STD_LOGIC_VECTOR(3 downto 0);
           num2 : in STD_LOGIC_VECTOR(3 downto 0);
           EN : in STD_LOGIC;
           equal : out STD_LOGIC;
           greater : out STD_LOGIC;
           lesser : out STD_LOGIC );
end component;

component MUX_2way_4bit
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC;
           Mux_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component MUX_4way_4bit
    Port (
       A1 : in STD_LOGIC_VECTOR (3 downto 0);
       A2 : in STD_LOGIC_VECTOR (3 downto 0);
       A3 : in STD_LOGIC_VECTOR (3 downto 0);
       A4 : in STD_LOGIC_VECTOR (3 downto 0);
       Selector : in STD_LOGIC_VECTOR (1 downto 0);
       Output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component ADDER_SUBTRACTOR
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           M : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Multiplier_2by2
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
           B : in STD_LOGIC_VECTOR (1 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;
```

```
signal not_B : std_logic_vector (3 downto 0);
signal Mux_1_out, Mux_2_out : std_logic_vector (3 downto 0);
signal A_AND_B : std_logic_vector (3 downto 0);
signal A_OR_B : std_logic_vector (3 downto 0);
signal A_plus_B : std_logic_vector (3 downto 0);
signal A_into_B : std_logic_vector (3 downto 0);
signal Flags : std_logic_vector (3 downto 0);

begin
    not_B <= NOT B;

    MUX_1 : MUX_2way_4bit
        port map(
            A => B,
            B => not_B,
            Selector => Selector(2),
            Mux_out => Mux_1_out
        );

    A_AND_B <= Mux_1_out AND A;
    A_OR_B <= Mux_1_out OR A;

    ADD_SUB_UNIT : ADDER_SUBTRACTOR
        port map(
            A => A,
            B => B,
            M => Selector(2),
            S => A_plus_B,
            Flag_Reg => Flags
        );


     Multiplier : Multiplier_2by2
        port map (
            A => A(1 downto 0),
            B => B(1 downto 0),
            Y => A_into_B
         );

     MUX_2 : MUX_4way_4bit
        port map(
            A1 => A_AND_B,
            A2 => A_OR_B,
            A3 => A_plus_B,
            A4 => A_into_B,
            Selector => Selector(1 downto 0),
            Output => Mux_2_out
         );

     Comparator : Comp_4_bit
```

```
        port map(
            num1 => A,
            num2=>B,
            EN => Comp_EN,
            equal => equal,
            greater => greater,
            lesser => lesser);


Flag_Reg(0) <= Flags(0) AND Flag_EN;
Flag_Reg(1) <= Flags(1) AND Flag_EN;
Flag_Reg(2) <= Flags(2) AND Flag_EN;
Flag_Reg(3) <= Flags(3) AND Flag_EN;
Y <= Mux_2_out;

end Behavioral;
```

- RTL Schematic diagram of the ALU :



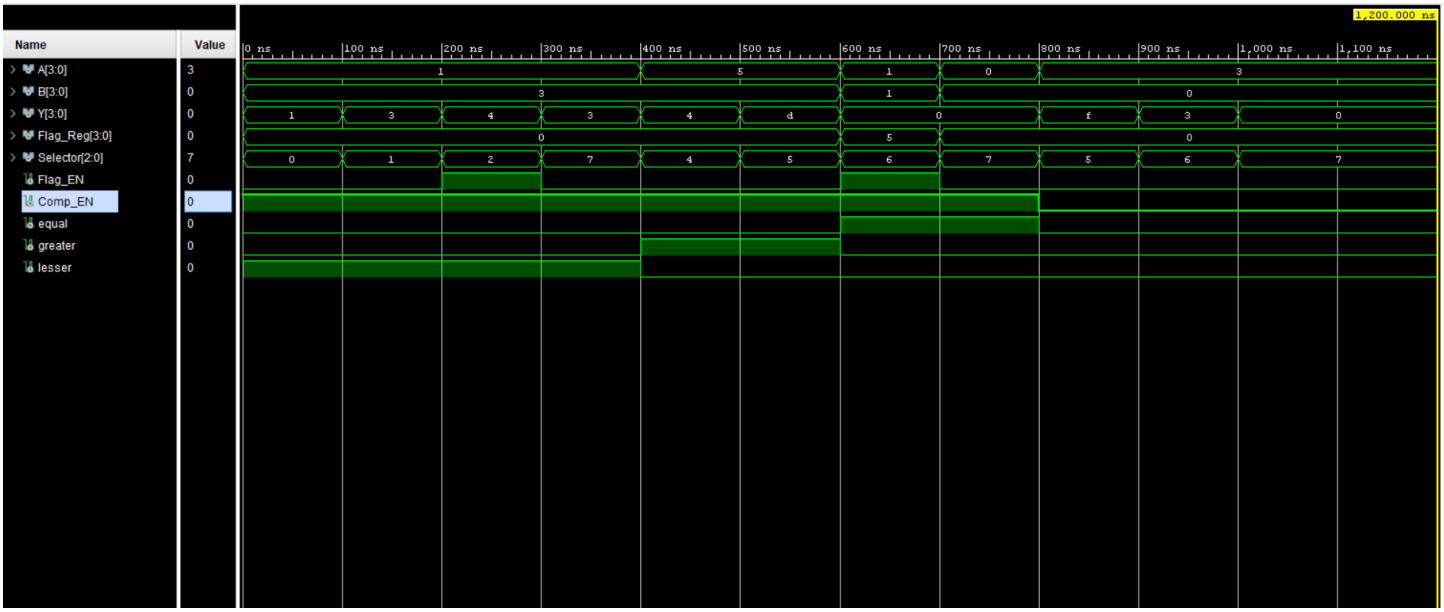
- Behavioral simulation Code for ALU :

```
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_ALU is
--  Port ( );
end TB_ALU;

architecture Behavioral of TB_ALU is

component ALU
    port (
            A : in std_logic_vector (3 downto 0);
            B : in std_logic_vector (3 downto 0);
            Flag_EN : in std_logic;
            Comp_EN : in std_logic;
            Selector : in std_logic_vector (2 downto 0);
            Y : out std_logic_vector (3 downto 0);
            Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0);
            equal : out STD_LOGIC;
            greater : out STD_LOGIC;
            lesser : out STD_LOGIC
          );
end component;

signal A, B, Y, Flag_Reg : STD_LOGIC_VECTOR (3 downto 0);
signal Selector : STD_LOGIC_VECTOR (2 downto 0);
signal Flag_EN, Comp_EN, equal, greater, lesser : STD_LOGIC;

begin
    UUT : ALU
        port map(
           A => A,
           B => B,
           Comp_EN => Comp_EN,
           Flag_EN => Flag_EN,
           Selector => Selector,
           Y => Y,
           Flag_Reg => Flag_Reg,
           equal => equal,
```

```
            greater => greater,
            lesser => lesser
        );

process begin
    A <= "0001";
    B <= "0011";
    Flag_EN <= '0';
    Comp_EN <= '1';

    Selector <= "000";
    wait for 100 ns;

    Selector <= "001";
    wait for 100 ns;

    Flag_EN <= '1';
    Selector <= "010";
    wait for 100 ns;

    Flag_EN <= '0';
    Selector <= "111";
    wait for 100 ns;

    A <= "0101";
    B <= "0011";
    Selector <= "100";
    wait for 100 ns;

    Selector <= "101";
    wait for 100 ns;

    A <= "0001";
    B <= "0001";
    Flag_EN <= '1';
    Selector <= "110";
    wait for 100 ns;

    A <= "0000";
    B <= "0000";
    Flag_EN <= '0';
    Selector <= "111";
    wait for 100 ns;

    A <= "0011";
    Comp_EN <= '0';
    Selector <= "101";
    wait for 100 ns;

    Selector <= "110";
```

```
    wait for 100 ns;

    Selector <= "111";
    wait;

end process;

end Behavioral;
```

- Timing Diagram for ALU :



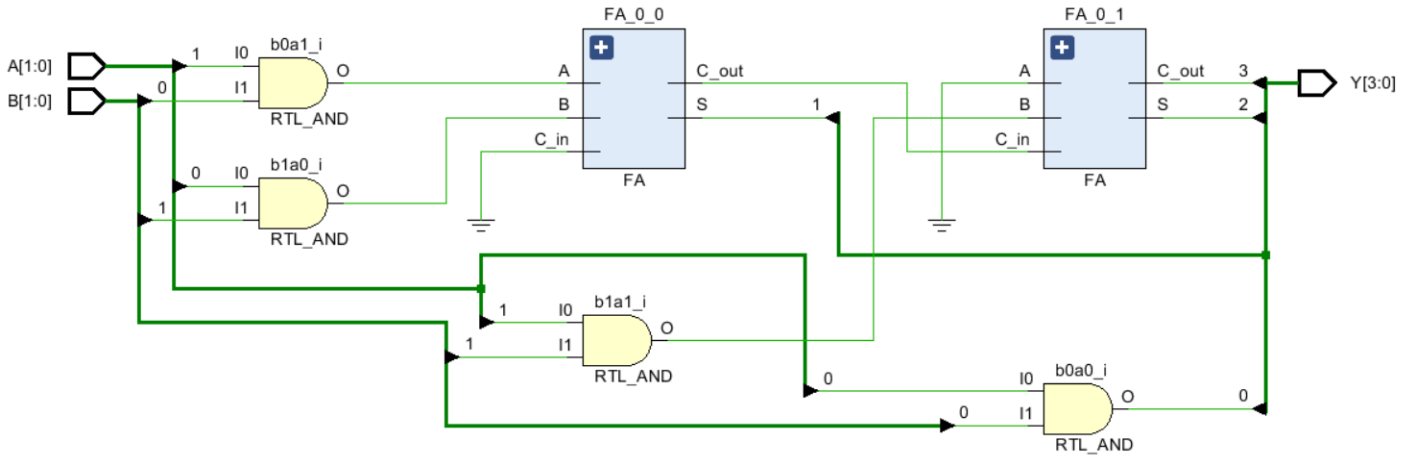
2. Multiplier

- Design source VHDL code of Comp_1_bit :

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

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

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

entity Multiplier_2by2 is
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
           B : in STD_LOGIC_VECTOR (1 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end Multiplier_2by2;

architecture Behavioral of Multiplier_2by2 is

component FA
    port(
        A : in std_logic;
        B : in std_logic;
        C_in : in std_logic;
        S : out std_logic;
        C_out : out std_logic
);
end component;

signal b0a0, b0a1, b1a0, b1a1 : std_logic;
signal s_0_0, s_0_1, c_0_0, c_0_1 : std_logic;

begin

FA_0_0: FA port map(
    A => b0a1,
    B => b1a0,
    C_in => '0',
    S => s_0_0,
    C_out => c_0_0
);

FA_0_1: FA port map(
    A => '0',
    B => b1a1,
```

```
    C_in => c_0_0,
    S => s_0_1,
    C_out => c_0_1
);

b0a0 <= A(0) and B(0);
b1a0 <= A(0) and B(1);
b0a1 <= A(1) and B(0);
b1a1 <= A(1) and B(1);

Y(0) <= b0a0;
Y(1) <= s_0_0;
Y(2) <= s_0_1;
Y(3) <= c_0_1;

end Behavioral;
```

- RTL Schematic diagram of the Multiplier:

