<u>Nanoprocessor Design Competition Report</u>

Team: Hashtag

- Talagala S.A. 220630D
- Gunawardana S.D.M.D. 220201N
- Samarasinghe Y.B.P. 220554X
- Arachchi K.A.K.N.K. 220039A

Content

Assigned Lab Task

- Design and develop a 4-bit arithmetic unit that can add and subtract signed integers.
- Design and develop k-way b-bit multiplexers.
- Develop a register bank consisting of eight registers.
- Develop a program read-only memory (ROM) utilizing a Lookup Table to store our Assembly program.
- Decode instructions to activate necessary components on the processor.
- Write an Assembly program to calculate the total of all integers between 1 and 3 and convert it into machine code.
- Verify their functionality via simulation and on the development board.

Assembly Program and its Machine Code Representation

Assembly Program

```
    MOVI    R7, 1   ; Move immediate value 1 to register R7
    MOVI    R6, 2   ; Move immediate value 2 to register R6
    MOVI    R5, 3   ; Move immediate value 3 to register R5
    ADD     R7, R6  ; Add the values of registers R7 and R6, store result in
R7
    ADD     R7, R5  ; Add the values of registers R7 and R5, store result in
R7
    MOVI    R0, 0   ; Move immediate value 0 to register R0
    JZR     R0, 6   ; Jump to instruction 6 if the value in R0 is zero
    END     ; End the program
```

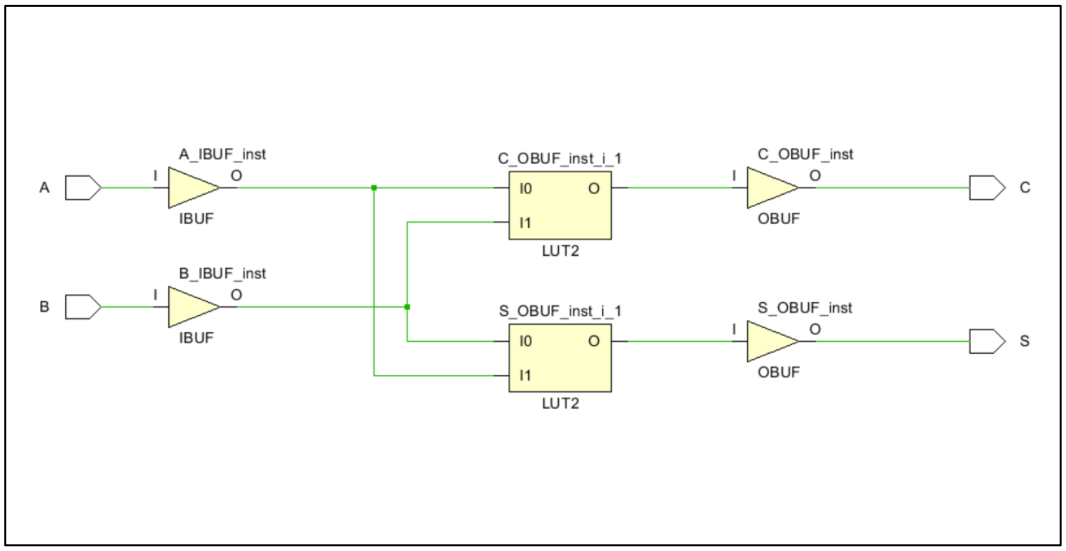
Machine Code

```
        "101110000001", -- MOVI R7, 1
        "101100000010", -- MOVI R6, 2
        "101010000011", -- MOVI R5, 3
        "001111100000", -- ADD R7, R6
        "001111010000", -- ADD R7, R5
        "101000000000", -- MOVI R0, 0
        "110000000101"  -- JZR R0, 6
```

Components

Half Adder

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/13/2024 02:36:15 PM
-- Design Name:
-- Module Name: HA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

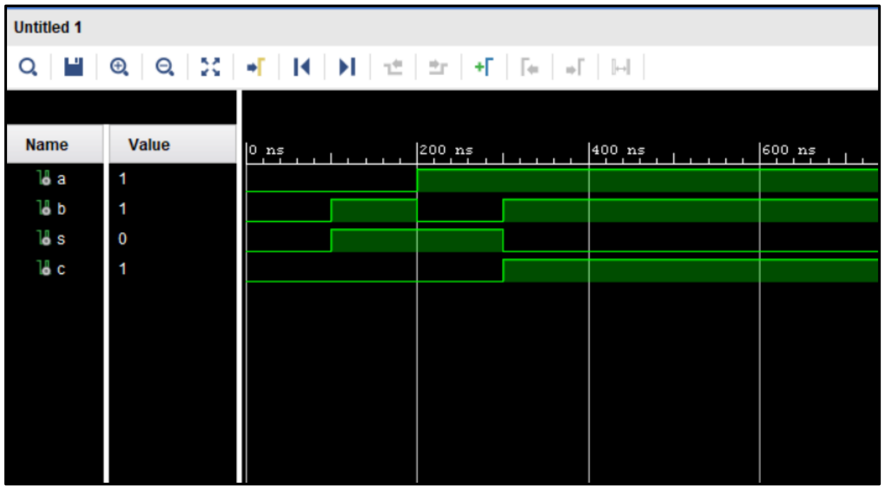
```vhdl
--
--------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity HA is
    Port ( A : in STD_LOGIC;    -- Input A
           B : in STD_LOGIC;    -- Input B
           S : out STD_LOGIC;   -- Output sum
           C : out STD_LOGIC);  -- Output carry
end HA;

architecture Behavioral of HA is

begin
    S<= A XOR B;    -- Sum is the XOR of inputs A and B
    C<= A AND B;    -- Carry is the AND of inputs A and B

end Behavioral;
```

Simulation Source File

```vhdl
--------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/16/2024 04:12:33 PM
-- Design Name:
-- Module Name: TB_HA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

```vhdl
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

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';
         b <= '0';

         WAIT FOR 100 ns;
         b <= '1';

         WAIT FOR 100 ns;
         b <= '0';
         a <= '1';

         WAIT FOR 100 ns;
         b <= '1';

         WAIT;

     end process;

end Behavioral;
```

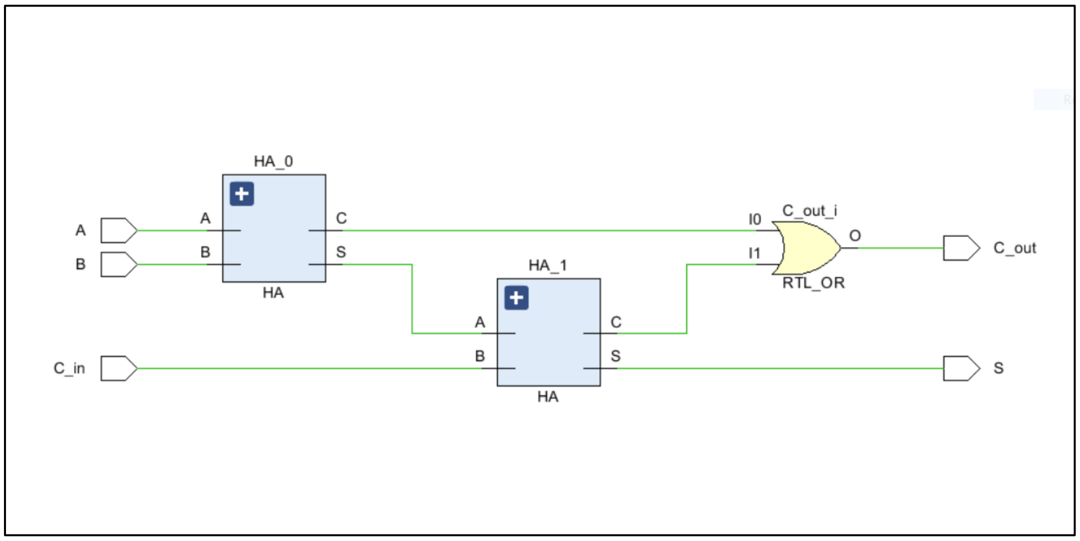
Timing Diagram



Full Adder

The full adder circuit is constructed by combining two half adders.

Elaborated Design Schematic



Design Source File

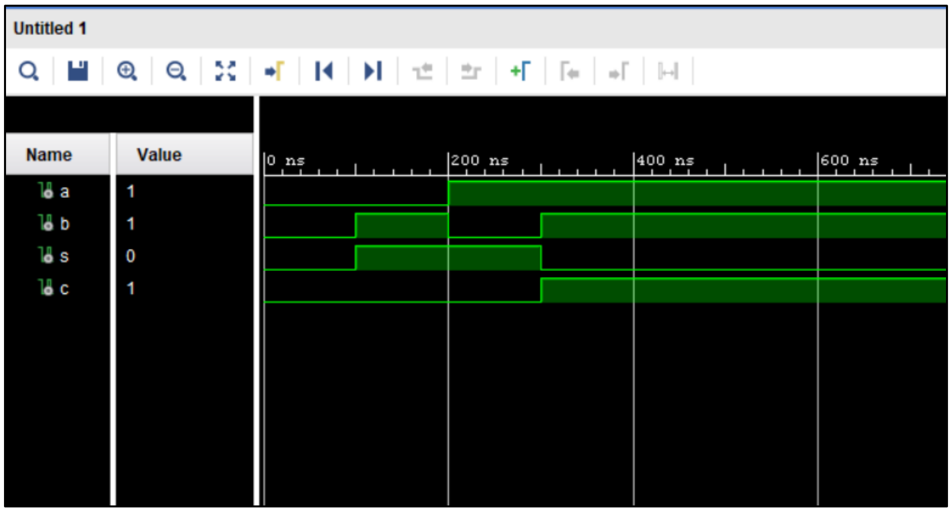
```vhdl
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/13/2024 02:38:20 PM
-- Design Name:
-- Module Name: FA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity FA is
    Port ( A : in STD_LOGIC;        -- Input A
           B : in STD_LOGIC;        -- Input B
           C_in : in STD_LOGIC;     -- Input carry-in
           S : out STD_LOGIC;       -- Output sum
           C_out : out STD_LOGIC);  -- Output carry-out
end FA;

architecture Behavioral of FA is
     component HA                     -- Half Adder component declaration
     port (
     A: in std_logic;                  -- Input A
     B: in std_logic;                  -- Input B
     S: out std_logic;                 -- Output sum
     C: out std_logic);                -- Output carry
     end component;
     SIGNAL HA0_S, HA0_C, HA1_S, HA1_C : std_logic;  -- Internal signals for
half adders

