# NanoProcessor Design

**Group Number** – 01

**Group Members :**


Parishith R – 220444K

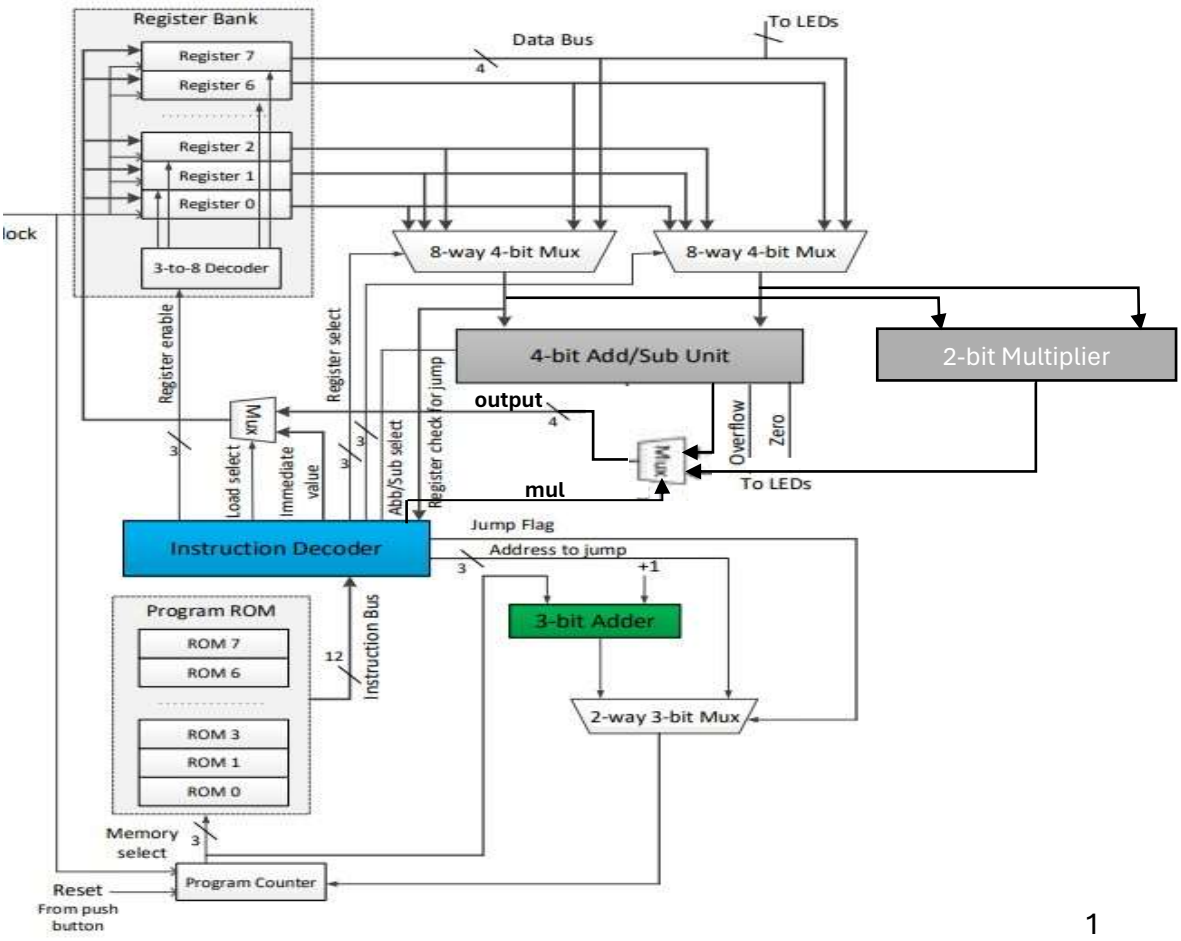
Kajaluxsan S – 220297K

Sukithan T – 220625R

Vithursanaa N – 220671D


## ❖ Introduction

We designed a basic Nano processor that can carry out a basic set of commands. For the Nano processor, we provide those instructions as a binary value (Eg: 010100000000) since it can only understand machine language. Our program will be hard coded into ROM. To reset the PC and Register Bank (which allows us to restart the program at any time), press one of the push buttons. We powered our nano processor with the slow clock. As a result, lower the clock rate so that it ticks every two or three seconds so that you can see the changes as our program runs. This is the nano processor's block diagram,



❖ **Purpose of this Lab**

- We can design and develop a 4-bit arithmetic unit that can add and subtract signed integers.
- We can decode instructions to activate necessary components on the processor.
- We can design and develop k-way b-bit multiplexers or tri-state busses.
- We can verify their functionality via simulation and on the Basys-3 development board.
- We can modify the nano processor to add more features.

Finally, we can be able to design a simple nano processor with these steps.

❖ **Lab Task**

We were asked to design a very simple nano processor capable of executing a simple set of given instructions.

We need to design some components for our nano processor.

- 4bit Adder and Subtractor unit
- 2bit Multiplier
- 3-bit adder
- 3-bit Program Counter (PC)
- K-way b-bit multiplexer
- 4-bit register
- Register Bank
- Program ROM
- Instruction Decoder
- Slow Clock
- Lookup Table 16 to 7
- Nano Processor\

## ❖ Components

### ➢ 4bit Adder and Subtractor unit

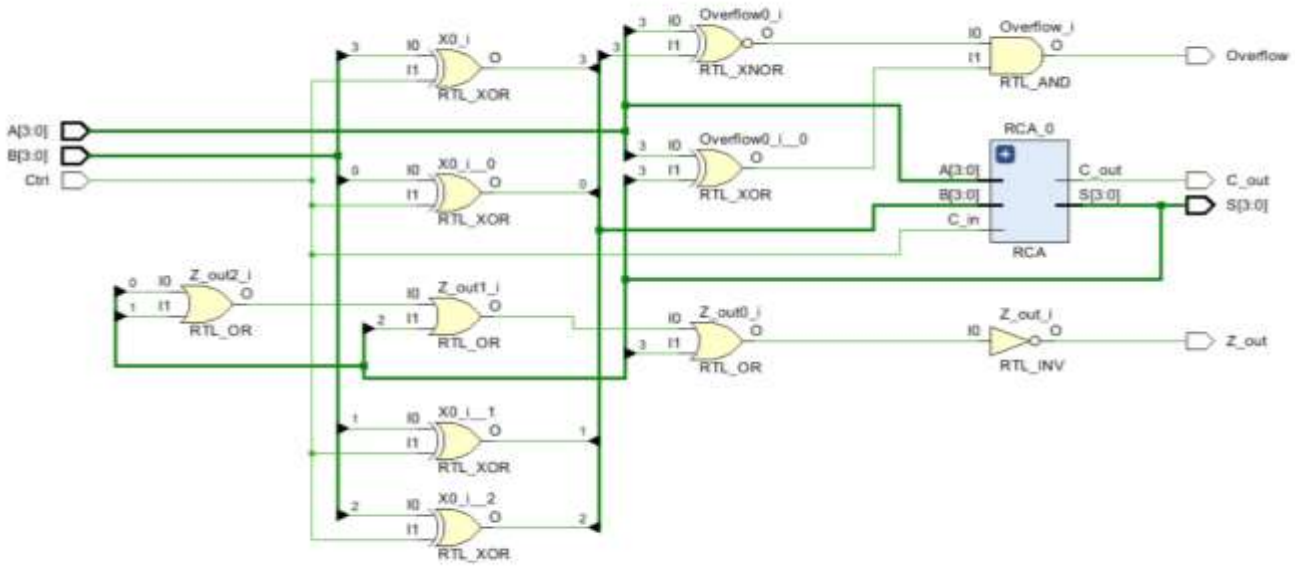
It can add and subtracting numbers represented using 2's complement. It has an overflow and zero flags.

- **VHDL Code**

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

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

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

entity Add_Sub is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           Ctrl : in STD_LOGIC;
           C_out : out STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Overflow : out STD_LOGIC;
           Z_out : out STD_LOGIC
           );
end Add_Sub;

architecture Behavioral of Add_Sub is

component RCA
 Port (A : in STD_LOGIC_VECTOR (3 downto 0);
       B : in STD_LOGIC_VECTOR (3 downto 0);
       C_in : in STD_LOGIC;
       S : out STD_LOGIC_VECTOR (3 downto 0);
       C_out : out STD_LOGIC
       );
end component;
SIGNAL X,Y: STD_LOGIC_VECTOR (3 downto 0);

begin
X(0) <= B(0) XOR Ctrl;
X(1) <= B(1) XOR Ctrl;
X(2) <= B(2) XOR Ctrl;
```

```
X(3) <= B(3) XOR Ctrl;
RCA_0 : RCA
    PORT MAP (
        A => A,
        B => X,
        C_in => Ctrl,
        S => Y,
        C_out => C_out
    );
Overflow <= (A(3) XNOR (Ctrl XOR B(3))) AND (A(3) XOR Y(3));
S <= Y;
Z_Out <= NOT( Y(0) OR Y(1) OR Y(2) OR Y(3));
end Behavioral;
```

- **Schematic Diagram**

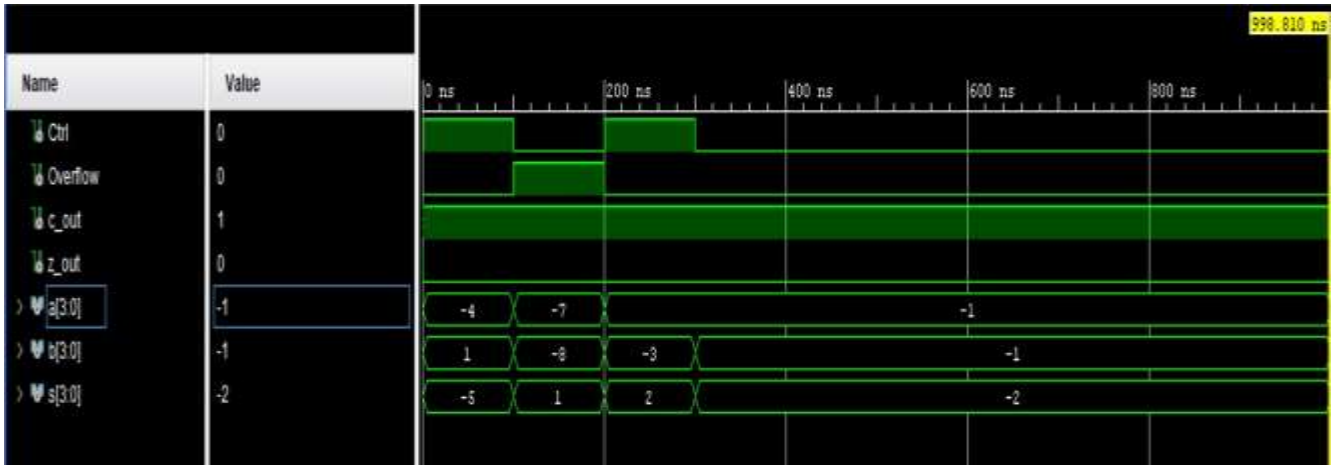



- **Simulation Source File**

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

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

architecture Behavioral of TB_Add_Sub is
component Add_Sub
     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
            B : in STD_LOGIC_VECTOR (3 downto 0);
            Ctrl : in STD_LOGIC;
```

```
            S : out STD_LOGIC_VECTOR (3 downto 0);
            Overflow : out STD_LOGIC;
            Z_out : out STD_LOGIC;
            C_out : out STD_LOGIC
            );
end component;
signal Ctrl,Overflow,c_out,z_out : STD_LOGIC;
signal a,b,s : STD_LOGIC_VECTOR (3 downto 0);
begin
UUT: Add_Sub
    PORT MAP (
         A => a,
         B => b,
         S => s,
         Ctrl =>Ctrl,
         Overflow =>Overflow,
         C_out => c_out,
         Z_out => z_out
    );
    process
            begin
                --- 220444K - 0b 0011 0101 1101 0001 1100
                --- 220297K - 0b 0011 0101 1100 1000 1001
                --- 220625R - 0b 0011 0101 1101 1101 0001
                --- 220671D - 0b 0011 0101 1101 1111 1111

                Ctrl <='1';
                A <= "1100"; -- -4
                B <= "0001"; -- 1
                wait for 100ns;
                Ctrl <= '0';
                A <= "1001"; -- -7
                B <= "1000"; -- -8
                wait for 100ns;
                Ctrl <= '1';
                A <= "1111"; -- 1
                B <= "1101"; -- -3
                wait for 100ns;
                Ctrl <='0';
                A <= "1111"; -- -1
                B <= "1111"; -- -1
                wait;
            end process;


end Behavioral;
```