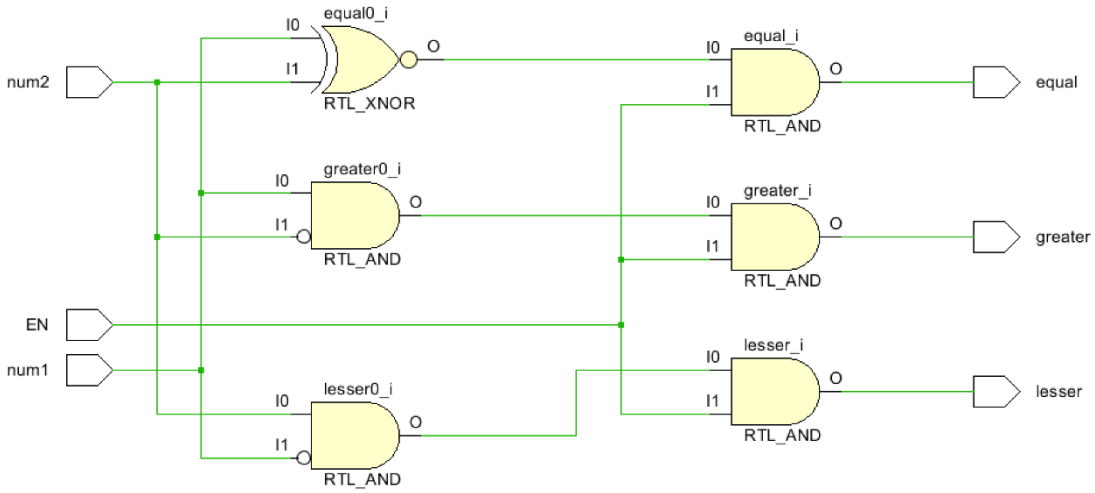


3. Comp_1_bit

- Design source VHDL code of Comp_1_bit :

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

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

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

entity Comp_1_bit is
        Port ( num1 : in STD_LOGIC;
               num2 : in STD_LOGIC;
               EN : in STD_LOGIC;
               equal : out STD_LOGIC;       --num1 = num2
               greater : out STD_LOGIC;     --num1 > num2
               lesser : out STD_LOGIC );    --num1 < num2
end Comp_1_bit;

architecture Behavioral of Comp_1_bit is

begin
    equal <= (num1 XNOR num2) AND EN;
    greater <= (num1 AND (NOT num2)) AND EN;
    lesser <= (num2 AND (NOT num1)) AND EN;
end Behavioral;
```

- RTL Schematic diagram of the Comp_1_bit :



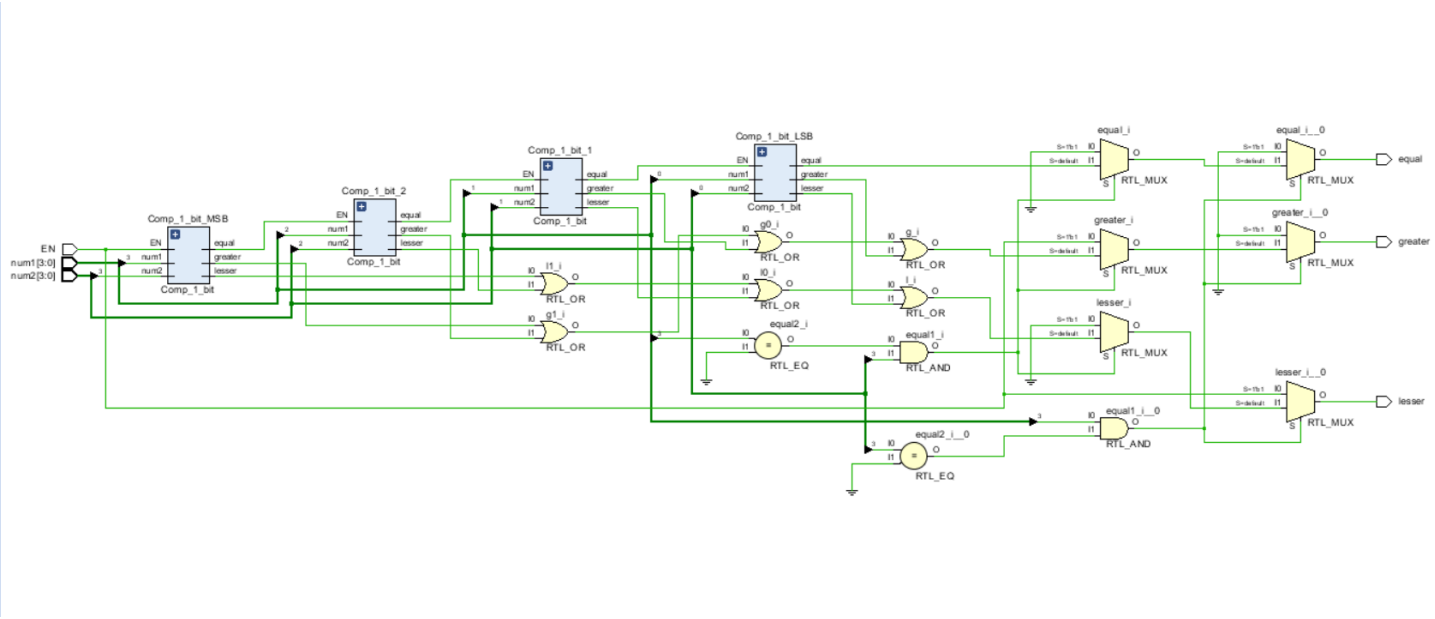
4. Comp_4_bit

- Design source VHDL code of Comp_4_bit :

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

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

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

entity Comp_4_bit is
    Port ( num1 : in STD_LOGIC_VECTOR(3 downto 0);
           num2 : in STD_LOGIC_VECTOR(3 downto 0);
           EN : in STD_LOGIC;
           equal : out STD_LOGIC;       --num1 = num2
           greater : out STD_LOGIC;     --num1 > num2
           lesser : out STD_LOGIC );    --num1 < num2
end Comp_4_bit;

architecture Behavioral of Comp_4_bit is

component Comp_1_bit is
        Port ( num1 : in STD_LOGIC;
               num2 : in STD_LOGIC;
               EN : in STD_LOGIC;
               equal : out STD_LOGIC;
               greater : out STD_LOGIC;
               lesser : out STD_LOGIC );
end component;

SIGNAL E_3,G_3,L_3,E_2,G_2,L_2,E_1,G_1,L_1,E_0,G_0,L_0,e,g,l : STD_LOGIC;

begin

    Comp_1_bit_MSB : Comp_1_bit
        Port map( num1 => num1(3),
               num2 => num2(3),
               EN => EN,
               equal => E_3,
               greater => G_3,
               lesser => L_3 );

    Comp_1_bit_2 : Comp_1_bit
       Port map( num1 => num1(2),
```

```
            num2 => num2(2),
            EN => E_3,
            equal => E_2,
            greater => G_2,
            lesser => L_2 );

    Comp_1_bit_1 : Comp_1_bit
      Port map( num1 => num1(1),
            num2 => num2(1),
            EN => E_2,
            equal => E_1,
            greater => G_1,
            lesser => L_1 );

    Comp_1_bit_LSB : Comp_1_bit
     Port map( num1 => num1(0),
            num2 => num2(0),
            EN => E_1,
            equal => E_0,
            greater => G_0,
            lesser => L_0 );

    e <= E_0;
    g <= G_3 OR G_2 OR G_1 OR G_0;
    l <= L_3 OR L_2 OR L_1 OR L_0;

    process (num1,num2,EN,e,g,l) begin
         if (num1(3)='1' AND num2(3)='0') then
             equal <= '0';
             greater <= '0';
             lesser <= EN;
         elsif (num1(3)='0' AND num2(3)='1') then
             equal <= '0';
             greater <= EN;
             lesser <= '0';
         else
             equal <= e;
             greater <= g;
             lesser <= l;
         end if;
    end process;

end Behavioral;
```

- RTL Schematic diagram of the Comp_4_bit :



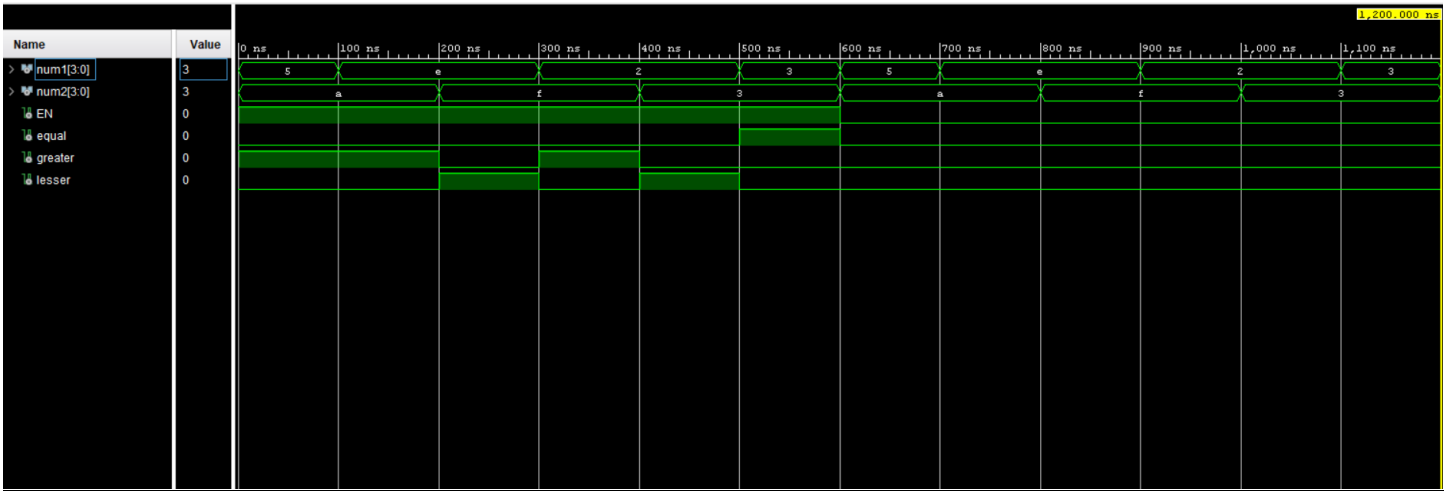
- Behavioral simulation Code for Comp_4_bit:

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

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

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

entity TB_Comp_4_bit is
--  Port ( );
end TB_Comp_4_bit;

architecture Behavioral of TB_Comp_4_bit is

component Comp_4_bit is
    Port ( num1 : in STD_LOGIC_VECTOR(3 downto 0);
           num2 : in STD_LOGIC_VECTOR(3 downto 0);
           EN : in STD_LOGIC;
           equal : out STD_LOGIC;        --num1 = num2
           greater : out STD_LOGIC;      --num1 > num2
           lesser : out STD_LOGIC );     --num1 < num2
end component;
```

```
SIGNAL num1, num2 : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL EN, equal, greater, lesser : STD_LOGIC;

begin
    UUT : Comp_4_bit
        Port map ( num1 => num1,
           num2 => num2,
           EN => EN,
           equal => equal,
           greater => greater,
           lesser => lesser);

    process begin

        EN <= '1';
        num1 <= "0101";
        num2 <= "1010";
        wait for 100ns;

        num1 <= "1110";
        wait for 100ns;

        num2 <= "1111";
        wait for 100ns;

        num1 <= "0010";
        wait for 100ns;

        num2 <= "0011";
        wait for 100ns;

        num1 <= "0011";
        wait for 100ns;

