GROUP 48

NANOPROCESSOR

220213D – Harinakshi W.B.S.S.

220303E – Kariyawasam U.G.R.T.

220407C – Muthumala V.D.W.

220419N – Navodi S.Y.A.C.

Assigned Lab Task

The assigned lab task is to design and develop a 4-bit nanoprocessor capable of executing 4 instructions. This includes several components that need to be developed or extended from the previous labs:

- **4-bit Add/Subtract unit:** Capable of adding and subtracting numbers represented using 2's complement.
- **3-bit Adder:** Used to increment the Program Counter.
- **3-bit Program Counter (PC):** Built using D Flip Flops with a clear/reset input.
- **k-way b-bit multiplexers or tri-state busses:** Includes building 2-way 3-bit and 2-way 4-bit multiplexers, as well as an 8-way 4-bit multiplexer. Alternatively, tri-state buffers can be used.
- **Register Bank:** Contains 8, 4-bit registers (R0 to R7), with R0 hardcoded to all 0s.
- **Program ROM:** Stores the Assembly program, built by extending the ROM-based LUT.
- **Buses:** Use 3, 4, and 12-bit buses (D(3 downto 0), I(11 downto 0), M(3 downto 0), R(3 downto 0)) to connect components.
- **Instruction Decoder:** Design and build the circuit to activate necessary components based on instructions.

Apart from the above components we decided to improve the nanoprocessor by allowing it to support more instructions using the below components

- **4-bit Comparator unit:** Compares 2, 4-bit numbers and signals the result through 3 distinct LEDs indicating whether the first number is greater, equal or lesser in value than the second number.
- **Logical Unit:** Performs logical AND, OR and XOR between 2 input numbers of 4 bit as well as checks whether the first number is odd or even according to the selected operation out of above 4.
- **Bit Shifter:** Performs left or right shifts on a 4-bit input number based on the shift direction and the number of digits specified in the input

The lab also includes steps for building the circuits, testing components using simulation, writing an Assembly program, connecting inputs and outputs (including LEDs and 7-segment display), testing on BASYS 3, and submitting a lab report detailing the project, VHDL codes, timing diagrams, conclusions, and team member contributions

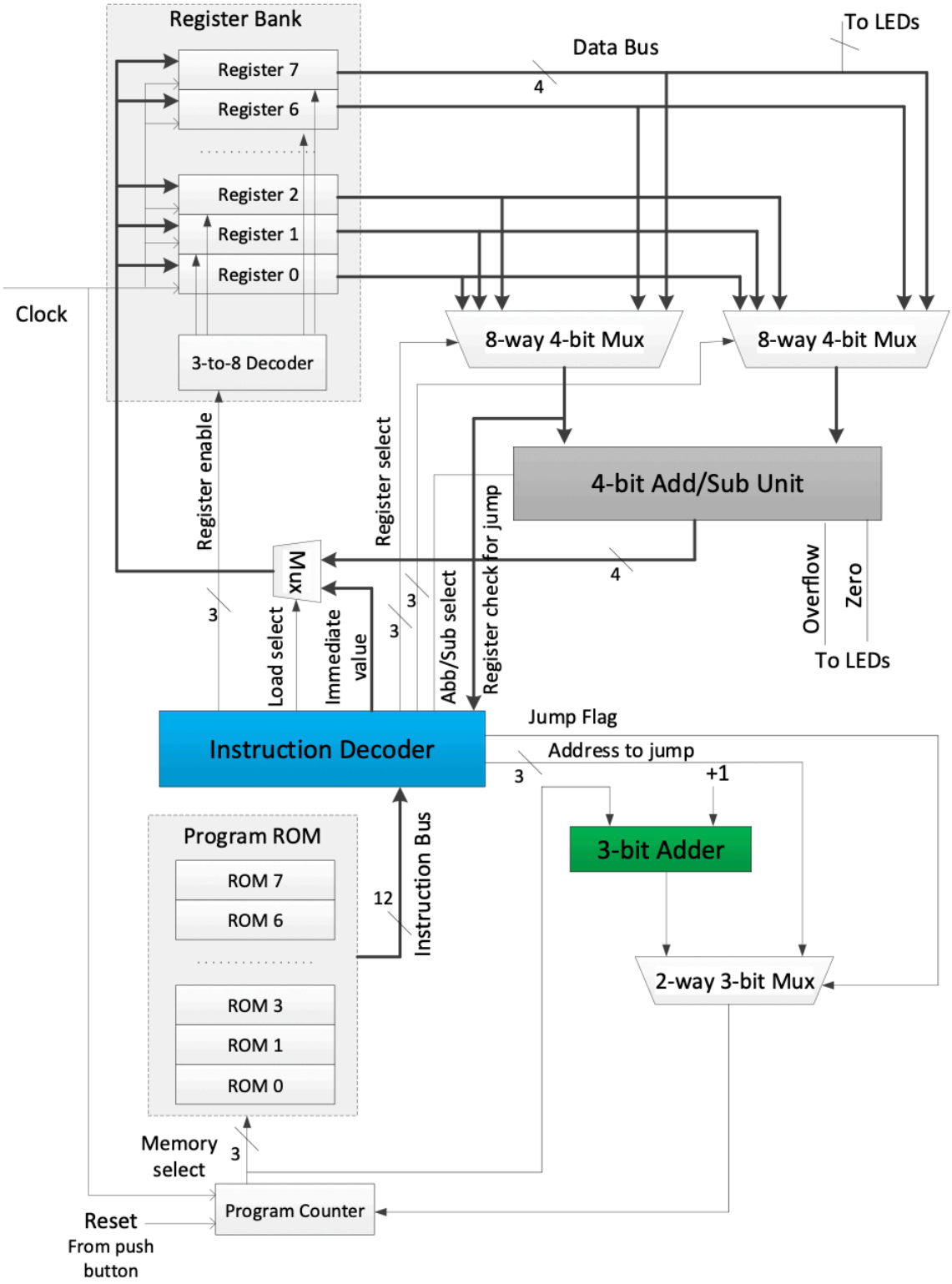
Calculating the total of integers between 1 and 3 using nanoprocessor

In the basic nanoprocessor binary instructions stored in the instruction bus consist of 12 bits. Then we optimized the nanoprocessor to support compare, perform logical operations and bit shifting and increased the instruction bus to 13 bits. Those instructions have the following structure.

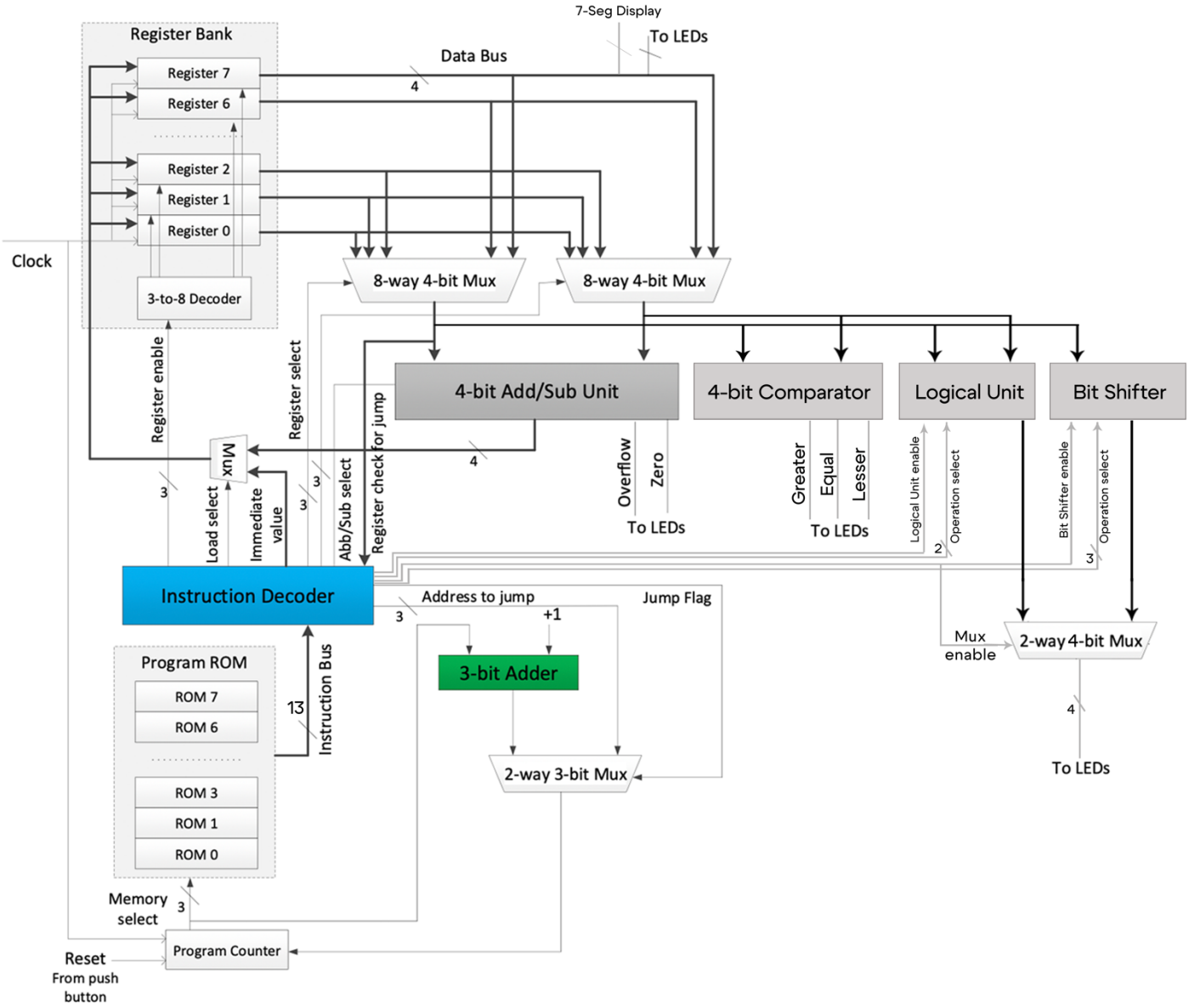
Instruction	Description	Format (13-bit instruction)
MOVI R, *d*	Move immediate value *d* to register R, i.e., *R* ← *d* R ∈ [0, 7], *d* ∈ [0, 15]	0 1 0 R R R 0 0 0 d d d d
ADD Ra, Rb	Add values in registers Ra and Rb and store the result in Ra, i.e., Ra ← Ra + Rb Ra, Rb ∈ [0, 7]	0 0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0
NEG R	2's complement of registers R, i.e., R ← – R R ∈ [0, 7]	0 0 1 R R R 0 0 0 0 0 0 0
JZR R, d	Jump if value in register R is 0, i.e., If R == 0 PC ← d; Else PC ← PC + 1; R ∈ [0, 7], *d* ∈ [0, 7]	0 1 1 R R R 0 0 0 0 d d d
CMP Ra, Rb	Compares the values in the registers Ra and Rb	1 0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0
LOG Ra, Rb, *d*	Performs the logical operation d between the values in registers Ra and Rb	1 0 1 Ra Ra Ra Rb Rb Rb 0 0 d d
SFT R, *d*, *n*	Shifts the bits of the value in register R by *n* places to the direction *d*	1 1 0 R R R 0 0 0 0 d n n

```
------------ machine language code for operations -------------

"0101110000001",      --- MOVI R7, 1 ; R7 <= 1
"0100100000010",      --- MOVI R2, 2 ; R2 <= 2
"0100110000011",      --- MOVI R3, 3 ; R3 <= 3
"0001110100000",      --- ADD R7, R2 ; R7 <= R2 + R7
"0001110110000",      --- ADD R7, R3 ; R7 <= R3 + R7
"0110000000110",      --- JZR R0, 6
"0110000000101",      --- JZR R0, 5
"0000000000000"
```

Block level diagram of the nanoprocessor



Block level diagram of the nanoprocessor after optimizing it to support more instructions

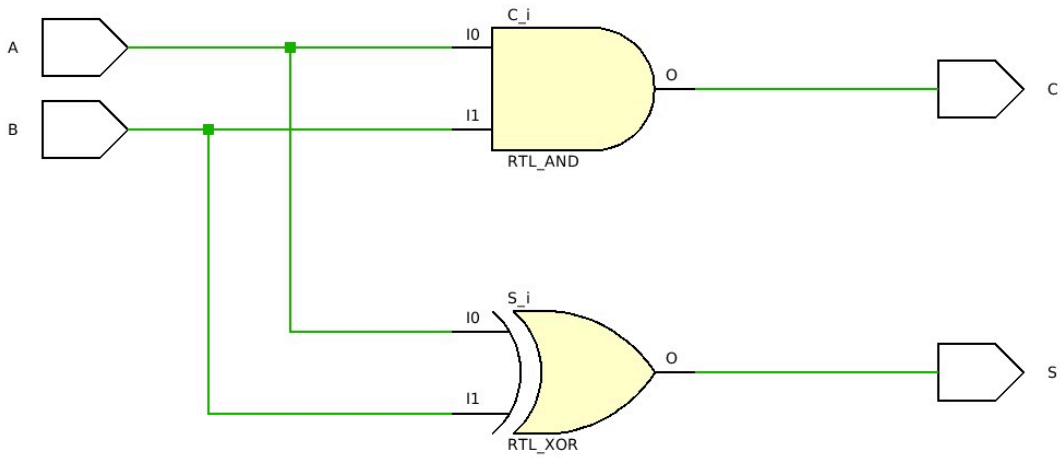


VHDL Design Codes, Design Schematics, Test Bench Codes and Timing Diagrams of the Components

Half Adder - Design

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

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

architecture Behavioral of HA is

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

end Behavioral;
```

Half Adder Design Schematic

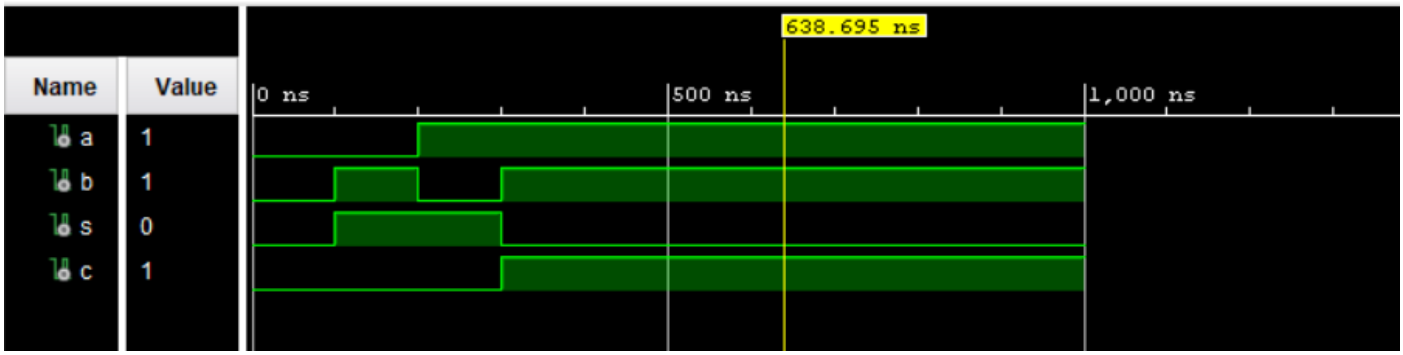


Half Adder - TB

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

architecture Behavioral of TB_HA is

 COMPONENT HA
    PORT( A, B : IN STD_LOGIC;
        S, C : OUT STD_LOGIC);
 END COMPONENT;
    SIGNAL a, b : std_logic;
    SIGNAL s, c : std_logic;
```

```
begin
    UUT: HA PORT MAP(
        A => a,
        B => b,
        S => s,
        C => c
    );
    process
     begin
     a <= '0'; -- set initial values
     b <= '0';
     WAIT FOR 100 ns; -- after 100 ns change inputs
     b <= '1';
     WAIT FOR 100 ns; --change again
     a <= '1';
     b <= '0';
     WAIT FOR 100 ns; --change again
     b <= '1';
     WAIT; -- will wait forever
    end process;
```

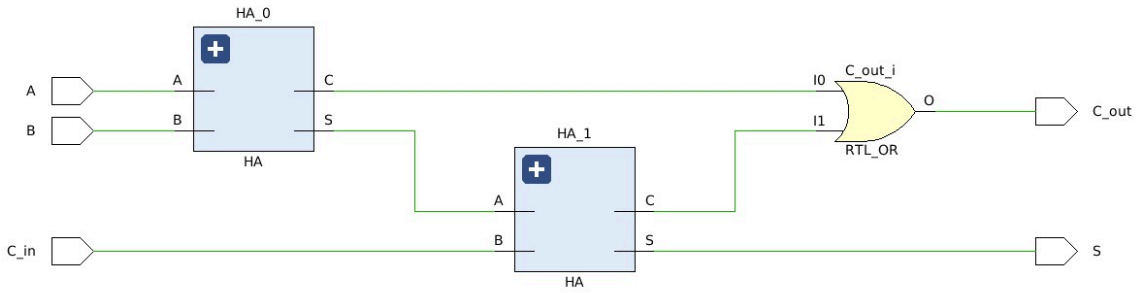
Half Adder Timing Diagram



Full Adder - Design

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

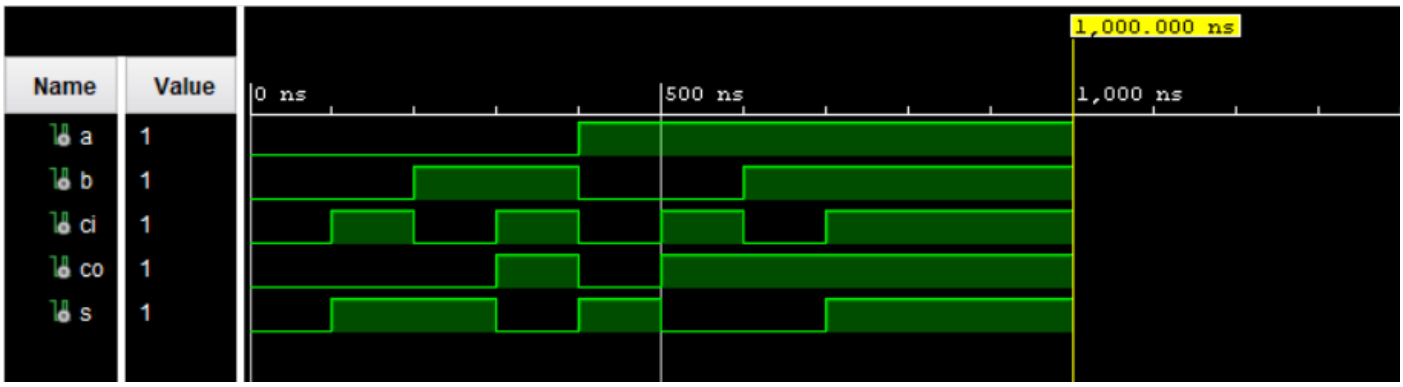
Full Adder Design Schematic



Full Adder - TB

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

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

architecture Behavioral of TB_FA is
 COMPONENT FA
    PORT( A, B, C_in : IN STD_LOGIC;
        C_out, S : OUT STD_LOGIC);
    END COMPONENT;
     SIGNAL a, b, ci : std_logic;
     SIGNAL co, s : std_logic;
begin
UUT: FA PORT MAP(
    A => a,
    B => b,
    C_in => ci,
    S => s,
    C_out => co);
process
      begin
      a <= '0'; -- set initial values
      b <= '0';
      ci <= '0';
      WAIT FOR 100 ns; -- after 100 ns change inputs
      ci <= '1';
      WAIT FOR 100 ns;
      b <= '1';
      ci <= '0';
      WAIT FOR 100 ns;
      ci <= '1';
      WAIT FOR 100 ns;
      a <= '1';
      b <= '0';
      ci <= '0';
      WAIT FOR 100 ns;
      ci <= '1';
      WAIT FOR 100 ns;
      b <= '1';
      ci <= '0';
      WAIT FOR 100 ns;
      ci <= '1';
      WAIT; -- will wait forever
     end process;
end Behavioral;
```

Full Adder Timing Diagram

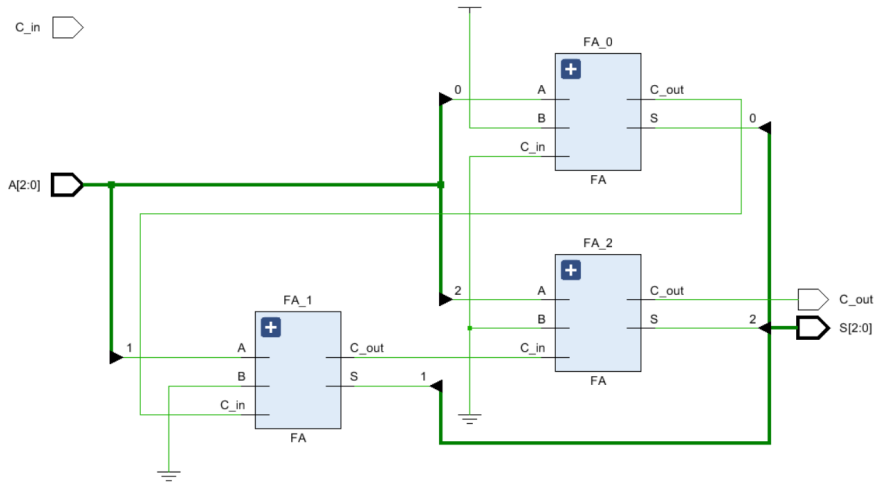


3-bit Adder - Design

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


entity Adder_3_B is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           C_in : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (2 downto 0);
           C_out : out STD_LOGIC);
end Adder_3_B;

architecture Behavioral of Adder_3_B 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 FA_C: std_logic_vector (2 downto 0);

begin

  FA_0 : FA
   port map (
     A => A(0),
     B => '1',
     C_in => '0',
     S => S(0),
     C_Out => FA_C(0));

  FA_1 : FA
   port map (
     A => A(1),
     B => '0',
     C_in => FA_C(0),
     S => S(1),
     C_Out => FA_C(1));
```