- **Timing Diagram**




➢ **3-bit Adder**

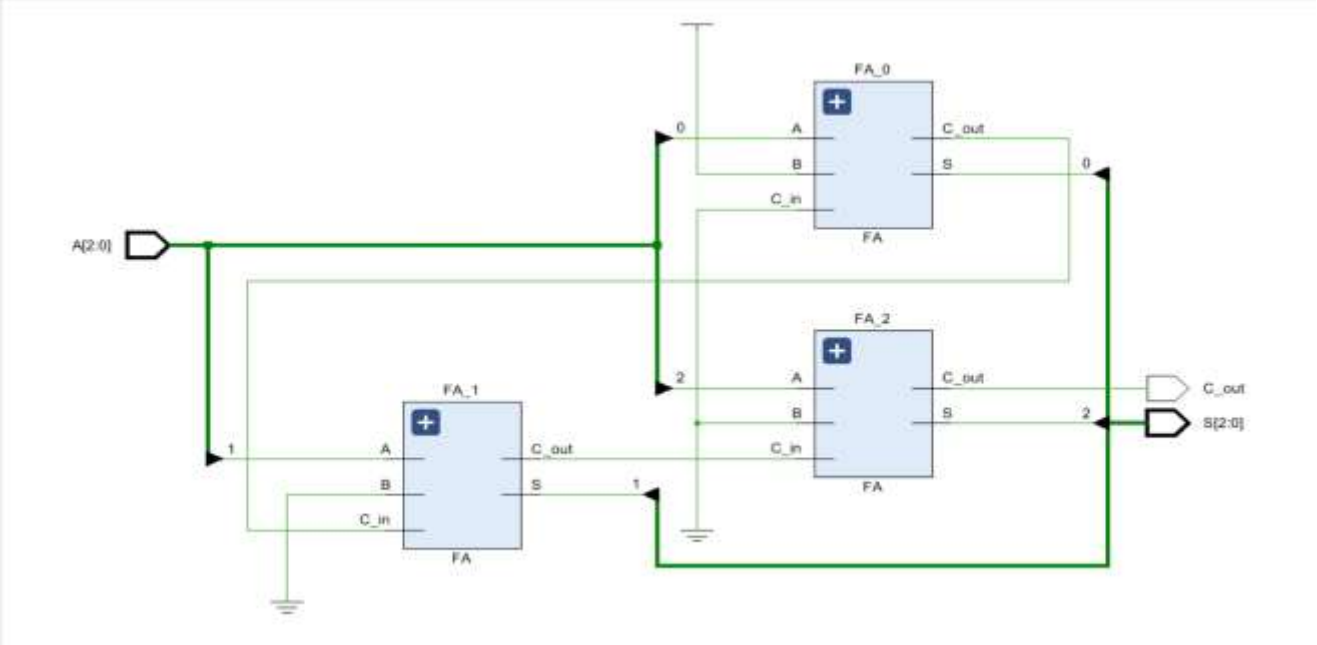
It is used to increment the program counter.

- **VHDL Code**

```
library IEEE;
use IEEE.STD_LOGIC_1164.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 FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C, FA3_S, FA3_C : std_logic;
begin
FA_0 : FA
    port map (
         A => A(0),
         B => '1',
         C_in => '0',
```

```
            S => S(0),
            C_Out => FA0_C);
    FA_1 : FA
        port map (
            A => A(1),
            B => '0',
            C_in => FA0_C,
            S => S(1),
            C_Out => FA1_C);
    FA_2 : FA
        port map (
            A => A(2),
            B => '0',
            C_in => FA1_C,
            S => S(2),
            C_Out => C_out);

    end Behavioral;
```

- **Schematic Diagram**

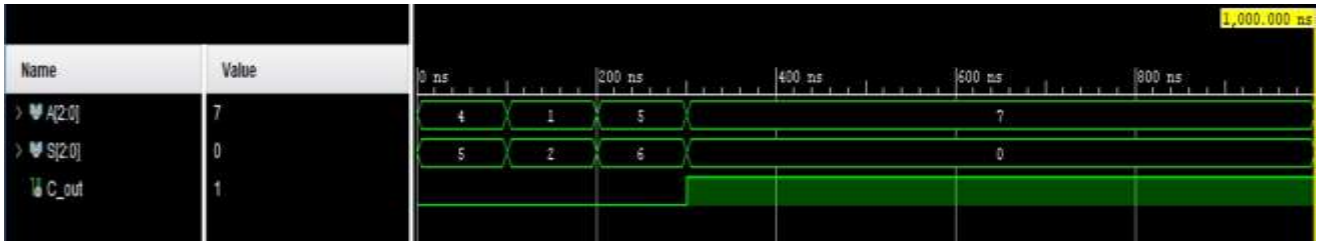



- **Simulation Source File**

```
library IEEE;
use IEEE.STD_LOGIC_1164.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
    );
    --- 220444K - 0b 0011 0101 1101 0001 1100
    --- 220297K - 0b 0011 0101 1100 1000 1001
    --- 220625R - 0b 0011 0101 1101 1101 0001
    --- 220671D - 0b 0011 0101 1101 1111 1111
    process
            begin
                A <= "100";
                wait for 100ns;
                A <= "001";
                wait for 100ns;
                A <= "101";
                wait for 100ns;
                A <= "111";
                wait;
            end process;
end Behavioral;
```

- **Timing Diagram**



➢ **3-bit Program Counter**

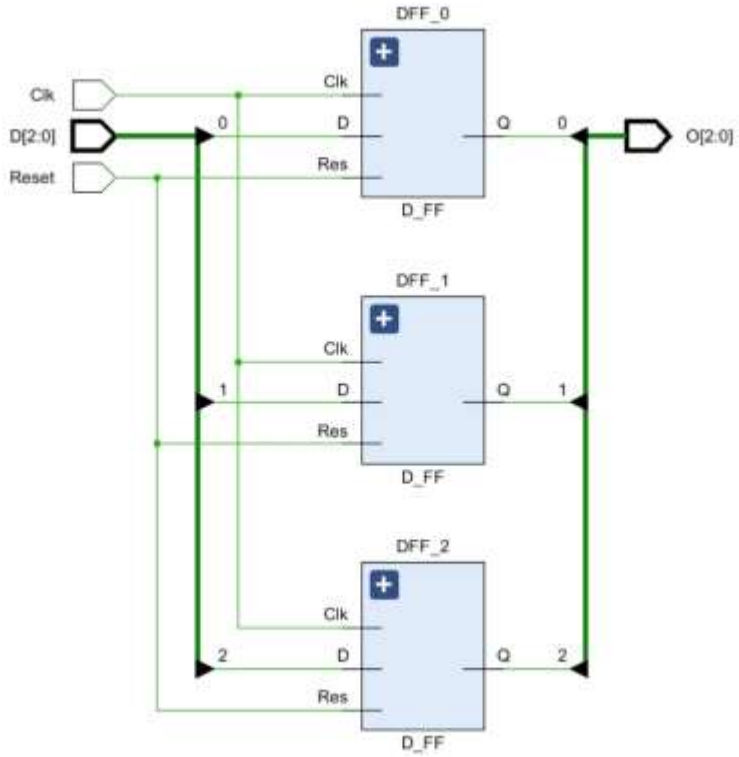
This is stored the address of the next instruction to executed. This is a type of a register.

- **VHDL Code**

```
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;
           D : in STD_LOGIC_VECTOR (2 downto 0);
           O : 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
           );
end component;
signal Res_Sig : STD_LOGIC:= '1';
begin
DFF_0 : D_FF
    PORT MAP (
        D => D(0),
        Q => O(0),
        Clk => Clk,
        Res => Res_Sig
    );
DFF_1 : D_FF
    PORT MAP (
        D => D(1),
        Q => O(1),
        Clk => Clk,
        Res => Res_Sig
    );
```

```
DFF_2 : D_FF
    PORT MAP (
        D => D(2),
        Q => O(2),
        Clk => Clk,
        Res => Res_Sig
    );
Res_Sig <= Reset;
end Behavioral;
```