begin
         HA_0 : HA                          -- Instantiation of first half
adder
             port map (
             A => A,
             B => B,
             S => HA0_S,
             C => HA0_C);
         HA_1 : HA                          -- Instantiation of second half
adder
             port map (
             A => HA0_S,
             B => C_in,
             S => HA1_S,
             C => HA1_C);
         S <= HA1_S;                        -- Output sum is the output of
the second half adder
         C_out <= HA0_C OR HA1_C;           -- Output carry-out is the OR of
carry-out from both half adders


     end Behavioral;
```

Simulation Source File

```
-----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/16/2024 04:15:23 PM
-- Design Name:
-- Module Name: TB_FA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----
```

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

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

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

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

architecture Behavioral of TB_FA is
        COMPONENT FA
        PORT ( A, B, C_in : IN STD_LOGIC;
               S, C_out : OUT STD_LOGIC);
        END COMPONENT;
        SIGNAL a, b, c_in : std_logic;
        SIGNAL s, c_out : std_logic;

begin
    UUT: FA PORT MAP (
         A => a,
         B => b,
         C_in => c_in,
         S => s,
         C_out => c_out);
    process
    begin
     a <= '0';
     b <= '0';
     c_in <= '0';
     WAIT FOR 100 ns;
     c_in <= '1';
     WAIT FOR 100 ns;
     b <= '1';
     c_in <= '0';
     WAIT FOR 100 ns;
     c_in <= '1';
     WAIT FOR 100 ns;
     a <= '1';
     b <= '0';
     c_in <= '0';
     WAIT FOR 100 ns;
     c_in <= '1';
     WAIT FOR 100 ns;
     b <= '1';
     c_in <= '0';
     WAIT FOR 100 ns;
     c_in <= '1';
     WAIT;

    end process;
end Behavioral;
```

Timing Diagram

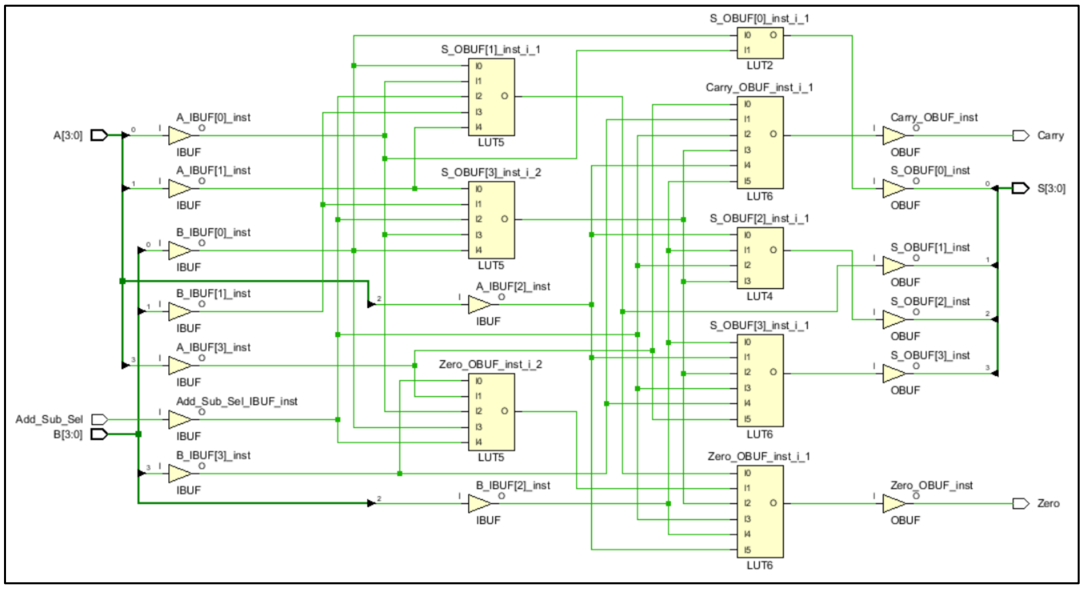


Name	Value
a	1
b	1
s	0
c	1

4-Bit Add/Subtract Unit

The 4-bit add/subtract unit is constructed by using 4 full adders.

Elaborated Design Source



Design Source File

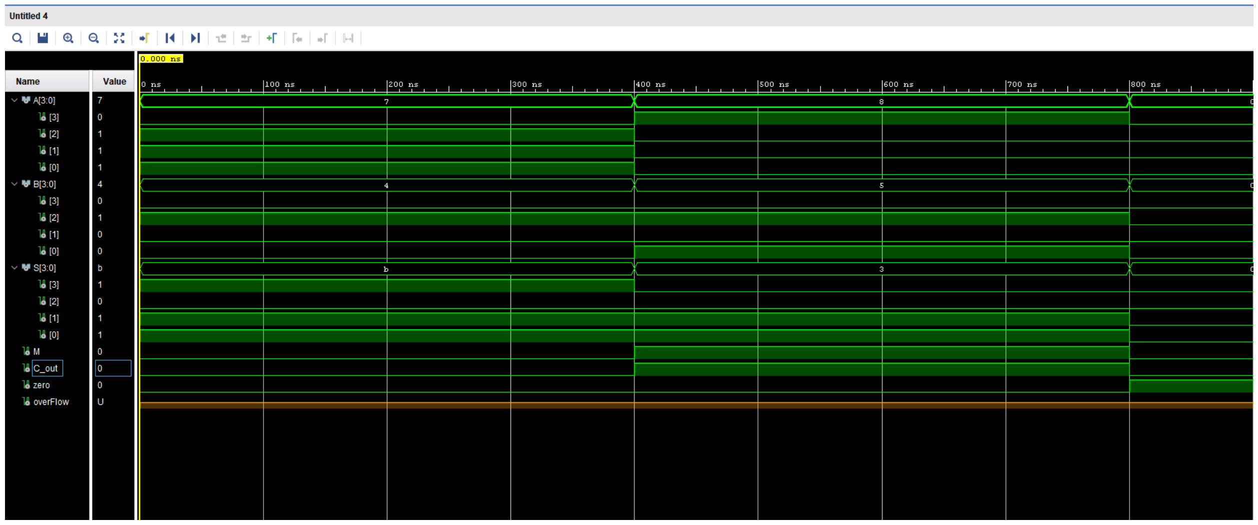
```vhdl
--------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/17/2024 01:08:05 PM
-- Design Name:
-- Module Name: Add_Sub_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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


entity Add_Sub_4bit is
    Port (
        A: in Std_Logic_Vector (3 downto 0);          -- Input A (4-bit)
        B: in Std_Logic_Vector (3 downto 0);          -- Input B (4-bit)
        S: out Std_Logic_Vector (3 downto 0);         -- Output
sum/difference (4-bit)
        Add_Sub_Sel : in STD_LOGIC;                   -- Selector for
addition/subtraction
        Carry : out STD_LOGIC;                        -- Output carry bit
        Zero: out Std_Logic                           -- Output zero flag
    );
end Add_Sub_4bit;

architecture Behavioral of Add_Sub_4bit is
    -- Component declaration for a full adder
    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;

    -- Internal signals
    Signal S_out, B_in, FA_C: Std_Logic_Vector (3 downto 0);
    Signal C_out: Std_Logic;
begin
    -- Full adder instances for each bit
    FA_0 : FA
        port map (
            A => A(0),
            B => B_in(0),
            C_in => Add_Sub_Sel,
            S => S_out(0),
            C_out => FA_C(0)
        );

    FA_1 : FA
        port map (
            A => A(1),
            B => B_in(1),
            C_in => FA_C(0),
            S => S_out(1),
            C_out => FA_C(1)
        );

    FA_2 : FA
        port map (
            A => A(2),
            B => B_in(2),
            C_in => FA_C(1),
            S => S_out(2),
            C_out => FA_C(2)
        );

    FA_3 : FA
        port map (
            A => A(3),
            B => B_in(3),
            C_in => FA_C(2),
            S => S_out(3),
            C_out => FA_C(3)
        );

    -- Logic for selecting the input B based on the operation
    B_in(0) <= Add_Sub_Sel XOR B(0);
    B_in(1) <= Add_Sub_Sel XOR B(1);
    B_in(2) <= Add_Sub_Sel XOR B(2);
    B_in(3) <= Add_Sub_Sel XOR B(3);
    -- Assigning carry out and zero flag
    C_out <= FA_C(3);
    Zero <= NOT (S_out(0) OR S_out(1) OR S_out(2) OR S_out(3) OR C_out);

    -- Output assignments
    S <= S_out;
    Carry <= C_out;
end Behavioral;
```

Simulation Source File

```
------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/22/2024 11:10:39 AM
-- Design Name:
-- Module Name: TB_Add_Sub_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Add_Sub_4bit is
--  Port ( );
end TB_Add_Sub_4bit;

architecture Behavioral of TB_Add_Sub_4bit is
```

```
component Add_Sub_4bit is
    Port (A: in Std_Logic_Vector (3 downto 0);
          B: in Std_Logic_Vector (3 downto 0);
          S: out Std_Logic_Vector (3 downto 0);
          Add_Sub_Sel : in STD_LOGIC;
          Carry : out STD_LOGIC;
          Zero: out Std_Logic);
end component;

signal A,B,S: STD_LOGIC_VECTOR(3 DOWNTO 0);
signal Add_Sub_Sel:STD_LOGIC;
signal C_out:std_logic;
signal zero,overFlow:std_logic;

begin

    UUT:Add_Sub_4bit PORT MAP(A,B,S,Add_Sub_Sel,C_out,zero);

    main: process begin

        -- Add
        Add_Sub_Sel<='0';
        A<="0111";
        B<="0100";
        wait for 200 ns;

        Add_Sub_Sel<='0';
        A<="0111";
        B<="0100";
        wait for 200 ns;

        -- Subtract
        Add_Sub_Sel<='1';
        A<="1000";
        B<="0101";
        wait for 200 ns;

        Add_Sub_Sel<='1';
        A<="1000";
        B<="0101";
        wait for 200 ns;

        -- Zero
        Add_Sub_Sel<='0';
        A<="0000";
        B<="0000";
        wait;

    end process;
end Behavioral;
```