        EN <= '0';
        num1 <= "0101";
        num2 <= "1010";
        wait for 100ns;

        num1 <= "1110";
        wait for 100ns;

        num2 <= "1111";
        wait for 100ns;

        num1 <= "0010";
        wait for 100ns;

        num2 <= "0011";
```

```
        wait for 100ns;

        num1 <= "0011";

        wait;

    end process;
end Behavioral;
```

- Timing Diagram for Comp_4_bit:



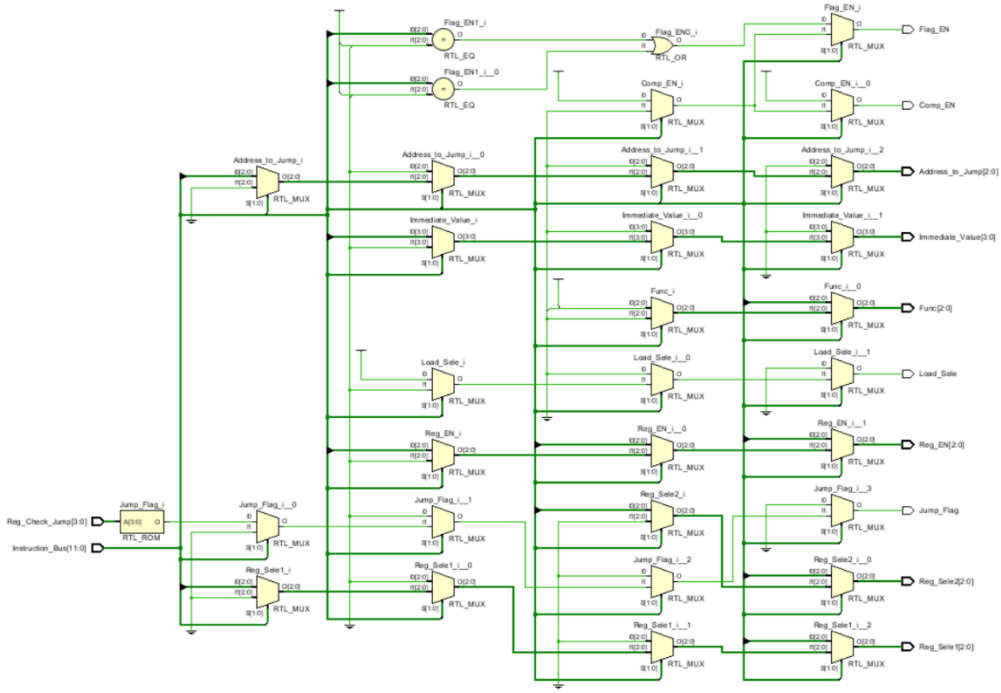
5. INSTRUCTION_DECORDER

- Design source VHDL code of INSTRUCTION_DECORDER :

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

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

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

entity INSTRUCTION_DECODER is
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Func : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Sele : out STD_LOGIC;
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Flag_EN : out STD_LOGIC; -- Enable the Flags in ALU
           Comp_EN : out STD_LOGIC; -- Enable the Comparator in ALU
           Jump_Flag : out STD_LOGIC;
           Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end INSTRUCTION_DECODER;

architecture Behavioral of INSTRUCTION_DECODER is

signal Operator : std_logic_vector (1 downto 0);
signal FuncValue : std_logic_vector (2 downto 0);

begin
    Operator <= Instruction_Bus(11 downto 10);

    process (Operator, Instruction_Bus, Reg_Check_Jump,FuncValue) begin
        Jump_Flag <= '0';
        Flag_EN <= '0';
        Comp_EN <= '0';
        Load_Sele <= '0';
        Reg_Sele1 <= "000";
        Reg_Sele2 <= "000";
        Immediate_Value <= "0000";
        Reg_EN <= "000";
        Address_to_Jump <= "000";
```

```
        Func <= "000";

        FuncValue <= Instruction_Bus(2 downto 0);

        if Operator = "00" then
            Comp_EN <= '1';
            Reg_Sele1 <= Instruction_Bus(9 downto 7);
            Reg_Sele2 <= Instruction_Bus(6 downto 4);
            Reg_EN <= Instruction_Bus(9 downto 7);
            Func <= FuncValue;
            if ( FuncValue = "010" or FuncValue = "110" ) then
                Flag_EN <= '1';
            end if;

        elsif Operator = "01" then
            Comp_EN <= '1';
            Reg_EN <= Instruction_Bus(9 downto 7);
            Flag_EN <= '1';
            Reg_Sele2 <= Instruction_Bus(9 downto 7);
            Func <= "110";

        elsif Operator = "10" then

            Immediate_Value <= Instruction_Bus(3 downto 0);
            Reg_EN <= Instruction_Bus(9 downto 7);
            Load_Sele <= '1';

        elsif Operator = "11" then
            Address_to_Jump <= Instruction_Bus(2 downto 0);
            Reg_Sele1 <= Instruction_Bus(9 downto 7);
            if Reg_Check_Jump = "0000" then
                Jump_Flag <= '1';
            end if;
        end if;
    end process;

end Behavioral;
```

- RTL Schematic diagram of the INSTRUCTION_DECORDER :



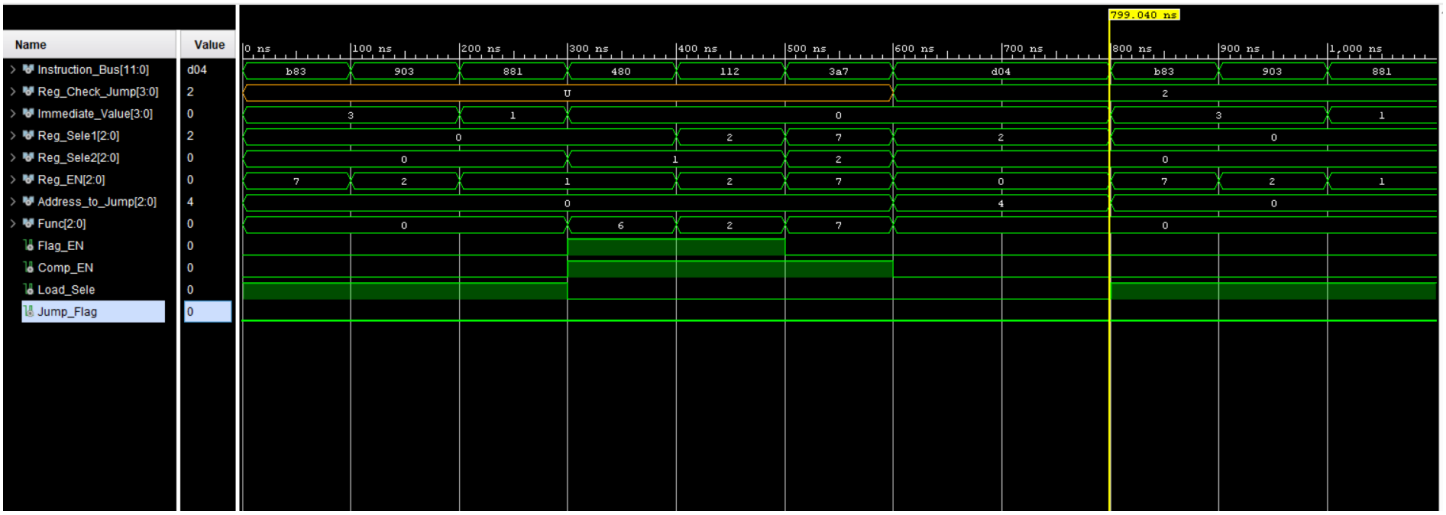
- Behavioral simulation Code for INSTRUCTION_DECORDER:

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

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

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

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

architecture Behavioral of TB_INSTRUCTION_DECODER is

component INSTRUCTION_DECODER
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
            Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
            Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
            Func : out STD_LOGIC_VECTOR (2 downto 0);
            Load_Sele : out STD_LOGIC;
            Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
            Flag_EN : out STD_LOGIC; -- Enable the Flags in ALU
            Comp_EN : out STD_LOGIC; -- Enable the Comparator in ALU
            Jump_Flag : out STD_LOGIC;
            Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal Instruction_Bus : STD_LOGIC_VECTOR (11 downto 0);
signal Reg_Check_Jump, Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Reg_Sele1, Reg_Sele2, Reg_EN, Address_to_Jump, Func : STD_LOGIC_VECTOR (2 downto
0);
signal Flag_EN,Comp_EN, Load_Sele, Jump_Flag : STD_LOGIC;

begin
UUT : INSTRUCTION_DECODER port map(
        Instruction_Bus => Instruction_Bus,
        Reg_Check_Jump => Reg_Check_Jump,
        Flag_EN => Flag_EN,
        Func => Func,
        Comp_EN => Comp_EN,
        Reg_Sele1 => Reg_Sele1,
        Reg_Sele2 => Reg_Sele2,
        Immediate_Value => Immediate_Value,
        Load_Sele => Load_Sele,
        Reg_EN => Reg_EN,
        Jump_Flag => Jump_Flag,
        Address_to_Jump => Address_to_Jump
    );


stimulus_process: process
    begin

    Instruction_Bus <= "101110000011";
    wait for 100 ns;

    Instruction_Bus <= "100100000011";
    wait for 100 ns;

    Instruction_Bus <= "100010000001";
    wait for 100 ns;

    Instruction_Bus <= "010010000000";
    wait for 100 ns;
```

```
    Instruction_Bus <= "000100010010";
    wait for 100 ns;

    Instruction_Bus <= "001110100111";
    wait for 100 ns;

    Instruction_Bus <= "110100000100";
    Reg_Check_Jump <= "0010";
    wait for 100 ns;

    Reg_Check_Jump <= "0010";
    wait for 100 ns;

    end process stimulus_process;

    end Behavioral;
```

- Timing Diagram for INSTRUCTION_DECORDER:



6. NANO_PROCESSOR

- Design source VHDL code of NANO_PROCESSOR:

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

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

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

entity NANO_PROCESSOR is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0);
           Comp_equal : out STD_LOGIC;
           Comp_greater : out STD_LOGIC;
           Comp_lesser : out STD_LOGIC);
end NANO_PROCESSOR;

architecture Behavioral of NANO_PROCESSOR is

component REGISTER_BANK
    Port (Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          MUX_Out : in STD_LOGIC_VECTOR (3 downto 0);
          Reset : in STD_LOGIC;
          R0_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R1_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R2_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R3_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R4_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R5_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R6_Out : out STD_LOGIC_VECTOR (3 downto 0);
          R7_Out : out STD_LOGIC_VECTOR (3 downto 0)
          );
end component;

component ALU
    port (
          A : in std_logic_vector (3 downto 0);
          B : in std_logic_vector (3 downto 0);
          Flag_EN : in std_logic;
          Comp_EN : in std_logic;
```