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

end Behavioral;
```

3bit Adder Design Schematic

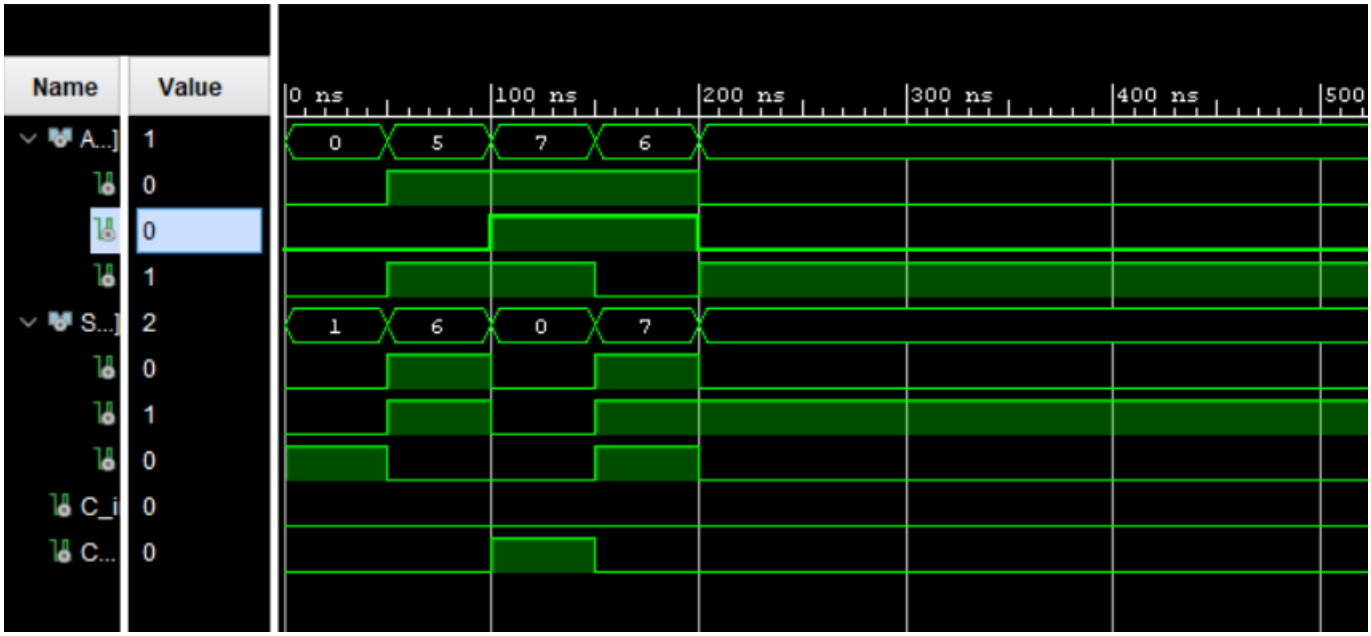


3-bit Adder - TB

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

entity TB_Adder_3_B is
--  Port ( );
end TB_Adder_3_B;

architecture Behavioral of TB_Adder_3_B is

component Adder_3_B
 Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
      C_in : in STD_LOGIC;
      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_in,C_out:std_logic;

begin
```

```
begin

UUT:Adder_3_B
    PORT MAP(
        A => A,
        C_in => C_in,
        S => S,
        C_out => C_out);
process

begin
    C_in <= '0';

    A <= "000";
    wait for 50 ns;

    A <= "101";
    wait for 50 ns;

    A <= "111";
    wait for 50 ns;

    A <= "110";
    wait for 50 ns;

    A <= "001";

    wait;
end process;

end Behavioral;
```

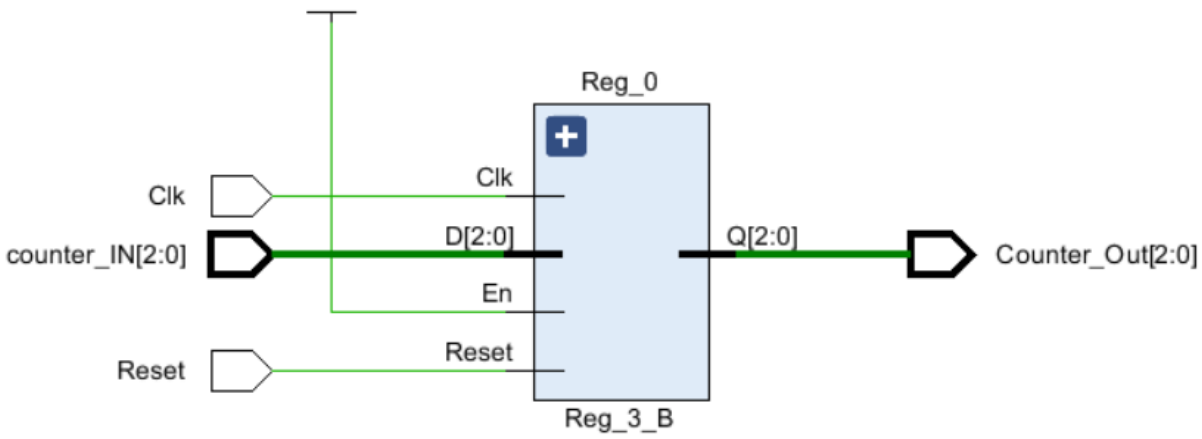
3bit Adder Timing Diagram



3-bit Program Counter (PC)

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

entity Program_Counter is
    Port ( Counter_IN : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Counter_Out : out STD_LOGIC_VECTOR (2 downto 0));
end Program_Counter;

architecture Behavioral of Program_Counter is

component Reg_3_B
 Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
        En : in STD_LOGIC;
        Clk : in STD_LOGIC;
        Reset : in STD_LOGIC;
        Q : out STD_LOGIC_VECTOR (2 downto 0));
 end component;

begin
    Reg_0 : Reg_3_B
        Port map (
            D => Counter_IN,
            EN => '1',
            Reset => Reset,
            Clk => Clk,
            Q => Counter_Out);

end Behavioral;
```

3bit Program Counter Design Schematic

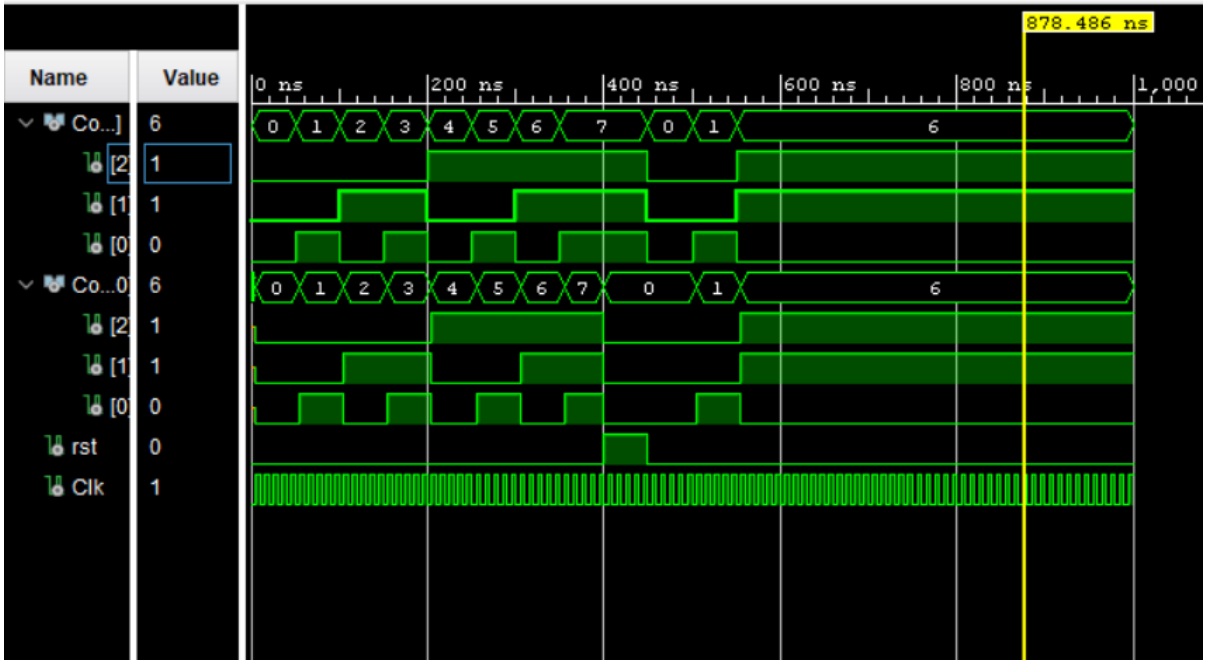


3-bit Program Counter (PC) - TB

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

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

architecture Behavioral of TB_Program_Counter is

Component Program_Counter
    Port ( Counter_IN : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Counter_Out : out STD_LOGIC_VECTOR (2 downto 0));
End Component;

    SIGNAL Count_in, Count_out : STD_LOGIC_VECTOR(2 downto 0);
    SIGNAL rst, Clk : STD_LOGIC := '0';

begin

UUT : Program_Counter PORT MAP(
        Counter_IN => Count_in,
        Reset => rst,
        Clk => Clk,
        Counter_Out => Count_out);

clock_process: process
        begin
            Clk <= '0';
            WAIT FOR 5 ns;
            Clk <= '1';
            WAIT FOR 5 ns;
      end process;

process
      begin
          -- rst = 0 means output will  be same as input otherwise output
will be 000
          rst <= '0';
          Count_in <= "000";
          wait for 50ns;

          Count_in <= "001";
          wait for 50ns;

           Count_in <= "010";
          wait for 50ns;

          Count_in <= "011";
          wait for 50ns;
```

```
            Count_in <= "100";
            wait for 50ns;

            Count_in <= "101";
            wait for 50ns;

            Count_in <= "110";
            wait for 50ns;

            Count_in <= "111";
            wait for 50ns;

            rst <= '1';
            wait for 50ns;
            rst <= '0';
            Count_in <= "000";
            wait for 50ns;

            Count_in <= "001";
            wait for 50ns;

            Count_in <= "110";
            wait for 50ns;

            wait; -- wait forever

    end process;

end Behavioral;
```

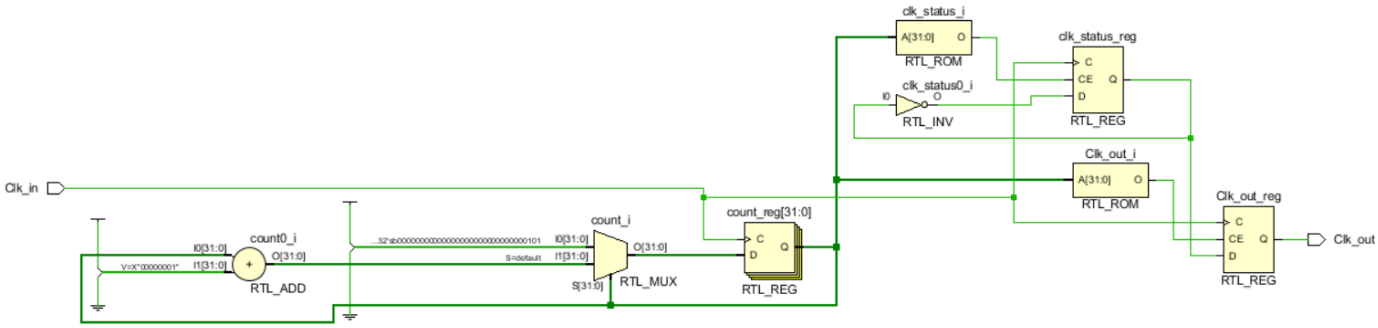
3bit Program Counter Timing Diagram



Slow down clock - Design

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

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

architecture Behavioral of Slow_Clk is

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

begin
    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1;
            if (count = 5) then
                clk_status <= not clk_status;
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

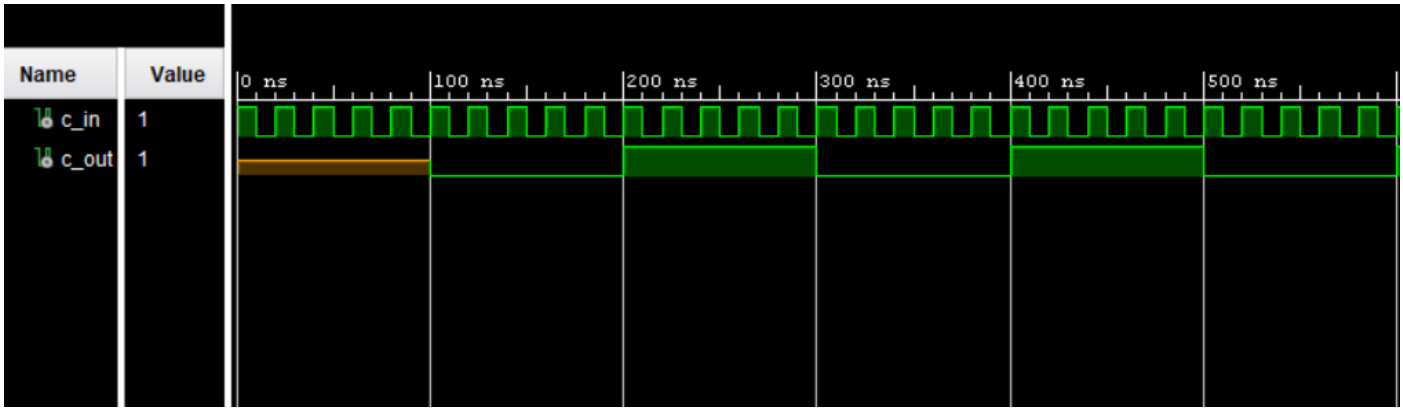
Slow down Clock Design Schematic



Slow down clock - TB

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


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

architecture Behavioral of TB_Slow_Clk is

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

signal c_in, c_out : std_logic;

begin
UTT: Slow_clk port map(
    Clk_in => c_in,
    Clk_out => c_out);

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

end Behavioral;
```

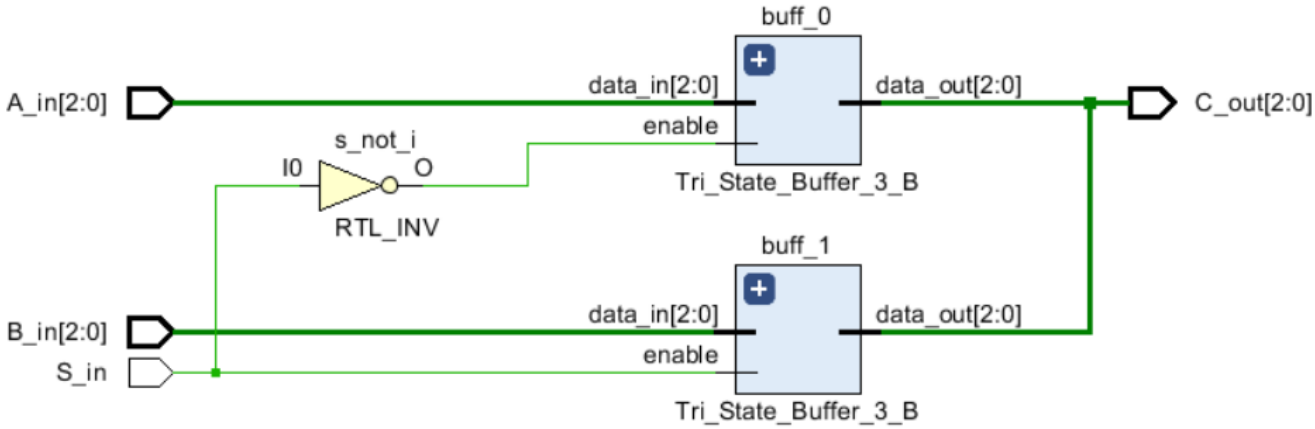
Slow down Clock Timing Diagram



2-way 3-bit Multiplexer - Design

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


entity Mux_2_W_3_B is
    Port ( A_in : in STD_LOGIC_VECTOR (2 downto 0);
           B_in : in STD_LOGIC_VECTOR (2 downto 0);
           S_in : in STD_LOGIC;
           C_out : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2_W_3_B;

architecture Behavioral of Mux_2_W_3_B is


component Tri_State_Buffer_3_B is
    Port ( data_in : in STD_LOGIC_VECTOR (2 downto 0);
          enable : in STD_LOGIC;
          data_out: out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal s,s_not: STD_LOGIC;

begin

s <= s_in;
s_not <= NOT s_in;

buff_0 : Tri_State_Buffer_3_B
    port map (
        data_in => A_in,
        enable => s_not,
        data_out => c_out);

buff_1 : Tri_State_Buffer_3_B
    port map (
        data_in => B_in,
;       enable => s,
        data_out => c_out);

end Behavioral
```

2-way 3-bit Multiplexer Design Schematic

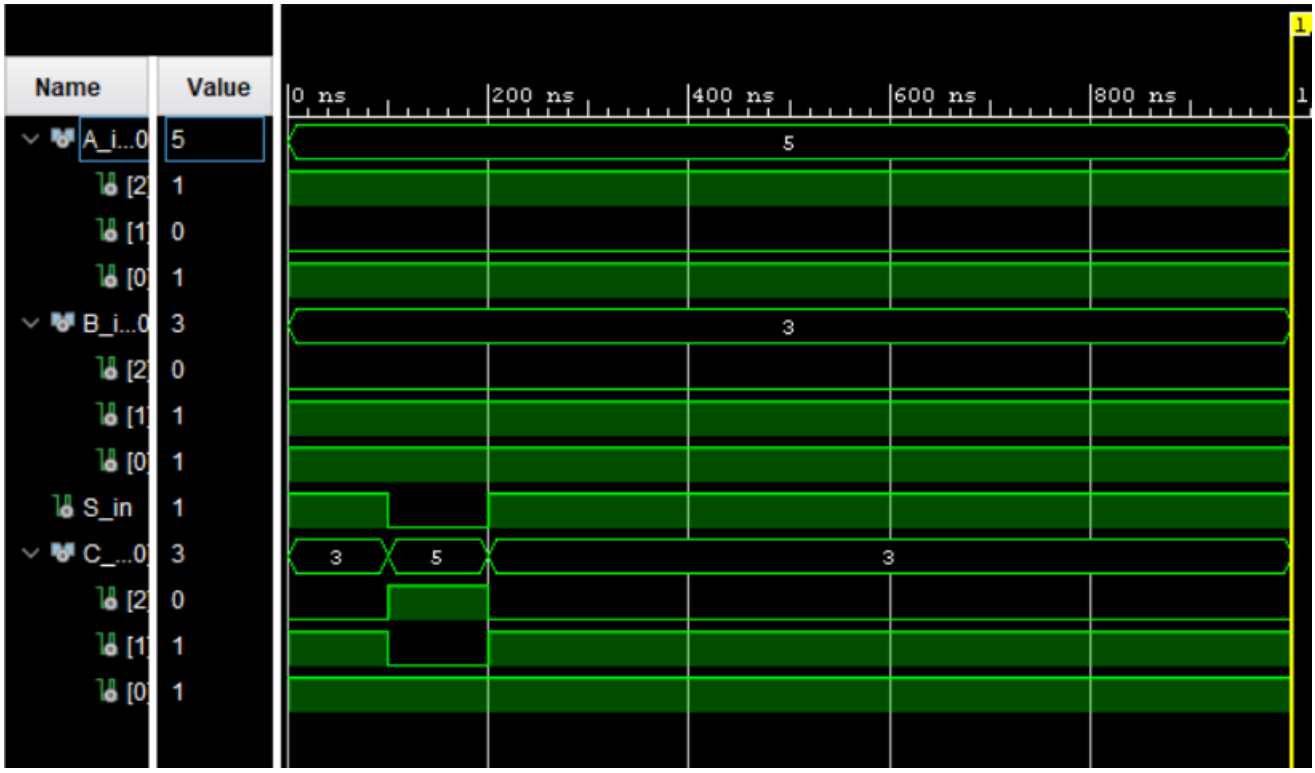


2-way 3-bit Multiplexer - TB

```
entity TB_Mux_2_W_3_B is
--  Port ( );
end TB_Mux_2_W_3_B;

architecture Behavioral of TB_Mux_2_W_3_B is

component Mux_2_W_3_B
     port(A_in : in STD_LOGIC_VECTOR (2 downto 0);
          B_in : in STD_LOGIC_VECTOR (2 downto 0);
          S_in : in STD_LOGIC;
          C_out : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal A_in : STD_LOGIC_VECTOR (2 downto 0);
signal B_in : STD_LOGIC_VECTOR (2 downto 0);
signal S_in : STD_LOGIC;
signal C_out : STD_LOGIC_VECTOR (2 downto 0);

begin

UUT: Mux_2_W_3_B
  PORT MAP(
    A_in => A_in,
    B_in => B_in,
    S_in => S_in,
    C_out => C_out);

process
    begin
      A_in <= "101";
      B_in <= "011";
      S_in <= '1';
```

```
        S_in <= '0';

        wait for 100 ns;
        S_in <= '1';
        wait;
    end process;

end Behavioral;
```