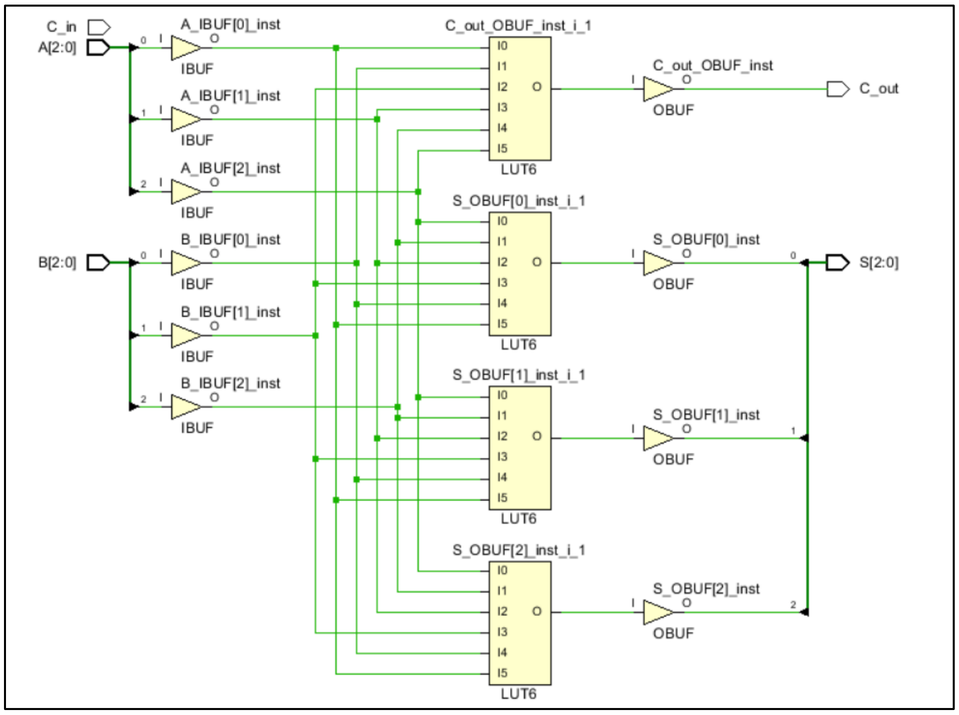
Timing Diagram



3-Bit Adder

The 3-bit adder is implemented by employing 3 full adders.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/20/2024 10:26:26 AM
-- Design Name:
-- Module Name: Adder_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Adder_3bit is
    Port ( A: in Std_Logic_Vector (2 downto 0);          -- Input A (3-bit)
           B: in Std_Logic_Vector (2 downto 0);          -- Input B (3-bit)
           S: out Std_Logic_Vector (2 downto 0);        -- Output sum (3-bit)
           C_in : in STD_LOGIC;                          -- Carry-in
           C_out : out STD_LOGIC);                       -- Carry-out
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);
```

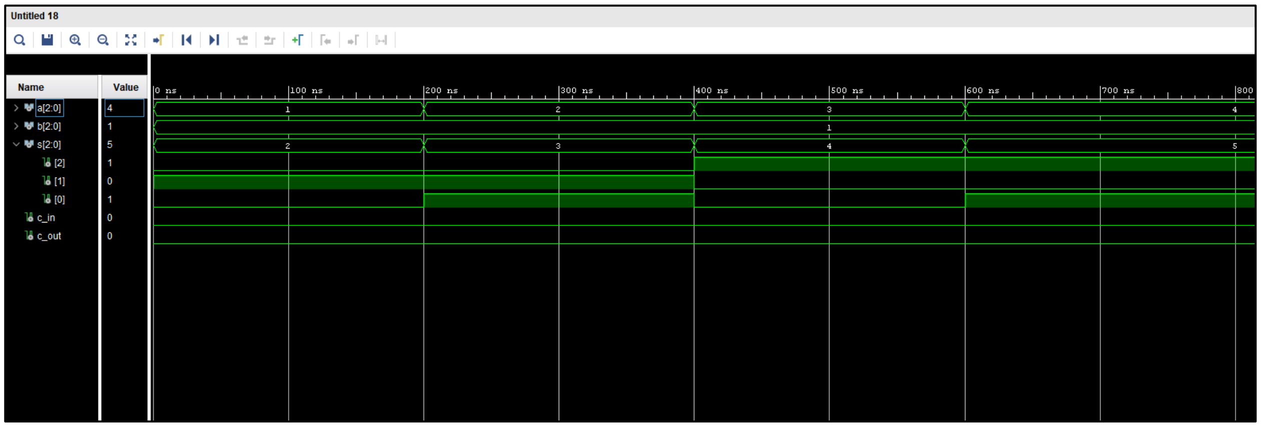
```
   end component;

SIGNAL FA_C: Std_Logic_Vector (2 downto 0);              -- Carry vector for
each bit
SIGNAL internal_S: Std_Logic_Vector (2 downto 0);        -- Internal sum
vector
SIGNAL internal_C_out: Std_Logic;                        -- Internal carry-
out

begin
   FA_0 : FA
           port map (
           A => A(0),                                    -- First bit of A
           B => B(0),                                    -- First bit of B
           C_in =>'0',                                   -- Initial carry-in
is 0
           S => internal_S(0),                           -- Output sum of
first bit
           C_out => FA_C(0));                            -- Carry-out of
first bit

   FA_1 : FA
       port map (
           A => A(1),                                    -- Second bit of A
           B => B(1),                                    -- Second bit of B
           C_in => FA_C(0),                              -- Carry-in from
first bit's FA
           S => internal_S(1),                           -- Output sum of
second bit
           C_out => FA_C(1));                            -- Carry-out of
second bit

   FA_2 : FA
       port map (
           A => A(2),                                    -- Third bit of A
           B => B(2),                                    -- Third bit of B
           C_in => FA_C(1),                              -- Carry-in from
second bit's FA
           S => internal_S(2),                           -- Output sum of
third bit
           C_out => internal_C_out);                     -- Internal carry-
out

-- Calculate final sum and carry-out
S <= (NOT(internal_C_out & internal_C_out & internal_C_out) AND internal_S)
OR ((internal_C_out & internal_C_out & internal_C_out) AND "000");
C_out <= internal_C_out;

end Behavioral;
```

Simulation Source File

```vhdl
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/20/2024 10:41:17 AM
-- Design Name:
-- Module Name: TB_Adder_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity 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);
             B: in Std_Logic_Vector (2 downto 0);
             S: out Std_Logic_Vector (2 downto 0);
             C_in : in STD_LOGIC;
             C_out : out STD_LOGIC);
        END COMPONENT;

        Signal a, b, s: Std_Logic_Vector (2 downto 0);
        SIGNAL c_in,c_out  : std_logic;
```

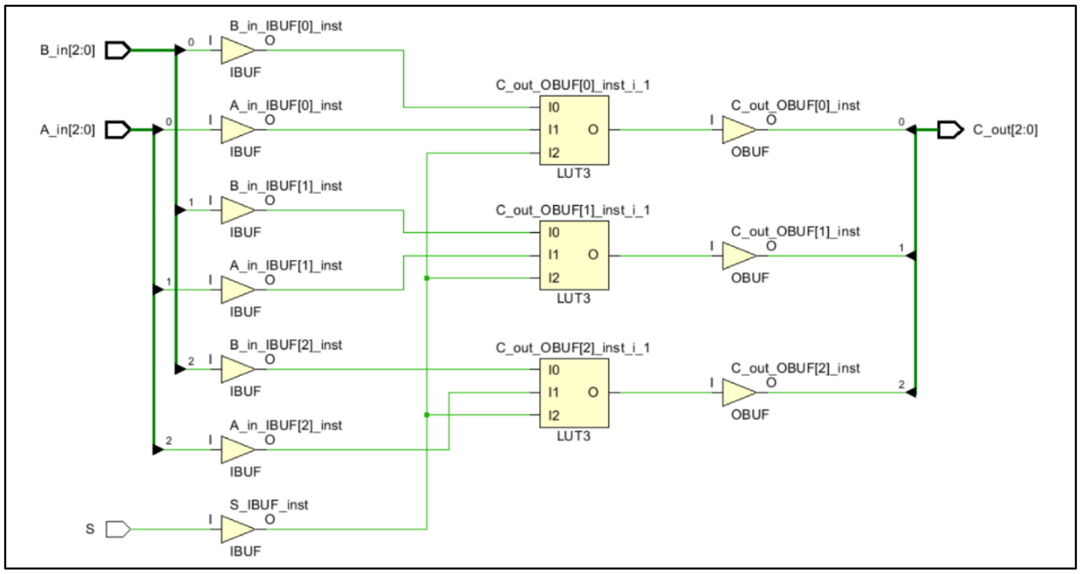
```
begin

    UUT: Adder_3bit  PORT MAP(A,B,S,c_in,c_out);

    main: process begin

            c_in <= '0';
            a <= "001";
            b <= "001";
            WAIT FOR 200 ns;
            a <= "010";
            Wait for 200 ns;
            a <= "011";
            Wait for 200 ns;
            a <= "100";
            Wait;