- **Schematic Diagram**

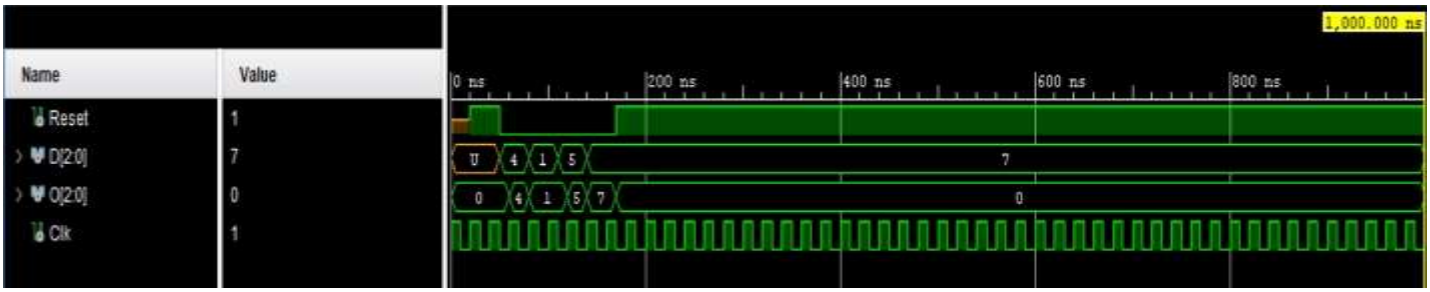



- **Simulation Source File**

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

```
            D : in STD_LOGIC_VECTOR (2 downto 0);
            O : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal Reset : STD_LOGIC;
signal D,O : STD_LOGIC_VECTOR (2 downto 0);
signal Clk : STD_LOGIC := '1';
begin
    UUT : PC
        PORT MAP(
            Clk => Clk,
            Reset => Reset,
            D => D,
            O => O
        );
    process
    begin
        wait for 10ns;
        Clk <= NOT(CLK);
    end process;

    --- 220444K - 0b 0011 0101 1101 0001 1100
    --- 220297K - 0b 0011 0101 1100 1000 1001
    --- 220625R - 0b 0011 0101 1101 1101 0001
    --- 220671D - 0b 0011 0101 1101 1111 1111

    process
    begin
        wait for 20ns;
        Reset <= '1';
        wait for 30ns;
        Reset <= '0';
        D <= "100";
        wait for 30ns;
        D <= "001";
        wait for 30ns;
        D <= "101";
        wait for 30ns;
        D <= "111";
        wait for 30ns;
        Reset <= '1';
        wait;
    end process;



end Behavioral;
```

- **Timing Diagram**



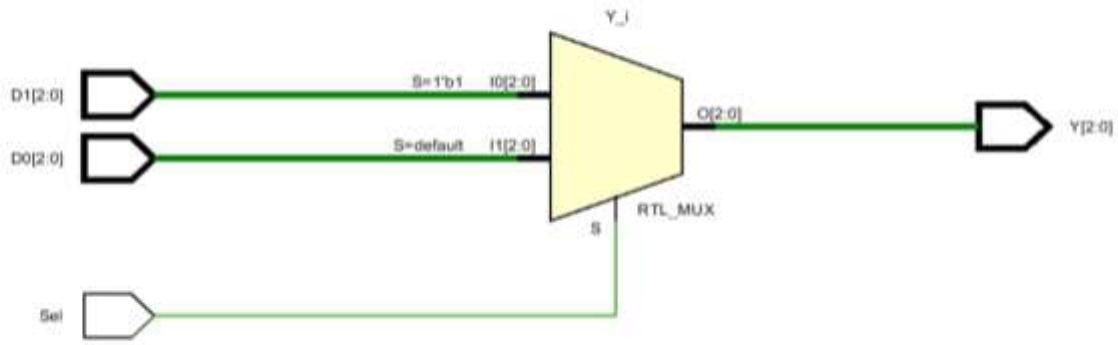

➢ **2-way-3-bit Multiplexer**

It can take in 2-inputs, each with 3-bits, and the output is a group of 3-bits. There are 1 control bits, and these control bits are used to select one of the 3 groups of 3.

- **VHDL Code**

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

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

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

entity Mux_2_to_1_3bit is
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2_to_1_3bit;

architecture Behavioral of Mux_2_to_1_3bit is

begin
      Y <= D1 when (Sel='1') else D0;
end Behavioral;
```

- **Schematic Diagram**

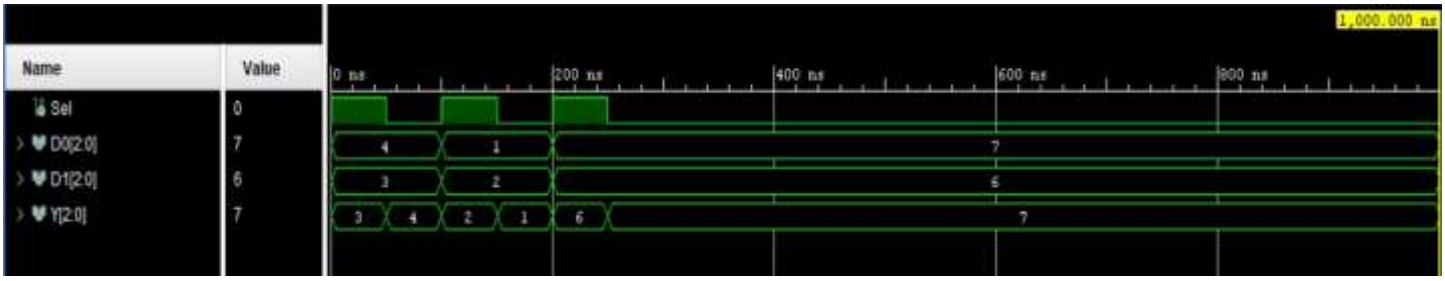



- **Simulation Source File**

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


entity TB_Mux_2_to_1_3bit is
--  Port ( );
end TB_Mux_2_to_1_3bit;

architecture Behavioral of TB_Mux_2_to_1_3bit is
component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal Sel : STD_LOGIC;
signal D0 : STD_LOGIC_VECTOR (2 downto 0);
signal D1 : STD_LOGIC_VECTOR (2 downto 0);
signal Y : STD_LOGIC_VECTOR (2 downto 0);
begin
UUT: Mux_2_to_1_3bit
      port map (
           Sel=>Sel,
           D0=>D0,
           D1=>D1,
           Y=>Y   );
```

```
--- 220444K - 0b 0011 0101 1101 0001 1100
--- 220297K - 0b 0011 0101 1100 1000 1001
--- 220625R - 0b 0011 0101 1101 1101 0001
--- 220671D - 0b 0011 0101 1101 1111 1111

 process
 begin
      D0 <= "100";
      D1 <= "011";
      Sel <= '1';

      wait for 50ns;
      Sel <= '0';

      wait for 50ns;
      D0 <= "001";
      D1 <= "010";
      Sel <= '1';

      wait for 50ns;
      Sel <= '0';

      wait for 50ns;
      D0 <= "111";
      D1 <= "110";
      Sel <= '1';

      wait for 50ns;
      Sel <= '0';
      wait;
 end process;

end Behavioral;
```

- **Timing Diagram**



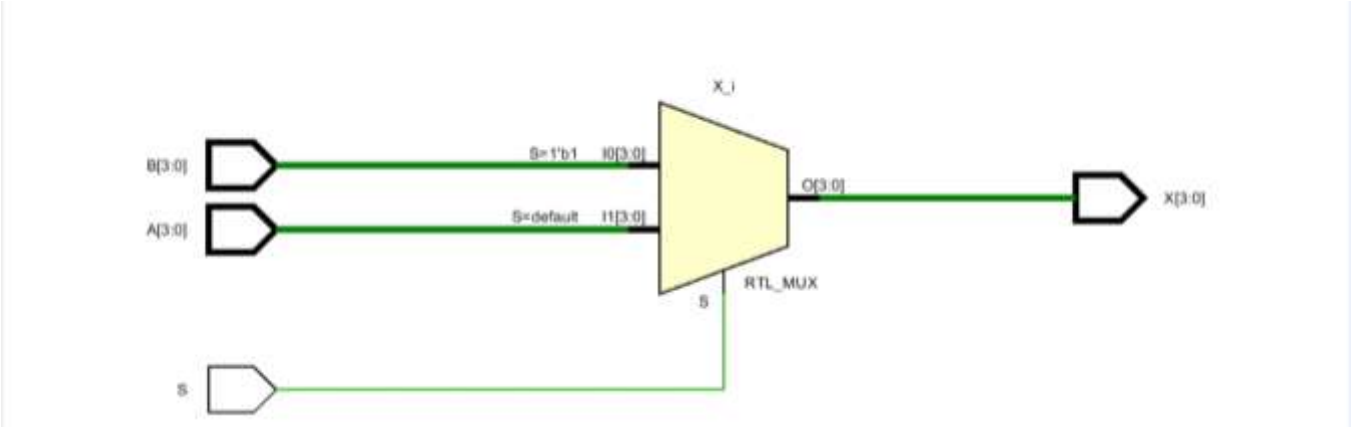
## ➢ 2-way-4-bit Multiplexer

It can take in 2-inputs, each with 4-bits, and the output is a group of 4-bits. There are 1 control bits, and these control bits are used to select one of the 2 groups of 4 bits.

- **VHDL Code**

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

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

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

entity Mux_2_to_1_4bit is
    Port ( S : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           X : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2_to_1_4bit;

architecture Behavioral of Mux_2_to_1_4bit is

begin
      X <= B when (S='1') else A;

end Behavioral;
```

- **Schematic Diagram**



- **Simulation Source File**

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

entity TB_Mux_2_to_1_4bit is
--  Port ( );
end TB_Mux_2_to_1_4bit;

architecture Behavioral of TB_Mux_2_to_1_4bit is
component Mux_2_to_1_4bit
    Port ( S : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           X : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal S : STD_LOGIC;
signal A : STD_LOGIC_VECTOR (3 downto 0);
signal B : STD_LOGIC_VECTOR (3 downto 0);
signal X : STD_LOGIC_VECTOR (3 downto 0);
begin
UUT : Mux_2_to_1_4bit
      port map (
           S=>S,
           A=>A,
           B=>B,
           X=>X   );

    --- 220444K - 0b 0011 0101 1101 0001 1100
    --- 220297K - 0b 0011 0101 1100 1000 1001
    --- 220625R - 0b 0011 0101 1101 1101 0001
    --- 220671D - 0b 0011 0101 1101 1111 1111

     process
     begin
          A <= "1100";
          B <= "0001";
          S <= '1';

          wait for 50ns;
          S <= '0';

          wait for 50ns;
          A <= "1001";
          B <= "1000";
          S <= '1';
          wait for 50ns;
          S <= '0';
```

```
            wait for 50ns;
            A <= "0001";
            B <= "1101";
            S <= '1';
            wait for 50ns;
            S <= '0';
            wait;

        end process;

    end Behavioral;
```

- **Timing Diagram**


## ➢ 8-way-4-bit Multiplexer

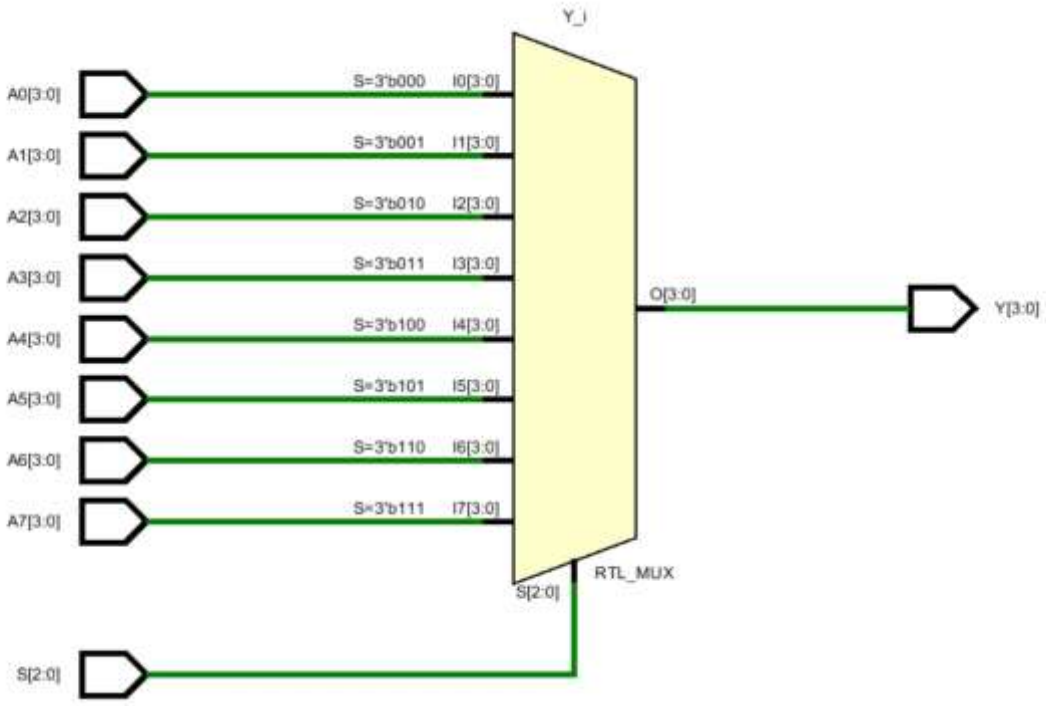
It can take in 8-inputs, each with 4-bits, and the output is a group of 4-bits. There are 3 control bits, and these control bits are used to select one of the 8 groups of 4 bits.