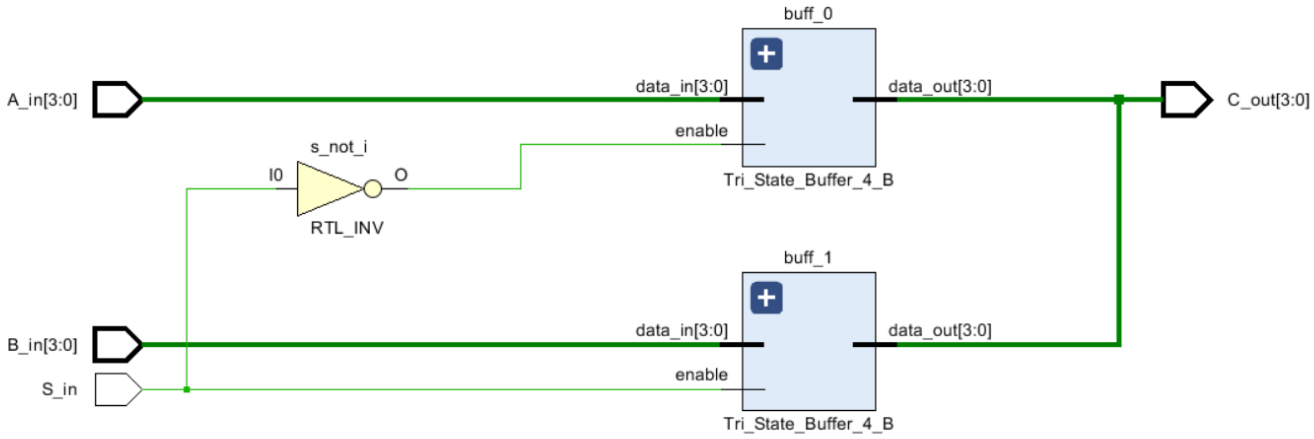
2-way 3-bit Multiplexer Timing Diagram



2-way 4-bit Multiplexer

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

entity Mux_2_W_4_B is
    Port ( S_in : in STD_LOGIC;
           A_in : in STD_LOGIC_VECTOR (3 downto 0);
           B_in : in STD_LOGIC_VECTOR (3 downto 0);
           C_out : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_2_W_4_B;

architecture Behavioral of Mux_2_W_4_B is

component Tri_State_Buffer_4_B is
    Port ( data_in : in STD_LOGIC_VECTOR (3 downto 0);
          enable : in STD_LOGIC;
          data_out: out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal s,s_not: STD_LOGIC;

begin

    s <= s_in;
    s_not <= NOT s_in;

    buff_0 : Tri_State_Buffer_4_B
        port map (
            data_in => A_in,
            enable => s_not,
            data_out => c_out);

    buff_1 : Tri_State_Buffer_4_B
        port map (
            data_in => B_in,
            enable => s,
            data_out => c_out);

end Behavioral;
```

2-way 4-bit Multiplexer Design Schematic

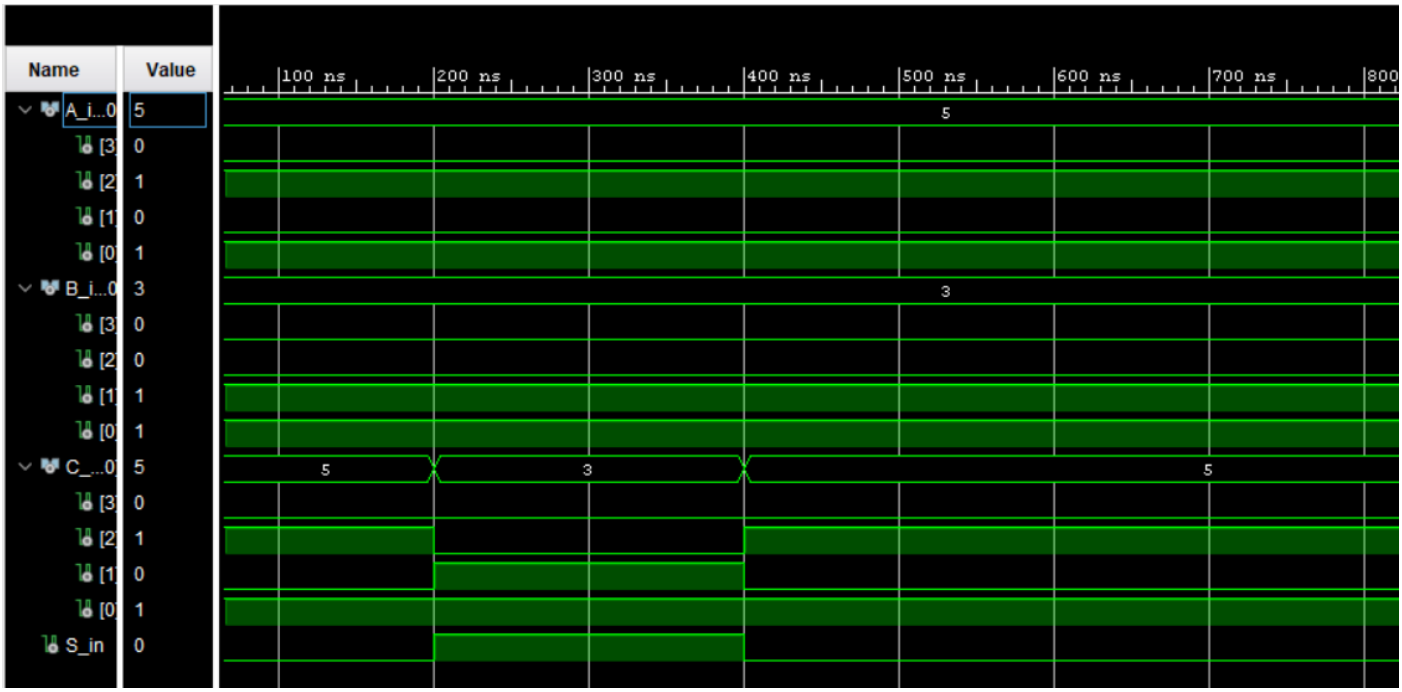


2-way 4-bit Multiplexer - TB

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

entity TB_Mux_2_W_4_B is
--  Port ( );
end TB_Mux_2_W_4_B;

architecture Behavioral of TB_Mux_2_W_4_B is


component Mux_2_W_4_B
           Port ( S_in : in STD_LOGIC;
           A_in : in STD_LOGIC_VECTOR (3 downto 0);
           B_in : in STD_LOGIC_VECTOR (3 downto 0);
           C_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL A_in,B_in,C_out:STD_LOGIC_VECTOR(3 downto 0);
SIGNAL S_in : STD_LOGIC ;

BEGIN
UUT: Mux_2_W_4_B
      PORT MAP(
      A_in => A_in,
         B_in => B_in,
         S_in => S_in,
         C_out => C_out) ;
```

```
process
begin

    A_in<="0101";
    B_in<="0011";
    S_in<='0';

    wait for 200 ns;
    S_in<='1';
    wait for 200 ns;
    S_in<='0';
    wait;
end process;

end Behavioral;
```

2-way 4-bit Multiplexer Timing Diagram

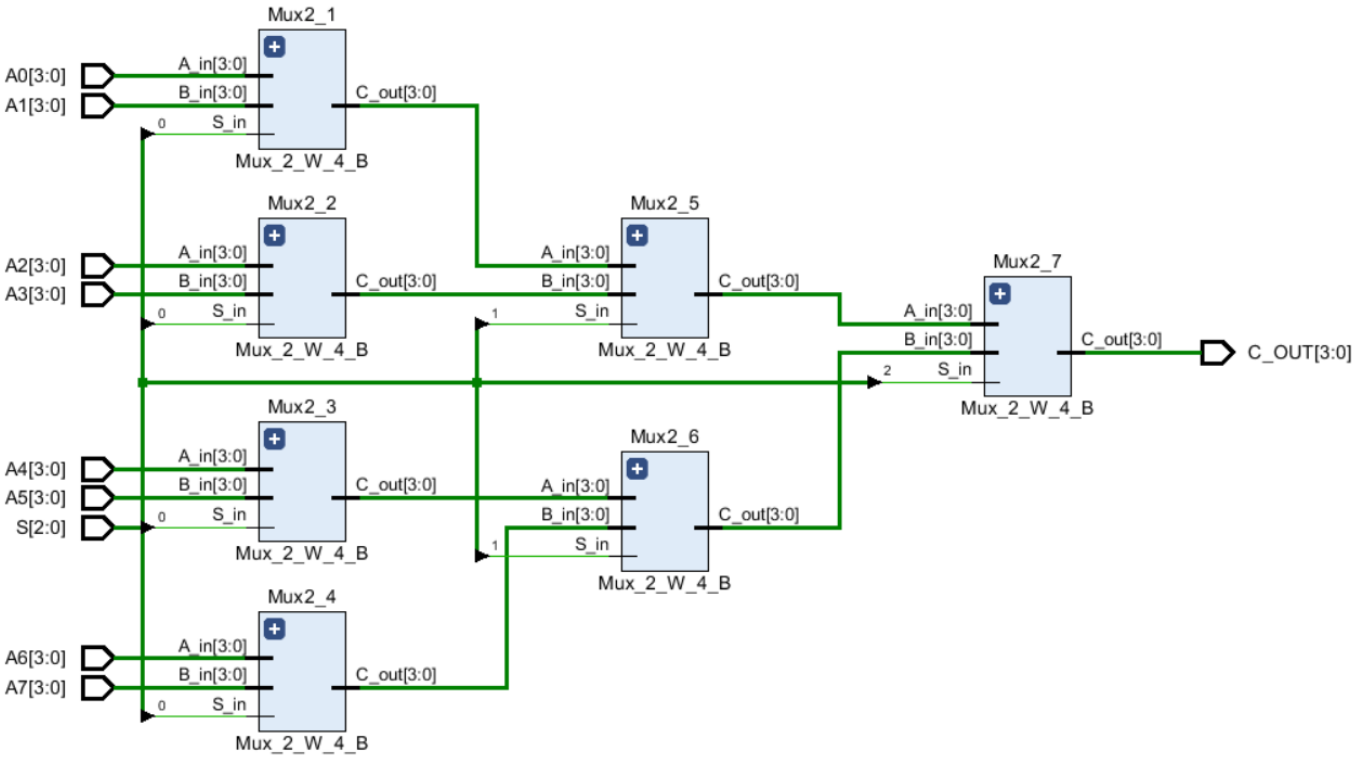


8-way 4-bit Multiplexer - Design

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

entity Mux_8_W_4_B is
    Port ( 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);
           C_OUT : out STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0));
end Mux_8_W_4_B;

architecture Behavioral of Mux_8_W_4_B is
    component Mux_2_W_4_B
        Port ( A_in : in STD_LOGIC_VECTOR (3 downto 0);
               B_in : in STD_LOGIC_VECTOR (3 downto 0);
               S_in : in STD_LOGIC;
               C_out : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    signal Mux_2_way_out_1, Mux_2_way_out_2, Mux_2_way_out_3,
Mux_2_way_out_4 : std_logic_vector (3 downto 0);
    signal Mux_2_way_out_5, Mux_2_way_out_6 : std_logic_vector (3 downto
0);

begin
    Mux2_1 : Mux_2_W_4_B
        port map (
            A_in => A0,
            B_in => A1,
            S_in => S(0),
            C_out => Mux_2_way_out_1);

    Mux2_2 : Mux_2_W_4_B
        port map (
            A_in => A2,
            B_in => A3,
            S_in => S(0),
            C_out => Mux_2_way_out_2);
```

```
    Mux2_3 : Mux_2_W_4_B
        port map (
            A_in => A4,
            B_in => A5,
            S_in => S(0),
            C_out => Mux_2_way_out_3);

    Mux2_4 : Mux_2_W_4_B
        port map (
            A_in => A6,
            B_in => A7,
            S_in => S(0),
            C_out => Mux_2_way_out_4);

    Mux2_5 : Mux_2_W_4_B
        port map (
            A_in => Mux_2_way_out_1,
            B_in => Mux_2_way_out_2,
            S_in => S(1),
            C_out => Mux_2_way_out_5);

    Mux2_6 : Mux_2_W_4_B
        port map (
            A_in => Mux_2_way_out_3,
            B_in => Mux_2_way_out_4,
            S_in => S(1),
            C_out => Mux_2_way_out_6);

    Mux2_7 : Mux_2_W_4_B
        port map (
            A_in => Mux_2_way_out_5,
            B_in => Mux_2_way_out_6,
            S_in => S(2),
            C_out => C_OUT);

end Behavioral;
```

8-way 4-bit Multiplexer Design Schematic

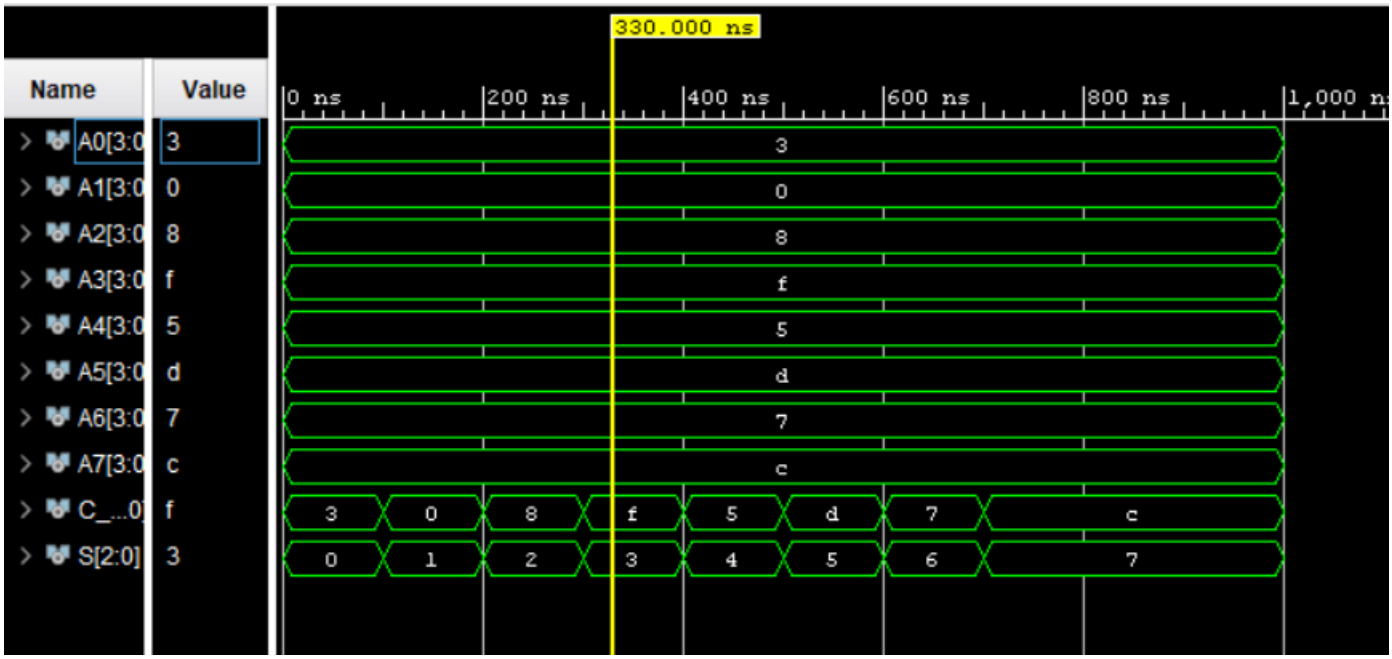


8-way 4-bit Multiplexer - TB

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

entity TB_Mux_8_W_4_B is
--  Port ( );
end TB_Mux_8_W_4_B;

architecture Behavioral of TB_Mux_8_W_4_B is

component MUX_8_W_4_B
    Port ( 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);
           C_OUT : out STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0));
end component;
```

```
signal A0,A1,A2,A3,A4,A5,A6,A7:STD_LOGIC_VECTOR(3 downto 0);
signal C_OUT:STD_LOGIC_VECTOR(3 downto 0);
signal S:STD_LOGIC_VECTOR(2 downto 0);

begin
UUT:MUX_8_W_4_B
    Port map(A0=>A0,
         A1=>A1,
         A2=>A2,
         A3=>A3,
         A4=>A4,
         A5=>A5,
         A6=>A6,
         A7=>A7,
         C_OUT=>C_OUT,
         S=>S);
  process
     begin

         A0 <= "0011";--3
         A1 <= "0000";--0
         A2 <= "1000";--8
         A3 <= "1111";--F
         A4 <= "0101";--5
         A5 <= "1101";--D
         A6 <= "0111";--7
         A7 <= "1100";--C
         S <= "000";

         WAIT FOR 100 ns;
         S <= "001";

         WAIT FOR 100 ns;
         S <= "010";

         WAIT FOR 100 ns;
         S <= "011";

         WAIT FOR 100 ns;
         S <= "100";

         WAIT FOR 100 ns;
         S <= "101";

         WAIT FOR 100 ns;
         S <= "110";

         WAIT FOR 100 ns;
         S <= "111";
         WAIT;
     end process;

end Behavioral;
```

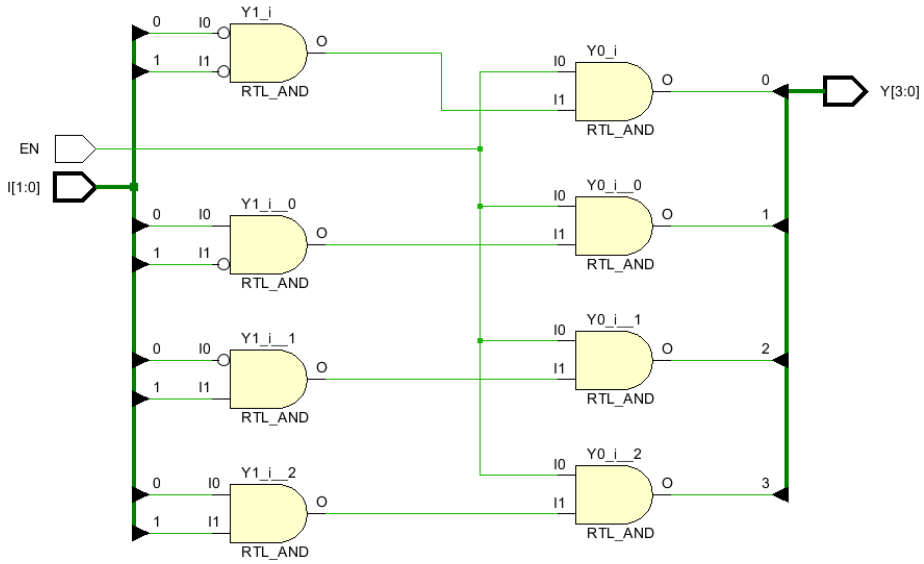
8-way 4-bit Multiplexer Timing Diagram



2-to-4 Decoder - Design

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

entity Decode_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 Decode_2_TO_4;

architecture Behavioral of Decode_2_TO_4 is

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

end Behavioral;
```

2-to-4 Decoder Design Schematic

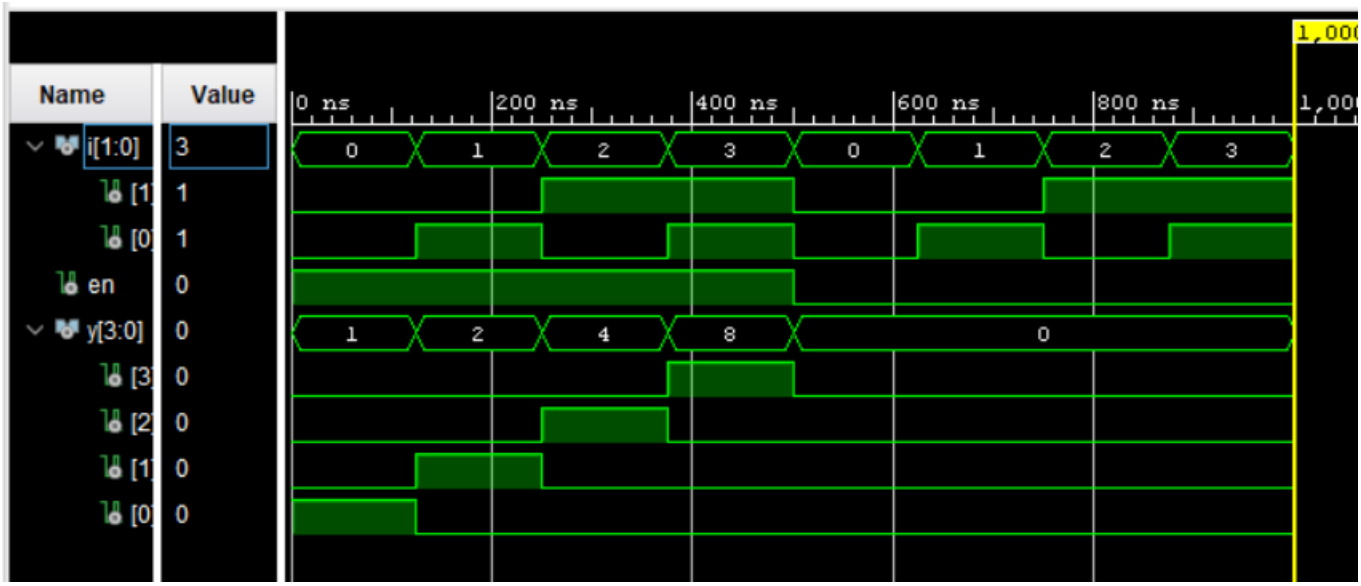


2-to-4 Decoder - TB

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

entity TB_Decode_2_To_4 is
--  Port ( );
end TB_Decode_2_To_4;

architecture Behavioral of TB_Decode_2_To_4 is

COMPONENT DECODE_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 i : STD_LOGIC_VECTOR (1 downto 0);
    SIGNAL en : STD_LOGIC;
    SIGNAL y : STD_LOGIC_VECTOR (3 downto 0);

begin
 UTT: DECODE_2_TO_4 PORT MAP(
    I => i,
    EN => en,
    Y => y
    );
```

```
process
     begin

          i <= "00";
          en <= '1';
          wait for 125 ns;
          i <= "01";
         wait for 125 ns;
          i <= "10";
          wait for 125 ns;
          i <= "11";
          wait for 125 ns;
           en <= '0';
           i <= "00";
          wait for 125 ns;
          i <= "01";
          wait for 125 ns;
          i <= "10";
          wait for 125 ns;
          i <= "11";
          wait;
    end process;
end Behavioral;
```