    end process;
end Behavioral;
```

Timing Diagram



2-Way 3-Bit Multiplexer

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/13/2024 04:31:04 PM
-- Design Name:
-- Module Name: Mux_2_Way_3_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

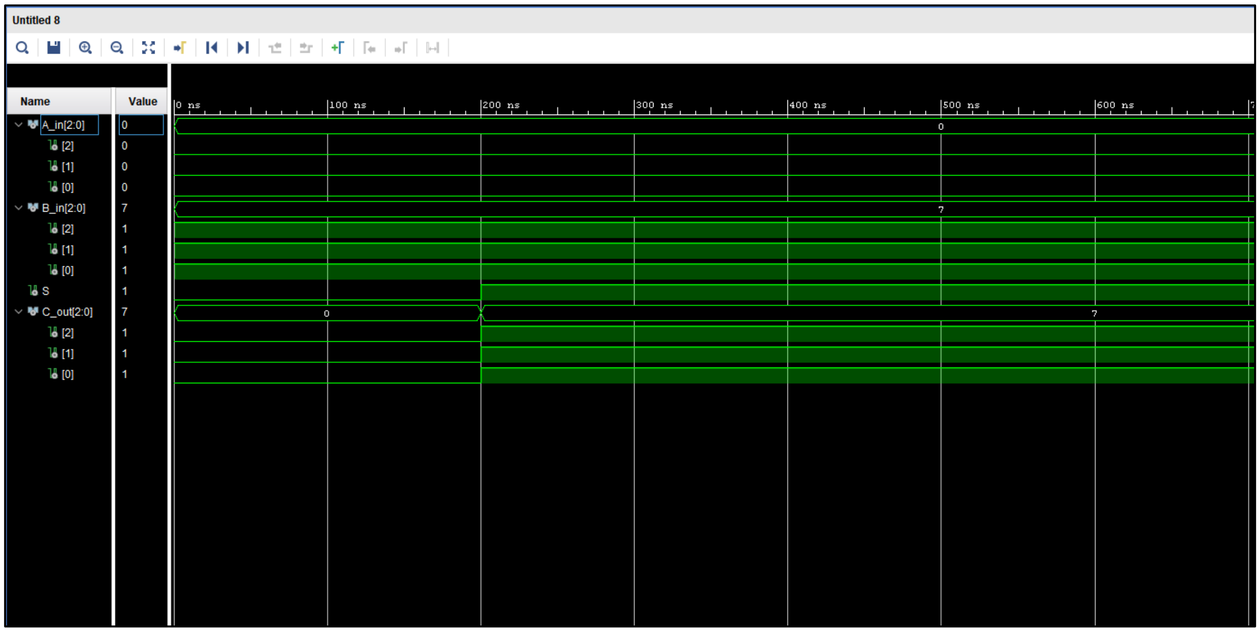
```
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Mux_2_Way_3_Bit is
    Port ( A_in : in STD_LOGIC_VECTOR (2 downto 0);        -- Input A
           B_in : in STD_LOGIC_VECTOR (2 downto 0);        -- Input B
           S  : in STD_LOGIC;                              -- Selection input
           C_out : out STD_LOGIC_VECTOR (2 downto 0));     -- Output
end Mux_2_Way_3_Bit;

architecture Behavioral of Mux_2_Way_3_Bit is

begin
    Process (A_in, B_in, S)
    Begin
        If S = '0' then                    -- If S is low
            C_out <= A_in;                  -- Output A
        Else                                -- If S is high
            C_out <= B_in;                  -- Output B
        End If;
End Process;

end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/22/2024 11:59:04 AM
-- Design Name:
-- Module Name: TB_Mux_2_W_3 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
```

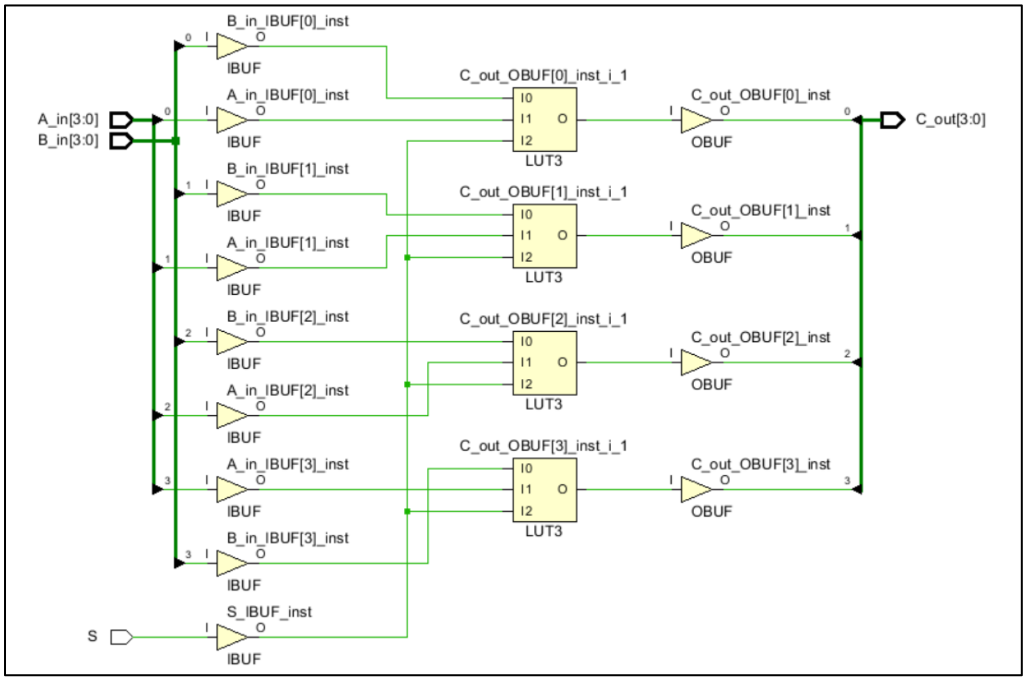
```
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Mux_2_W_3 is
--  Port ( );
end TB_Mux_2_W_3;

architecture Behavioral of TB_Mux_2_W_3 is

component Mux_2_Way_3_Bit is
    Port ( A_in : in STD_LOGIC_VECTOR (2 downto 0);
           B_in : in STD_LOGIC_VECTOR (2 downto 0);
           S : 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 : STD_LOGIC;
signal C_out : STD_LOGIC_VECTOR (2 downto 0);

begin

    UUT: Mux_2_Way_3_Bit PORT MAP(A_in, B_in, S, C_out);

    main: process begin

        A_in <= "000";
        B_in <= "111";

        S <= '0';
        wait for 200 ns;  -- Switch to next bus
        S <= '1';

        wait;

    end process;
end Behavioral;
```

Timing Diagram



2-Way 4-Bit Multiplexer

Elaborated Design Schematic



Design Source File

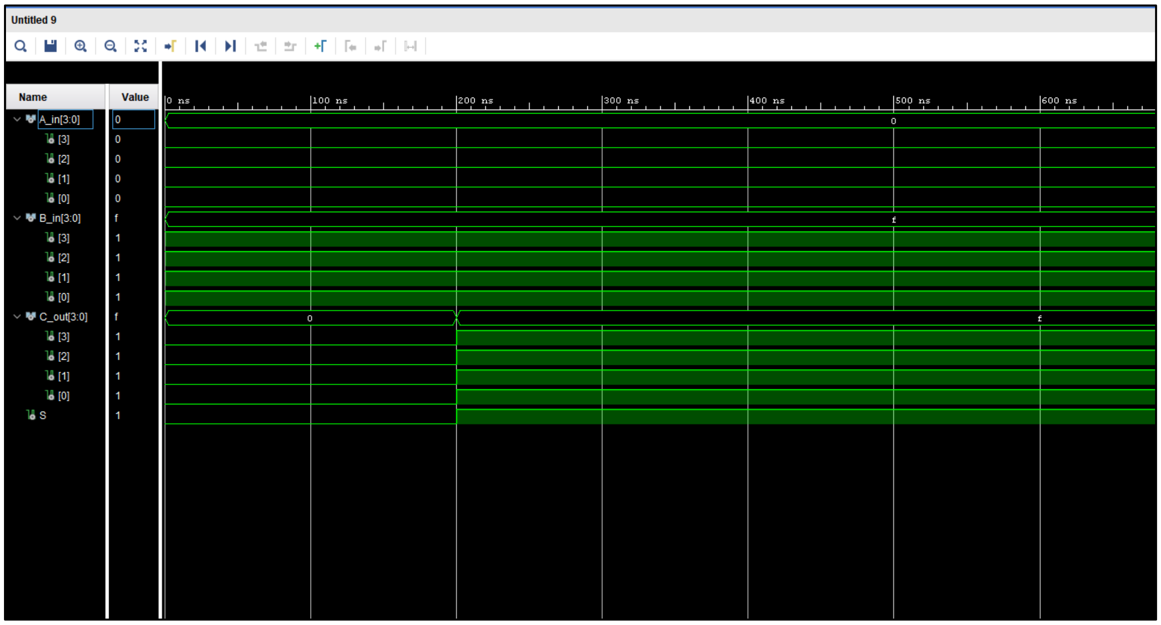
```
------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/13/2024 01:28:17 PM
-- Design Name:
-- Module Name: Mux_2_Way_4_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Mux_2_Way_4_Bit is
    Port (A_in: in Std_Logic_Vector (3 downto 0);       --Input A
          B_in: in Std_Logic_Vector (3 downto 0);       --Input B
          S: in Std_Logic;                              --Selection input
          C_out: out Std_Logic_Vector (3 downto 0));    --Output
end Mux_2_Way_4_Bit;

architecture Behavioral of Mux_2_Way_4_Bit is

begin
    Process (A_in, B_in, S)
    Begin
        If S = '0' then                                 --If S is low
            C_out <= A_in;                              --Output A
        Else                                            --If S is high
            C_out <= B_in;                              --Output B
        End If;
    End Process;
end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/22/2024 12:03:46 PM
-- Design Name:
-- Module Name: TB_Mux_2_W_4 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Mux_2_W_4 is
--  Port ( );
end TB_Mux_2_W_4;

architecture Behavioral of TB_Mux_2_W_4 is

component Mux_2_Way_4_Bit is
    Port (A_in: in Std_Logic_Vector (3 downto 0);
          B_in: in Std_Logic_Vector (3 downto 0);
          S: in Std_Logic;
          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 : STD_LOGIC ;

begin
    UUT: Mux_2_Way_4_Bit PORT MAP(A_in, B_in, S, C_out);

    main: process begin

        A_in<="0000";
        B_in<="1111";