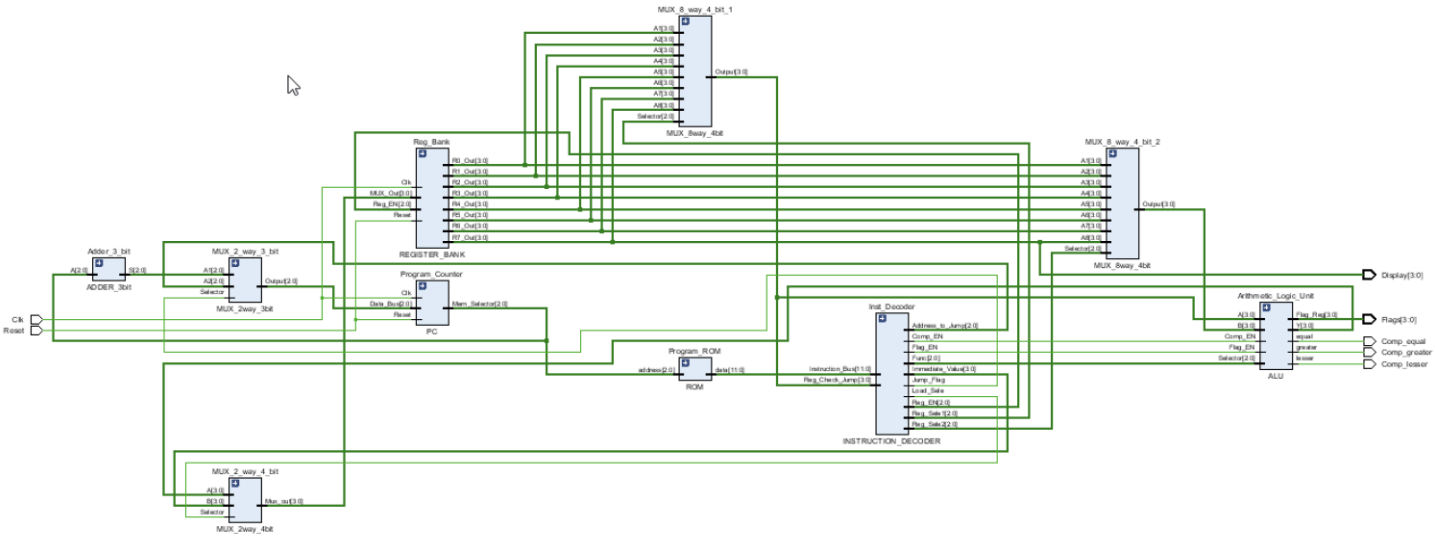
```
          Selector : in std_logic_vector (2 downto 0);
          Y : out std_logic_vector (3 downto 0);
          Flag_Reg : out STD_LOGIC_VECTOR (3 downto 0);
          equal : out STD_LOGIC;
          greater : out STD_LOGIC;
          lesser : out STD_LOGIC
          );
end component;

component ADDER_3bit
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC);
end component;

component PC
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Data_Bus: in STD_LOGIC_VECTOR (2 downto 0);
           Mem_Selector : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component INSTRUCTION_DECODER
    Port ( Instruction_Bus : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Sele1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Sele2 : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Func : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Sele : out STD_LOGIC;
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Flag_EN : out STD_LOGIC;
           Comp_EN : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address_to_Jump : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component ROM
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component MUX_2way_3bit
    Port ( A1 : in STD_LOGIC_VECTOR (2 downto 0);
           A2 : in STD_LOGIC_VECTOR (2 downto 0);
           Selector : in STD_LOGIC;
           Output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component MUX_2way_4bit
```

```
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC;
           Mux_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component MUX_8way_4bit
    Port ( A1 : in STD_LOGIC_VECTOR (3 downto 0);
           A2 : in STD_LOGIC_VECTOR (3 downto 0);
           A3 : in STD_LOGIC_VECTOR (3 downto 0);
           A4 : in STD_LOGIC_VECTOR (3 downto 0);
           A5 : in STD_LOGIC_VECTOR (3 downto 0);
           A6 : in STD_LOGIC_VECTOR (3 downto 0);
           A7 : in STD_LOGIC_VECTOR (3 downto 0);
           A8 : in STD_LOGIC_VECTOR (3 downto 0);
           Selector : in STD_LOGIC_VECTOR (2 downto 0);
           Output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal Load_sele, Jump_Flag, Flag_EN_ALU, Comp_EN_ALU : std_logic;

signal Mem_Selector, Output_2_3bit_MUX ,Address_to_Jump, Output_3bit_Adder :
std_logic_vector(2 downto 0);
signal Selector_8_4bit_MUX_1, Selector_8_4bit_MUX_2, Reg_EN: std_logic_vector(2 downto
0);

signal Instruction_Bus: std_logic_vector(11 downto 0);

signal Output_8_4bit_MUX_1,Output_8_4bit_MUX_2, Immediate_Value ,Output_2_4bit_MUX :
std_logic_vector(3 downto 0);
signal R0_Out,R1_Out,R2_Out,R3_Out,R4_Out,R5_Out,R6_Out,R7_Out : std_logic_vector (3
downto 0);

signal ALU_Output, Flag_Reg  : std_logic_vector (3 downto 0);
signal Func  : std_logic_vector (2 downto 0);
begin

    Program_Counter : PC
        Port map (
            Clk => Clk,
            Reset => Reset,
            Data_Bus => Output_2_3bit_MUX,
            Mem_Selector => Mem_Selector);

     Program_ROM : ROM
         Port map (
            address => Mem_Selector,
            data => Instruction_Bus);

     MUX_2_way_3_bit : MUX_2way_3bit
```

```
        Port map (
            A1 => Output_3bit_Adder,
            A2 => Address_to_Jump,
            Selector => Jump_Flag,
            Output => Output_2_3bit_MUX);

    Inst_Decoder : INSTRUCTION_DECODER
        Port map (
            Instruction_Bus  => Instruction_Bus,
            Reg_Check_Jump => Output_8_4bit_MUX_1,
            Func => Func,
            Reg_Sele1 => Selector_8_4bit_MUX_1,
            Reg_Sele2 => Selector_8_4bit_MUX_2,
            Immediate_Value => Immediate_Value,
            Load_Sele => Load_sele,
            Reg_EN => Reg_EN,
            Flag_EN => Flag_EN_ALU,
            Comp_EN => Comp_EN_ALU,
            Jump_Flag => Jump_Flag,
            Address_to_Jump => Address_to_Jump);

    Reg_Bank : REGISTER_BANK
        Port map (
            Reg_EN => Reg_EN,
            Clk => Clk,
            MUX_Out => Output_2_4bit_MUX,
            Reset => Reset,
            R0_Out => R0_Out,
            R1_Out => R1_Out,
            R2_Out => R2_Out,
            R3_Out => R3_Out,
            R4_Out => R4_Out,
            R5_Out => R5_Out,
            R6_Out => R6_Out,
            R7_Out => R7_Out
            );

    MUX_2_way_4_bit : MUX_2way_4bit
        Port map (
            A => ALU_Output,
            B => Immediate_Value,
            Selector => Load_sele,
            Mux_out => Output_2_4bit_MUX );

     MUX_8_way_4_bit_1 : MUX_8way_4bit
        Port map (
            A1=>R0_Out,
            A2=>R1_Out,
            A3=>R2_Out,
            A4=>R3_Out,
```

```
            A5=>R4_Out,
            A6=>R5_Out,
            A7=>R6_Out,
            A8=>R7_Out,
            Selector => Selector_8_4bit_MUX_1,
            Output => Output_8_4bit_MUX_1 );

    MUX_8_way_4_bit_2 : MUX_8way_4bit
       Port map (
            A1=>R0_Out,
            A2=>R1_Out,
            A3=>R2_Out,
            A4=>R3_Out,
            A5=>R4_Out,
            A6=>R5_Out,
            A7=>R6_Out,
            A8=>R7_Out,
            Selector => Selector_8_4bit_MUX_2,
            Output => Output_8_4bit_MUX_2);

    Arithmetic_Logic_Unit : ALU
       port map(
            A => Output_8_4bit_MUX_1,
            B => Output_8_4bit_MUX_2,
            Selector => Func,
            Flag_EN => Flag_EN_ALU,
            Comp_EN => Comp_EN_ALU,
            Y => ALU_Output,
            Flag_Reg => Flag_Reg,
            equal => Comp_equal,
            greater => Comp_greater,
            lesser => Comp_lesser);


    Adder_3_bit : ADDER_3bit
       Port map (
            A => Mem_Selector,
            S => Output_3bit_Adder );

Display <= R7_Out;
Flags <= Flag_Reg;

end Behavioral;
```

- RTL Schematic diagram of the NANO_PROCESSOR:

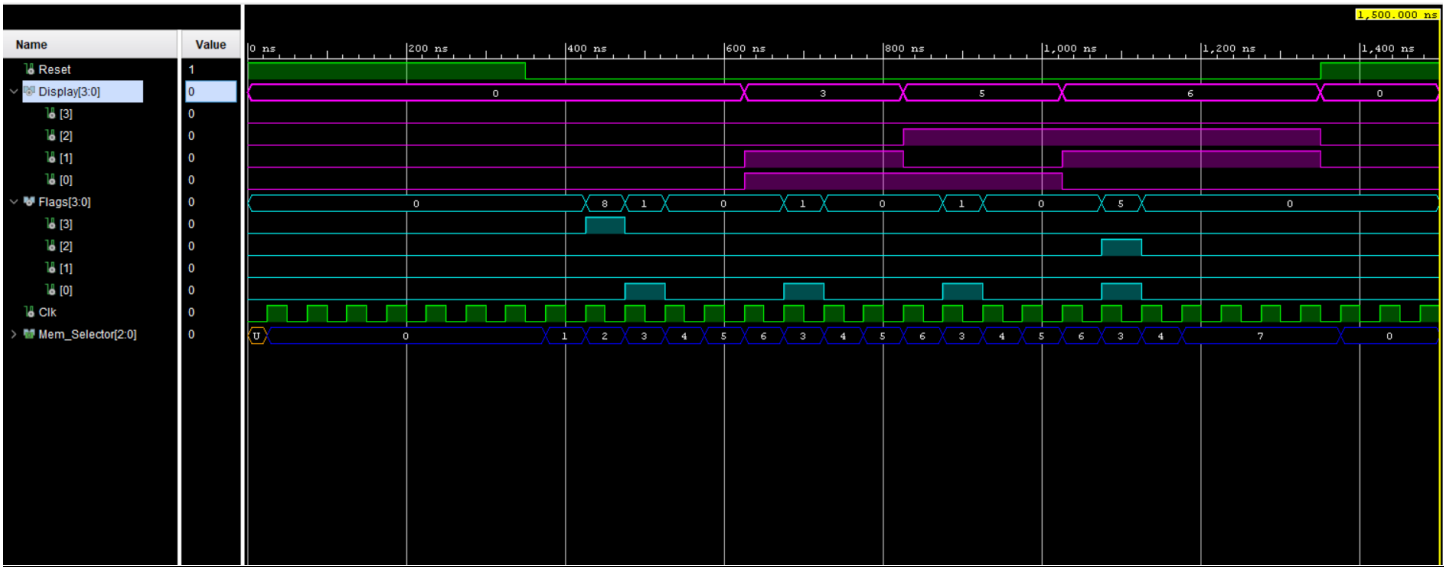


- Behavioral simulation Code for NANO_PROCESSOR:

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

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

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

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

architecture Behavioral of TB_NANO_PROCESSOR is
```

```
component NANO_PROCESSOR
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0));
end component;

signal Reset : std_logic;
signal Flags : std_logic_vector (3 downto 0);
signal Display : std_logic_vector (3 downto 0);
signal Clk : std_logic := '1';

begin
    UUT : NANO_PROCESSOR
        port map (
            Clk => Clk,
            Reset => Reset,
            Display => Display,
            Flags => Flags);

    process begin
        Clk <= not Clk;
        wait for 25 ns;
    end process;

    process begin
        Reset <= '1';
        wait for 350 ns;

        Reset <= '0';
        wait for 1000 ns;