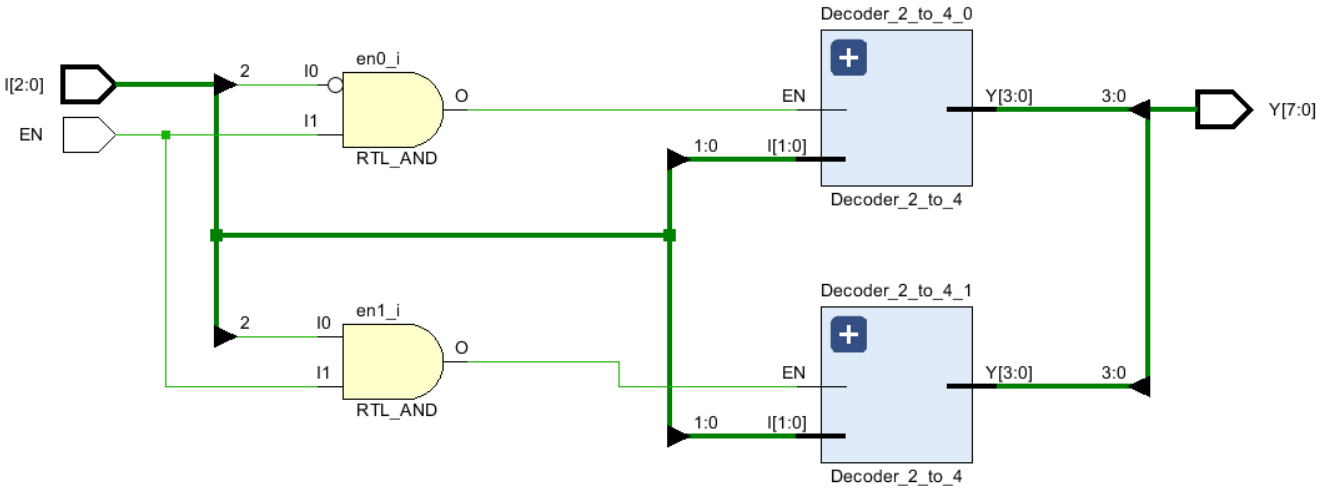
2-to-4 Decoder Timing Diagram



3-to-8 Decoder - Design

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

entity Decoder_3_TO_8 is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (7 downto 0));
end Decoder_3_TO_8;

architecture Behavioral of Decoder_3_TO_8 is

 component Decode_2_to_4
 port(
    I: in STD_LOGIC_VECTOR;
    EN: in STD_LOGIC;
    Y: out STD_LOGIC_VECTOR );
 end component;

    signal I0,I1 : STD_LOGIC_VECTOR (1 downto 0);
    signal Y0,Y1 : STD_LOGIC_VECTOR (3 downto 0);
    signal en0,en1, I2 : STD_LOGIC;

begin
 Decode_2_to_4_0 : Decode_2_to_4
 port map(
    I => I0,
    EN => en0,
    Y => Y0 );

 Decode_2_to_4_1 : Decode_2_to_4
 port map(
    I => I1,
    EN => en1,
    Y => Y1 );

     en0 <= NOT(I(2)) AND EN;
     en1 <= I(2) AND EN;

     I0 <= I(1 downto 0);
     I1 <= I(1 downto 0);
     I2 <= I(2);

     Y(3 downto 0) <= Y0;
     Y(7 downto 4) <= Y1;

end Behavioral;
```

3-to-8 Decoder Design Schematic

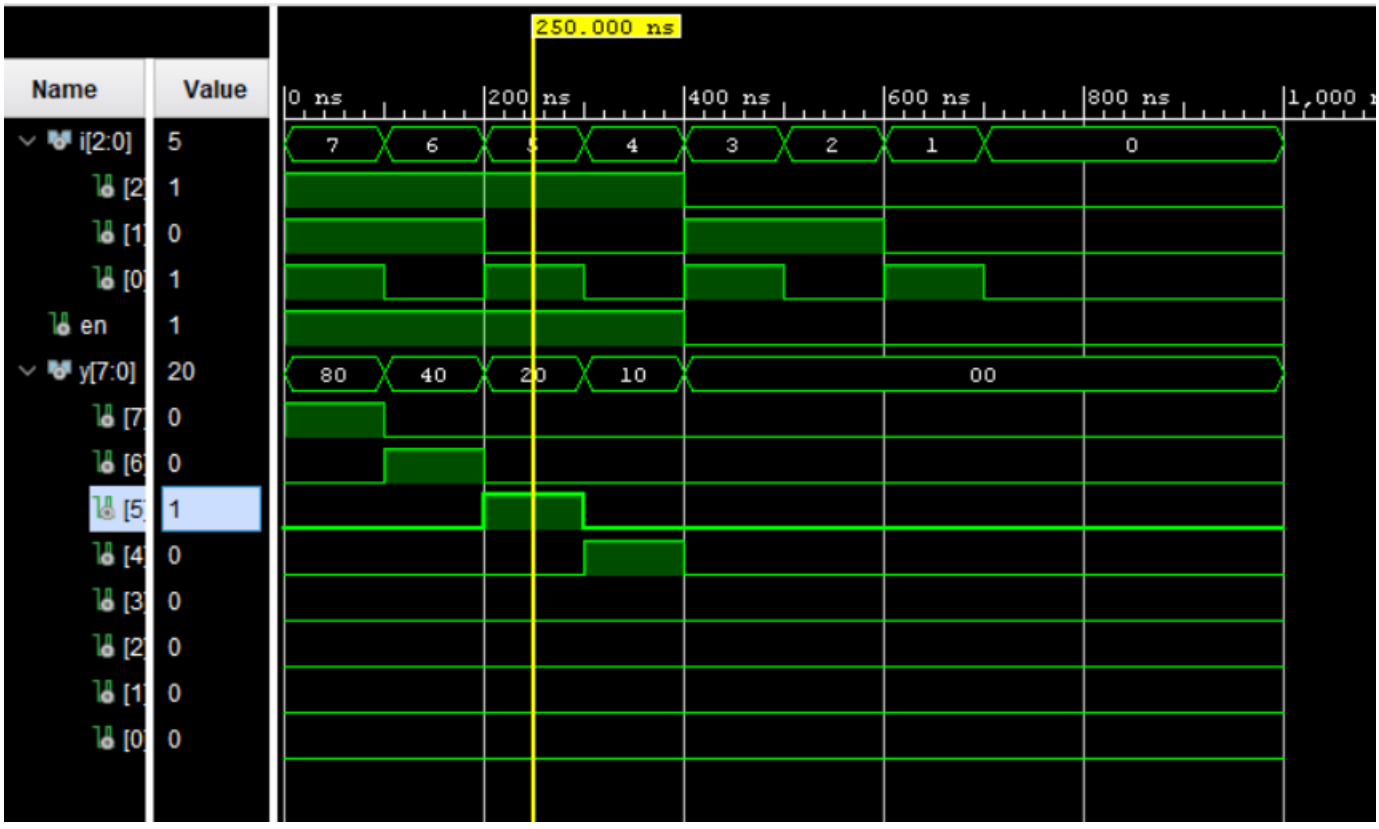


3-to-8 Decoder - TB

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

entity TB_Decode_3_To_8 is
--  Port ( );
end TB_Decode_3_To_8;

architecture Behavioral of TB_Decode_3_To_8 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;
    SIGNAL i : STD_LOGIC_VECTOR (2 downto 0);
    SIGNAL en : STD_LOGIC;
    SIGNAL y : STD_LOGIC_VECTOR (7 downto 0);
begin
    UTT: DECODER_3_TO_8 PORT MAP(
        I => i,
        EN => en,
        Y => y);
```

```
    process
    begin
        i <= "111";
        en <= '1';
        wait for 100 ns;
        i <= "110";
        wait for 100 ns;
        i <= "101";
        wait for 100 ns;
        i <= "100";
        wait for 100 ns;
        i <= "011";
        en <= '0';
        wait for 100 ns;
        i <= "010";
        wait for 100 ns;
        i <= "001";
        wait for 100 ns;
        i <= "000";

        wait;
        end process;
end Behavioral;
```

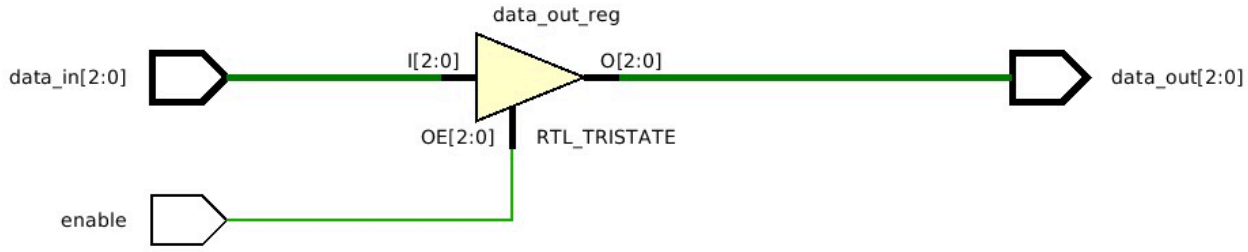
3-to-8 Decoder Timing Diagram



3-bit tri-state Buffer - Design

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


entity Tri_State_Buffer_3_B is
     Port ( data_in : in STD_LOGIC_VECTOR (2 downto 0);
          enable : in STD_LOGIC;
          data_out: out STD_LOGIC_VECTOR (2 downto 0));
end Tri_State_Buffer_3_B;

architecture Behavioral of Tri_State_Buffer_3_B is

begin

    data_out <= data_in WHEN(Enable='1') else "ZZZ";

end Behavioral;
```

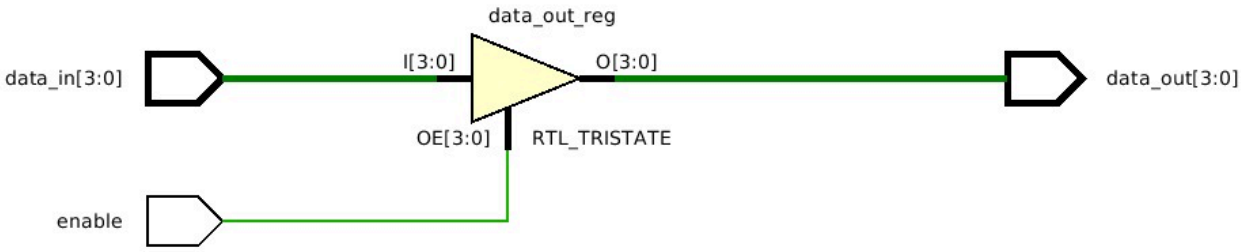
3-bit tri-state Buffer Design Schematic



4-bit tri-state Buffer - Design

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


entity Tri_State_Buffer_4_B is
     Port ( data_in : in STD_LOGIC_VECTOR (3 downto 0);
          enable : in STD_LOGIC;
          data_out: out STD_LOGIC_VECTOR (3 downto 0));
end Tri_State_Buffer_4_B;

architecture Behavioral of Tri_State_Buffer_4_B is

begin

    data_out <= data_in WHEN(Enable='1') else "ZZZ";

end Behavioral;
```

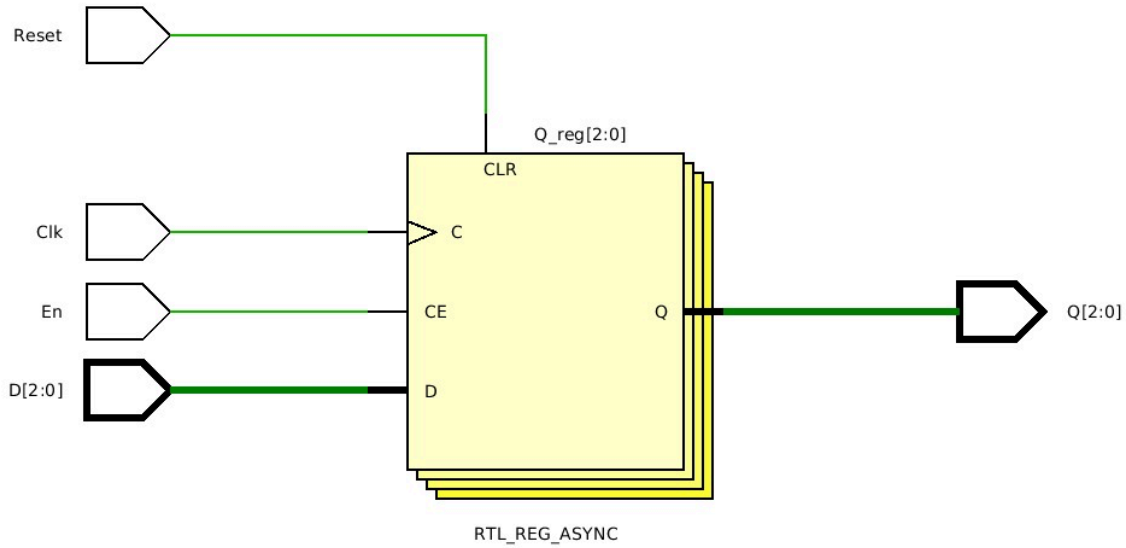
4-bit tri-state Buffer Design Schematic



3-bit Register - Design

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

entity Reg_3_B is
    Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end Reg_3_B;

architecture Behavioral of Reg_3_B is

begin

process (Clk, Reset) begin
    if (Reset = '1') then
        Q <= "000"; -- reset the register asynchronously
    else if (rising_edge(Clk)) then -- respond when clock rises
        if En = '1' then -- Enable should be set
            Q <= D;
        end if;
    end if;
    end if;
end process;
end Behavioral;
```

3-bit Register Design Schematic

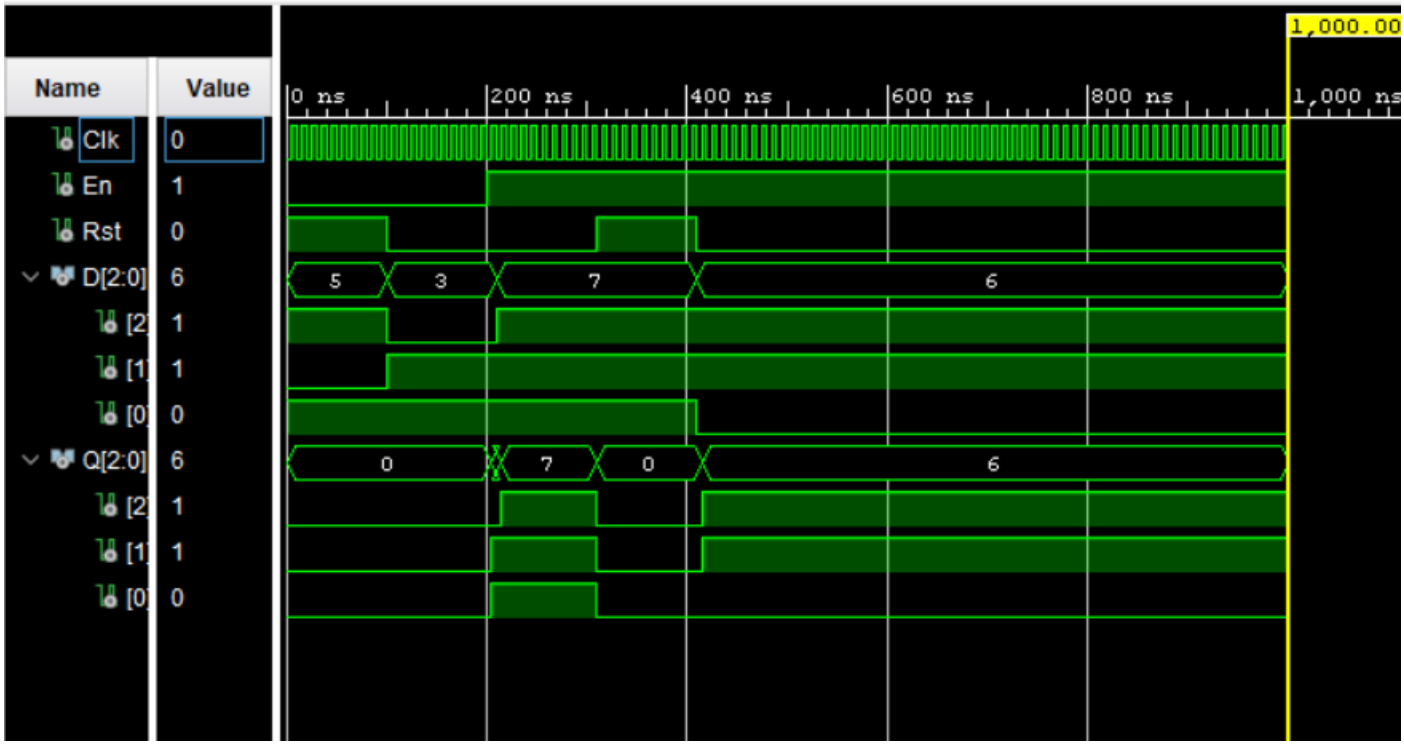


3-bit Register - TB

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

entity TB_Reg_3_B is
--  Port ( );
end TB_Reg_3_B;

architecture Behavioral of TB_Reg_3_B is


component Reg_3_B is
    Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end component;

SIGNAL Clk, En, Rst : STD_LOGIC;
SIGNAL D : STD_LOGIC_VECTOR(2 downto 0);
SIGNAL Q : STD_LOGIC_VECTOR(2 downto 0);

begin

UUT : Reg_3_B PORT MAP (
        Clk => Clk,
        Reset => Rst,
        En => En,
        D => D,
        Q => Q
    );


    clock_process : process -- background clock process
    begin

        Clk <= '0';
        wait for 5ns;
        Clk <= '1';
        wait for 5ns;

    end process;
```

```
    reg_process : process
    begin
        Rst <= '1';
        En <= '0';
        D <= "101";
        wait for 100ns;

        Rst <= '0';
        D <= "011";
        wait for 100ns;

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

        D <= "111";
        wait for 100ns;

        -- reset the register
        Rst <= '1';
        wait for 100ns;
        Rst <= '0';

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

        wait; -- wait forever

    end process;

end Behavioral;
```

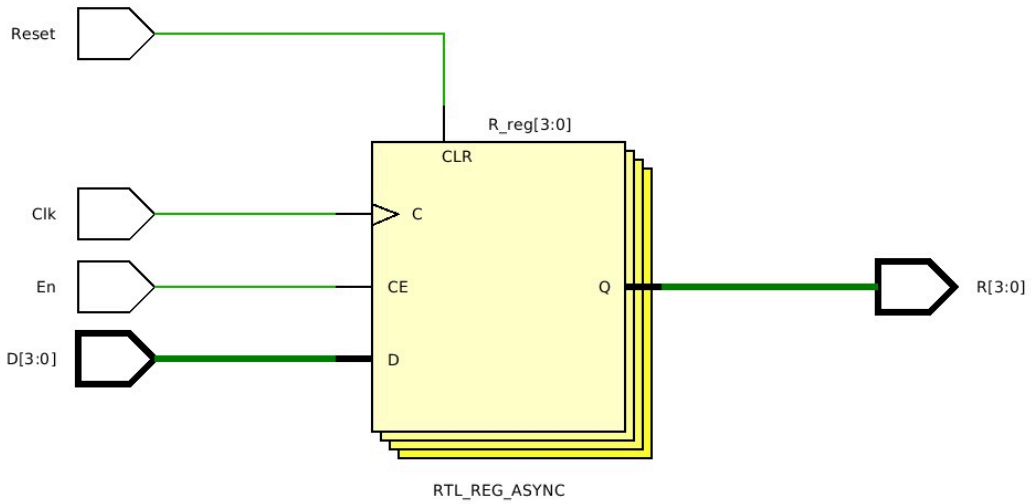
3-bit Register Timing Diagram



4-bit Register

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

entity Reg_4_B is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           R : out STD_LOGIC_VECTOR (3 downto 0));
end Reg_4_B;

architecture Behavioral of Reg_4_B is

begin
process (Clk, Reset) begin
    if (Reset = '1') then
        R <= "0000"; -- reset the register asynchronously
    else if (rising_edge(Clk)) then -- respond when clock rises
        if En = '1' then -- Enable should be set
            R <= D;
        end if;
    end if;
    end if;
end process;
end Behavioral;
```

4-bit Register Design Schematic

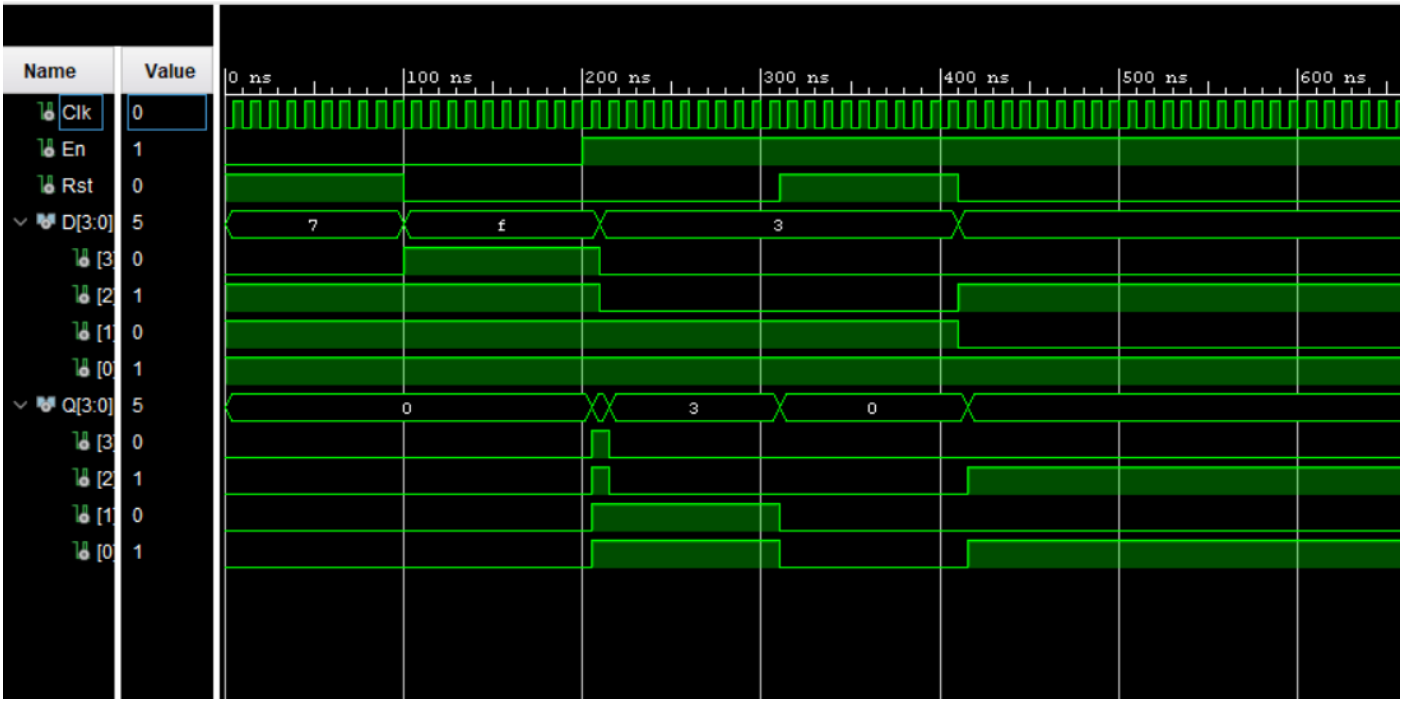


4-bit Register - TB

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

entity TB_Reg_4_B is
--  Port ( );
end TB_Reg_4_B;

architecture Behavioral of TB_Reg_4_B is

component Reg_4_B is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           R : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL Clk, En, Rst : STD_LOGIC;
SIGNAL D : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL Q : STD_LOGIC_VECTOR(3 downto 0);

begin

UUT : Reg_4_B PORT MAP (
        Clk => Clk,
        Reset => Rst,
        En => En,
        D => D,
        R => Q
    );


    clock_process : process -- background clock process
    begin

        Clk <= '0';
        wait for 5ns;
        Clk <= '1';
        wait for 5ns;

    end process;
```

```
    reg_process : process
    begin
        Rst <= '1';
        En <= '0';
        D <= "0111";
        wait for 100ns;

        Rst <= '0';
        D <= "1111";
        wait for 100ns;

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

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

        -- reset the register
        Rst <= '1';
        wait for 100ns;
        Rst <= '0';

        D <= "0101";
        wait for 100ns;

        wait; -- wait forever

    end process;

end Behavioral;
```