- **VHDL Code**

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

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

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

entity Mux_8_to_1_4bit is
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           A0 : in STD_LOGIC_VECTOR (3 downto 0);
           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);
            Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8_to_1_4bit;

architecture Behavioral of Mux_8_to_1_4bit is
begin
    process(A0,A1,A2,A3,A4,A5,A6,A7,S)
    begin
    case S is
    when "000" => Y <= A0;
    when "001" => Y <= A1;
    when "010" => Y <= A2;
    when "011" => Y <= A3;
    when "100" => Y <= A4;
    when "101" => Y <= A5;
    when "110" => Y <= A6;
    when "111" => Y <= A7;
    when others => NULL;
    end case;
end process;
```

- **Schematic Diagram**

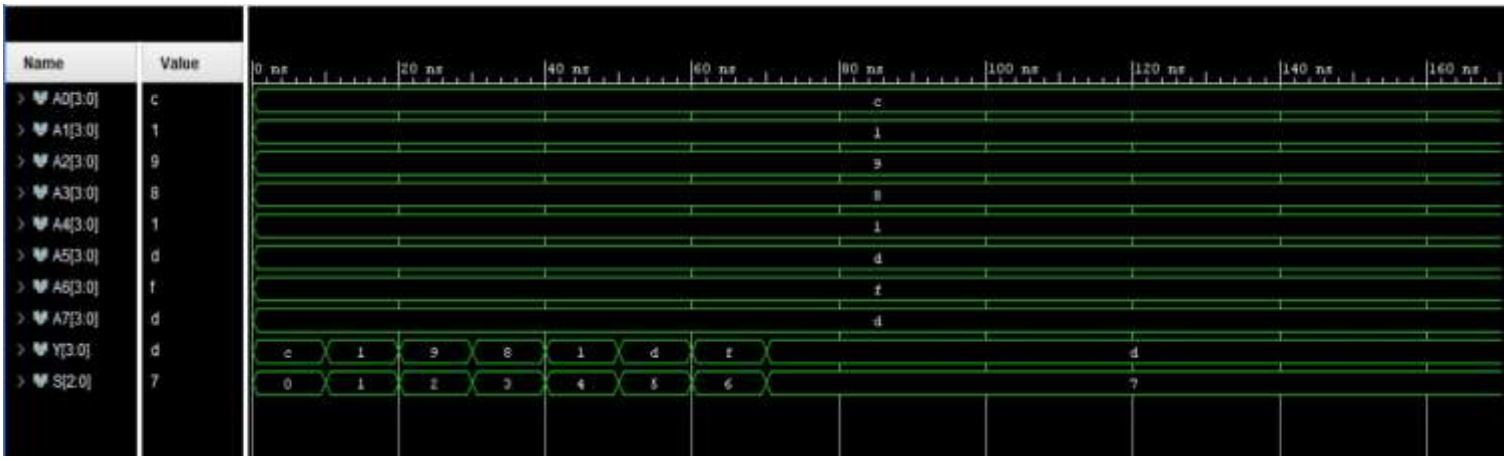


- **Simulation Source File**

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


entity TB_Mux_8_to_1_4bit is
--  Port ( );
end TB_Mux_8_to_1_4bit;

architecture Behavioral of TB_Mux_8_to_1_4bit is
component Mux_8_to_1_4bit
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           A0 : in STD_LOGIC_VECTOR (3 downto 0);
           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);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal A0,A1,A2,A3,A4,A5,A6,A7,Y: STD_LOGIC_VECTOR (3 downto 0);
signal S : STD_LOGIC_VECTOR (2 downto 0);
begin
UUT : Mux_8_to_1_4bit
     port map (
     S => S,
     A0 => A0,
     A1 => A1,
     A2 => A2,
     A3 => A3,
     A4 => A4,
     A5 => A5,
     A6 => A6,
     A7 => A7,
     Y => Y         );

    --- 220444K - 0b 0011 0101 1101 0001 1100
    --- 220297K - 0b 0011 0101 1100 1000 1001
    --- 220625R - 0b 0011 0101 1101 1101 0001
    --- 220671D - 0b 0011 0101 1101 1111 1111
     process
     begin
         A0 <= "1100";
         A1 <= "0001";
         A2 <= "1001";
```

```
        A3 <= "1000";
        A4 <= "0001";
        A5 <= "1101";
        A6 <= "1111";
        A7 <= "1101";
        S <= "000";

        wait for 10ns;
        S <= "001";

        wait for 10ns;
        S <= "010";

        wait for 10ns;
        S <= "011";

        wait for 10ns;
        S <= "100";

        wait for 10ns;
        S <= "101";

        wait for 10ns;
        S <= "110";

        wait for 10ns;
        S <= "111";
        wait;

    end process;
end Behavioral;
```

- **Timing Diagram**



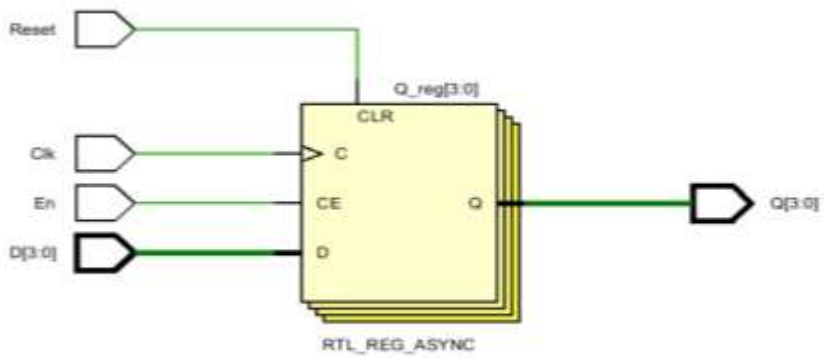
## ➢ 4-bit Register

It can store a 4 bit data temporarily.

- **VHDL Code**

```
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_4bit 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 Register_4bit;

architecture Behavioral of Register_4bit is

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

end Behavioral;
```

- **Schematic Diagram**




- **Simulation Source File**

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

entity TB_Register_4bit is
--  Port ( );
end TB_Register_4bit;

architecture Behavioral of TB_Register_4bit is

component Register_4bit
    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 D : STD_LOGIC_VECTOR (3 downto 0):="0000";
signal En,Clk : STD_LOGIC :='0';
signal Reset : STD_LOGIC :='1';
signal Q : STD_LOGIC_VECTOR (3 downto 0);

begin

UUT: Register_4bit
    PORT MAP (
        D => D,
        En => En,
        Clk => Clk,
```

```
            Reset => Reset,
            Q => Q
        );
    process
    begin
        wait for 50ns;
        Clk <= NOT(Clk);
    end process;

    process
    begin
        wait for 20ns;
        Reset <= NOT(Reset);
        Wait for 5ns;
        En <= '1';
        wait;
    end process;