        Reset <= '1';
        wait;

     end process;
end Behavioral;
```

- Timing Diagram for NANO_PROCESSOR:



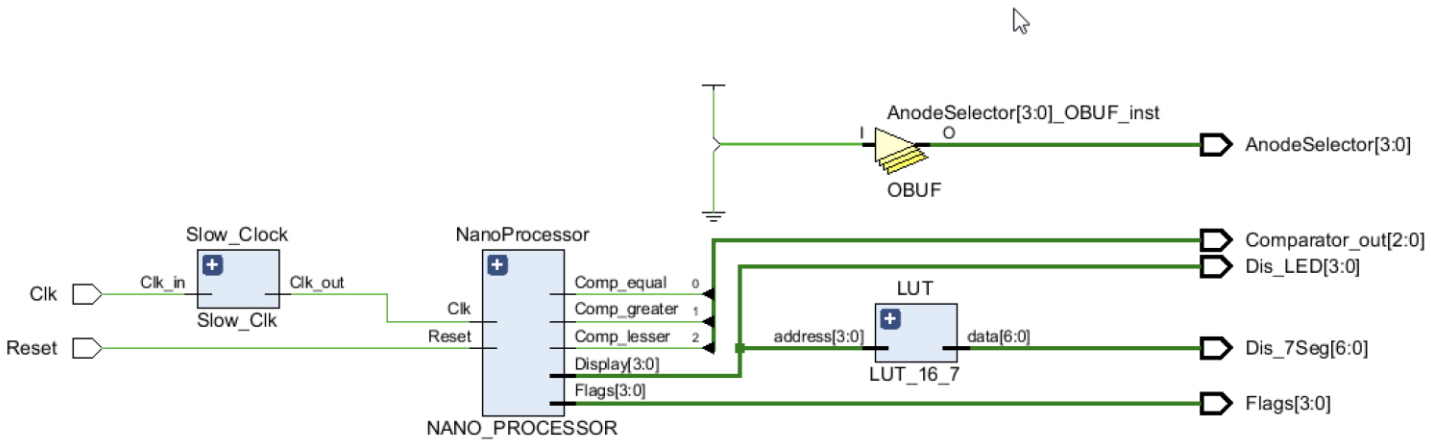
7. MAIN

- Design source VHDL code of MAIN :

```
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 MAIN is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Dis_LED : out STD_LOGIC_VECTOR (3 downto 0);
           Dis_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           Flags : out STD_LOGIC_VECTOR (3 downto 0);
           Comparator_out : out STD_LOGIC_VECTOR (2 downto 0);
           AnodeSelector : out STD_LOGIC_VECTOR (3 downto 0)
          );
end MAIN;

architecture Behavioral of MAIN is

component NANO_PROCESSOR is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Display : out STD_LOGIC_VECTOR (3 downto 0);
           Flags : out STD_LOGIC_VECTOR( 3 downto 0);
           Comp_equal : out STD_LOGIC;
           Comp_greater : out STD_LOGIC;
           Comp_lesser : out STD_LOGIC);
end component;

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

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

signal Display_out : STD_LOGIC_VECTOR (3 downto 0);
```

```
signal SlowClk_out : std_logic;

begin
    Slow_Clock : Slow_Clk
        Port map(
            Clk_in => Clk,
            Clk_out => SlowClk_out
         );

    NanoProcessor : NANO_PROCESSOR
         PORT MAP(
             Clk => SlowClk_out,
             Reset => Reset,
             Display => Display_out,
             Flags => Flags,
             Comp_equal => Comparator_out(0),
             Comp_greater => Comparator_out(1),
             Comp_lesser => Comparator_out(2)
          );


    LUT : LUT_16_7
        port map(
            address => Display_out,
            data => Dis_7Seg
         );

   Dis_LED <= Display_out;
   AnodeSelector <= "1110";

end Behavioral;
```

- RTL Schematic diagram of the MAIN :

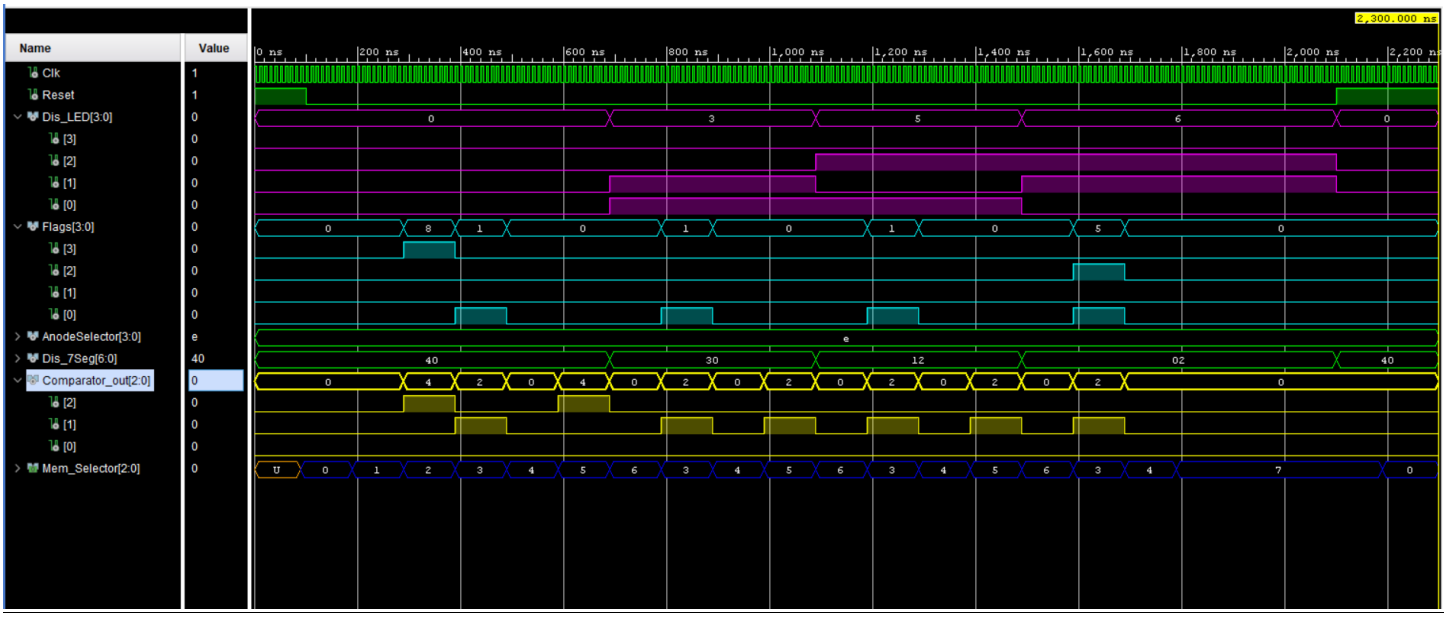


- Behavioral simulation Code for MAIN :

```
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_MAIN is
--  Port ( );
end TB_MAIN;

architecture Behavioral of TB_MAIN is

component MAIN
        Port (  Clk : in STD_LOGIC;
                Reset : in STD_LOGIC;
                Dis_LED : out STD_LOGIC_VECTOR (3 downto 0);
                Dis_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
                Flags : out STD_LOGIC_VECTOR (3 downto 0);
                Comparator_out : out STD_LOGIC_VECTOR (2 downto 0);
                AnodeSelector : out STD_LOGIC_VECTOR (3 downto 0)
              );
end component;

signal Clk : std_logic := '0';
signal Reset : std_logic;
signal Dis_LED, Flags, AnodeSelector : std_logic_vector (3 downto 0);
signal Dis_7Seg : std_logic_vector (6 downto 0);
signal Comparator_out : std_logic_vector (2 downto 0);

begin
    UUT : MAIN
        port map (
            Clk => Clk,
            Reset => Reset,
            Dis_LED => Dis_LED,
            Dis_7Seg => Dis_7Seg,
            Comparator_out => Comparator_out,
            Flags => Flags,
            AnodeSelector => AnodeSelector
         );

process begin
```

```
    Clk <= not Clk;
    wait for 5 ns;
end process;

process begin
    Reset <= '1';
    wait for 100 ns;

    Reset <= '0';
    wait for 2000 ns;

    Reset <= '1';
    wait;
end process;

end Behavioral;
```

- Timing Diagram for MAIN :

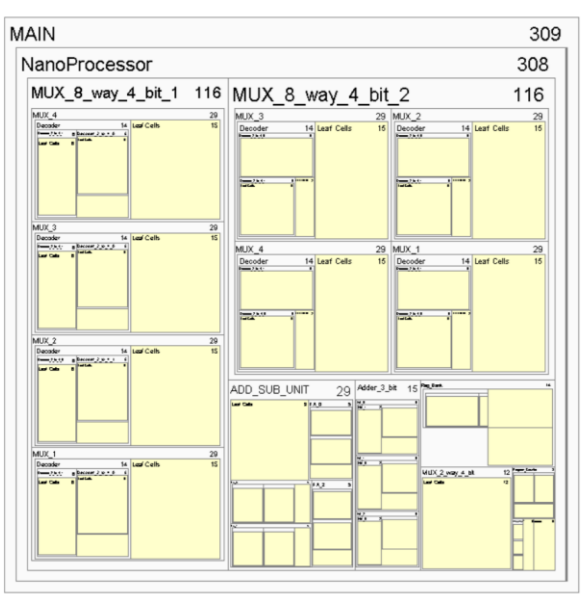


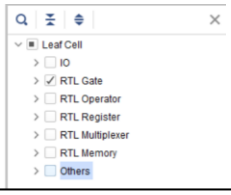
- Constraint File

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

## LEDs
set_property PACKAGE_PIN U16 [get_ports {Dis_LED[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {Dis_LED[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {Dis_LED[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {Dis_LED[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_LED[3]}]

set_property PACKAGE_PIN U14 [get_ports {Comparator_out[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Comparator_out[0]}]
set_property PACKAGE_PIN V14 [get_ports {Comparator_out[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Comparator_out[1]}]
set_property PACKAGE_PIN V13 [get_ports {Comparator_out[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Comparator_out[2]}]

set_property PACKAGE_PIN P3 [get_ports {Flags[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[3]}]
set_property PACKAGE_PIN N3 [get_ports {Flags[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[1]}]
set_property PACKAGE_PIN P1 [get_ports {Flags[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[2]}]
set_property PACKAGE_PIN L1 [get_ports {Flags[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Flags[0]}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {Dis_7Seg[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {Dis_7Seg[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {Dis_7Seg[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {Dis_7Seg[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {Dis_7Seg[4]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {Dis_7Seg[5]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {Dis_7Seg[6]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Dis_7Seg[6]}]
```

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


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


set_property CFGBVS VCCO [current_design]

set_property CONFIG_VOLTAGE 3.3 [current_design]
```

- Optimization caused by the MUXes with tri-state
- We reduce the RTL logic gates count by a significant number. However, there is a slight increment in resource utilization (LUTs).
- ***Before using tri-state buffers for the multiplexes (for version 1)***

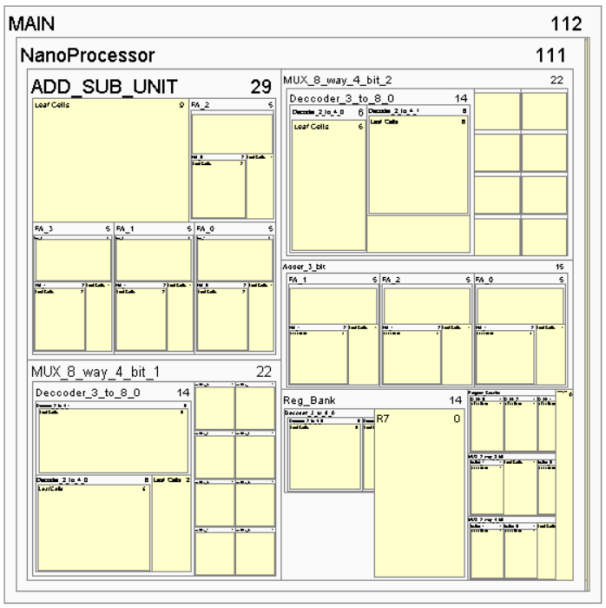


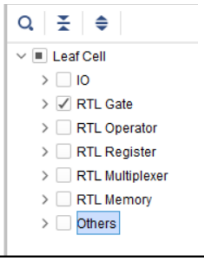