4-bit Register Timing Diagram



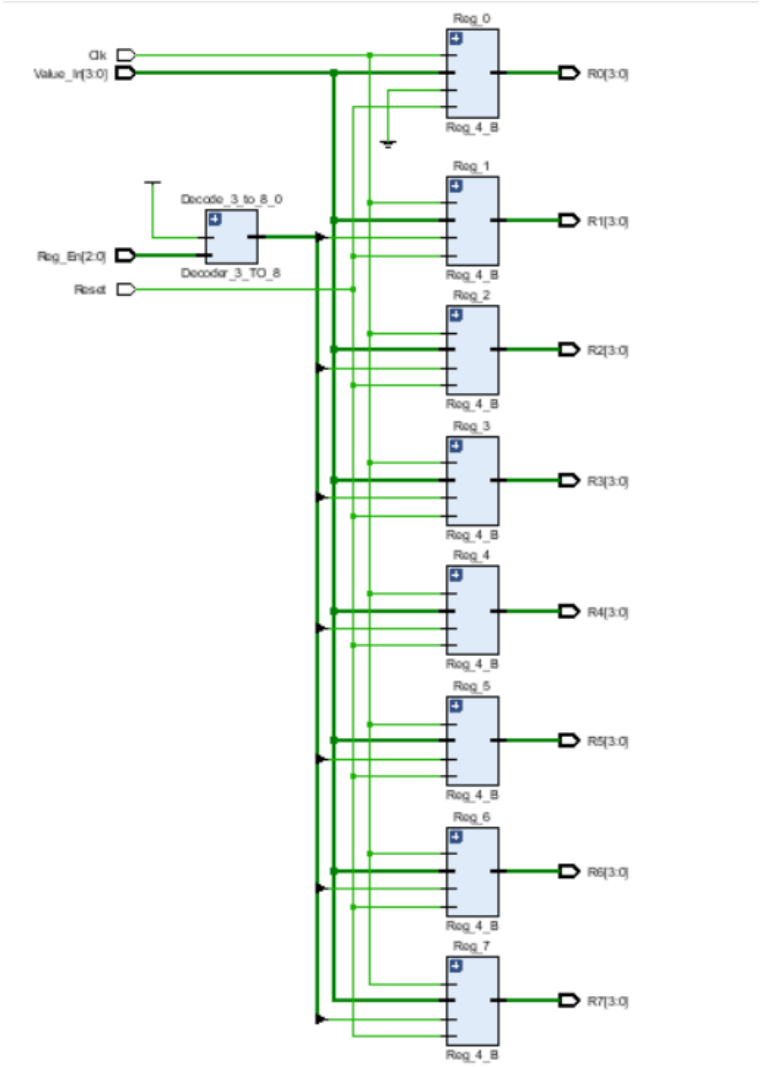
Register Bank - Design

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

entity Register_Bank is
     Port ( Value_In : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC;
           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));
end Register_Bank;

architecture Behavioral of Register_Bank is

component Decoder_3_to_8
 port(
    I: in STD_LOGIC_VECTOR;
    EN: in STD_LOGIC;
    Y: out STD_LOGIC_VECTOR );
 end component;

 component Reg_4_B
 Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
        En : in STD_LOGIC;
        Clk : in STD_LOGIC;
        Reset : in STD_LOGIC;
        R : out STD_LOGIC_VECTOR (3 downto 0) := "0000");
 end component;

 signal reg_en_out : std_logic_vector (7 downto 0);

begin
    Decode_3_to_8_0 : Decoder_3_to_8
        port map(
            I => Reg_En,
            EN => '1',
            Y => reg_en_out );
```

```
Reg_0 : Reg_4_B  -- Read only register with a initial value of 0000
     Port map (
         D => Value_In,
         EN => '0',
         Reset => Reset,
         Clk => Clk,
         R => R0);

 Reg_1 : Reg_4_B
    Port map (
         D => Value_In,
         EN => reg_en_out(1),
         Reset => Reset,
         Clk => Clk,
         R => R1);

  Reg_2 : Reg_4_B
     Port map (
         D => Value_In,
         EN => reg_en_out(2),
         Reset => Reset,
         Clk => Clk,
         R => R2);

  Reg_3 : Reg_4_B
     Port map (
         D => Value_In,
         EN => reg_en_out(3),
         Reset => Reset,
         Clk => Clk,
         R => R3);

  Reg_4 : Reg_4_B
     Port map (
         D => Value_In,
         EN => reg_en_out(4),
         Reset => Reset,
         Clk => Clk,
         R => R4);

 Reg_5 : Reg_4_B
     Port map (
         D => Value_In,
         EN => reg_en_out(5),
         Reset => Reset,
         Clk => Clk,
         R => R5);
```

```
    Reg_6 : Reg_4_B
        Port map (
            D => Value_In,
            EN => reg_en_out(6),
            Reset => Reset,
            Clk => Clk,
            R => R6);

    Reg_7 : Reg_4_B
        Port map (
            D => Value_In,
            EN => reg_en_out(7),
            Reset => Reset,
            Clk => Clk,
            R => R7);

end Behavioral;
```

Register Bank Design Schematic

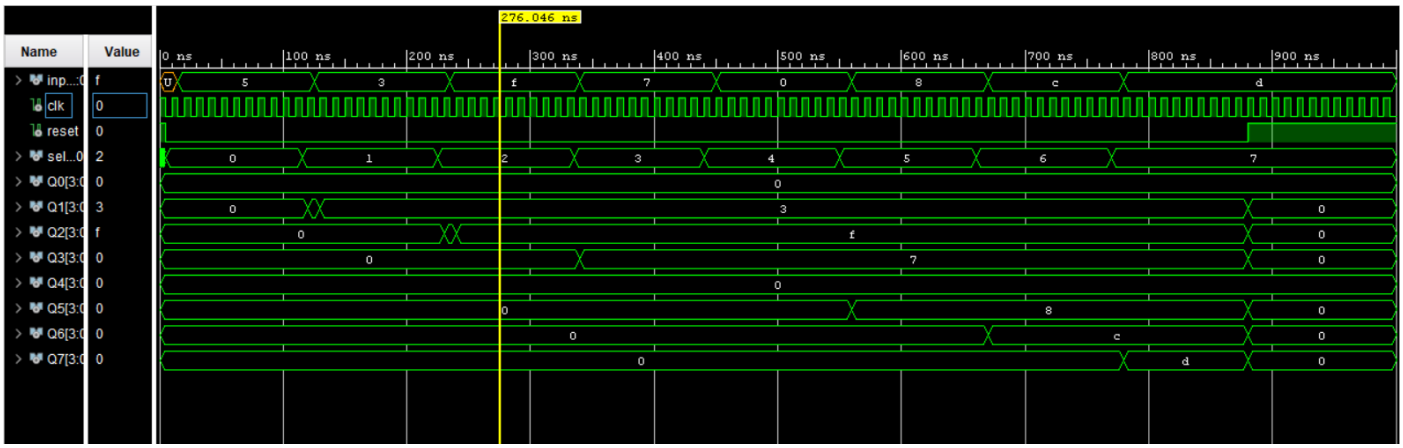


Register Bank - TB

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

architecture Behavioral of TB_Reg_Bank is


component Register_Bank
    Port ( Value_In : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC;
           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));
END COMPONENT;

signal input : STD_LOGIC_VECTOR(3 downto 0);
signal clk,reset : STD_LOGIC := '0';
signal selecter : STD_LOGIC_VECTOR(2 downto 0);
signal Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7 : STD_LOGIC_VECTOR(3 downto 0);

begin

UUT: Register_Bank
    PORT MAP (
       Value_In=> input,
       Clk => clk,
       Reg_En => selecter,
       Reset => reset,
       R0 => Q0,
       R1 => Q1,
       R2 => Q2,
       R3 => Q3,
       R4 => Q4,
       R5 => Q5,
       R6 => Q6,
       R7 => Q7 );

  process
  begin
      Clk <= NOT(Clk);
      wait for 5 ns;

  end process;
```

```
process
  begin
      reset <= '1';
      wait for 5 ns;
       reset <= '0';

      selecter <= "000";
      wait for 10 ns;
      input <= "0101";    --5

      wait for 100 ns;
      selecter <= "001";
      wait for 10 ns;
      input <= "0011";    --3

      wait for 100 ns;
      selecter <= "010";
      wait for 10 ns;
      input <= "1111";    --F

      wait for 100 ns;
      selecter <= "011";
      wait for 5 ns;
      input <= "0111";    --7

      wait for 100 ns;
      selecter <= "100";
      wait for 10 ns;
      input <= "0000";    --0

      wait for 100 ns;
      selecter <= "101";
      wait for 10 ns;
      input <= "1000";    --8

      wait for 100 ns;
      selecter <= "110";
      wait for 10 ns;
      input <= "1100";    --C

      wait for 100 ns;
      selecter <= "111";
      wait for 10 ns;
      input <= "1101";    --D

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

  end process;

end Behavioral;
```

Register Bank Timing Diagram

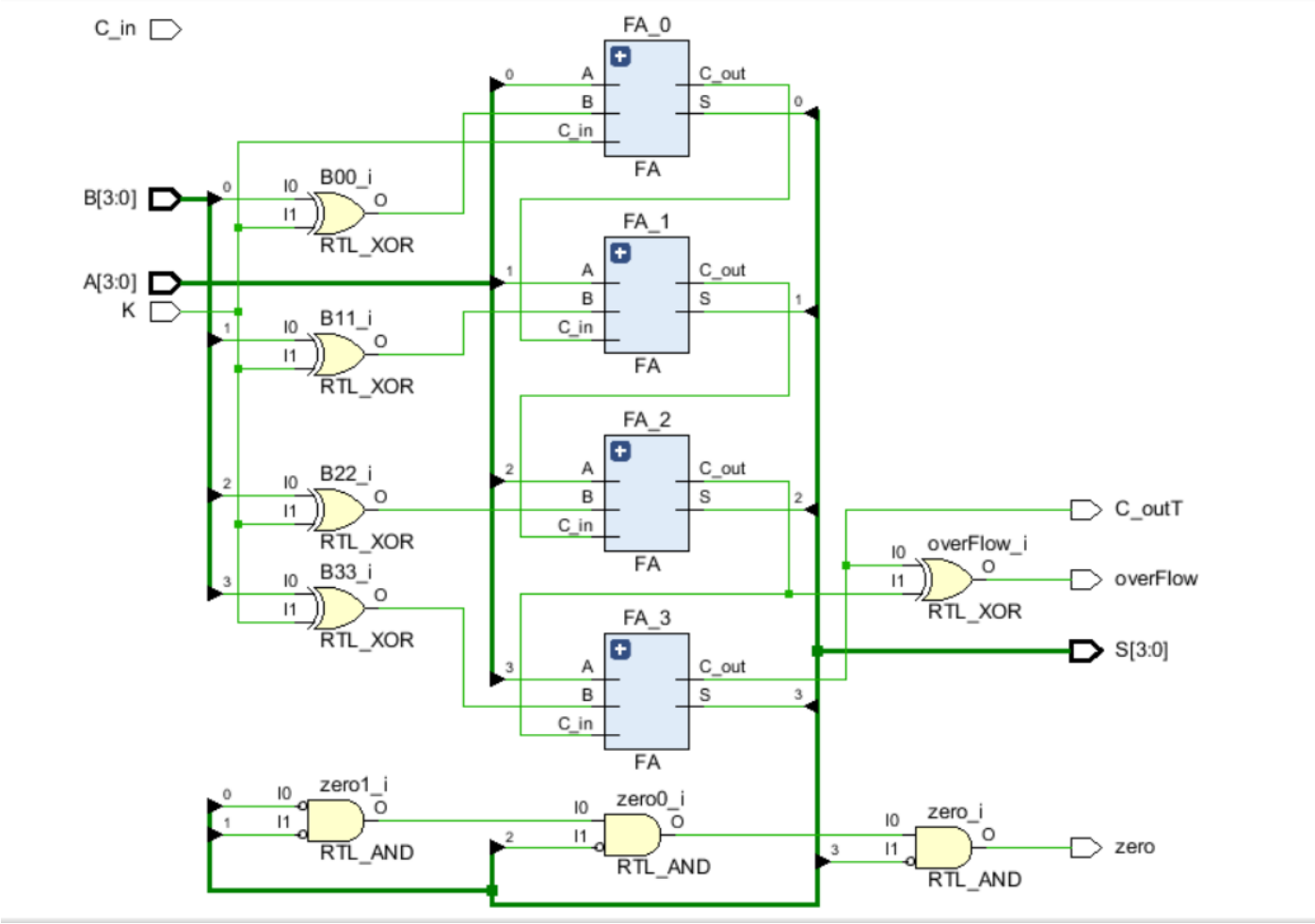


4-bit Add/Subtract Unit - Design

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

entity Add_Sub_Unit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C_in : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           C_outT : out STD_LOGIC;
           overFlow:out Std_logic;
           zero:out STD_LOGIC;
           K : in STD_LOGIC);
end Add_Sub_Unit;

architecture Behavioral of Add_Sub_Unit is

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

   SIGNAL FA0_C, FA1_C, FA2_C, FA3_C  : std_logic;
   SIGNAL SO : std_logic_vector (3 downto 0);
   SIGNAL carry:std_logic;
   SIGNAL B00:std_logic:=(B(0) XOR K);
   SIGNAL B11:std_logic:=(B(1)XOR K);
   SIGNAL B22:std_logic:=(B(2) XOR K);
   SIGNAL B33:std_logic:=(B(3) XOR K);
```

```
begin
    B00 <= (B(0) XOR K);
    B11 <= (B(1) XOR K);
    B22 <= (B(2) XOR K);
    B33 <= (B(3) XOR K);

  FA_0 : FA
   port map (
       A => A(0),
       B => B00,
       C_in => k, -- Set to ground
       S => SO(0),
       C_Out => FA0_C);

  FA_1 : FA
    port map (
       A => A(1),
       B => B11,
       C_in => FA0_C,
       S => SO(1),
       C_Out => FA1_C);

   FA_2 : FA
   port map (
       A => A(2),
       B => B22,
       C_in => FA1_C,
       S => SO(2),
       C_Out => FA2_C);

   FA_3 : FA
   port map (
       A => A(3),
       B => B33,
       C_in => FA2_C,
       S => SO(3),
       C_Out => carry);

    overFlow <= carry XOR FA2_C;
    C_outT <= carry;
    S <= SO;

    Zero <= (NOT SO(0)) AND (NOT SO(1)) AND (NOT SO(2)) AND (NOT SO(3));

end Behavioral;
```

4-bit Add/Subtract Unit Design Schematic

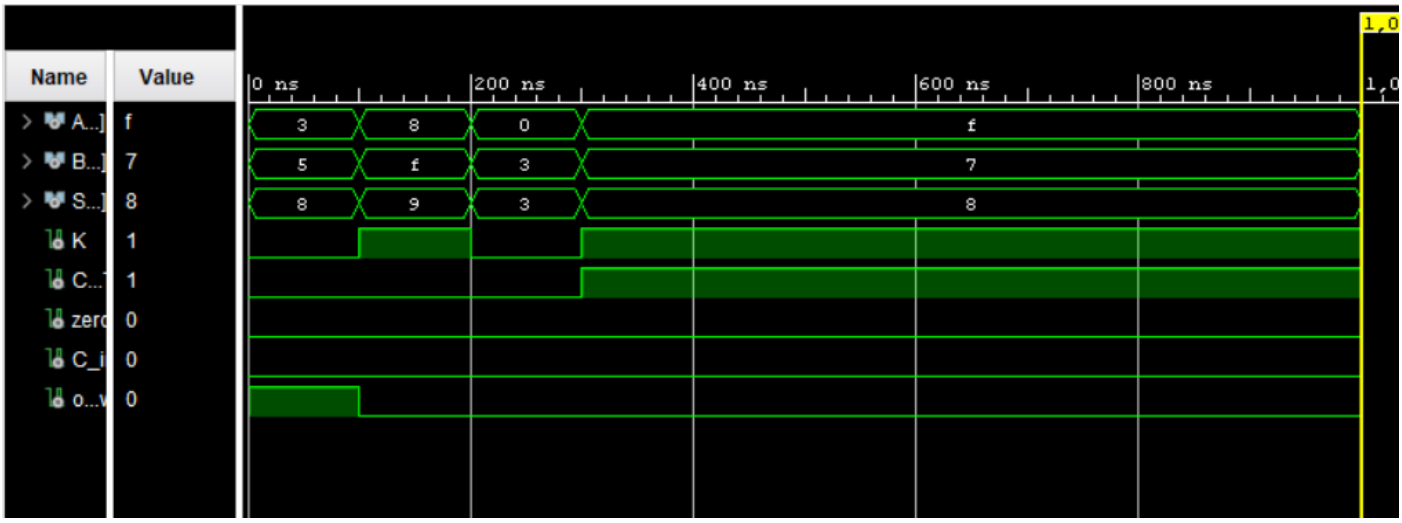


4-bit Add/Subtract Unit - TB

```
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_Unit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C_in : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           C_outT : out STD_LOGIC;
           overFlow:out Std_logic;
           zero:out STD_LOGIC;
           K : in STD_LOGIC);
end component;
```

```
signal A,B,S: STD_LOGIC_VECTOR(3 DOWNTO 0);
signal K:STD_LOGIC;
signal C_outT:std_logic;
signal zero,C_in,overFlow:std_logic;

begin

UUT:Add_Sub_Unit PORT MAP(
    A => A,
    B => B,
    C_in => C_in,
    S => S,
    C_outT => C_outT,
    overFlow => overflow,
    zero=> zero,
    K => K);
process
    begin
        C_in <= '0';
     --add 5 to 3
        K <= '0';
        A <= "0011";
        B <= "0101";
        wait for 100 ns;

    --subtract 15 from 8
        K <= '1';
        A <= "1000";
        B <= "1111";
        wait for 100 ns;

    --add 3 to 0
        K <= '0';
        A <= "0000";
        B <= "0011";
        wait for 100 ns;

    --subtract 7 from 15
        K <= '1';
        A <= "1111";
        B <= "0111";
        wait;
    end process;
end Behavioral;
```

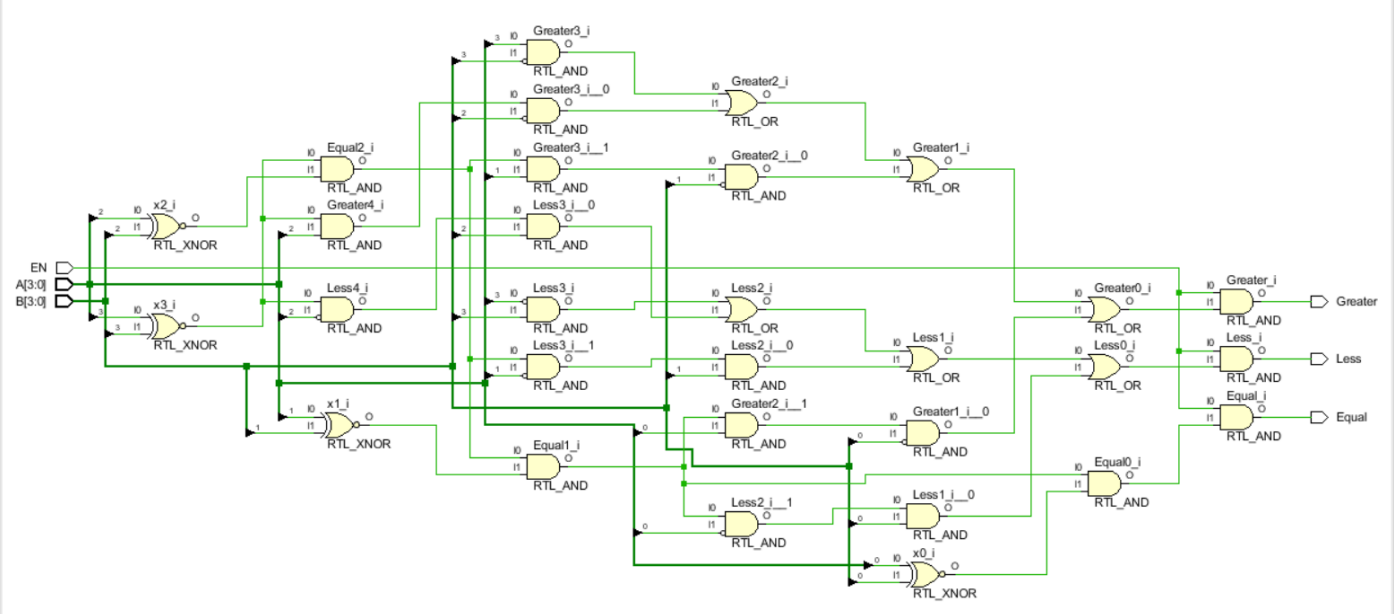
4-bit Add/Subtract Unit Timing Diagram



4-bit Comparator - Design

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

entity Comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           Equal : out STD_LOGIC;
           Greater : out STD_LOGIC;
           Less : out STD_LOGIC);
end Comparator;

architecture Behavioral of Comparator is
    signal x0: std_logic:=A(0) xnor B(0);
    signal x1: std_logic:=A(1) xnor B(1);
    signal x2: std_logic:=A(2) xnor B(2);
    signal x3: std_logic:=A(3) xnor B(3);

begin
    x0 <= A(0) xnor B(0);
    x1 <=A(1) xnor B(1);
    x2 <=A(2) xnor B(2);
    x3 <=A(3) xnor B(3);

    Equal<= EN AND (x3 and x2 and x1 and x0);
    Greater<= EN AND ((A(3)AND (NOT B(3)))  OR (x3 and A(2) AND (NOT B(2)))
OR (X3 AND X2 AND A(1)  AND (NOT B(1)))OR (X3 AND X2 AND X1 AND A(0)
AND(NOT(B(0))))) ;
    Less<= EN AND ((not(A(3))AND B(3))  OR (x3 and not(A(2)) AND B(2)) OR
(X3 AND X2 AND not(A(1))  AND B(1))OR (X3 AND X2 AND X1 AND not( A(0)) AND
B(0))) ;

end Behavioral;
```