        S<='0';
        wait for 200 ns;  -- Switch to next bus
        S<='1';

        wait;

    end process;
end Behavioral;
```

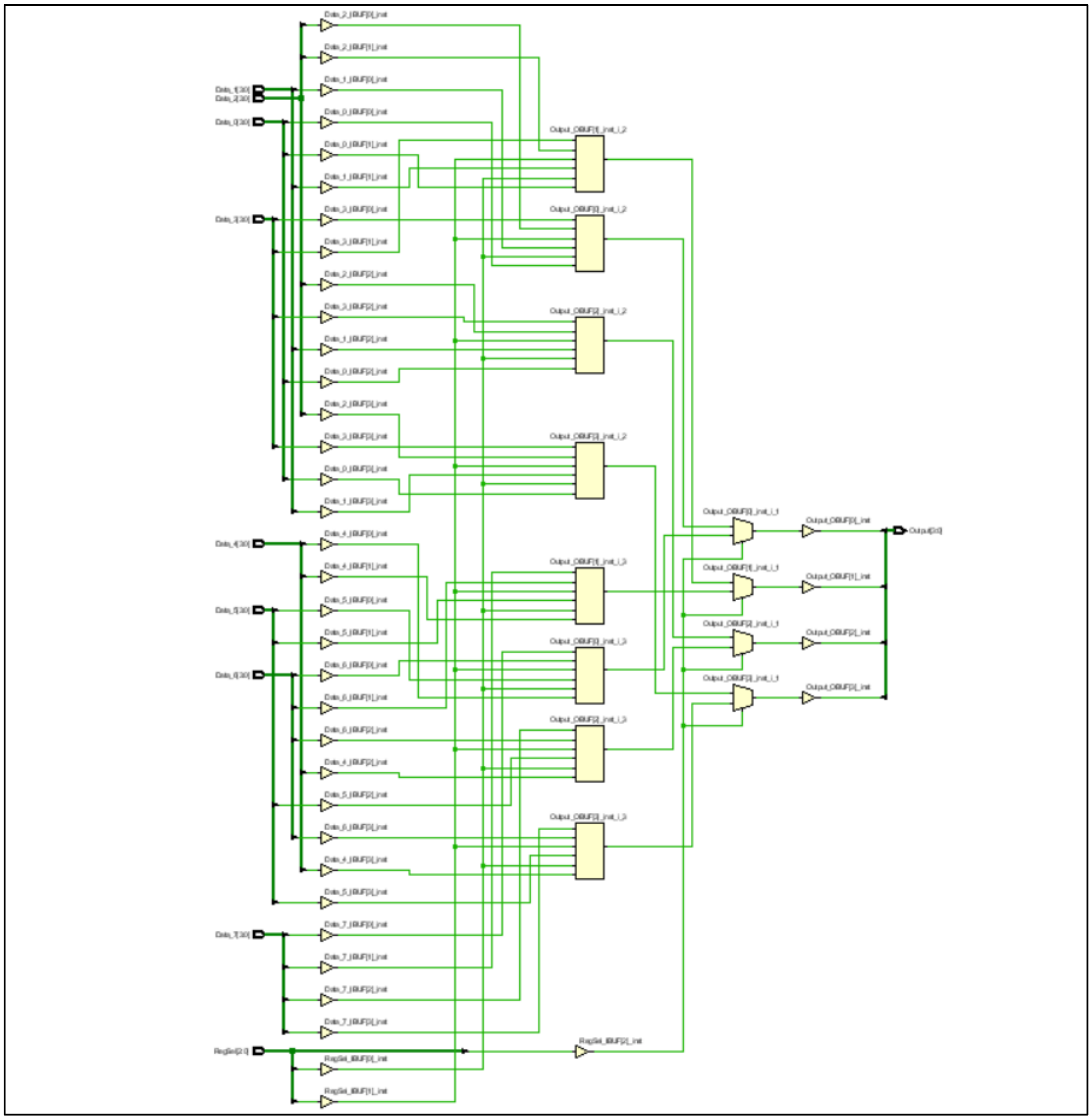
Timing Diagram



8-Way 4-Bit Multiplexer

The eight-way four-bit multiplexer is assembled using 7, two-way four-bit multiplexers.

Elaborated Design Schematic



Design Source File

```
------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/11/2024 10:00:07 PM
-- Design Name:
-- Module Name: Mux_8_Way_4_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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


entity Mux_8_Way_4_Bit is

    Port (Data_0, Data_1, Data_2, Data_3, Data_4, Data_5, Data_6, Data_7  :
in STD_LOGIC_VECTOR (3 downto 0);   -- Input data

          RegSel : in STD_LOGIC_VECTOR (2 downto 0);        -- Selection
input
          Output : out STD_LOGIC_VECTOR (3 downto 0));      -- Output data
end Mux_8_Way_4_Bit;

architecture Behavioral of Mux_8_Way_4_Bit is
    COMPONENT Mux_2_Way_4_Bit
    PORT(A_in: in STD_LOGIC_VECTOR (3 downto 0);
         B_in: in STD_LOGIC_VECTOR (3 downto 0);
         S: in STD_LOGIC;
```

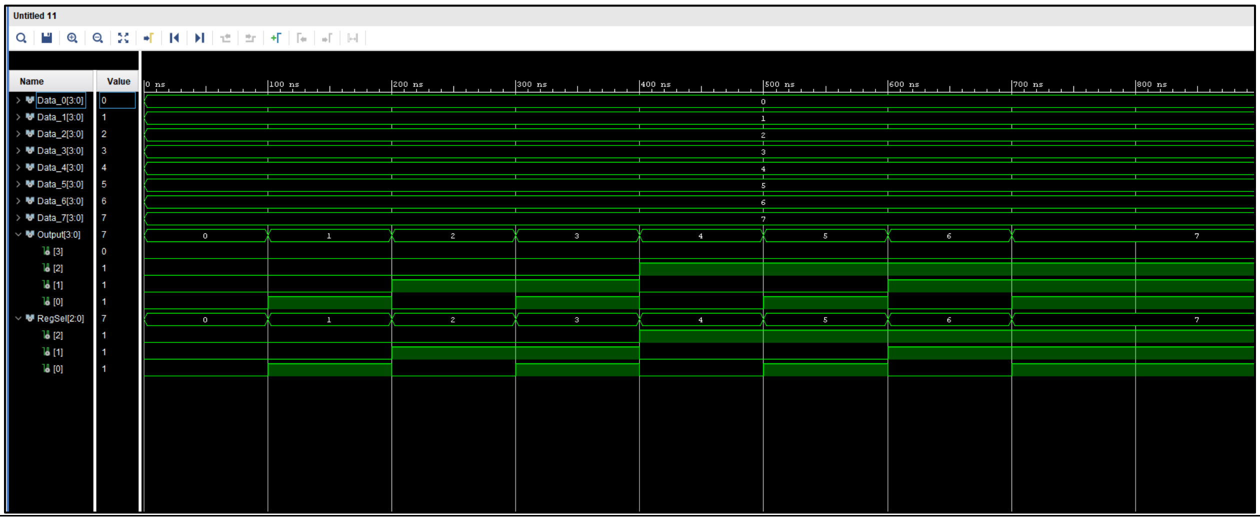
```
            C_out: out STD_LOGIC_VECTOR (3 downto 0));
        END COMPONENT;
        SIGNAL C_out_0_0, C_out_0_1, C_out_0_2, C_out_0_3, C_out_1_0, C_out_1_1:
    STD_LOGIC_VECTOR (3 downto 0);     -- Intermediate signals for multiplexers

    begin
        -- First stage of multiplexers
        Mux_2_Way_4_Bit_0_0: Mux_2_Way_4_Bit
        Port Map (A_in => Data_0,
                  B_in => Data_1,
                  S => RegSel(0),
                  C_out => C_out_0_0);

        Mux_2_Way_4_Bit_0_1: Mux_2_Way_4_Bit
        Port Map (A_in => Data_2,
                  B_in => Data_3,
                  S => RegSel(0),
                  C_out => C_out_0_1);

        Mux_2_Way_4_Bit_0_2: Mux_2_Way_4_Bit
        Port Map (A_in => Data_4,
                  B_in => Data_5,
                  S => RegSel(0),
                  C_out => C_out_0_2);

        Mux_2_Way_4_Bit_0_3: Mux_2_Way_4_Bit
        Port Map (A_in => Data_6,
                  B_in => Data_7,
                  S => RegSel(0),
                  C_out => C_out_0_3);

        -- Second stage of multiplexers
        Mux_2_Way_4_Bit_1_0: Mux_2_Way_4_Bit
        Port Map (A_in => C_out_0_0,
                  B_in => C_out_0_1,
                  S => RegSel(1),
                  C_out => C_out_1_0);

        Mux_2_Way_4_Bit_1_1: Mux_2_Way_4_Bit
        Port Map (A_in => C_out_0_2,
                  B_in => C_out_0_3,
                  S => RegSel(1),
                  C_out => C_out_1_1);

        -- Third stage of multiplexers
        Mux_2_Way_4_Bit_2_0: Mux_2_Way_4_Bit
        Port Map (A_in => C_out_1_0,
                  B_in => C_out_1_1,
                  S => RegSel(2),
                  C_out => Output);


    end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/22/2024 11:13:13 PM
-- Design Name: 
-- Module Name: TB_Mux_8_W_4 - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Mux_8_W_4 is
--  Port ( );
end TB_Mux_8_W_4;

architecture Behavioral of TB_Mux_8_W_4 is

component Mux_8_Way_4_Bit is
    Port (Data_0, Data_1, Data_2, Data_3, Data_4, Data_5, Data_6, Data_7  : 
in STD_LOGIC_VECTOR (3 downto 0);
          RegSel : in STD_LOGIC_VECTOR (2 downto 0);
          Output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

signal 
Data_0,Data_1,Data_2,Data_3,Data_4,Data_5,Data_6,Data_7:STD_LOGIC_VECTOR(3 
downto 0);
```

```vhdl
signal Output:STD_LOGIC_VECTOR(3 downto 0);
signal RegSel:STD_LOGIC_VECTOR(2 downto 0);

begin

    UUT:Mux_8_Way_4_Bit Port map(Data_0, Data_1, Data_2, Data_3, Data_4,
Data_5, Data_6, Data_7, RegSel, Output);

    main: process begin

        Data_0<="0000";
        Data_1<="0001";
        Data_2<="0010";
        Data_3<="0011";
        Data_4<="0100";
        Data_5<="0101";
        Data_6<="0110";
        Data_7<="0111";