        --- 220444K - 0b 0011 0101 1101 0001 1100
        --- 220297K - 0b 0011 0101 1100 1000 1001
        --- 220625R - 0b 0011 0101 1101 1101 0001
        --- 220671D - 0b 0011 0101 1101 1111 1111
    process
    begin
         D <= "1100";
         wait for 100ns;
         D <= "1001";
         wait for 100ns;
         D <= "0001";
         wait for 100ns;
         D <= "1111";
         wait;
    end process;
    end Behavioral;
```

- **Timing Diagram**

## ➢ Register Bank

It contains 4-bit,8-bit registers. It also has 3-8 decoder for enabling the registers.

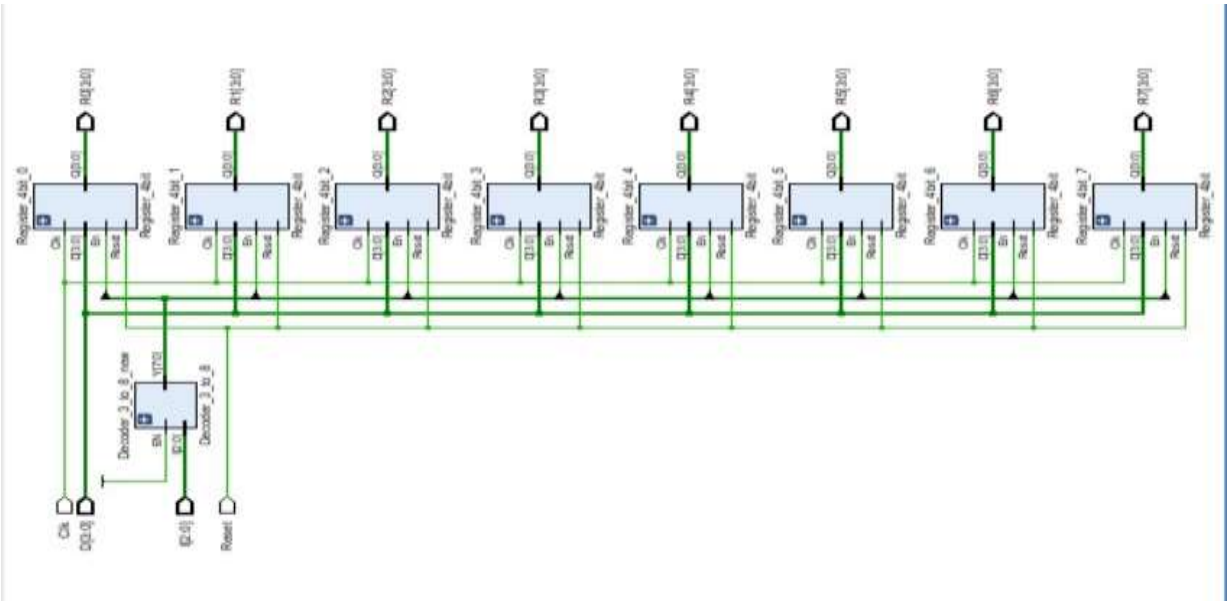
- **VHDL Code**

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

entity Register_Bank is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           R0 : out STD_LOGIC_VECTOR (3 downto 0);
           R1 : out STD_LOGIC_VECTOR (3 downto 0);
           R2 : out STD_LOGIC_VECTOR (3 downto 0);
           R3 : out STD_LOGIC_VECTOR (3 downto 0);
           R4 : out STD_LOGIC_VECTOR (3 downto 0);
           R5 : out STD_LOGIC_VECTOR (3 downto 0);
           R6 : out STD_LOGIC_VECTOR (3 downto 0);
           R7 : out STD_LOGIC_VECTOR (3 downto 0);
           I : in STD_LOGIC_VECTOR (2 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 Register_4bit
    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 EN,Reg_S : STD_LOGIC:='1';
signal Y : STD_LOGIC_VECTOR(7 downto 0);
```

```
begin
Decoder_3_to_8_new:Decoder_3_to_8
 PORT MAP (
        I=>I,
        EN=>EN,
        Y=>Y
        );

Register_4bit_0 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(0),
    Clk => Clk,
    Reset => Reg_S,
    Q => R0
    );

Register_4bit_1 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(1),
    Clk => Clk,
    Reset => Reg_S,
    Q => R1
    );

Register_4bit_2 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(2),
    Clk => Clk,
    Reset => Reg_S,
    Q => R2
    );

Register_4bit_3 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(3),
    Clk => Clk,
    Reset => Reg_S,
    Q => R3
    );

Register_4bit_4 : Register_4bit
 PORT MAP(
    D => D,
    En => Y(4),
```

```
        Clk => Clk,
        Reset => Reg_S,
        Q => R4
        );
    Register_4bit_5 : Register_4bit
     PORT MAP(
        D => D,
        En => Y(5),
        Clk => Clk,
        Reset => Reg_S,
        Q => R5
        );
    Register_4bit_6 : Register_4bit
     PORT MAP(
        D => D,
        En => Y(6),
        Clk => Clk,
        Reset => Reg_S,
        Q => R6
        );
    Register_4bit_7 : Register_4bit
     PORT MAP(
        D => D,
        En => Y(7),
        Clk => Clk,
        Reset => Reg_S,
        Q => R7
        );
    Reg_S <= Reset;
    end Behavioral;
```

- **Schematic Diagram**

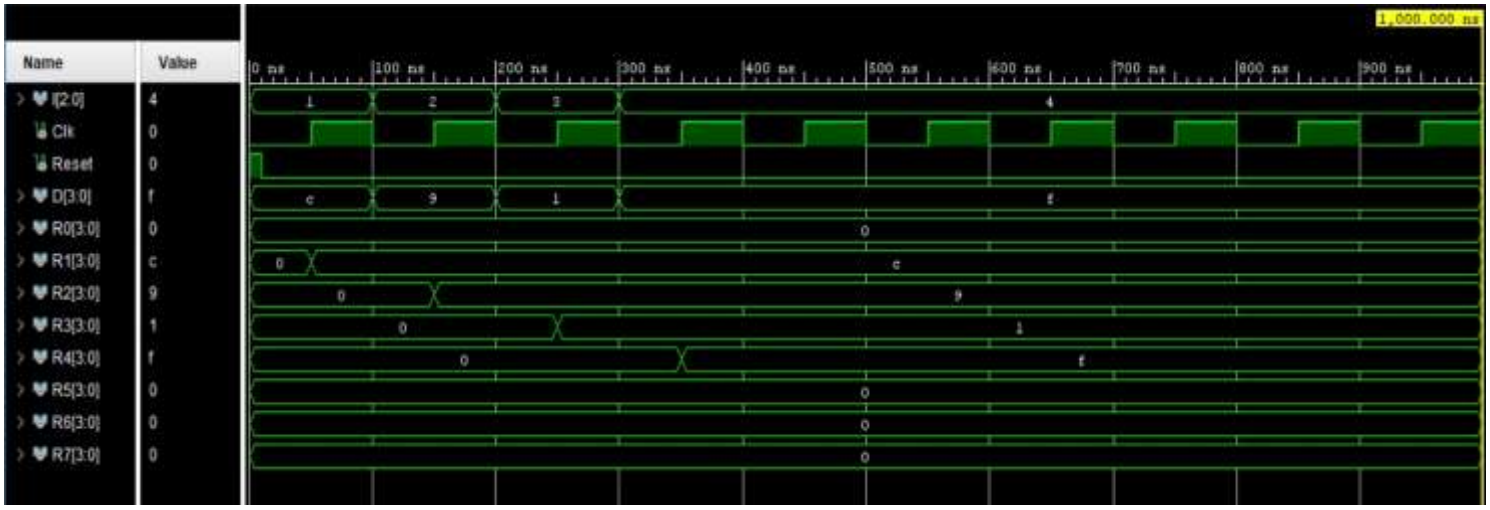


- **Simulation Source File**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_Register_Bank is
end TB_Register_Bank;
architecture Behavioral of TB_Register_Bank is
component Register_Bank
    PORT ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           R0 : out STD_LOGIC_VECTOR (3 downto 0);
           R1 : out STD_LOGIC_VECTOR (3 downto 0);
           R2 : out STD_LOGIC_VECTOR (3 downto 0);
           R3 : out STD_LOGIC_VECTOR (3 downto 0);
           R4 : out STD_LOGIC_VECTOR (3 downto 0);
           R5 : out STD_LOGIC_VECTOR (3 downto 0);
           R6 : out STD_LOGIC_VECTOR (3 downto 0);
           R7 : out STD_LOGIC_VECTOR (3 downto 0);
           I : in STD_LOGIC_VECTOR (2 downto 0)
    );
end component;

signal I : STD_LOGIC_VECTOR (2 downto 0);
signal Clk : STD_LOGIC := '0';
signal Reset : STD_LOGIC := '1';
signal D : STD_LOGIC_VECTOR (3 downto 0);
signal R0,R1,R2,R3,R4,R5,R6,R7 : STD_LOGIC_VECTOR (3 downto 0);
begin
UUT : Register_Bank
    PORT MAP(
            Clk => Clk,
            Reset => Reset,
            D => D,
            R0 => R0,
            R1 => R1,
            R2 => R2,
            R3 => R3,
            R4 => R4,
            R5 => R5,
            R6 => R6,
            R7 => R7,
            I => I
    );
```

```
process
begin
    wait for 50ns;
    Clk <= NOT(Clk);
end process;
process
begin
    wait for 10ns;
    Reset <= NOT(Reset);
    wait;
end process;
    --- 220444K - 0b 0011 0101 1101 0001 1100
    --- 220297K - 0b 0011 0101 1100 1000 1001
    --- 220625R - 0b 0011 0101 1101 1101 0001
    --- 220671D - 0b 0011 0101 1101 1111 1111
process
begin
      I <= "001"; --grey code
      D <= "1100";
      wait for 100ns;
      I <= "010";
      D <= "1001";
      wait for 100ns;
      I <= "011";
      D <= "0001";
      wait for 100ns;
      I <= "100";
      D <= "1111";
      wait;
end process;
end Behavioral;
```

- **Timing Diagram**



## ➢ Program Rom

This is a ROM (Read Only Memory) which has programmed with certain instructions to run the whole processor as needed. The instructions will be transferred to Instruction Decoder to decode.

Assembly code:

0 => MOVI R1, 3;

1 => MOVI R2, 1;

2 => NEG R2;

3 => ADD R7, R1;

4 => ADD R1, R2;

5 => JZR R1, 7;

6 => JZR R0, 3;

7 => JZR R0, 7;

**Machine Code**

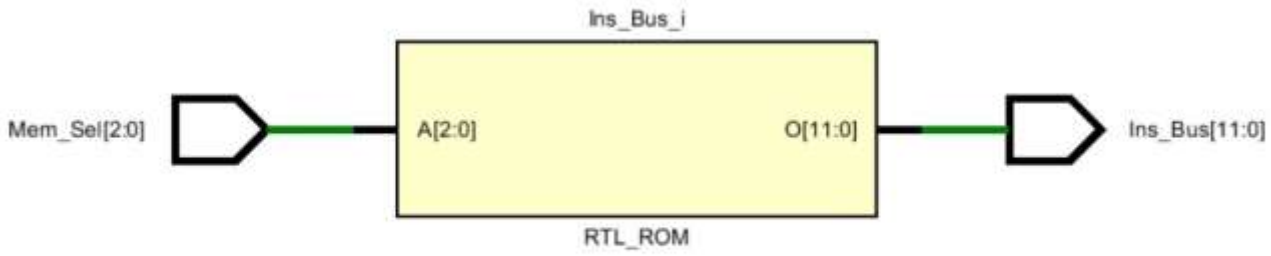
| ROM | | | Instruction | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S2 | S1 | S0 | I (11) | I (10) | I (9) | I (8) | I (7) | I (6) | I (5) | I (4) | I (3) | I (2) | I (1) | I (0) |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

- **VHDL Code**

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


entity Program_ROM is
    Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Ins_Bus : out STD_LOGIC_VECTOR (12 downto 0));
end Program_ROM;
```

```
architecture Behavioral of Program_ROM is
type rom_type is array (0 to 7) of STD_LOGIC_VECTOR(12 downto 0);
    signal sevenSegment_ROM : rom_type := (
    "0100100000011",-- Mov R2, 3
    "0101110000010",-- Mov R7, 2
    "0011110100000",-- Sub R7,R2
    "1011110000000",-- Neg R7
    "0100110000010",-- Mov R3, 2
    "0001110110000",-- Add R7, R3
    "1001110110000",-- Mul R7, R3
    "0110000000001"-- JMZ( check R0 )
    );
begin
Ins_Bus <= sevenSegment_ROM(to_integer(unsigned(Mem_Sel)));
end Behavioral;
```