4-bit Comparator Design Schematic

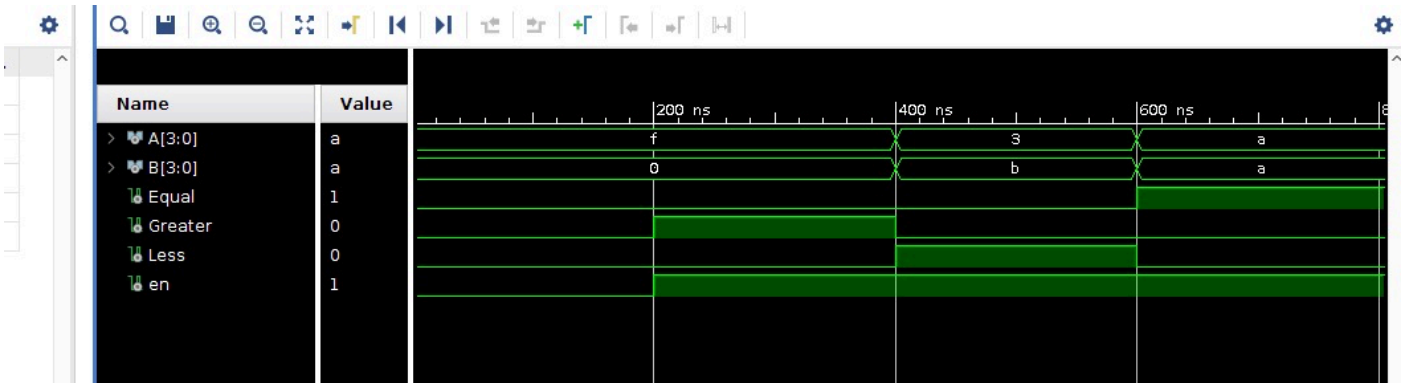


4-bit Comparator - TB

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

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

architecture Behavioral of TB_Comparator is


component Comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
       B : in STD_LOGIC_VECTOR (3 downto 0);
       EN : in STD_LOGIC;
       Equal : out STD_LOGIC;
       Greater : out STD_LOGIC;
       Less : out STD_LOGIC);
end component;

signal A,B:STD_LOGIC_VECTOR(3 downto 0);
signal Equal,Greater,Less, en : STD_LOGIC;
```

```
begin
UUT:Comparator
        PORT MAP(
              A => A,
              B => B,
              en => en,
              Equal => Equal,
              Greater => Greater,
              Less => Less);

    PROCESS
        BEGIN
        EN <= '0';
        A<="1111";
        B<="0000";
        wait for 200ns;

        En <= '1';
        wait for 200ns;

        A<="0011";
        B<="1011";
        wait for 200ns;


        A<="1010";
        B<="1010";
        wait for 200 ns;
        WAIT;
    end process;

end Behavioral;
```

4-bit Comparator Timing Diagram

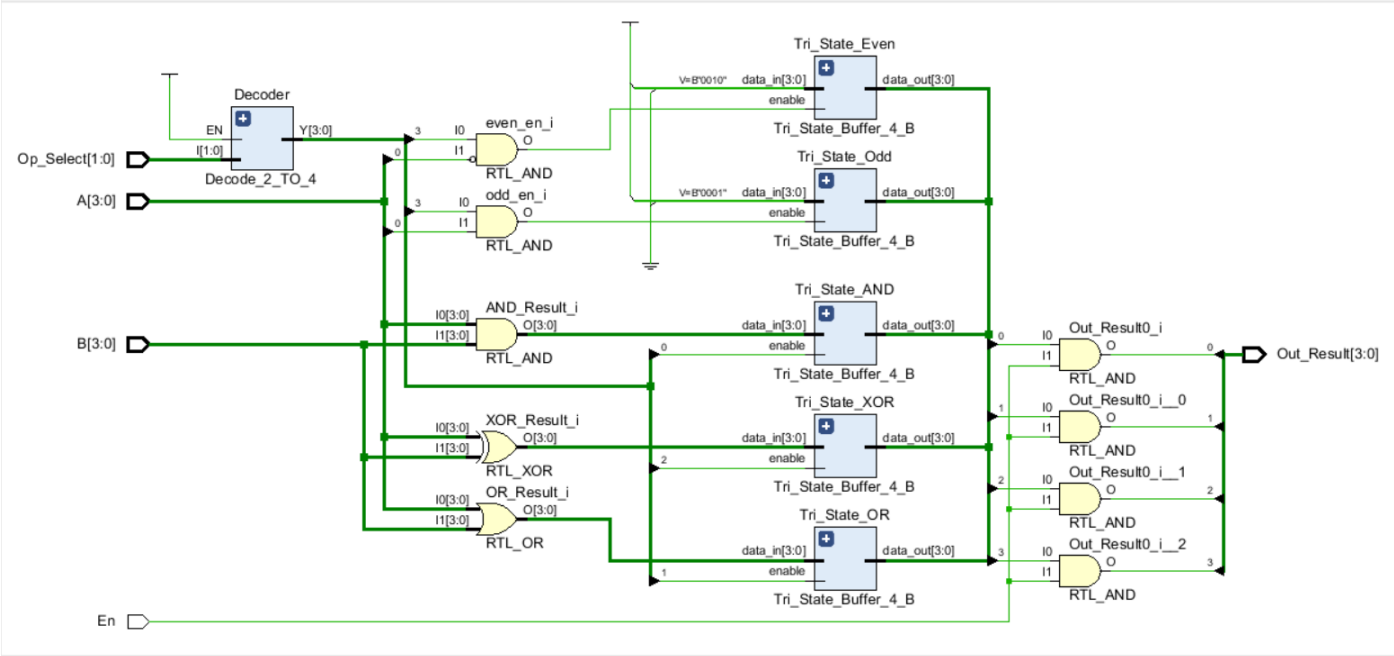


Logical Operations Unit - Design

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

entity Logical_Unit is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B : in STD_LOGIC_VECTOR (3 downto 0);
        En : in STD_LOGIC;
        Op_Select : in STD_LOGIC_VECTOR (1 downto 0);
        Out_Result : out STD_LOGIC_VECTOR (3 downto 0)
    );
end Logical_Unit;

architecture Behavioral of Logical_Unit is

    component Decode_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;

    component Tri_State_Buffer_4_B
        port (
            data_in : in STD_LOGIC_VECTOR (3 downto 0);
            enable : in STD_LOGIC;
            data_out: out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    signal AND_Result : STD_LOGIC_VECTOR(3 downto 0);
    signal OR_Result : STD_LOGIC_VECTOR(3 downto 0);
    signal XOR_Result : STD_LOGIC_VECTOR(3 downto 0);
    signal OddEvenCheck_Result : STD_LOGIC_VECTOR(3 downto 0);

    signal Decoder_Out : STD_LOGIC_VECTOR(3 downto 0);
    signal Final_Out : STD_LOGIC_VECTOR(3 downto 0);

    signal odd_en, even_en : STD_LOGIC;
```

```
    begin

    Decoder: Decode_2_to_4
        port map ( I => Op_Select,
                   En => '1',
                   Y => Decoder_Out);

    AND_Result <= A and B;
    OR_Result <= A or B;
    XOR_Result <= A xor B;

    odd_en <= Decoder_Out(3) AND A(0);
    even_en <= Decoder_Out(3) AND NOT A(0);


    Tri_State_AND: Tri_State_Buffer_4_B
        port map (data_in => AND_Result,
                  enable => Decoder_Out(0),
                  data_out => Final_Out);

    Tri_State_OR: Tri_State_Buffer_4_B
        port map (data_in => OR_Result,
                  enable => Decoder_Out(1),
                  data_out => Final_Out);

    Tri_State_XOR: Tri_State_Buffer_4_B
        port map (data_in => XOR_Result,
                  enable => Decoder_Out(2),
                  data_out => Final_Out);

    Tri_State_Odd: Tri_State_Buffer_4_B
        port map (data_in => "0001",   -- output 1 if odd and 2 if even
                  enable => odd_en,
                  data_out => Final_Out);

    Tri_State_Even : Tri_State_Buffer_4_B
        port map (data_in => "0010",
                  enable => even_en,
                  data_out => Final_Out);

    Out_Result(0) <= Final_Out(0) AND EN;
    Out_Result(1) <= Final_Out(1) AND EN;
    Out_Result(2) <= Final_Out(2) AND EN;
    Out_Result(3) <= Final_Out(3) AND EN;

end Behavioral;
```

Logic Operations Unit Design Schematic

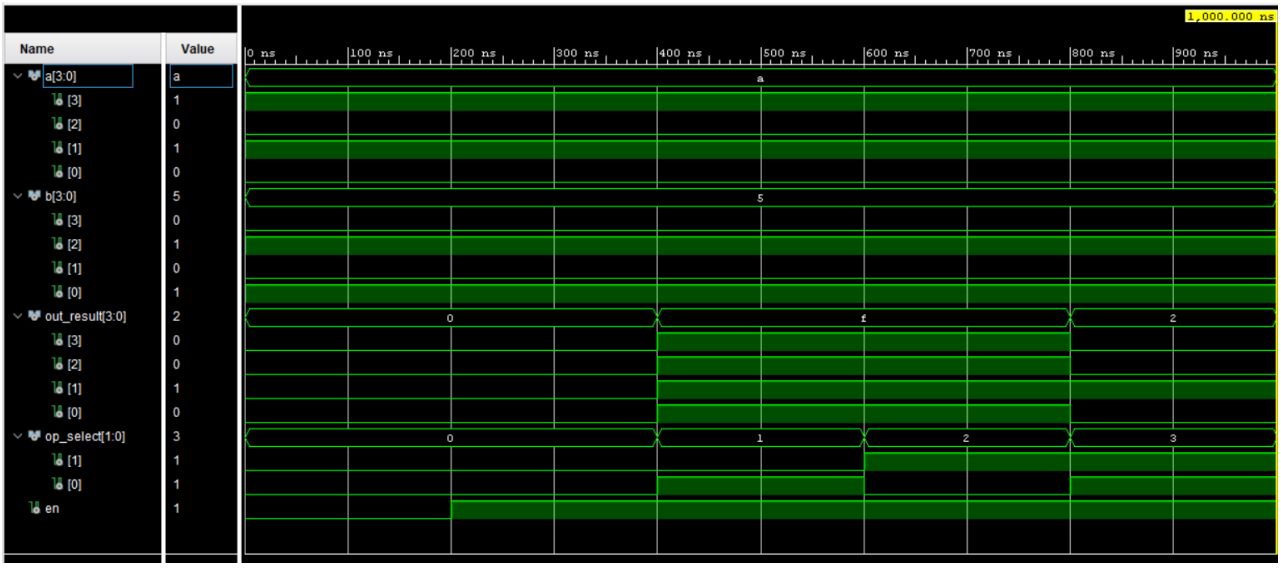


Logical Operations Unit - TB

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

architecture Behavioral of TB_Logical_Unit is

Component Logical_Unit is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B : in STD_LOGIC_VECTOR (3 downto 0);
        EN : in STD_LOGIC;
        Op_Select : in STD_LOGIC_VECTOR (1 downto 0);
        Out_Result : out STD_LOGIC_VECTOR (3 downto 0)
    );
end component;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_Logical_Unit is
--  Port ( );
end TB_Logical_Unit;
```

```
begin
    UTT : Logical_Unit
        port map (
            A => a,
            B => b,
            EN => en,
            Op_Select => op_select,
            out_result => out_result);


    Process
        begin
            a <= "1010";
            b <= "0101";
            en <= '0';
            op_select <= "00";
            wait for 200ns;

            en <= '1';
            wait for 200ns;

            op_select <= "01";
            wait for 200ns;

            op_select <= "10";
            wait for 200ns;

            op_select <= "11";
            wait;
        end process;

end Behavioral;
```

Logic Operations Unit Timing Diagram



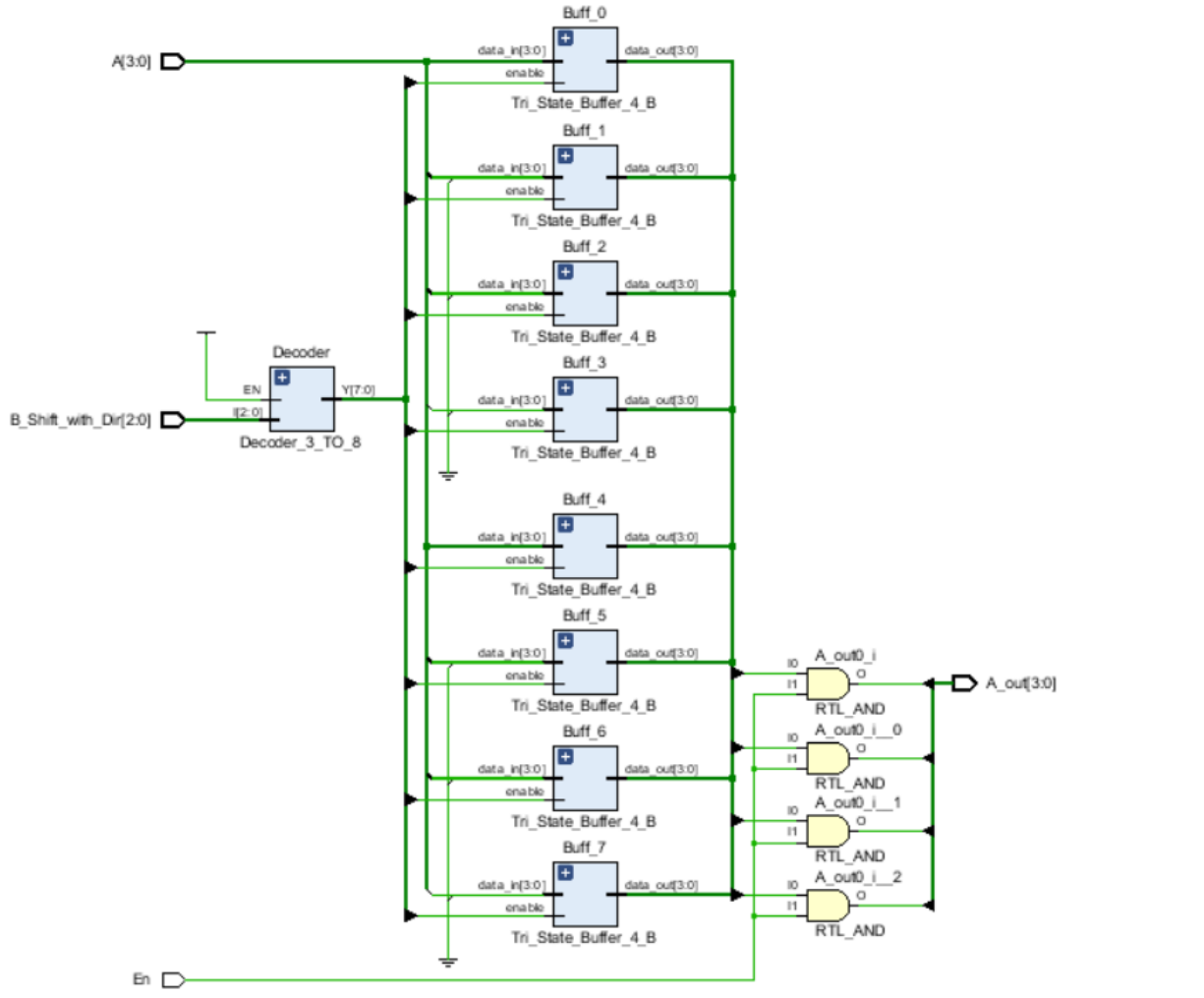
Bit Shift - Design

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

entity Bit_Shift is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B_Shift_with_Dir : in STD_LOGIC_VECTOR (2 downto 0); -- MSB shows
direction to shift and next 2 shows no of digits
        En : in STD_LOGIC;
        A_out : out STD_LOGIC_VECTOR (3 downto 0));
end Bit_Shift;

architecture Behavioral of Bit_Shift 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 Tri_State_Buffer_4_B
    port (
        data_in : in STD_LOGIC_VECTOR (3 downto 0);
        enable : in STD_LOGIC;
        data_out: out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal Decoder_Out : STD_LOGIC_VECTOR (7 downto 0);
signal Lshift_0, Lshift_1, Lshift_2, Lshift_3 :  STD_LOGIC_VECTOR (3 downto
0);
signal Rshift_0, Rshift_1, Rshift_2, Rshift_3 :  STD_LOGIC_VECTOR (3 downto
0);
signal Final_Out : STD_LOGIC_VECTOR (3 downto 0);

begin

Decoder: Decoder_3_to_8
    port map ( I => B_Shift_with_Dir,
               En => '1',
               Y => Decoder_Out);

Lshift_0 <= A;

Lshift_1(0) <= '0';
Lshift_1(1) <= A(0);
Lshift_1(2) <= A(1);
Lshift_1(3) <= A(2);
```

```
Lshift_2(0) <= '0';
Lshift_2(1) <= '0';
Lshift_2(2) <= A(0);
Lshift_2(3) <= A(1);

Lshift_3(0) <= '0';
Lshift_3(1) <= '0';
Lshift_3(2) <= '0';
Lshift_3(3) <= A(0);

Rshift_0 <= A;

Rshift_1(0) <= A(1);
Rshift_1(1) <= A(2);
Rshift_1(2) <= A(3);
Rshift_1(3) <= '0';

Rshift_2(0) <= A(2);
Rshift_2(1) <= A(3);
Rshift_2(2) <= '0';
Rshift_2(3) <= '0';

Rshift_3(0) <= A(3);
Rshift_3(1) <= '0';
Rshift_3(2) <= '0';
Rshift_3(3) <= '0';

Buff_0 : Tri_State_Buffer_4_B
    port map (
        data_in => Lshift_0,
        enable => Decoder_Out(0),
        data_out => Final_Out);

 Buff_1 : Tri_State_Buffer_4_B
    port map (
        data_in => Lshift_1,
        enable => Decoder_Out(1),
        data_out => Final_Out);

 Buff_2 : Tri_State_Buffer_4_B
    port map (
        data_in => Lshift_2,
        enable => Decoder_Out(2),
        data_out => Final_Out);


 Buff_3 : Tri_State_Buffer_4_B
    port map (
        data_in => Lshift_3,
        enable => Decoder_Out(3),
        data_out => Final_Out);
```

```
Buff_4 : Tri_State_Buffer_4_B
    port map (
        data_in => Rshift_0,
        enable => Decoder_Out(4),
        data_out => Final_Out);

 Buff_5 : Tri_State_Buffer_4_B
    port map (
        data_in => Rshift_1,
        enable => Decoder_Out(5),
        data_out => Final_Out);

 Buff_6 : Tri_State_Buffer_4_B
    port map (
        data_in => Rshift_2,
        enable => Decoder_Out(6),
        data_out => Final_Out);

 Buff_7 : Tri_State_Buffer_4_B
    port map (
        data_in => Rshift_3,
        enable => Decoder_Out(7),
        data_out => Final_Out);

A_out(0) <= Final_Out(0) AND EN;
A_out(1) <= Final_Out(1) AND EN;
A_out(2) <= Final_Out(2) AND EN;
A_out(3) <= Final_Out(3) AND EN;

end Behavioral;
```

Bit Shift Design Schematic

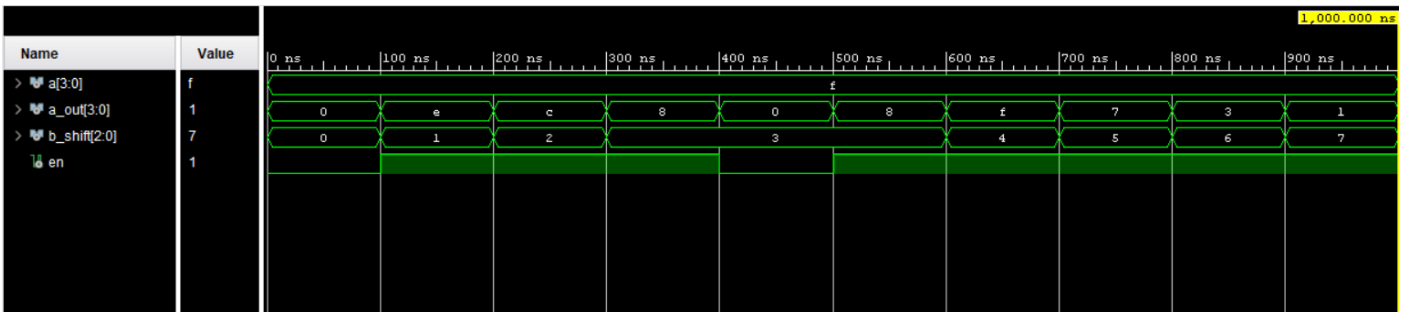


Bit Shift - TB

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

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

architecture Behavioral of TB_Bit_Shift is
component Bit_Shift is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B_Shift_with_Dir : in STD_LOGIC_VECTOR (2 downto 0);
        En : in STD_LOGIC;
        A_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal a, a_out : STD_LOGIC_VECTOR (3 downto 0);
signal b_shift : STD_LOGIC_VECTOR (2 downto 0);
signal en : STD_LOGIC;
```