        -- Switching between busses.
        RegSel<="000";
        WAIT FOR 100 ns;
        RegSel<="001";
        WAIT FOR 100 ns;
        RegSel<="010";
        WAIT FOR 100 ns;
        RegSel<="011";
        WAIT FOR 100 ns;
        RegSel<="100";
        WAIT FOR 100 ns;
        RegSel<="101";
        WAIT FOR 100 ns;
        RegSel<="110";
        WAIT FOR 100 ns;
        RegSel<="111";
        WAIT;

    end process;
end Behavioral;
```

Timing Diagram

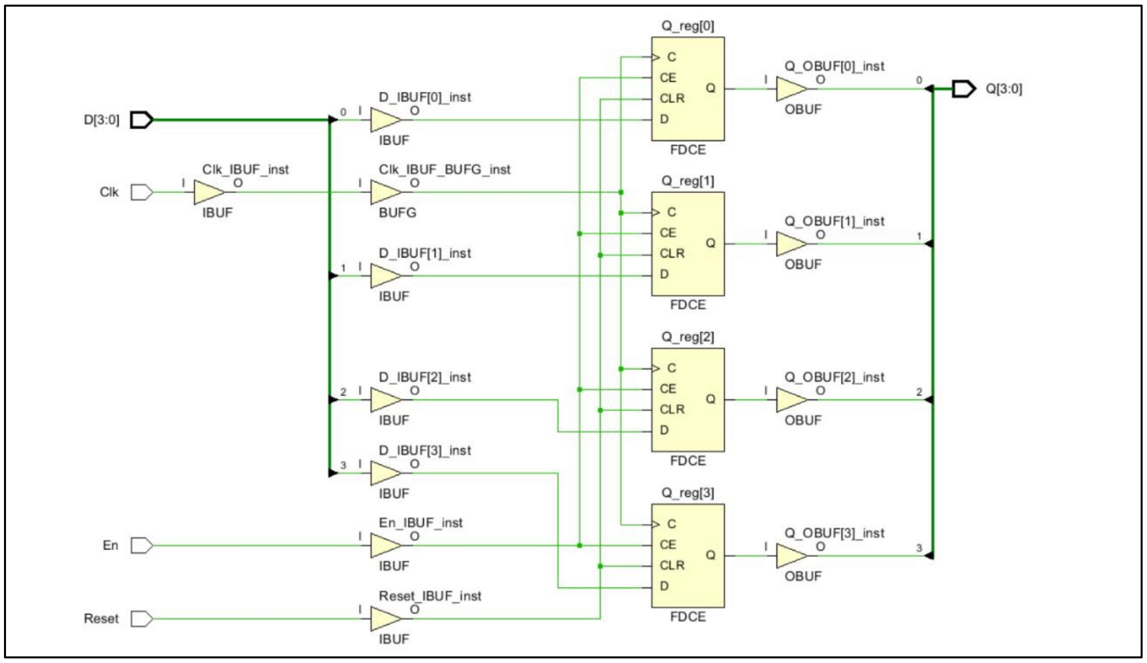


Name	Value
Data_0[3:0]	0
Data_1[3:0]	1
Data_2[3:0]	2
Data_3[3:0]	3
Data_4[3:0]	4
Data_5[3:0]	5
Data_6[3:0]	6
Data_7[3:0]	7
Output[3:0]	7
[3]	0
[2]	1
[1]	1
[0]	1
RegSel[2:0]	7
[2]	1
[1]	1
[0]	1

Register

Register employs the logic of a D flip-flop.

Elaborated Design Schematic



Design Source File

```
------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/12/2024 02:01:31 PM
-- Design Name:
-- Module Name: Reg - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
```

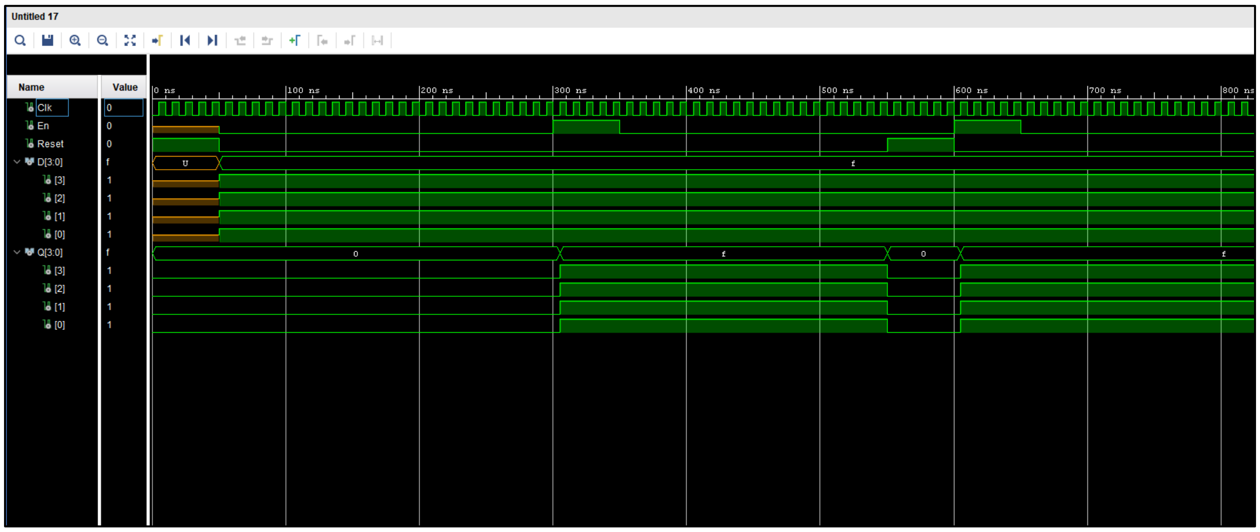
```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of Reg is

begin
process (Clk, Reset) begin
    if (Reset = '1') then                   -- Reset condition
        Q <= "0000";                        -- Reset the register
    else if (rising_edge(Clk)) then         -- Respond when clock rises
        if En = '1' then                    -- Check if enable signal is set
            Q <= D;                         -- Update output with input data
        end if;
    end if;
    end if;
end process;
end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/23/2024 10:55:35 AM
-- Design Name:
```

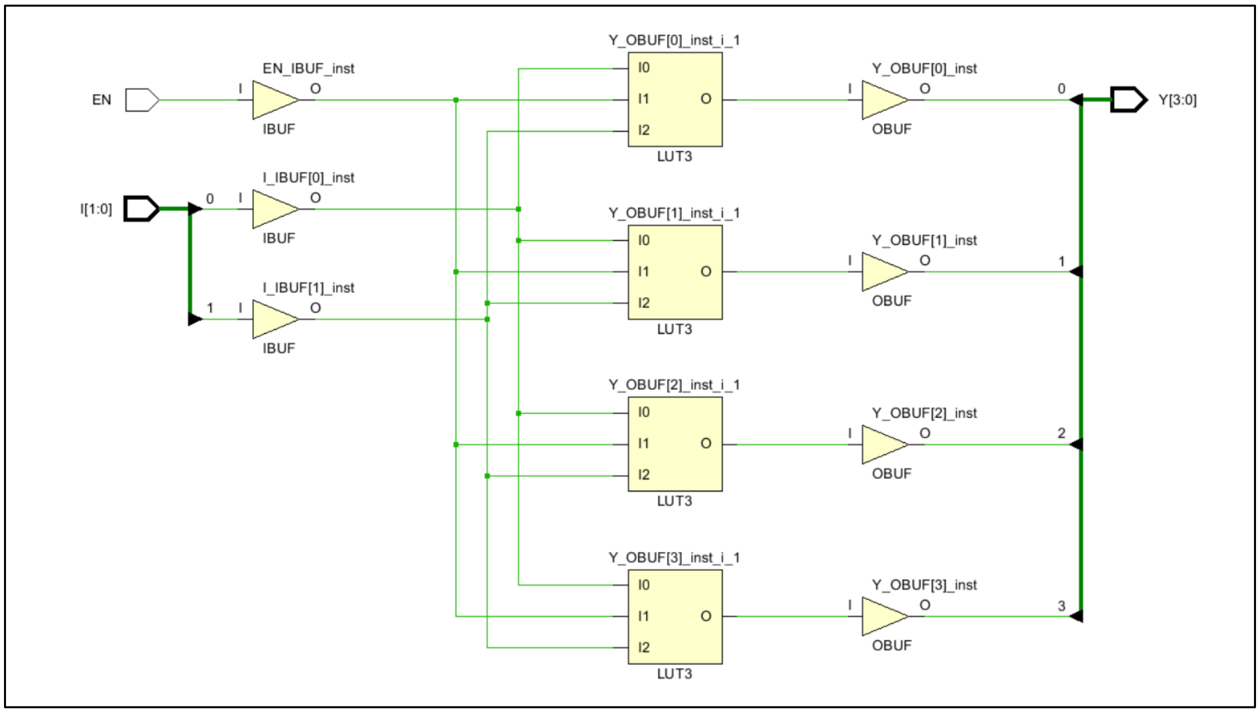
```vhdl
-- Module Name: TB_Reg_4_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Reg_4_Bit is
--  Port ( );
end TB_Reg_4_Bit;

architecture Behavioral of TB_Reg_4_Bit is

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

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

begin

    UUT : Reg PORT MAP (D, Reset, En, Clk, Q);

    clock : process
    begin
        Clk <= '0';
        wait for 5ns;
        Clk <= '1';
        wait for 5ns;
    end process;
    main : process begin

        Reset <= '1';    -- Reset
        wait for 50ns;
        Reset <= '0';

        En <= '0';
        D <= "1111";
        wait for 50ns;

        wait for 200ns;

        En <= '1';
        D <= "1111";
        wait for 50ns;
        En <= '0';

        wait for 200ns;

        Reset <= '1';    -- Reset
        wait for 50ns;
        Reset <= '0';