- **Schematic Diagram**

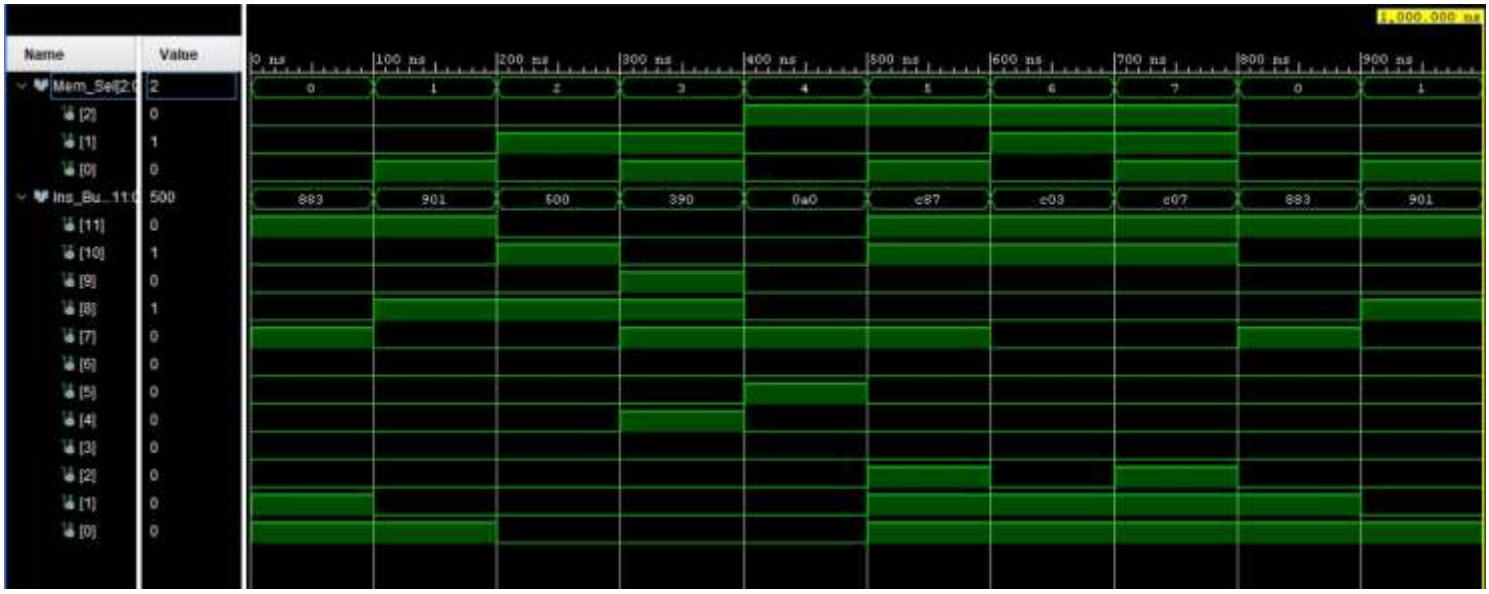



- **Simulation Source File**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Program_ROM_TB is
--  Port ( );
end Program_ROM_TB;

architecture Behavioral of Program_ROM_TB is
component Program_ROM
    Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Ins_Bus : out STD_LOGIC_VECTOR (11 downto 0));
end component;
signal Mem_Sel :STD_LOGIC_VECTOR (2 downto 0);
signal Ins_Bus : STD_LOGIC_VECTOR (11 downto 0);
begin
```

```
UUT: Program_ROM
    PORT MAP(
        Mem_Sel => Mem_Sel,
        Ins_Bus => Ins_Bus
    );

process
begin
    Mem_Sel <= "000";
    wait for 100 ns;
    Mem_Sel <= "001";
    wait for 100 ns;
    Mem_Sel <= "010";
    wait for 100 ns;
    Mem_Sel <= "011";
    wait for 100 ns;
    Mem_Sel <= "100";
    wait for 100 ns;
    Mem_Sel <= "101";
    wait for 100 ns;
    Mem_Sel <= "110";
    wait for 100 ns;
    Mem_Sel <= "111";
    wait for 100 ns;
end process;
end Behavioral;
```

- **Timing Diagram**



➢ **Instruction Decoder**

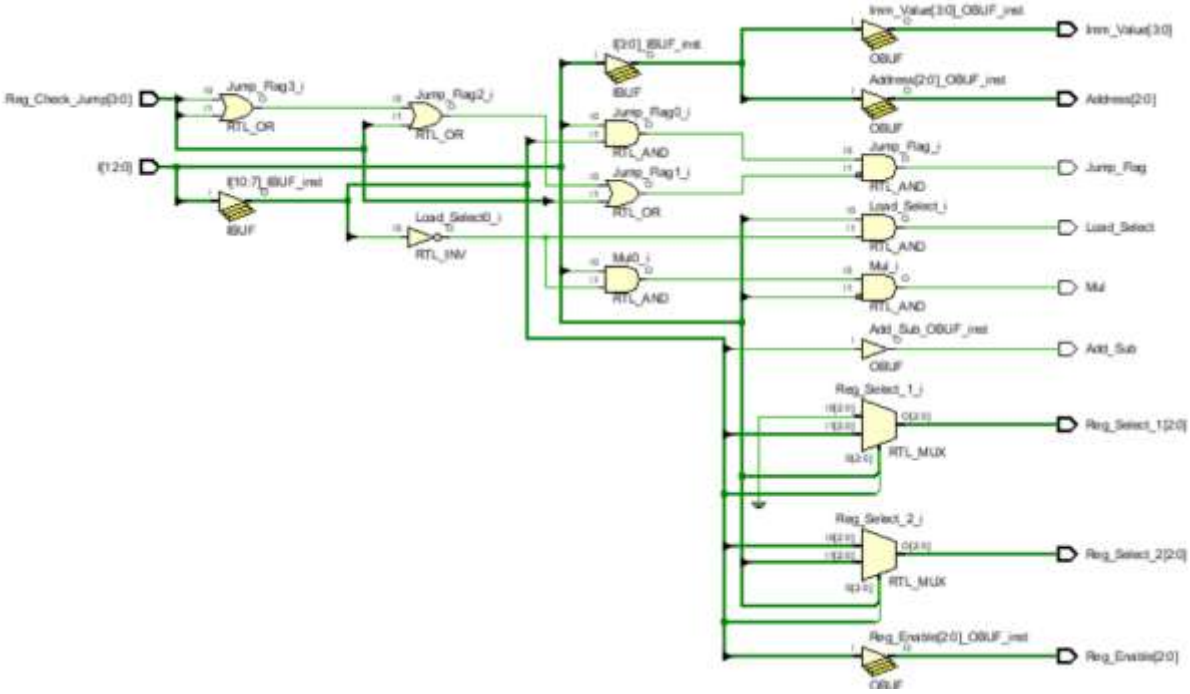
It activates the necessary components for the instruction execute.

- **VHDL Code**

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

entity Instruction_Decoder is
    Port ( I : in STD_LOGIC_VECTOR (12 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : out STD_LOGIC;
           Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address : out STD_LOGIC_VECTOR (2 downto 0);
           Mul: out STD_LOGIC
           );
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is

component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal Ins : STD_LOGIC_VECTOR (2 downto 0);
signal RegA : STD_LOGIC_VECTOR (2 downto 0);
signal RegB : STD_LOGIC_VECTOR (2 downto 0);
signal Data : STD_LOGIC_VECTOR (3 downto 0);
signal Sel: STD_LOGIC;

begin
Ins <= I(12 downto 10);
RegA <= I(9 downto 7);
RegB <= I(6 downto 4);


with Ins select Reg_select_1<=
        "000" when "101",
        RegA when others;
```

```
with Ins select Reg_select_2<=
        RegA when "101",
        RegB when others;


Data <= I(3 downto 0);

Sel <= NOT(Ins(1)) AND Ins(0);
Load_Select <= Ins(1) AND NOT (Ins(0)) ;
Add_Sub <= Ins(0);

Jump_Flag <= Ins(1) AND Ins(0) AND NOT( Reg_Check_Jump(3) OR
Reg_Check_Jump(2) OR Reg_Check_Jump(1)OR Reg_Check_Jump(0));
Reg_Enable <= RegA;
Imm_Value <= Data;
Address <= Data(2 downto 0);
Mul <= Ins(2) and not(Ins(0)) and not(Ins(1));

end Behavioral;
```

- **Schematic Diagram**

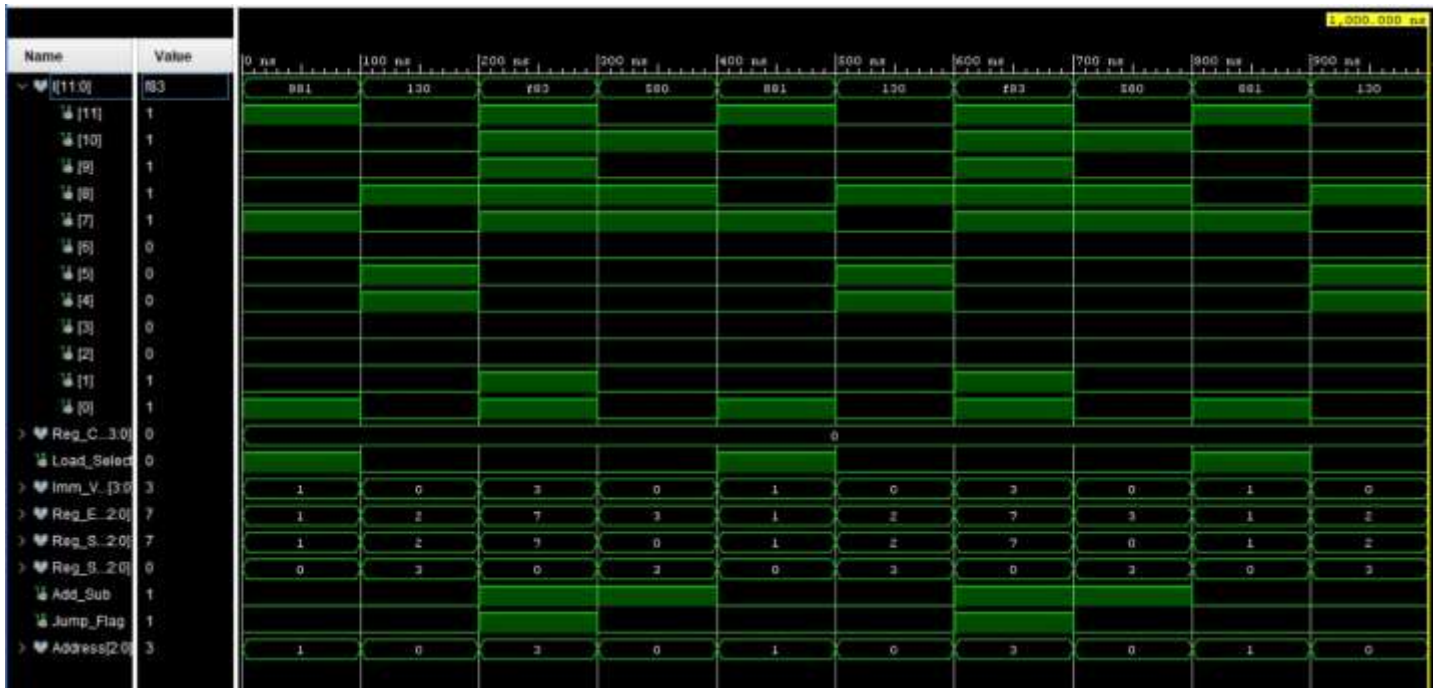


- **Simulation Source File**

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

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

architecture Behavioral of TB_Instruction_Decoder is
component Instruction_Decoder
    Port ( I : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
           Load_Select : out STD_LOGIC;
           Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub, mul : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Address : out STD_LOGIC_VECTOR (2 downto 0)
           );
end component;
signal I : STD_LOGIC_VECTOR (11 downto 0);
signal Reg_Check_Jump : STD_LOGIC_VECTOR (3 downto 0);
signal Load_Select  : STD_LOGIC;
signal Imm_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Reg_Enable :  STD_LOGIC_VECTOR (2 downto 0);
signal Reg_Select_1 :  STD_LOGIC_VECTOR (2 downto 0);
signal Reg_Select_2 : STD_LOGIC_VECTOR (2 downto 0);
signal Add_Sub : STD_LOGIC;
signal Jump_Flag :STD_LOGIC;
signal Address : STD_LOGIC_VECTOR (2 downto 0);
begin
UUT: Instruction_Decoder
    Port Map (
       I => I,
       Reg_Check_Jump => Reg_Check_Jump,
       Load_Select =>  Load_Select,
       Imm_Value=>  Imm_Value,
       Reg_Enable=>   Reg_Enable,
       Reg_Select_1=> Reg_Select_1,
       Reg_Select_2=> Reg_Select_2,
       Add_Sub=> Add_Sub,
       Mul => mul;
       Jump_Flag=> Jump_Flag,
       Address=> Address
    );
```