```
begin

UTT : Bit_Shift
    port map (
        A => a,
        B_Shift_with_Dir => b_shift,
        En => en,
        A_out => a_out);

PROCESS
    begin
        a <= "1111";

        en <= '0';

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

        en <= '1';
        b_shift <= "001";
        wait for 100ns;

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

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

        en <= '0';
        waitfor 100ns;

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

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

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

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

        b_shift <= "111";
        wait for 100ns;

     wait; -- wait forever

    end process;

end Behavioral;
```

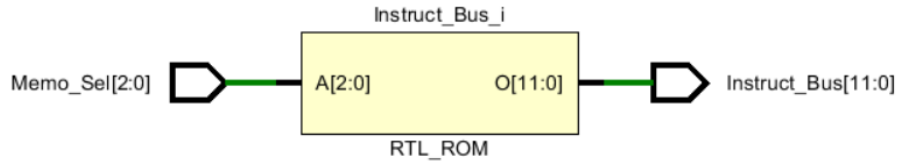
Bit Shift Timing Diagram



Program Rom - Design

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

entity Program_Rom is
    Port ( Memo_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Instruct_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(11 downto 0);
    signal program_ROM : rom_type := (

        "0100010000110",
        "0100100001010",
        "0100110000011",
        "1000010100000",

        "1010000010001",
        "1010100000011",

        "1100110000001",
        "0000000000000"
        );

begin
    Instruct_Bus <= program_ROM(to_integer(unsigned(Memo_Sel)));

end Behavioral;
```

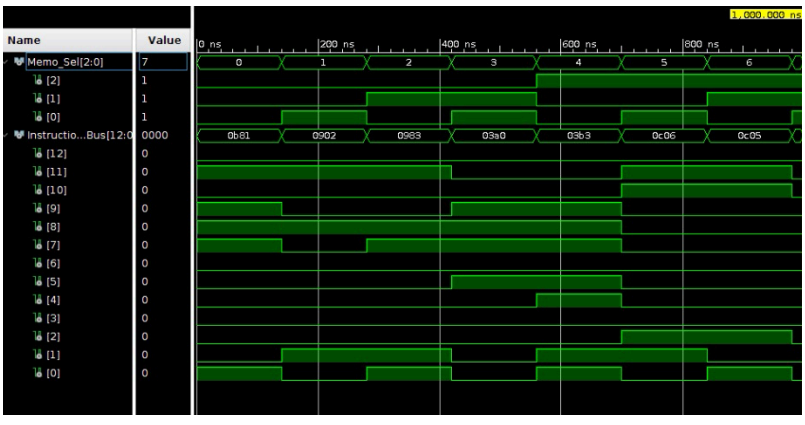
Program Rom Design Schematic



Program Rom - TB

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

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

architecture Behavioral of TB_Programm_Rom is

component Program_Rom is
    Port ( Memo_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Instruction_Bus : out STD_LOGIC_VECTOR (12 downto 0));
end component;

signal Memo_Sel : STD_LOGIC_VECTOR (2 downto 0);
signal Instruction_Bus : STD_LOGIC_VECTOR (12 downto 0);

begin

UTT : Program_Rom
    port map (
        Memo_sel => memo_sel,
        Instruction_Bus => Instruction_bus);

    rom_process : process
        begin
            memo_sel <= "000";
            wait for 100ns;

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

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

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

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

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

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

            memo_sel <= "111";
            wait;
    end process;

end Behavioral;
```

Program Rom Timing Diagram

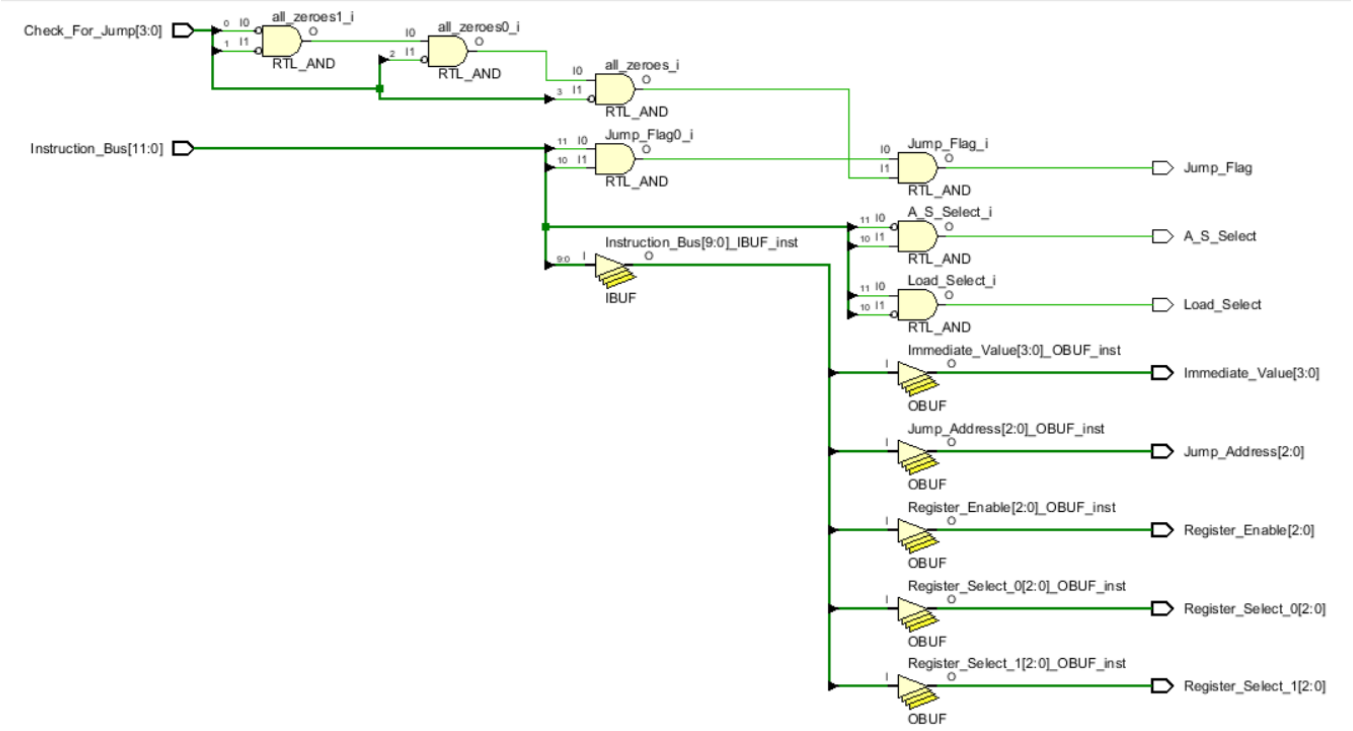


Instruction Decoder - Design

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

entity Instruction_Decoder is
      Port ( Instruction_Bus : in STD_LOGIC_VECTOR (12 downto 0);
             Check_For_Jump : in STD_LOGIC_VECTOR (3 downto 0);
             Register_Enable : out STD_LOGIC_VECTOR (2 downto 0);
             Load_Select : out STD_LOGIC;
             Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
             Register_Select_0 : out STD_LOGIC_VECTOR (2 downto 0);
             Register_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
             A_S_Select : out STD_LOGIC;
             Jump_Flag : out STD_LOGIC;
             Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);
             Comparator_En : out STD_LOGIC;
             Logical_unit_en : out STD_LOGIC;
             Logical_Operation_Select : out STD_LOGIC_VECTOR(1 Downto 0);
             Bit_Shift_En : out STD_LOGIC;
             Bit_Shift_with_Dir : out STD_LOGIC_VECTOR (2 downto 0));
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is

signal all_zeroes : STD_LOGIC;

begin

     all_zeroes <= NOT(Check_For_Jump(0)) AND NOT(Check_For_Jump(1)) AND
NOT(Check_For_Jump(2)) AND NOT(Check_For_Jump(3));

    -- Register_Enable <= Instruction_bus (9 downto 7);

     Register_Enable(0) <= Instruction_bus (7) AND NOT Instruction_bus(12);
     Register_Enable(1) <= Instruction_bus (8) AND NOT Instruction_bus(12);
     Register_Enable(2) <= Instruction_bus (9) AND NOT Instruction_bus(12);

     Load_Select <= NOT Instruction_bus(12) AND Instruction_bus(11) AND
NOT(Instruction_bus(10));

     Immediate_Value <= Instruction_bus(3 downto 0);

     Register_Select_0 <= Instruction_bus(9 downto 7);

     Register_Select_1 <= Instruction_bus(6 downto 4);

     A_S_Select <= NOT Instruction_bus(12) AND NOT(Instruction_bus(11)) AND
Instruction_bus(10);
```

```
     Jump_Flag <= NOT Instruction_bus(12) AND Instruction_bus(11) AND
Instruction_bus(10) AND all_zeroes;

     Jump_Address <= Instruction_bus(2 downto 0);

     Comparator_En <= Instruction_bus(12) AND NOT Instruction_bus(11) AND
NOT Instruction_bus(10);

     Logical_unit_en <= Instruction_bus(12) AND NOT Instruction_bus(11) AND
Instruction_bus(10);

     Logical_Operation_Select <= Instruction_bus(1 downto 0);

     Bit_Shift_En <= Instruction_bus(12) AND Instruction_bus(11) AND NOT
Instruction_bus(10);

     Bit_Shift_with_Dir <= Instruction_bus(2 downto 0);
end Behavioral;
```

Instruction Decoder Design Schematic

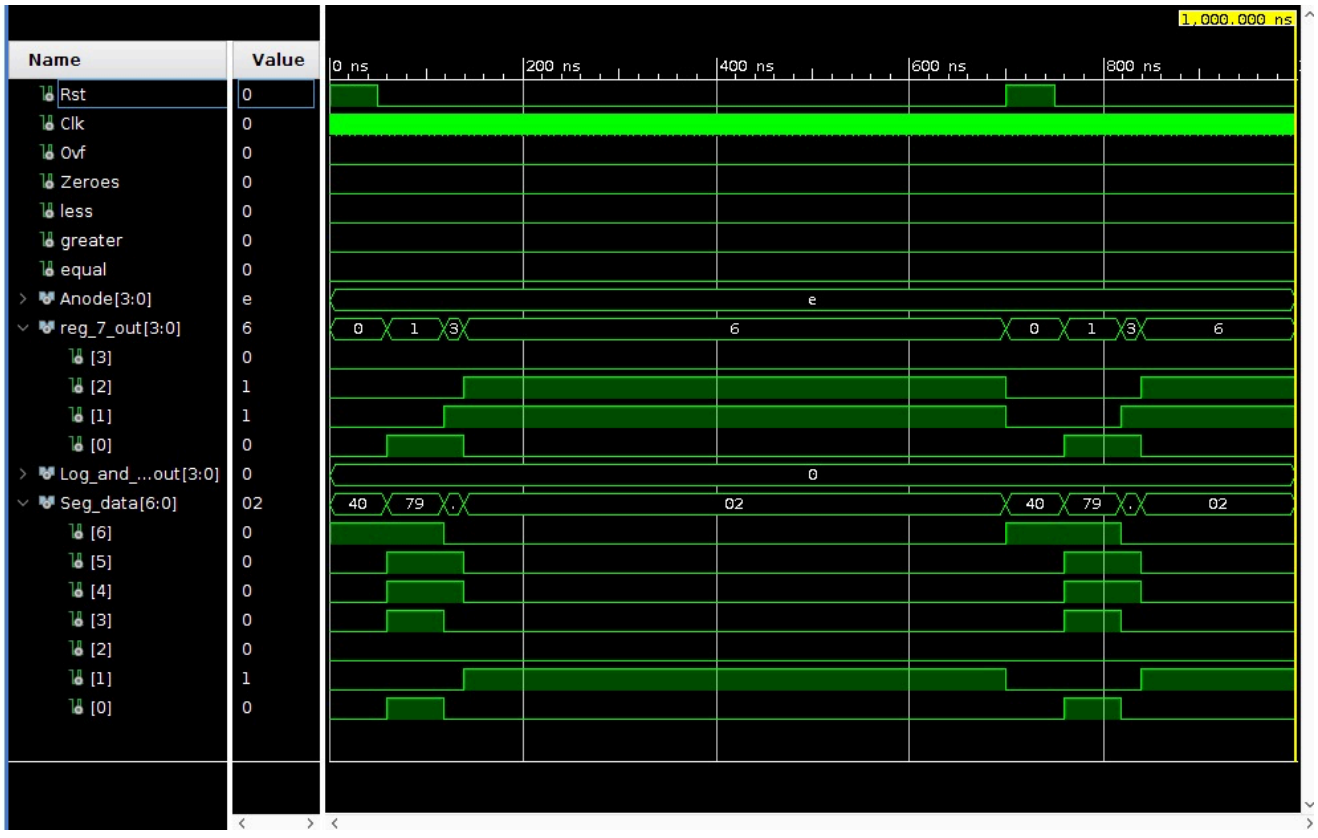


Instruction Decoder - TB

```
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 is
      Port ( Instruction_Bus : in STD_LOGIC_VECTOR (12 downto 0);
             Check_For_Jump : in STD_LOGIC_VECTOR (3 downto 0);
             Register_Enable : out STD_LOGIC_VECTOR (2 downto 0);
             Load_Select : out STD_LOGIC;
             Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
             Register_Select_0 : out STD_LOGIC_VECTOR (2 downto 0);
             Register_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
             A_S_Select : out STD_LOGIC;
             Jump_Flag : out STD_LOGIC;
             Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);
             Comparator_En : out STD_LOGIC;
             Logical_unit_en : out STD_LOGIC;
              Logical_Operation_Select : out STD_LOGIC_VECTOR(1 Downto 0);
              Bit_Shift_En : out STD_LOGIC;
              Bit_Shift_with_Dir : out STD_LOGIC_VECTOR (2 downto 0));
end component;

    SIGNAL ins_bus : STD_LOGIC_VECTOR(12 downto 0);
    SIGNAL jmp_check, im_val : STD_LOGIC_VECTOR(3 downto 0);
    SIGNAL reg_enb, reg_sel_0, reg_sel_1, jmp_addr, B_Shift_with_Dir:
STD_LOGIC_VECTOR(2 downto 0);
    SIGNAL load_sel, add_sub_sel, jmp, Comparator_En, Bit_Shift_En,
Logical_unit_en : STD_LOGIC;
    SIGNAL Logical_Operation_Select :  STD_LOGIC_VECTOR (1 downto 0);

begin
    UUT : Instruction_decoder
    PORT MAP (
        Instruction_Bus => ins_bus,
        Check_For_Jump => jmp_check,
        Register_Enable => reg_enb,
        Load_Select => load_sel,
        Immediate_Value => im_val,
        Register_Select_0 => reg_sel_0,
        Register_Select_1 => reg_sel_1,
        A_S_Select => add_sub_sel,
        Jump_Flag => jmp,
        Jump_Address => jmp_addr,
        Comparator_En => Comparator_En,
        Logical_unit_en => Logical_unit_en,
        Logical_Operation_Select => Logical_Operation_Select,
        Bit_Shift_En => Bit_Shift_En,
        Bit_Shift_with_Dir => B_Shift_with_Dir);
```

```
    --IndexNumber
     --220213D = 11 0101 1100 0011 0101
     --220419N = 11 0101 1101 0000 0011
     --220303e = 11 0101 1100 1000 1111
     --220407C = 11 0101 1100 1111 0111

         process
         begin
             jmp_check <= "0000";
         -- MOVI R1, 10
             ins_bus <= "0100010001010";
             wait for 100ns;

             -- MOVI R2, 1
             ins_bus <= "0100100000101";
             wait for 100ns;

             -- NEG R2
             ins_bus <= "1010010110011";
             wait for 100ns;

             -- JZR 1
             ins_bus <= "1100110000001";
             wait;
         end process;

end Behavioral
```

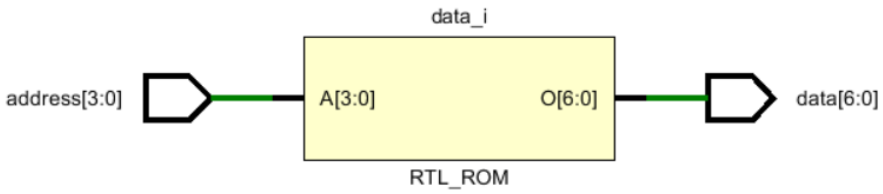
Instruction Decoder Timing Diagram



Seven Segment Display – Design

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

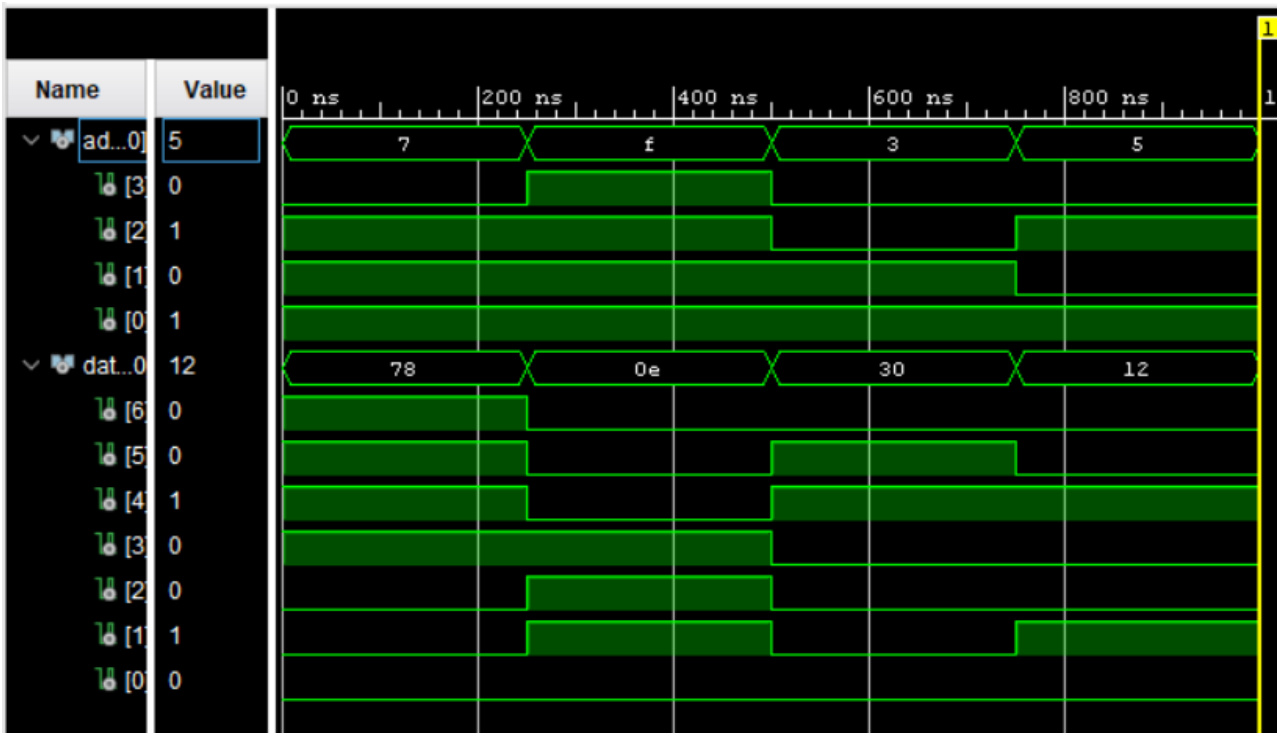
Seven segment display Design Schematic



Seven Segment Display – Design

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

entity TB_LUT_16_7 is
--  Port ( );
end TB_LUT_16_7;

architecture Behavioral of TB_LUT_16_7 is

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

signal address : std_logic_vector(3 downto 0);
signal data : std_logic_vector (6 downto 0);

begin
UUT : LUT_16_7 port map (
    address => address,
    data => data);


process
    begin
        address <= "0111";
        wait for 250 ns;

        address <= "1111";
        wait for 250 ns;

        address <= "0011";
        wait for 250 ns;

        address <= "0101";
        wait;
    end process;

end Behavioral;
```

Seven segment display Timing Diagram



Nanoprocessor - Design

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

entity Main_Process is
    Port (
        Clk_In : in STD_LOGIC;
        Reset : in STD_LOGIC;
        Overflow_Flag : out STD_LOGIC;
        Zero_Flag : out STD_LOGIC;
        seg_data: out STD_LOGIC_VECTOR (6 downto 0);
        anode : out STD_LOGIC_VECTOR (3 downto 0);
        Reg_7_Out : out std_logic_vector(3 downto 0);
        Equal : out STD_LOGIC;
        Greater : out STD_LOGIC;
        Less : out STD_LOGIC;
        Log_and_Shift_out : out STD_LOGIC_VECTOR (3 downto 0)
    );
end Main_Process;

architecture Behavioral of Main_Process is

component Instruction_Decoder is
      Port ( Instruction_Bus : in STD_LOGIC_VECTOR (12 downto 0);
             Check_For_Jump : in STD_LOGIC_VECTOR (3 downto 0);
             Register_Enable : out STD_LOGIC_VECTOR (2 downto 0);
             Load_Select : out STD_LOGIC;
             Immediate_Value : out STD_LOGIC_VECTOR (3 downto 0);
             Register_Select_0 : out STD_LOGIC_VECTOR (2 downto 0);
             Register_Select_1 : out STD_LOGIC_VECTOR (2 downto 0);
             A_S_Select : out STD_LOGIC;
             Jump_Flag : out STD_LOGIC;
             Jump_Address : out STD_LOGIC_VECTOR (2 downto 0);
             Comparator_En : out STD_LOGIC;
             Logical_unit_en : out STD_LOGIC;
             Logical_Operation_Select : out STD_LOGIC_VECTOR(1 Downto 0);
             Bit_Shift_En : out STD_LOGIC;
             Bit_Shift_with_Dir : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux_8_W_4_B is
    Port ( 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);
           C_OUT : out STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0));
end component;
```

```
component Mux_2_W_3_B is
    Port ( A_in : in STD_LOGIC_VECTOR (2 downto 0);
           B_in : in STD_LOGIC_VECTOR (2 downto 0);
           S_in : in STD_LOGIC;
           C_out : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux_2_W_4_B is
    Port ( S_in : in STD_LOGIC;
           A_in : in STD_LOGIC_VECTOR (3 downto 0);
           B_in : in STD_LOGIC_VECTOR (3 downto 0);
           C_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component Register_Bank is
     Port ( Value_In : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           Reg_En : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC;
           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));
end component;

component Add_Sub_Unit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          C_in : in STD_LOGIC;
          S : out STD_LOGIC_VECTOR (3 downto 0);
          C_outT : out STD_LOGIC;
          overFlow:out Std_logic;
          zero:out STD_LOGIC;
          K : in STD_LOGIC);
end component;

component Program_Rom is
    Port ( Memo_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Instruct_Bus : out STD_LOGIC_VECTOR (12 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 is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end component;

component Comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           EN : in STD_LOGIC;
           Equal : out STD_LOGIC;
           Greater : out STD_LOGIC;
           Less : out STD_LOGIC);
end component;

component Logical_Unit is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B : in STD_LOGIC_VECTOR (3 downto 0);
        En : in STD_LOGIC;
        Op_Select : in STD_LOGIC_VECTOR (1 downto 0);
        Out_Result : out STD_LOGIC_VECTOR (3 downto 0)
    );
End component;
```