        En <= '1';
        D <= "1111";
        wait for 50ns;
        En <= '0';

        wait;
    end process;
end Behavioral;
```

Timing Diagram



2 to 4 Decoder

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/20/2024 02:00:48 PM
-- Design Name:
-- Module Name: Decoder_2_to_4 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

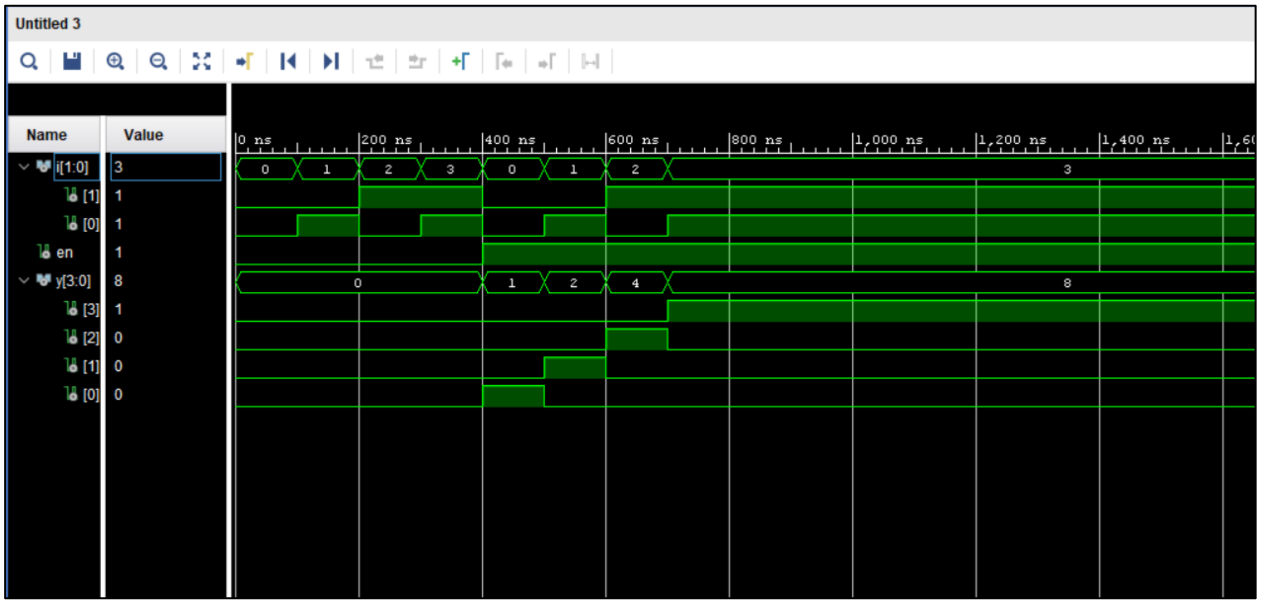
```
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Decoder_2_to_4 is
    Port ( I : in STD_LOGIC_VECTOR (1 downto 0);    -- Input vector with 2
bits
           EN : in STD_LOGIC;                        -- Enable input
           Y : out STD_LOGIC_VECTOR (3 downto 0));  -- Output vector with 4
bits
end Decoder_2_to_4;

architecture Behavioral of Decoder_2_to_4 is

begin

    Y(0) <= EN AND NOT I(0) AND NOT I(1);           -- Output Y(0) is enabled
if EN is high and both input bits are low
    Y(1) <= EN AND I(0) AND NOT I(1);               -- Output Y(1) is enabled
if EN is high, I(0) is high, and I(1) is low
    Y(2) <= EN AND NOT I(0) AND I(1);               -- Output Y(2) is enabled
if EN is high, I(0) is low, and I(1) is high
    Y(3) <= EN AND I(0) AND I(1);                   -- Output Y(3) is enabled
if EN is high and both input bits are high


end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/20/2024 02:19:47 PM
-- Design Name:
-- Module Name: TB_Decoder_2_to_4 - Behavioral
-- Project Name:
```

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


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Decoder_2_to_4 is
--  Port ( );
end TB_Decoder_2_to_4;

architecture Behavioral of TB_Decoder_2_to_4 is
    COMPONENT Decoder_2_to_4
    PORT (I: In std_logic_vector(1 downto 0);
          EN: In std_logic;
          Y: Out std_logic_vector(3 downto 0));
    END COMPONENT;

    SIGNAL i: std_logic_vector(1 downto 0);
    SIGNAL en: std_logic;
    SIGNAL y: std_logic_vector(3 downto 0);

begin
    UUT: Decoder_2_to_4 PORT MAP(
        I => i,
        EN => en,
        Y => y);

process
begin
        en <= '0';
        i(0) <= '0';
        i(1) <= '0';

        WAIT for 100ns;
        i(0) <= '1';

        WAIT for 100ns;
        i(1) <= '1';
        i(0) <= '0';

        WAIT for 100ns;
        i(0) <= '1';

        WAIT for 100ns;
        en <= '1';
        i(0) <= '0';
        i(1) <= '0';

        WAIT for 100ns;
        i(0) <= '1';

        WAIT for 100ns;
        i(1) <= '1';
        i(0) <= '0';

        WAIT for 100ns;
        i(0) <= '1';

        WAIT;

end process;
end Behavioral;
```

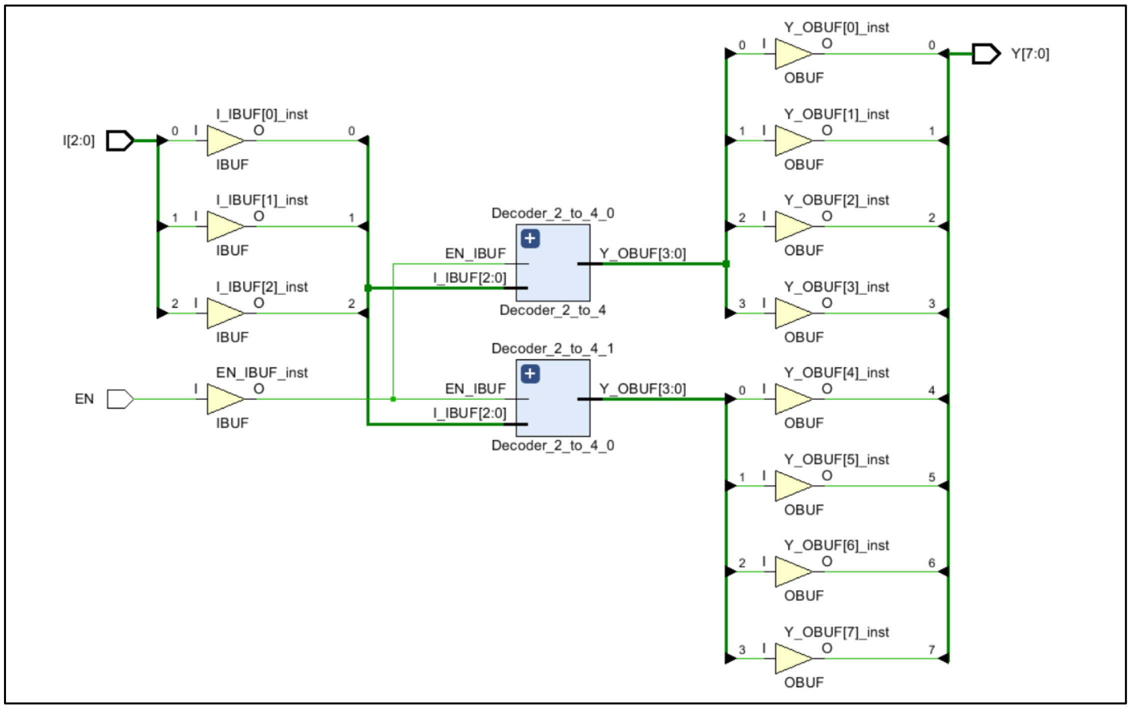
Timing Diagram



3 to 8 Decoder

Three to eight decoder is implemented by integrating 2, two to four decoders.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company: 
-- Engineer: 
-- 
-- Create Date: 02/20/2024 03:36:30 PM
-- Design Name: 
-- Module Name: Decoder_3_to_8 - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
```

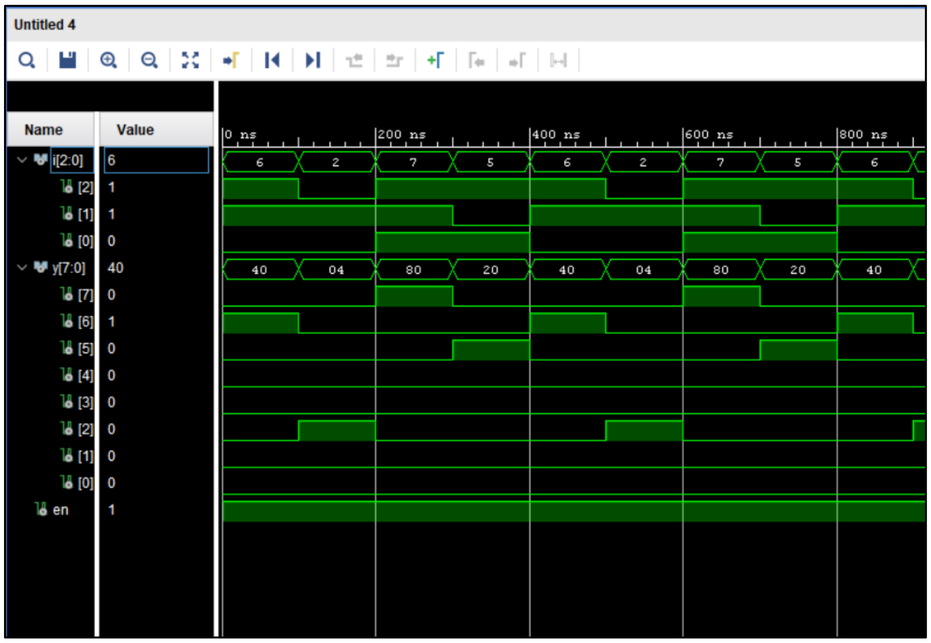
```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Decoder_3_to_8 is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);    -- Input vector with 3
bits
           EN : in STD_LOGIC;                       -- Enable input
           Y : out STD_LOGIC_VECTOR (7 downto 0));  -- Input vector with 8
bits
end Decoder_3_to_8;

architecture Behavioral of Decoder_3_to_8 is
    COMPONENT Decoder_2_to_4                        -- Instantiating a 2 to 4
decoder component
        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);  -- Signals for splitting
input
     SIGNAL Y0,Y1 : std_logic_vector (3 downto 0);  -- Signals for storing
outputs of the first and second decoders
     SIGNAL EN0,EN1,I2 : std_logic;                 -- Signals for enabling
the first and second decoders

begin
    Decoder_2_to_4_0: Decoder_2_to_4                -- Instantiation of the
first 2 to 4 decoder
    PORT MAP (I => I0,
              EN => EN0,
              Y => Y0);

    Decoder_2_to_4_1: Decoder_2_to_4                -- Instantiation of the
second 2 to 4 decoder
    PORT MAP (I => I1,
              EN => EN1,
              Y => Y1);

    -- Enable signals for the first and second decoders
    EN0 <= NOT(I(2)) AND EN;
    EN1 <= I(2) AND EN;

    -- Splitting input I into two parts for the two decoders
    I0 <= I(1 downto 0);
    I1 <= I(1 downto 0);
    I2 <= I(2);

    Y(3 downto 0) <= Y0;                          -- Assigning the lower 4
bits of the output from the first decoder
    Y(7 downto 4) <= Y1;                          -- Assigning the upper 4
bits of the output from the second decoder

end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/21/2024 11:22:50 AM
-- Design Name:
-- Module Name: TB_Decoder_3_to_8 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Decoder_3_to_8 is
```