```
process
begin
Reg_Check_Jump <= "0000";
I <= "100010000001";
wait for 100ns;
I <= "000100110000";
wait for 100ns;
I <= "111110000011";
wait for 100ns;
I <= "010110000000";
wait for 100ns;
end process;

end Behavioral;
```

- **Timing Diagram**



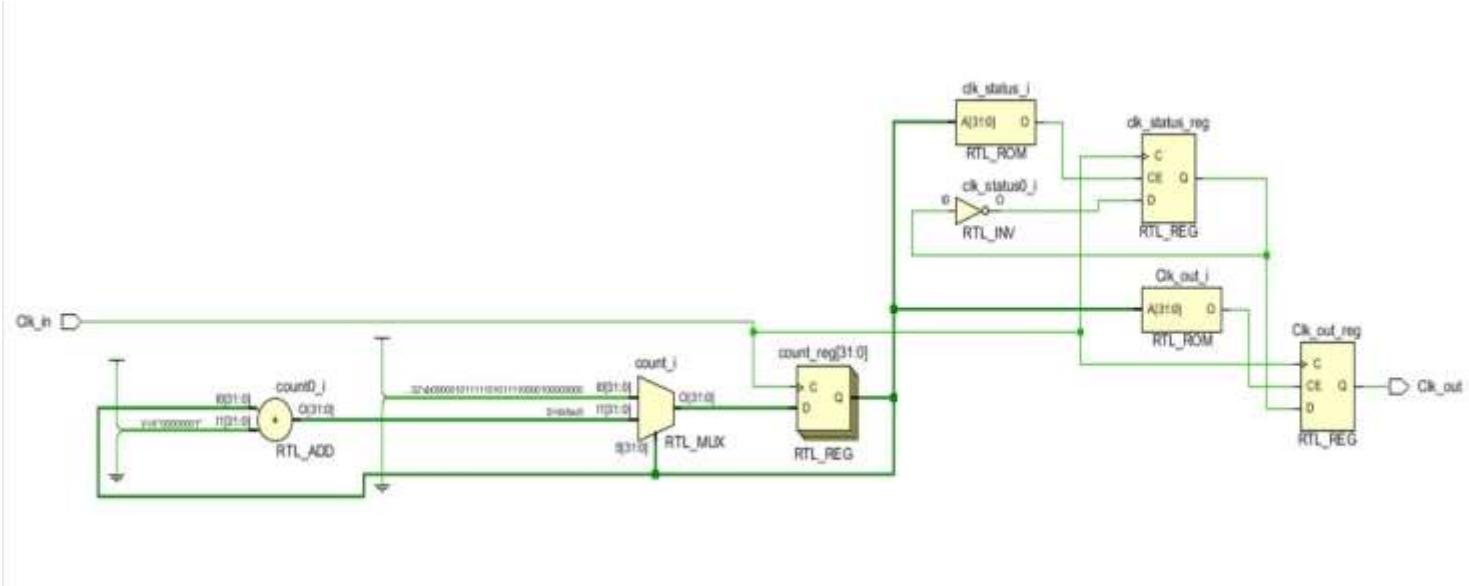
## ➢ Slow Clock

It is helped to slowdown the input clock signal.

- **VHDL Code**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;
architecture Behavioral of Slow_Clk is
signal count : integer := 1;
signal clk_status : std_logic := '0';

begin
    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1; --50000000
            if (count = 100000000)then
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

- **Schematic Diagram**



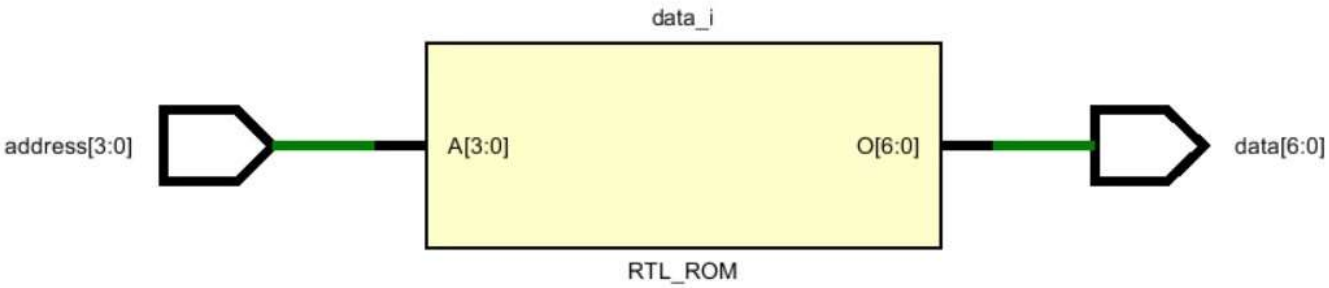
## ➢ LUT 16 to 7

7-segment Display is the component that shows the output in a LED screen on a BASYS-3 board. The output from the processor is mapped to this component as input and the stored data in an inbuilt ROM for the current input address will be the output from the display through LEDs.

- **VHDL Code**

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

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

architecture Behavioral of LUT_16_7 is
type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);
signal sevenSegment_ROM : rom_type := (
"1000000", -- 0
"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;
```

- **Schematic Diagram**




## ➢ Nano processor

- **VHDL Code**

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

entity NanoProcessor is
    Port (Clk : in STD_LOGIC;
          Reset : in STD_LOGIC;
          Reg : out STD_LOGIC_VECTOR (3 downto 0);
          Zero : out STD_LOGIC;
          Overflow : out STD_LOGIC;
          Display : out STD_LOGIC_VECTOR (6 downto 0);
          Anode: out STD_LOGIC_VECTOR (3 downto 0);
          Carry: out STD_LOGIC  );
end NanoProcessor;

architecture Behavioral of NanoProcessor is

component LUT_16_7
    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;

component Add_Sub
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            B : in STD_LOGIC_VECTOR (3 downto 0);
            Ctrl : in STD_LOGIC;
            C_out : out STD_LOGIC;
            S : out STD_LOGIC_VECTOR (3 downto 0);
            Overflow : out STD_LOGIC;
            Z_out : out STD_LOGIC
            );
end component;

component Register_Bank
    Port ( Clk : in STD_LOGIC;
            Reset : in STD_LOGIC;
            D : in STD_LOGIC_VECTOR (3 downto 0);
            R0 : out STD_LOGIC_VECTOR (3 downto 0);
            R1 : out STD_LOGIC_VECTOR (3 downto 0);
            R2 : out STD_LOGIC_VECTOR (3 downto 0);
            R3 : out STD_LOGIC_VECTOR (3 downto 0);
            R4 : out STD_LOGIC_VECTOR (3 downto 0);
            R5 : out STD_LOGIC_VECTOR (3 downto 0);
            R6 : out STD_LOGIC_VECTOR (3 downto 0);
            R7 : out STD_LOGIC_VECTOR (3 downto 0);
            I : in STD_LOGIC_VECTOR (2 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;
            D : in STD_LOGIC_VECTOR (2 downto 0);
            O : out STD_LOGIC_VECTOR (2 downto 0));
end component;
component Instruction_Decoder
    Port ( I : in STD_LOGIC_VECTOR (12 downto 0);
            Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
            Load_Select : out STD_LOGIC;
            Imm_Value : out STD_LOGIC_VECTOR (3 downto 0);
            Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
            Reg_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
            Reg_Select_2 : out STD_LOGIC_VECTOR (2 downto 0);
            Add_Sub : out STD_LOGIC;
            Jump_Flag : out STD_LOGIC;
            Address : out STD_LOGIC_VECTOR (2 downto 0);
            Mul : out STD_LOGIC
```

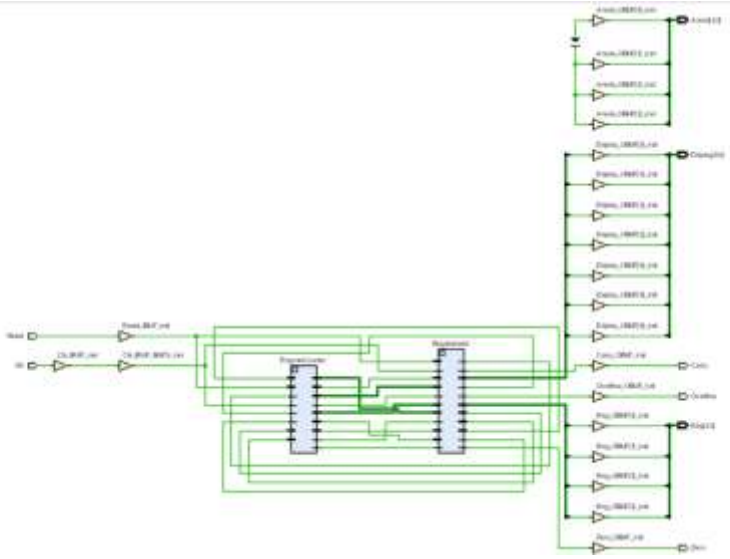
```
           );
end component;
component Mux_2_to_1_3bit
    Port ( Sel : in STD_LOGIC;
           D0 : in STD_LOGIC_VECTOR (2 downto 0);
           D1 : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;
component Mux_2_to_1_4bit
    Port ( S : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           X : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component Mux_8_to_1_4bit
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           A0 : in STD_LOGIC_VECTOR (3 downto 0);
           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);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;
component Program_ROM
    Port ( Mem_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Ins_Bus : out STD_LOGIC_VECTOR (12 downto 0));
end component;