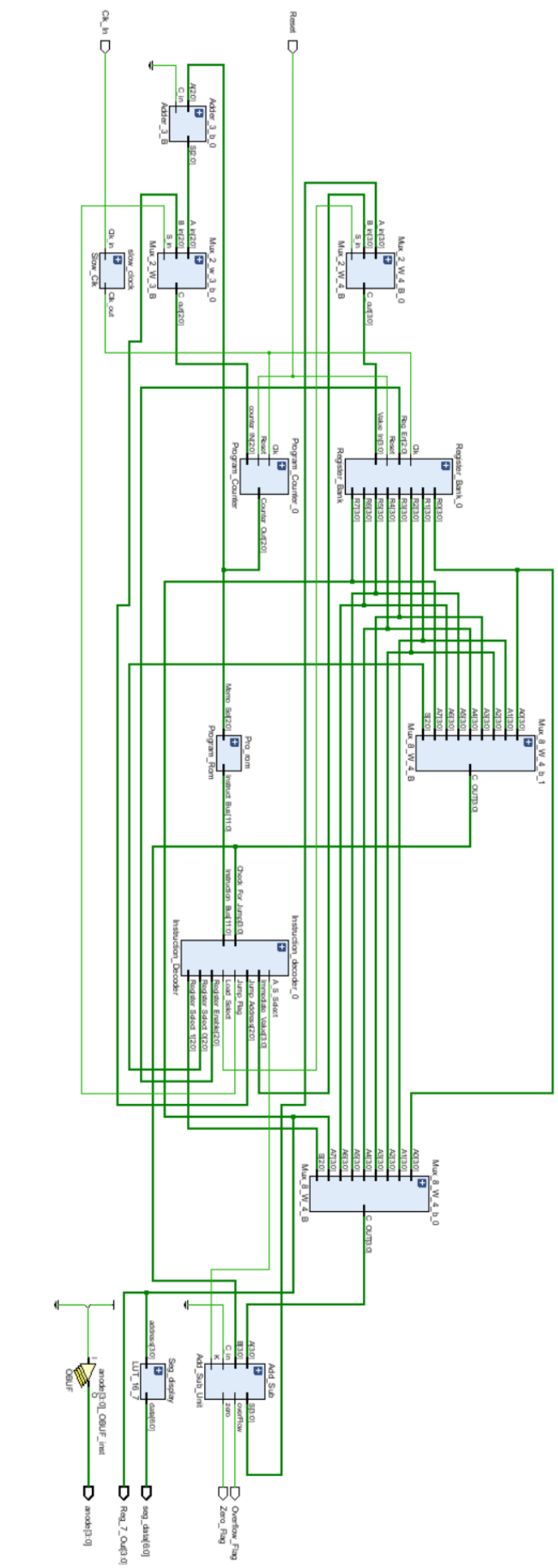
```
component Bit_Shift is
    Port (
        A : in STD_LOGIC_VECTOR (3 downto 0);
        B_Shift_with_Dir : in STD_LOGIC_VECTOR (2 downto 0);
        En : in STD_LOGIC;
        A_out : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal Instruction_Bus : STD_LOGIC_VECTOR (12 downto 0);
signal Check_For_Jump : STD_LOGIC_VECTOR (3 downto 0);
signal Register_Enable : STD_LOGIC_VECTOR (2 downto 0);
signal Immediate_Value : STD_LOGIC_VECTOR (3 downto 0);
signal Register_Select_0, Register_Select_1 : STD_LOGIC_VECTOR (2 downto
0);
signal A_S_Select,Jump_Flag, Load_Select: STD_LOGIC;
signal Jump_Address : STD_LOGIC_VECTOR (2 downto 0);

signal reg_bank_in : STD_LOGIC_VECTOR (3 downto 0);
signal clk : STD_LOGIC;
signal Data_bus_0, Data_bus_1, Data_bus_2, Data_bus_3, Data_bus_4,
Data_bus_5, Data_bus_6, Data_bus_7 : STD_LOGIC_VECTOR (3 downto 0);

signal mux_0_out, mux_1_out : STD_LOGIC_VECTOR (3 downto 0);

signal add_sub_out : STD_LOGIC_VECTOR( 3 downto 0);
signal c_out_add_sub : STD_LOGIC;

signal adder_3_bit_out, next_address : STD_LOGIC_VECTOR (2 downto 0);
signal adder_3_bit_carry_out : STD_LOGIC;

signal memory_select : STD_LOGIC_VECTOR (2 downto 0);

signal zero, over, bit_Shift_en : STD_LOGIC;
signal Comparator_En, Logical_unit_en : STD_LOGIC;
signal Logical_Operation_Select : STD_LOGIC_VECTOR (1 downto 0);

signal B_Shift_with_Dir : STD_LOGIC_VECTOR (2 downto 0);
signal A_out, Out_result : STD_LOGIC_VECTOR(3 downto 0);

begin
Instruction_decoder_0 : Instruction_decoder
    port map (
         Instruction_Bus => Instruction_Bus,
         Check_For_Jump => Check_for_jump,
         Register_Enable => Register_enable,
         Load_Select => Load_Select,
         Immediate_Value => Immediate_Value,
         Register_Select_0 => Register_Select_0,
         Register_Select_1 => Register_Select_1,
         A_S_Select => A_S_Select,
         Jump_Flag => Jump_Flag,
         Jump_Address => Jump_Address,
         Comparator_En => Comparator_En,
         Logical_Unit_En => Logical_unit_en,
         Logical_Operation_Select => Logical_Operation_Select,
         Bit_Shift_En => bit_Shift_en,
         Bit_Shift_with_Dir => B_Shift_with_Dir);
```

```
Register_Bank_0 : Register_Bank
    port map (
           Value_In => reg_bank_in,
           Clk => Clk,
           Reg_En => Register_enable,
           Reset => Reset,
           R0 => Data_bus_0,
           R1 => Data_bus_1,
           R2 => Data_bus_2,
           R3 => Data_bus_3,
           R4 => Data_bus_4,
           R5 => Data_bus_5,
           R6 => Data_bus_6,
           R7 => Data_bus_7);

Mux_8_W_4_b_0 : Mux_8_W_4_B
    port map (
           A0 => Data_bus_0,
           A1 => Data_bus_1,
           A2 => Data_bus_2,
           A3 => Data_bus_3,
           A4 => Data_bus_4,
           A5 => Data_bus_5,
           A6 => Data_bus_6,
           A7 => Data_bus_7,
           C_OUT => mux_0_out,
           S => Register_Select_1);

Check_For_Jump <= mux_1_out;

Mux_8_W_4_b_1 : Mux_8_W_4_B
    port map (
           A0 => Data_bus_0,
           A1 => Data_bus_1,
           A2 => Data_bus_2,
           A3 => Data_bus_3,
           A4 => Data_bus_4,
           A5 => Data_bus_5,
           A6 => Data_bus_6,
           A7 => Data_bus_7,
           C_OUT => mux_1_out,
           S => Register_Select_0);

Add_Sub : Add_Sub_Unit
    port map (
          A => mux_0_out,
          B  => mux_1_out,
          C_in  => '0',
          S  => add_sub_out,
          C_outT  => c_out_add_sub,
          overFlow  => over,
          zero  => zero,
          K => A_S_Select);
```

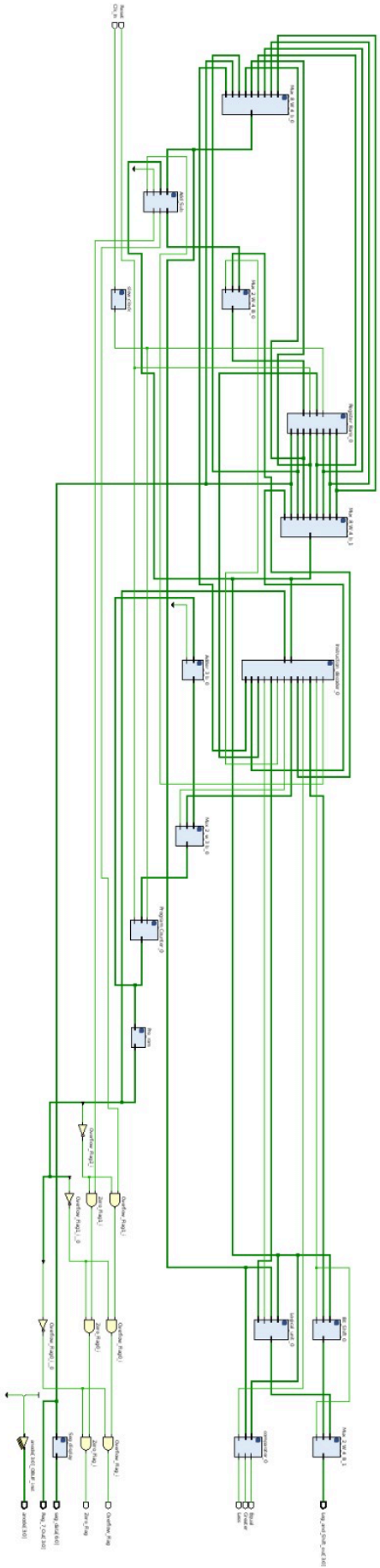
```
Mux_2_W_4_B_0 : Mux_2_W_4_B
    port map (
           S_in => Load_Select,
           A_in => add_sub_out,
           B_in => Immediate_Value,
           C_out => reg_bank_in);

Adder_3_b_0 : Adder_3_B
    port map (
           A => memory_select,
           C_in => '0',
           S => adder_3_bit_out,
           C_out => adder_3_bit_carry_out);

Mux_2_w_3_b_0 : Mux_2_W_3_B
    port map (
           A_in => adder_3_bit_out,
           B_in => Jump_Address,
           S_in => Jump_Flag,
           C_out => next_address);


Program_Counter_0 : Program_counter
    port map (
          Counter_IN => next_address,
          Reset => Reset,
          Clk => CLK,
          Counter_Out => memory_select);

Pro_rom : Program_Rom
   port map (
        Memo_Sel => memory_select,
        Instruct_Bus => Instruction_Bus);

slow_clock : Slow_clk
    port map(
        Clk_in => Clk_in,
        CLk_out => clk);

Seg_display : LUT_16_7
    port map(
        address => Data_bus_7,
        data => seg_data);

comparator_0 :  Comparator
    Port map (
           A => mux_1_out,
           B => mux_0_out,
           EN => comparator_en,
           Equal => Equal,
           Greater => Greater,
           Less => Less);
```

```
Logical_unit_0 :  Logical_Unit
      Port map (
           A => mux_1_out,
           B => mux_0_out,
           En => Logical_unit_en,
           Op_Select => Logical_Operation_Select,
           Out_Result => Out_Result);

 Bit_Shift_0 :  Bit_Shift
       Port map (
           A => mux_1_out,
           B_Shift_with_Dir => B_Shift_with_Dir,
           En => Bit_Shift_En,
           A_out => A_out);

Mux_2_W_4_B_1 : Mux_2_W_4_B
       port map (
              S_in => Bit_Shift_En,
              A_in => Out_Result,
              B_in => A_out,
              C_out => Log_and_Shift_out);

Zero_Flag <= zero AND NOT instruction_bus(12) AND NOT instruction_bus(11)
AND NOT instruction_bus(10);
Overflow_Flag <= over AND NOT instruction_bus(12) AND NOT
instruction_bus(11) AND NOT instruction_bus(10);
anode <= "1110";
Reg_7_Out <= Data_bus_7;

end Behavioral;
```

Nanoprocessor Design Schematic (Basic)



Nanoprocessor Design Schematic (After Optimizing)

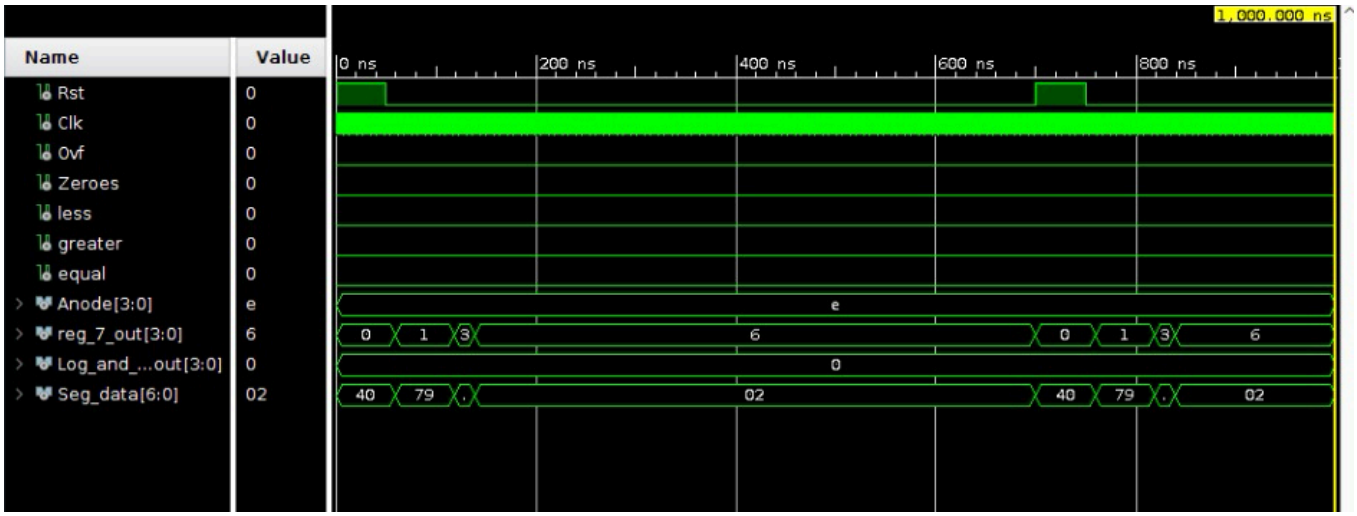


Nanoprocessor – TB

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

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

architecture Behavioral of TB_Main_Process is

component Main_Process is
    Port (
        Clk_In : in STD_LOGIC;
        Reset : in STD_LOGIC;
        Overflow_Flag : out STD_LOGIC;
        Zero_Flag : out STD_LOGIC;
        seg_data: out STD_LOGIC_VECTOR (6 downto 0);
        Reg_7_Out : out std_logic_vector (3 downto 0);
        anode : out STD_LOGIC_VECTOR (3 downto 0);
        Equal : out STD_LOGIC;
        Greater : out STD_LOGIC;
        Less : out STD_LOGIC;
        Log_and_Shift_out : out STD_LOGIC_VECTOR (3 downto 0)
    );
end component;

SIGNAL Rst, Clk, Ovf, Zeroes, less, greater, equal : STD_LOGIC;
SIGNAL  Anode : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL  reg_7_out, Log_and_Shift_out : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL Seg_data : STD_LOGIC_VECTOR(6 downto 0);

begin
    UTT : Main_Process
        port map (
            clk_In => clk,
            Reset => rst,
            Overflow_Flag => ovf,
            Zero_Flag => zeroes,
            seg_data => seg_data,
            anode => anode,
            Reg_7_Out => reg_7_out,
            Equal => equal,
            Less => less,
            Greater => greater,
            Log_and_Shift_out => Log_and_Shift_out
            );
```

```
    clk_process : process -- clock processor
        begin
            Clk <= '0';
            wait for 1ns;

            Clk <= '1';
            wait for 1ns;
    end process;

    process
        begin

            Rst <= '1';
            wait for 50ns;
            Rst <= '0';
            wait for 850ns; --wait forever

        end process;

end Behavioral;
```

Nanoprocessor Timing Diagram



Constraints File

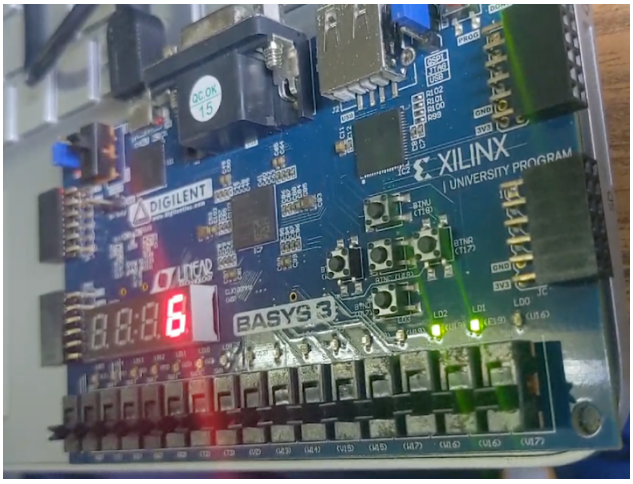
Constraints

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

## LEDs
set_property PACKAGE_PIN U16 [get_ports {Reg_7_Out[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Reg_7_Out[0]}]
set_property PACKAGE_PIN E19 [get_ports {Reg_7_Out[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Reg_7_Out[1]}]
set_property PACKAGE_PIN U19 [get_ports {Reg_7_Out[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Reg_7_Out[2]}]
set_property PACKAGE_PIN V19 [get_ports {Reg_7_Out[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Reg_7_Out[3]}]

set_property PACKAGE_PIN W18 [get_ports {Equal}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Equal}]
set_property PACKAGE_PIN U15 [get_ports {Greater}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Greater}]
set_property PACKAGE_PIN U14 [get_ports {Less}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Less}]

set_property PACKAGE_PIN V14 [get_ports {Log_and_Shift_out[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Log_and_Shift_out[0]}]
set_property PACKAGE_PIN V13 [get_ports {Log_and_Shift_out[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Log_and_Shift_out[1]}]
set_property PACKAGE_PIN V3 [get_ports {Log_and_Shift_out[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Log_and_Shift_out[2]}]
set_property PACKAGE_PIN W3 [get_ports {Log_and_Shift_out[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Log_and_Shift_out[3]}]

set_property PACKAGE_PIN P1 [get_ports {Zero_Flag}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Zero_Flag}]
set_property PACKAGE_PIN L1 [get_ports {Overflow_Flag}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Overflow_Flag}]

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

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

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

Conclusions from the Lab

This project successfully designed and implemented a 4-bit nanoprocessor capable of executing a core set of instructions. We achieved this by building and integrating various components, including an arithmetic logic unit (ALU) with extended functionality beyond addition and subtraction. It can process comparisons, bit shifting, and logical operators beyond the baseline requirements.

We effectively connected multiplexers and tri-state buffers to control data flow within the processor. The implemented instruction decoder accurately activates necessary components based on the executed instruction.



Furthermore, we wrote and executed an assembly program to calculate the sum of integers between 1 and 3, showcasing the nanoprocessor's ability to perform calculations. We successfully executed on the BASYS 3 board and validated the functionality of our design.

Throughout the project, each team member made significant contributions, From designing and testing individual components to integrating them into the final system, everyone played a vital role in the success of our nanoprocessor.

So in conclusion, We designed a nano processor that fulfills base requirements and includes enhanced functionalities. Through this project, we developed technical skills and fostered teamwork, problem-solving abilities, and a deeper understanding of computer organization and digital design principles.

Contributions by Team Members

Member	Contribution
220213D Harinakshi W.B.S.S.	▪ 4-bit Add/Sub Unit ▪ 4-bit Comparator ▪ 2-4 Decoder ▪ Test Benches for all Components
220303E Kariyawasam U.G.R.T.	▪ 3-bit Adder ▪ 2-way 4-bit Multiplexer ▪ 8-way 4-bit Multiplexer ▪ Logical Unit ▪ Bit Shifter ▪ Final Lab Report
220407C Muthumala V.D.W.	▪ 3-bit Buffer ▪ Register ▪ Register Bank ▪ Program Counter ▪ Program Rom ▪ Instruction Decoder ▪ Nanoprocessor
220419N Navodi S.Y.A.C.	▪ 4-bit Buffer ▪ 2-way 3-bit Multiplexer ▪ 4-bit Comparator ▪ 3-8 Decoder ▪ Test Benches for all Components
Whole Team	▪ Testing and verifying through Basys3 board