```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |   46 |     0 |     20800 |  0.22 |
|   LUT as Logic          |   46 |     0 |     20800 |  0.22 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |   49 |     0 |     41600 |  0.12 |
|   Register as Flip Flop |   49 |     0 |     41600 |  0.12 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

When 8-way-4-bit mux is implemented by using 8:1 mux

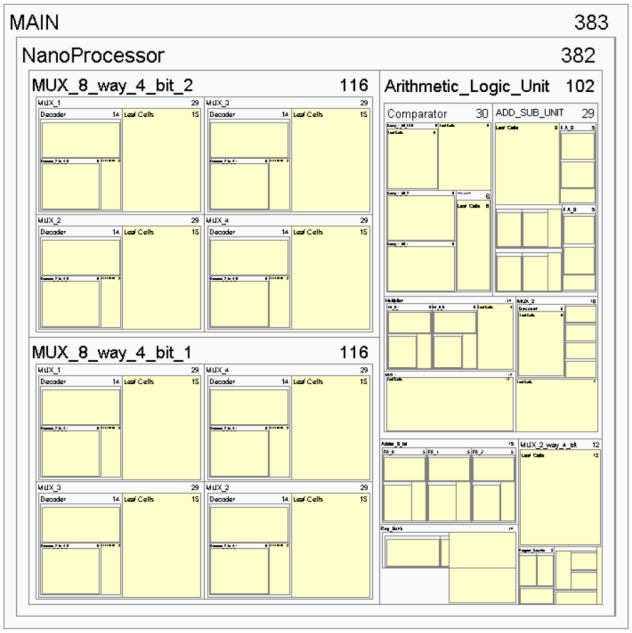
- ***After using tri-state buffers for the multiplexes***

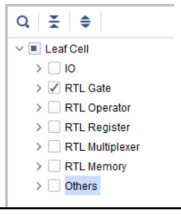





```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |   74 |     0 |     20800 |  0.36 |
|   LUT as Logic          |   74 |     0 |     20800 |  0.36 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |   49 |     0 |     41600 |  0.12 |
|   Register as Flip Flop |   49 |     0 |     41600 |  0.12 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

- ***Before using tri-state buffers for the multiplexes (for version 2)***



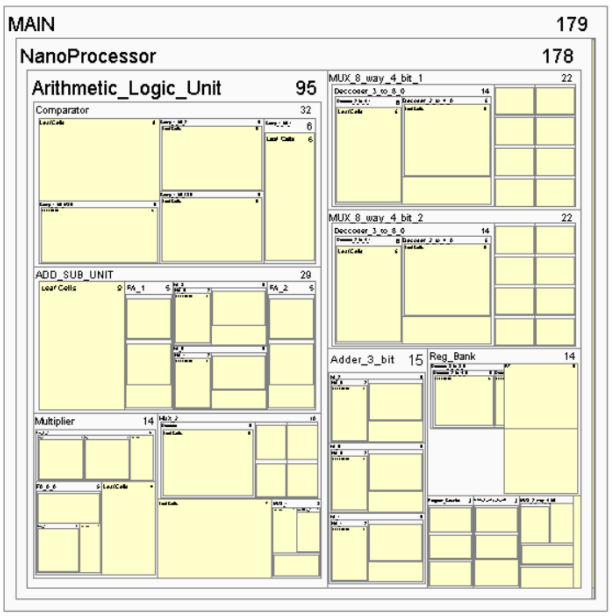


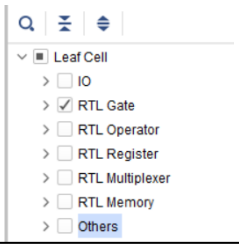
1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	62	0	20800	0.30
LUT as Logic	62	0	20800	0.30
LUT as Memory	0	0	9600	0.00
Slice Registers	50	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	1	0	41600	<0.01
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

When 8-way-4-bit mux is implemented by using 8:1 mux

- ***After using tri-state buffers for the multiplexes***





1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	110	0	20800	0.53
LUT as Logic	110	0	20800	0.53
LUT as Memory	0	0	9600	0.00
Slice Registers	49	0	41600	0.12
Register as Flip Flop	49	0	41600	0.12
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

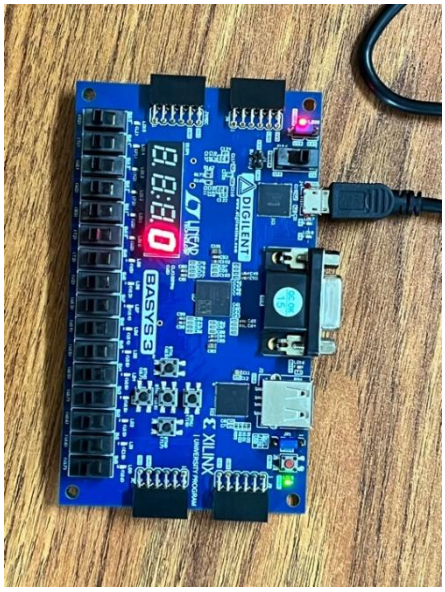
- Extended Tasks To Assure the Operativity of the ALU:

- **Task 1**

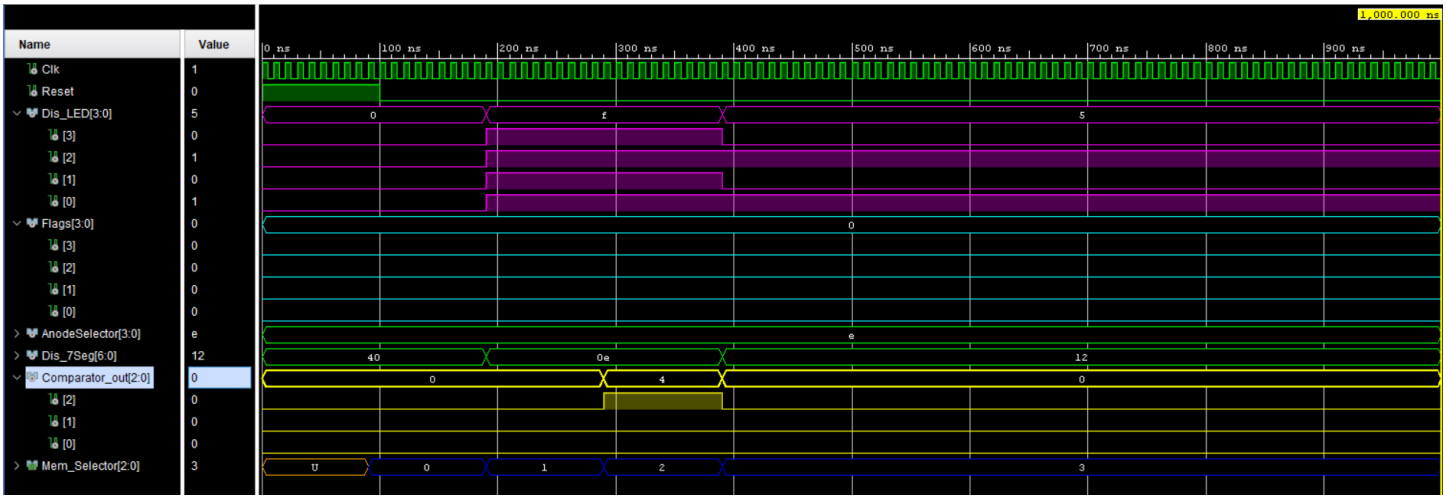
- Assembly Code:

```
;      process => (-1) AND 5 = 5

MOVI R7,-1      ; R7 <- -1
MOVI R1,5       ; R1 <- 5
AND  R7,R1      ; R7 <- R7 AND R1
JZR  R0,3       ; Jump to line 4 if R0 = 0
```

- Functionality :



- Timing Diagram:

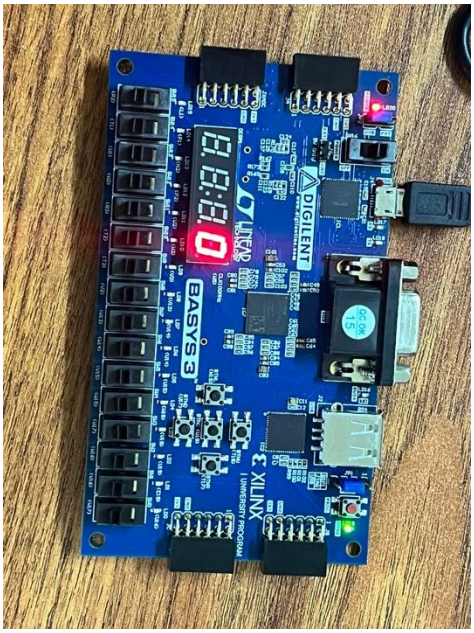
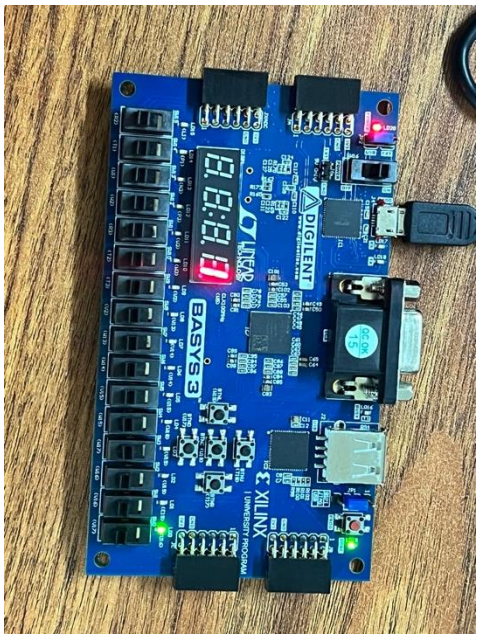
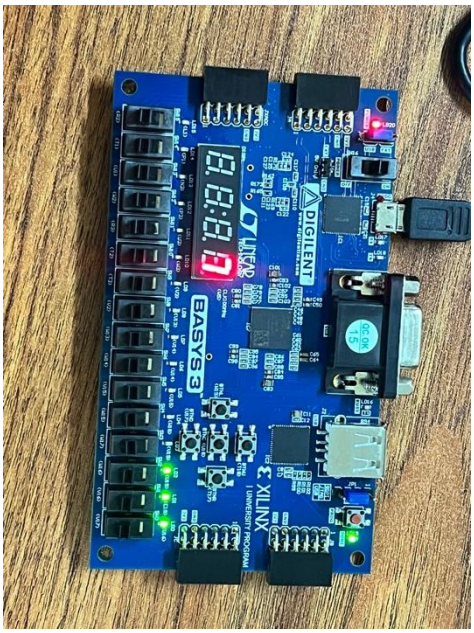


- ## Task 2

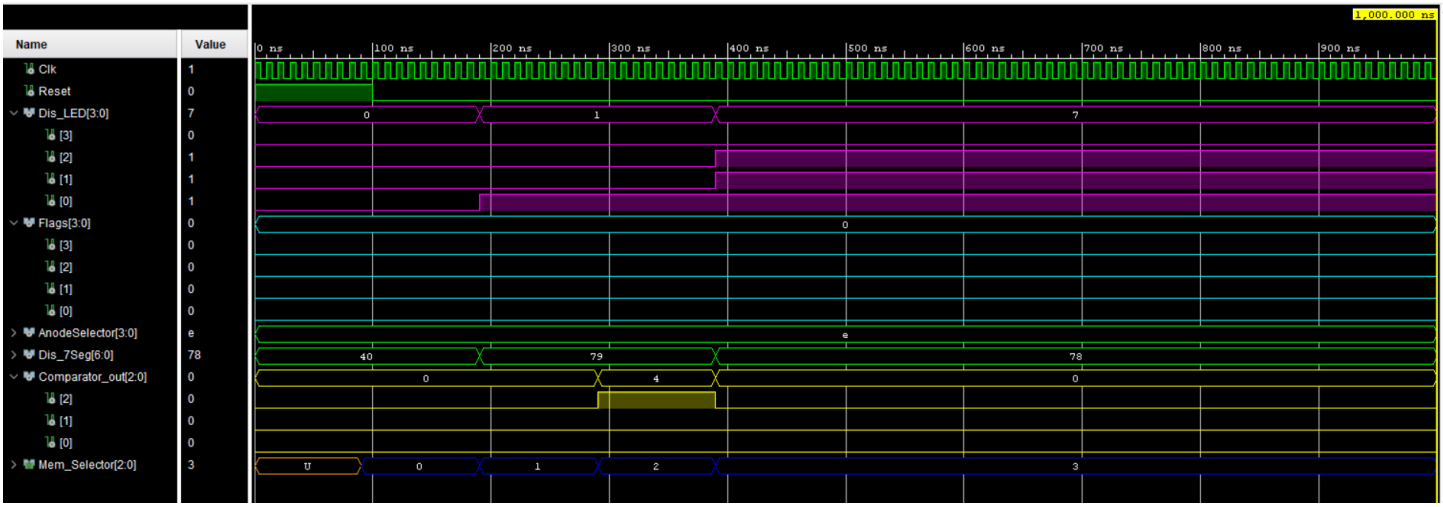
- Assembly Code:

```
;      process => 1 OR 7 = 7

MOVI R7,1      ; R7 <- 1
MOVI R1,7      ; R1 <- 7
OR   R7,R1     ; R7 <- R7 OR  R1
JZR  R0,3      ; Jump to line 4 if R0 = 0
```

- Functionality :

- Timing Diagram:



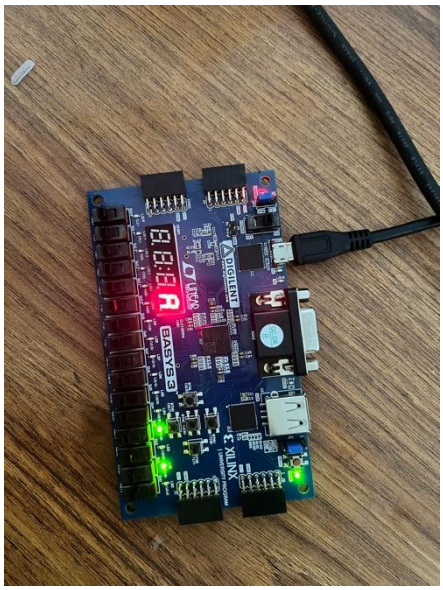
- ## Task 3

- Assembly Code:

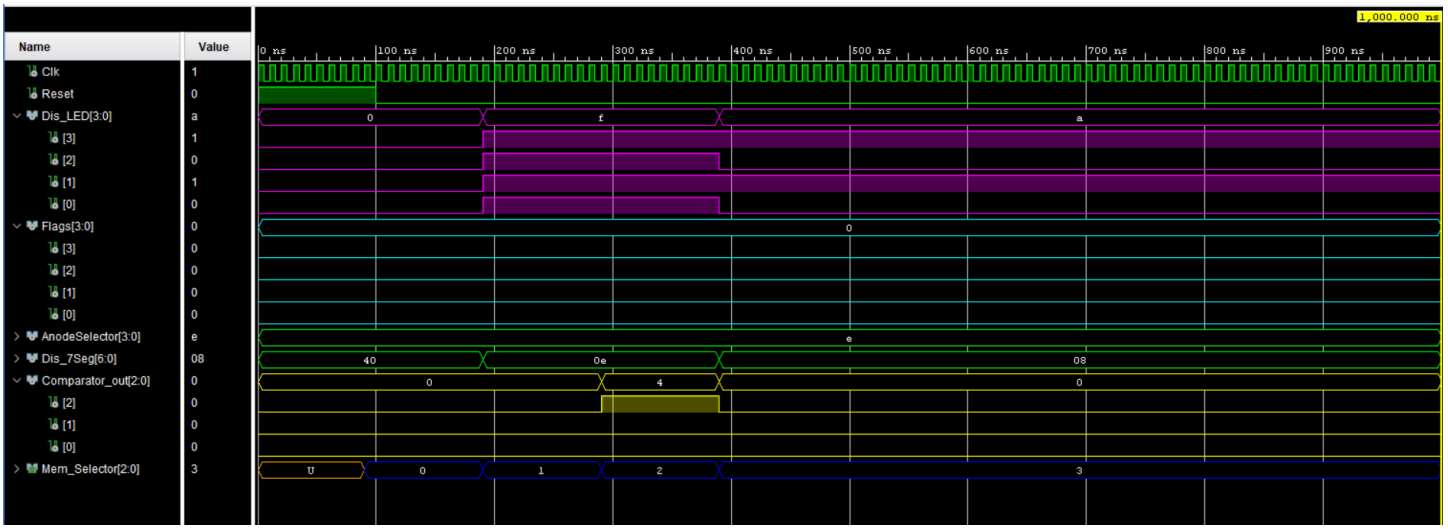
```
;     process => (-1) ANDN 5 = (-6)

MOVI R7,-1     ; R7 <- -1
MOVI R1,5      ; R1 <- 5
ANDN R7,R1     ; R7 <- R7 AND NOT(R1)
JZR  R0,3      ; Jump to line 4 if R0 = 0
```

- Functionality :





- Timing Diagram:

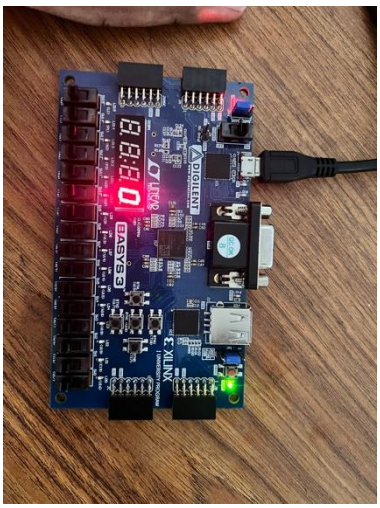
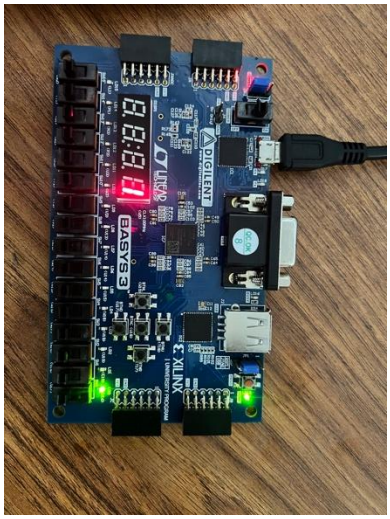


- ## Task 4

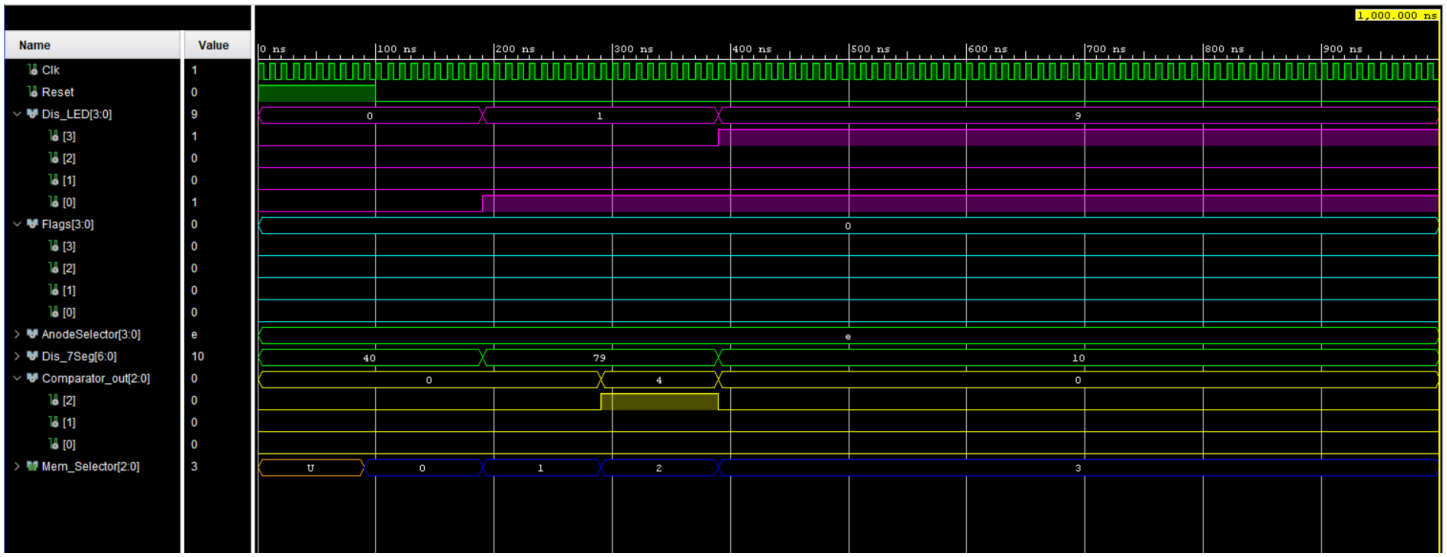
- Assembly Code:

```
;      process => 1 ORN 7 = (-7)

MOVI R7,1       ; R7 <- 1
MOVI R1,7       ; R1 <- 7
ORN  R7,R1      ; R7 <- R7 OR NOT(R1)
JZR  R0,3       ; Jump to line 4 if R0 = 0
```

- Functionality :


- Timing Diagram:

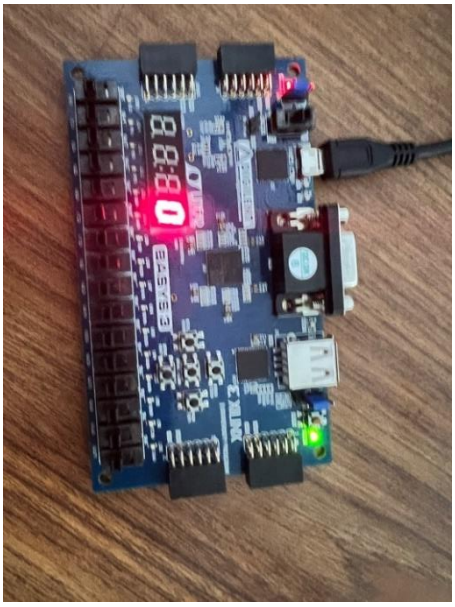


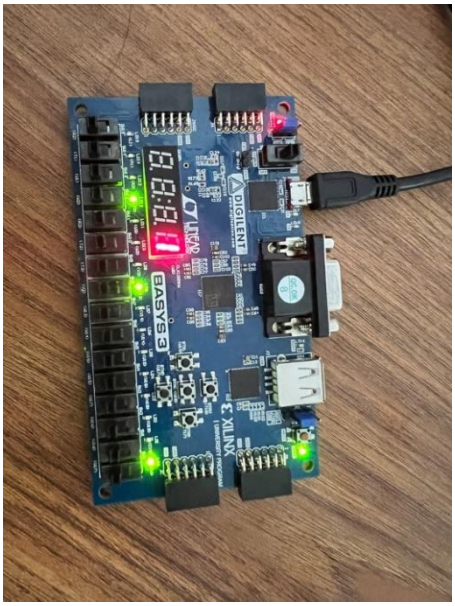
- ## Task 5

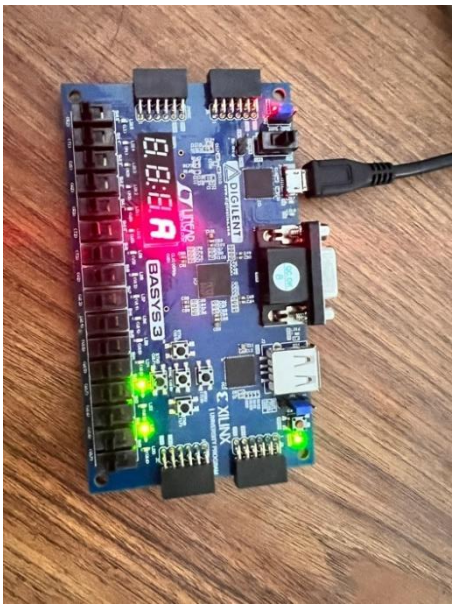
- Assembly Code:

```
;      process => 1 - 7 = (-6)

MOVI  R7,1      ; R7 <- 1
MOVI  R1,7      ; R1 <- 7
SUB   R7,R1     ; R7 <- R7 - R1
JZR   R0,3      ; Jump to line 4 if R0 = 0
```

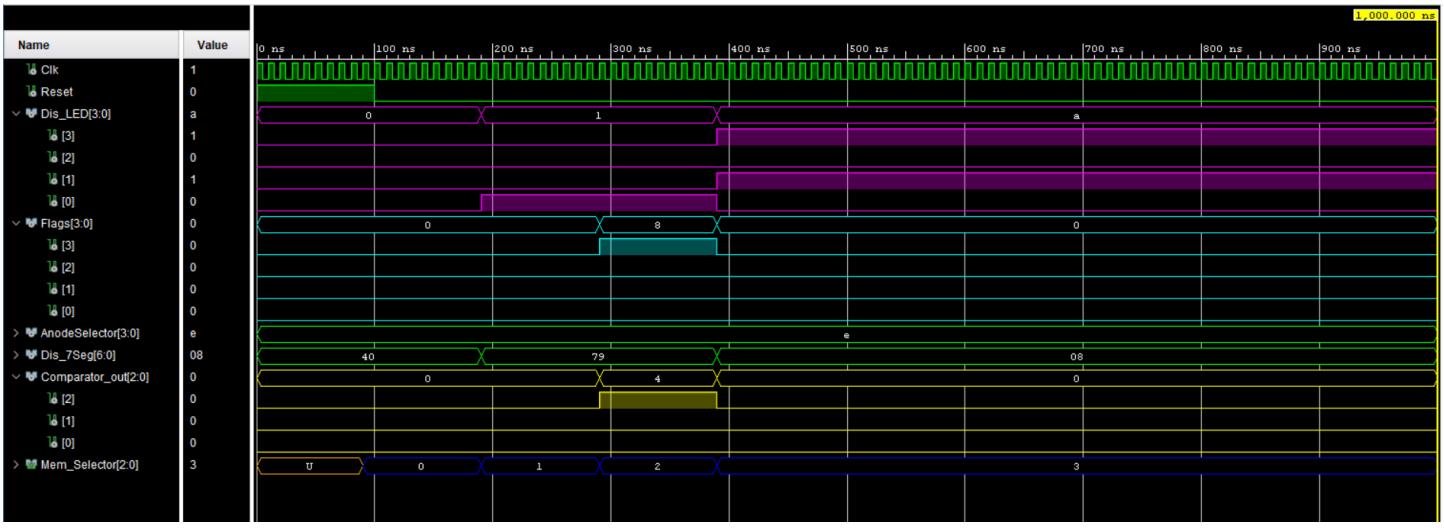
- Functionality :







- Timing Diagram:

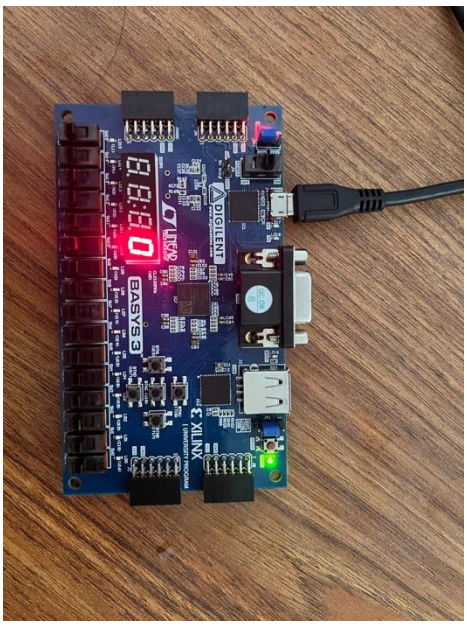


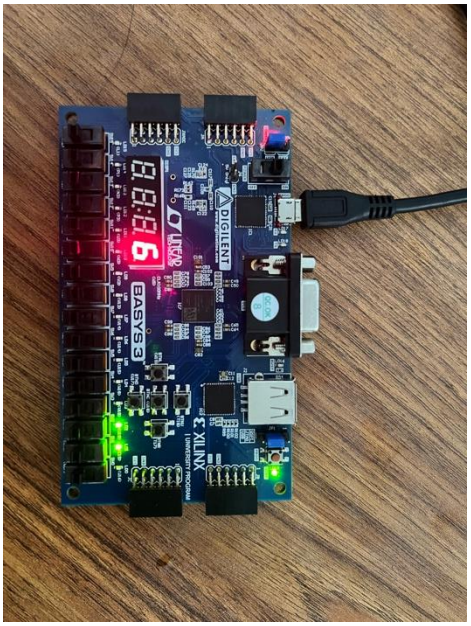
- ## Task 6

- Assembly Code:

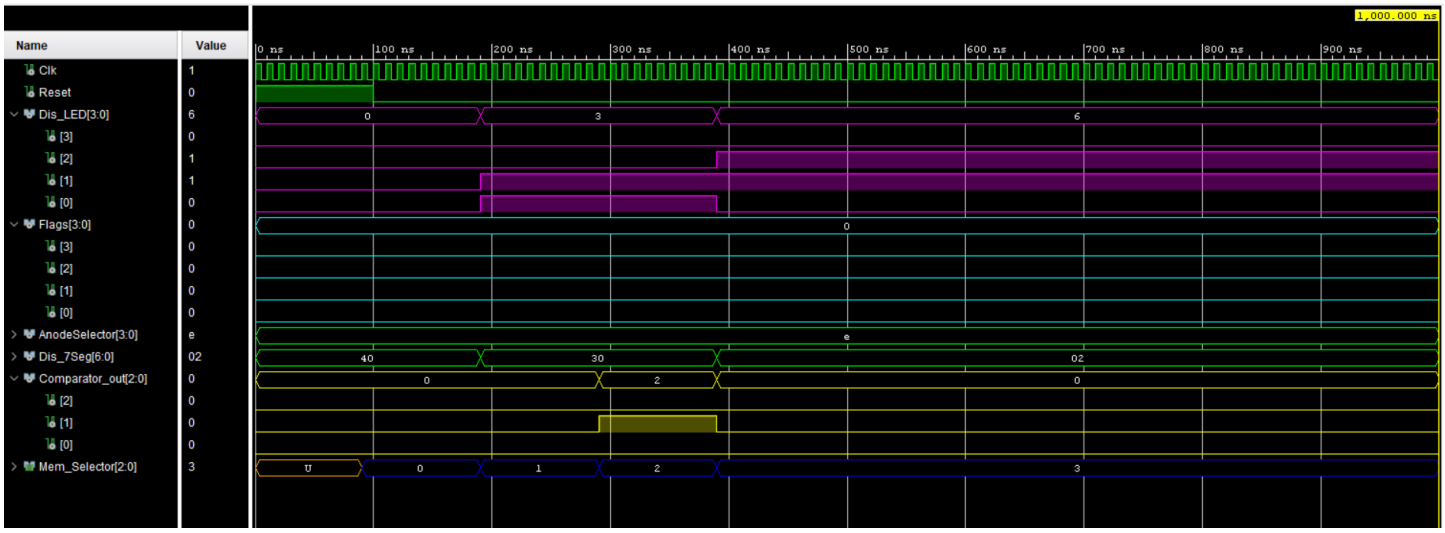
```
;      process => 3 x 2 = 6

MOVI R7,3      ; R7 <- 7
MOVI R1,2      ; R1 <- 2
MUL  R7,R1     ; R7 <- R7 x R1
JZR  R0,3      ; Jump to line 4 if R0 = 0
```

- Functionality :





- Timing Diagram:



Future Suggestions:

- In our extended instruction set, there are some instructions we didn't use for the design and those can be used to extend the design with extra functionalities.

Constraints

- In our Multiplier in ALU, We can only multiply numbers in this set {0,0,1,1,2,2,3} because we can only use 3 bits from the 4 bits. So the multiplied value that can be displayed has to be between -7 to 7.

Contributions

Member	Individual Contribution	Worked Time
RATHNAYAKE R.M.I.B. 220526N	• Instruction Decoder • 4-bit Add/Sub Unit • Arithmetic Logic Unit • Multiplier. • Design and port mapping of Nano processor and Main. • Test bench files of Instruction Decoder, 4-bit Add/Sub Unit, Arithmetic Logic Unit, Multiplier, Nano processor. • Report assembling by getting each component's TB files and time diagrams of the minimum required Nano processor. • Constraint files Basys3.xdc	45 hours
JAYATHUNGA R.D.S.S. 220277B	• Program rom • Program Counter • Arithmetic Logic Unit. • Design and port mapping of Nano processor and Main. • Test bench files of Program rom, Program Counter, Arithmetic Logic Unit, Main. • Extended Tasks of ALU. • Report assembling by getting each component's Source files and elaborated design files of the minimum required Nano processor.	40 hours

DISSANAYAKE C.N. 220133G	• Register bank • Register 4-bit • 3-bit adder • Comp 1 bit • Comp 4 bit. • Test bench files of Register bank, 3-bit adder, Comp 4-bit. • Design and port mapping of Nano processor and Main. • Report assembling by getting each component's Source files, elaborated design files, TB files, and time diagrams of the extended Nano processor. • Report finalize.	40 hours
RAVISHAN A.D.P. 220532E	• Mux 2 way 4 bit • Mux 2 way 3 bit • Mux 8 way 4 bit. • Tri-state buffer. • Test bench files of Mux 2 way 4 bit, Mux 2 way 3 bit, Mux 8 way 4 bit, Tri-state buffer. • Design and port mapping of Nano processor and Main. • Report assembling by getting each component's Source files, elaborated design files, TB files, and time diagrams of the extended Nano processor. • Report finalize.	35 hours

• Conclusion

- The nano processor design task was done by using our module contents and all the labs we have done before. There we learned how to design various types of components for specific tasks and assemble them all to make a large design to fulfill given tasks. Since this was a group project, we learned how to divide responsibilities with each other and how to do group work with unity, and also we learned how to face problems as a group. While we doing this project we had many problems as a team. The different components were designed by different members of our group and when we assembled them to make the main design, it was a much difficult task for us to make through. Also, we made some decisions to make this design with different ideas, we designed all Multiplexers by using Tri State Buffers instead of the logic we learned in the labs. Also, we first designed the instruction decoder without setting default values to the outputs which caused some unexpected behaviors of the component and then we solved it by supplying default values for the outputs. The design finally came up with better results after solving the problems we had.