-- Me
component Multiplier_2
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 Load_Select,Add_Sub_Selector,Jump_Flag,Overflow_0,Z_out,mul :
STD_LOGIC;
signal Ins_Bus : STD_LOGIC_VECTOR (12 downto 0);
signal O,b,Add_Out,Reg_Enable,Reg_Select_1,Reg_Select_2,c_out_0,S0 :
STD_LOGIC_VECTOR (2 downto 0);
signal Address : STD_LOGIC_VECTOR (2 downto 0);
signal Imm_Value,R0,R1,R2,R3,R4,R5,R6,R7,S,X,Y1,Y2,Y,output :
STD_LOGIC_VECTOR (3 downto 0);
--signal R: STD_LOGIC_VECTOR (3 downto 0);
signal Clk_out : STD_LOGIC;
```

```
begin

SlowClock : Slow_Clk
    Port map (
        Clk_in => Clk,
        Clk_out => Clk_out
    );

LUT :LUT_16_7
    Port map (
        address => R7,
        Data => Display
    );

ProgramCounter : PC
    Port Map(
        Clk => Clk,
        Reset => Reset,
        D => b,
        O => O
    );

Multiplier2 : Multiplier_2
     Port Map (
       A => Y1(1 downto 0),
       B => Y2(1 downto 0),
       Y=>  Y
    );

ProgramRom : Program_Rom
    Port Map(
        Ins_Bus => Ins_Bus,
        Mem_Sel => O
    );
InstructionDecoder: Instruction_Decoder
    Port Map(
       I => Ins_Bus,
       Reg_Check_Jump => Y1,
       Load_Select => Load_Select,
       Imm_Value  => Imm_Value,
       Reg_Enable => Reg_Enable,
       Reg_Select_1 => Reg_Select_1,
       Reg_Select_2 => Reg_Select_2,
       Add_Sub => Add_Sub_Selector,
       Jump_Flag => Jump_Flag,
       Address => Address,
       mul => mul
```

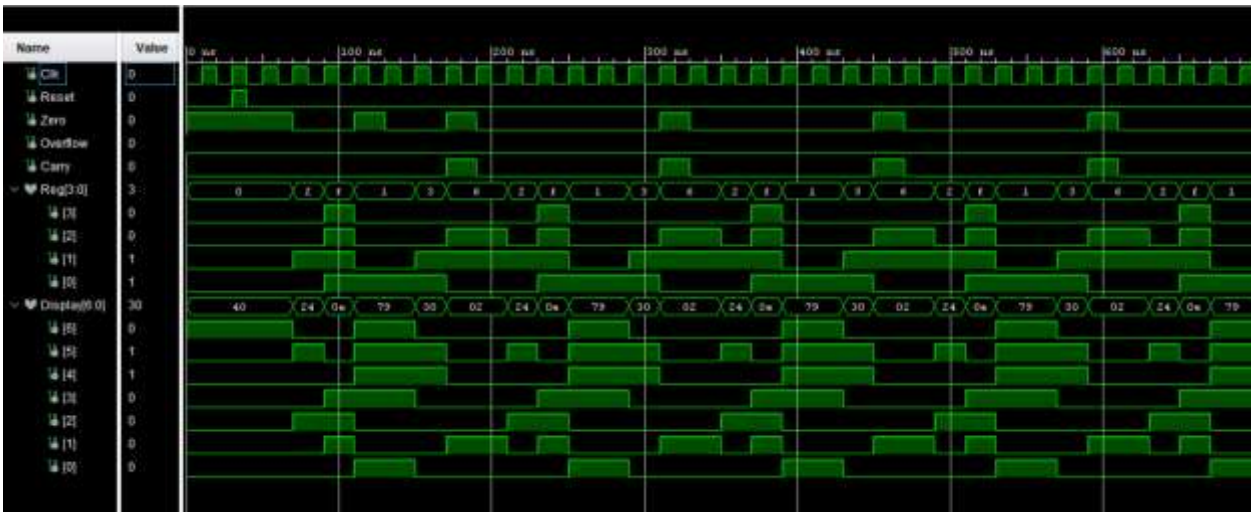
```
    );
Mux_2_to_1_4bit_0 : Mux_2_to_1_4bit
        Port Map (
           S => mul,
           A => S,
           B => Y,
           X => output
        );
Mux_2_to_1_4bit_1 : Mux_2_to_1_4bit
    Port Map (
       S => Load_Select,
       A => output,
       B => Imm_Value,
       X => X
    );
Registerbank : Register_Bank
    Port Map (
       Clk => Clk,
       Reset => Reset,
       D => X,
       R0 => R0,
       R1 => R1,
       R2 => R2,
       R3 => R3,
       R4 => R4,
       R5 => R5,
       R6 => R6,
       R7 => R7,
       I => Reg_Enable
    );
Mux_8_to_1_4bit_0 : Mux_8_to_1_4bit
    Port Map (
       S => Reg_Select_1,
       A0 => R0,
       A1 => R1,
       A2 => R2,
       A3 => R3,
       A4 => R4,
       A5 => R5,
       A6 => R6,
       A7 => R7,
       Y => Y1
    );
Mux_8_to_1_4bit_1 : Mux_8_to_1_4bit
    Port Map (
       S => Reg_Select_2,
       A0 => R0,
       A1 => R1,
```

```
        A2 => R2,
        A3 => R3,
        A4 => R4,
        A5 => R5,
        A6 => R6,
        A7 => R7,
        Y => Y2
    );
AddSub : Add_Sub
    Port Map (
        A => Y1,
        B => Y2,
        Ctrl => Add_Sub_Selector,
        C_out => Carry,
        S => S,
        Overflow => Overflow_0,
        Z_out => Z_out
    );
Mux_2_to_1_3bit_0 : Mux_2_to_1_3bit
    Port map (
        Sel => Jump_Flag,
        D0 => Add_Out,
        D1 => Address,
        Y => b
    );
Adder3bit : Adder_3bit
    Port Map(
        A => O,
        S => Add_Out
    );
Reg <= R7;
Zero <= Z_out;
Overflow <= Overflow_0;
Anode <= "1110";
end Behavioral;
```

- **Schematic Diagram**



- **Timing Diagram**



❖ **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 {Reg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Reg[0]}]
set_property PACKAGE_PIN E19 [get_ports {Reg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Reg[1]}]
set_property PACKAGE_PIN U19 [get_ports {Reg[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Reg[2]}]
set_property PACKAGE_PIN V19 [get_ports {Reg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Reg[3]}]
set_property PACKAGE_PIN N3 [get_ports {Overflow}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Overflow}]
set_property PACKAGE_PIN P1 [get_ports {Zero}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]
set_property PACKAGE_PIN L1 [get_ports {Carry}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Carry}]


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

#set_property PACKAGE_PIN V7 [get_ports dp]
    #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {Anode[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode[1]}]
```

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


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

## ❖ Extra Features added

We added a 2 bit multiplier which multiply 2bit numbers.

### 2 bit Multiplier

### Source file

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Multiplier_2 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_2;
architecture Behavioral of Multiplier_2 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 <= B(0) and A(0);
b1a0 <= B(1) and A(0);
b0a1 <= B(0) and A(1);
b1a1 <= B(1) and A(1);

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

end Behavioral;
```

**TB file**

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

entity TB_Multiplier_2 is
end TB_Multiplier_2;

architecture Behavioral of TB_Multiplier_2 is

component Multiplier_2 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 component;

SIGNAL A,B:STD_LOGIC_VECTOR (1 downto 0);
SIGNAL Y :STD_LOGIC_VECTOR (3 downto 0);

begin

UUT:Multiplier_2 PORT MAP(A,B,Y);

process
begin
                --- 220444K - 0b 11 01 01 11 01 00 01 11 00
                --- 220297K - 0b 11 01 01 11 00 10 00 10 01
```

```
                --- 220625R - 0b 11 01 01 11 01 11 01 00 01
                --- 220671D - 0b 11 01 01 11 01 11 11 11 11

A <= "11";
B <= "00";
WAIT FOR 100 ns;
A <= "10";
B <= "01";
WAIT FOR 100ns;
A <= "00";
B <= "01";
WAIT FOR 100ns;
A <= "11";
B <= "11";
WAIT FOR 100ns;

end process;
end Behavioral;
```

**Timing Diagram**


Also we separately added a subtractor which can directly subtract two numbers from one another

❖ **Strategies to increase optimization**

We created the instruction decoder without any use of clock input. Initially, we created the instruction decoder which will decode the instruction for the rising edge of the clock. At that time we used conditional statements to decode the instructions.

After that we designed the instruction decoder without any usage of clock. We analyzed the logic implementation which will be correct according to all four instructions. As a result, the instruction will be decoded immediately as the instruction arrives in the input port and the time delay for decoding has been removed.

In order to decode the negation statement we used the 2-way-3bit multiplexer for sending data to the register selector because we have to send the data to the second 8- way-4bit multiplexer.

❖ **LUT Counts**

1. Slice Logic

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs* | 31 | 0 | 20800 | 0.15 |
| LUT as Logic | 31 | 0 | 20800 | 0.15 |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| Slice Registers | 15 | 0 | 41600 | 0.04 |
| Register as Flip Flop | 15 | 0 | 41600 | 0.04 |
| Register as Latch | 0 | 0 | 41600 | 0.00 |
| F7 Muxes | 0 | 0 | 16300 | 0.00 |
| F8 Muxes | 0 | 0 | 8150 | 0.00 |

❖ **Contribution of each member**

| **Name** | **Components** |
|---|---|
| R.Parishith | 4bit Add/ Subract unit<br>3-bit Adder Instruction<br>3-bit Programming Counter<br>Multiplier<br>Decoder (All) |
| S.Kajaluxsan | 2-way-3bit Multiplexer<br>2-way-4bit Multiplexer<br>8-way-4bit Multiplexer<br>Instruction Decoder (All) |
| T.Sukithan | 4-bit Register<br>Register bank<br>Slow Clock<br>Instruction Decoder (All) |
| N.Vithursanaa | Program ROM LUT 16 to 7<br>Instruction Decoder (All) |

## ❖ Conclusion

• This circuit required the creation/extension of various components.

• We created a 4-bit arithmetic unit that can only handle smaller tasks, such as adding and subtracting 4-bit signed numbers. We are limited to working with integers between -7 to 8.

• Instead of connecting so many wires, we used busses which simplified the design circuit.

• We recalled several past lab activities in order to develop the nano processor (4bit RCA-Lab3, D flipflop-Lab5, 3 to 8 decoder, multiplexer-Lab4, ROM-based LUT□Lab7) which make our work easier.

• We hardcoded our program to ROM because microprocessor only understands the machine language.

• Middle push button is used to reset the PC and register bank.

• We used the slow clock to drive our nano processor in order to be able to detect changes. • By verifying each and every component's functionality via simulation and on the development board, we confirmed the proper work of nano processor.

• Through this work, we gained a wide knowledge about how microprocessor worked internally. Our teamwork abilities, such as coordination and communication, are improved through this project. Five of us divided up the tasks, which we later merged to create a nano processor