```
--  Port ( );
end TB_Decoder_3_to_8;

architecture Behavioral of TB_Decoder_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 y : std_logic_vector(7 downto 0);
 SIGNAL en: std_logic;

begin
    UUT: Decoder_3_to_8
    PORT MAP(
        I => i,
        EN => en,
        Y => y);

    process
    begin
        en <= '1';
        --Output 110
        i <= "110"; WAIT FOR 100ns;
        --Output 010
        i <= "010"; WAIT FOR 100ns;
        --Output 111
        i <= "111"; WAIT FOR 100ns;
        --Output 101
        i <= "101"; WAIT FOR 100ns;
    end process;
end Behavioral;
```

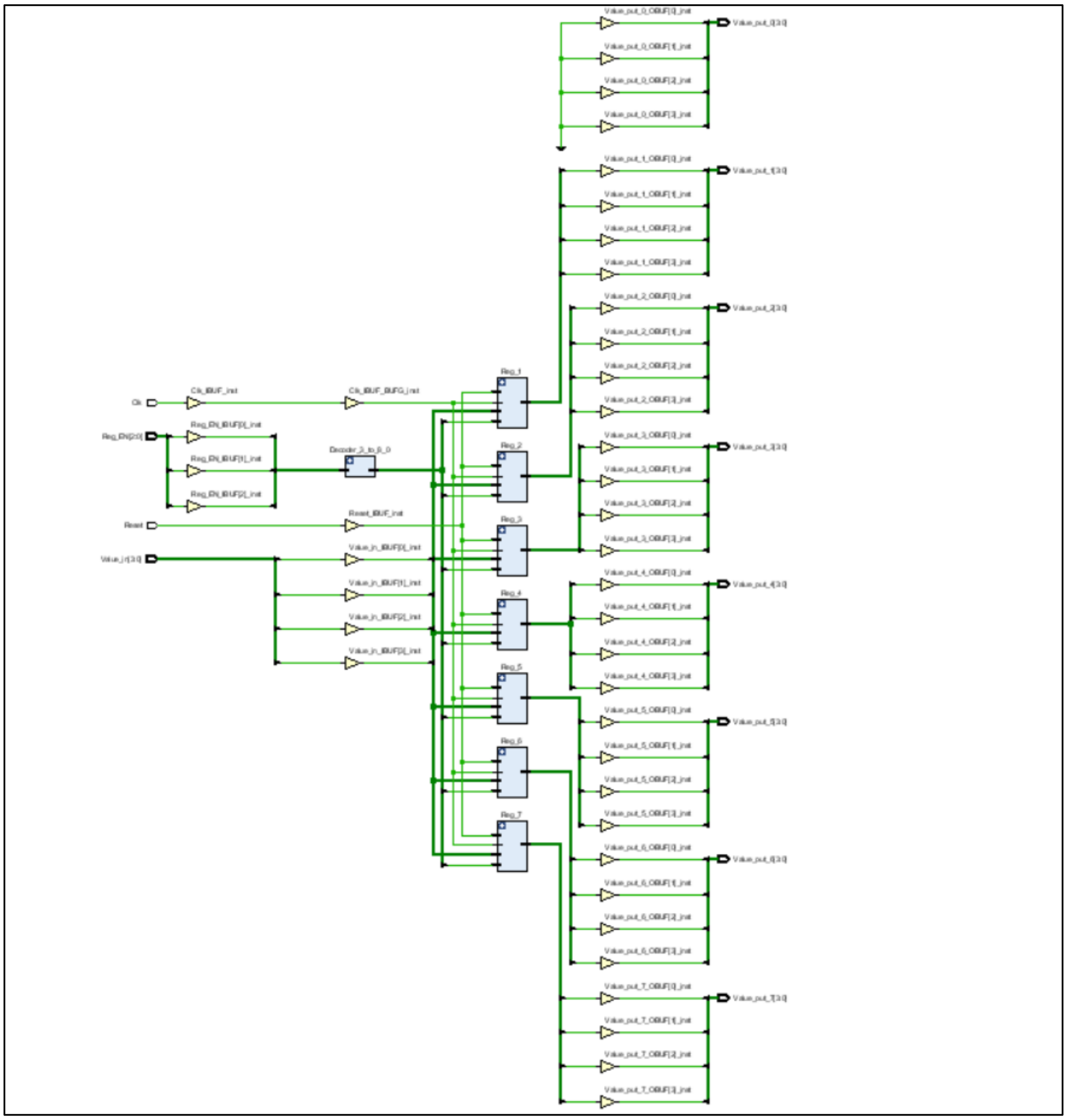
Timing Diagram



Register Bank

8 registers and 1, three to eight decoders are assembled to create the register bank.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
------
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/13/2024 05:05:19 PM
-- Design Name: 
-- Module Name: Register_Bank - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------
------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Register_Bank is
    Port (
        Value_in    : in  Std_Logic_Vector (3 downto 0);  -- Input values to
be written into registers
        Reg_EN      : in  Std_Logic_Vector (2 downto 0);  -- Register enable
signals
        Clk         : in  Std_Logic;                      -- Clock signal
        Reset       : in  Std_Logic;                      -- Reset signal
        Value_out_0 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 0
        Value_out_1 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 1
        Value_out_2 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 2
        Value_out_3 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 3
```

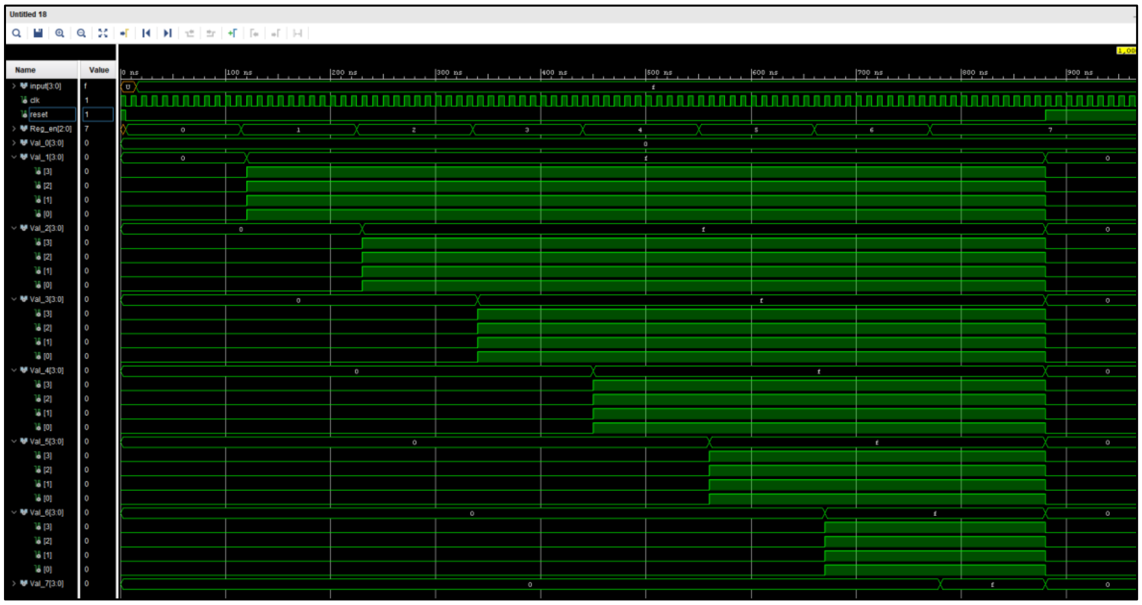
```
        Value_out_4 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 4
        Value_out_5 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 5
        Value_out_6 : out Std_Logic_Vector (3 downto 0);  -- Output value of
register 6
        Value_out_7 : out Std_Logic_Vector (3 downto 0)   -- Output value of
register 7
    );
end Register_Bank;

architecture Behavioral of Register_Bank is
    -- Component declarations
    Component Decoder_3_to_8
        Port (
            I  : in  STD_LOGIC_VECTOR (2 downto 0);   -- Input select lines
            EN : in  STD_LOGIC;                       -- Enable signal
            Y  : out STD_LOGIC_VECTOR (7 downto 0)    -- Output selection
lines
        );
    End Component;

    Component Reg
        Port (
            D     : in  STD_LOGIC_VECTOR (3 downto 0);  -- Input data
            Reset : in  STD_LOGIC;                      -- Reset signal
            En    : in  STD_LOGIC;                      -- Enable signal
            Clk   : in  STD_LOGIC;                      -- Clock signal
            Q     : out STD_LOGIC_VECTOR (3 downto 0)   -- Output data
        );
    End Component;

    -- Internal signal declaration
    Signal Y0 : Std_Logic_Vector (7 downto 0);

begin
    -- Decoder instantiation
    Decoder_3_to_8_0 : Decoder_3_to_8
    Port Map (
        I  => Reg_EN,   -- Input select lines from Reg_EN
        EN => '1',      -- Enable signal is always high
        Y  => Y0        -- Output selection lines connected to Y0
    );

    -- Register instantiations
    Reg_0 : Reg
    Port Map (
        D     => "0000",  -- Register 0 value is always 0000 and it's a read-
only register
        Reset => Reset,   -- Reset signal
        En    => Y0(0),   -- Enable signal from Y0(0)
        Clk   => Clk,     -- Clock signal
        Q     => Value_out_0  -- Output value of register 0
    );

    Reg_1 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(1),     -- Enable signal from Y0(1)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_1  -- Output value of register 1
    );

    Reg_2 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(2),     -- Enable signal from Y0(2)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_2  -- Output value of register 2
    );

    Reg_3 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(3),     -- Enable signal from Y0(3)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_3  -- Output value of register 3
    );

    Reg_4 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(4),     -- Enable signal from Y0(4)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_4  -- Output value of register 4
    );

    Reg_5 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(5),     -- Enable signal from Y0(5)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_5  -- Output value of register 5
    );

    Reg_6 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(6),     -- Enable signal from Y0(6)
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_6  -- Output value of register 6
    );

    Reg_7 : Reg
    Port Map (
        D     => Value_in,  -- Input data from Value_in
        Reset => Reset,     -- Reset signal
        En    => Y0(7),     -- Enable signal from Y0(7)
```

```
        Clk   => Clk,       -- Clock signal
        Q     => Value_out_7  -- Output value of register 7
    );

end Behavioral;
```

Simulation Source File

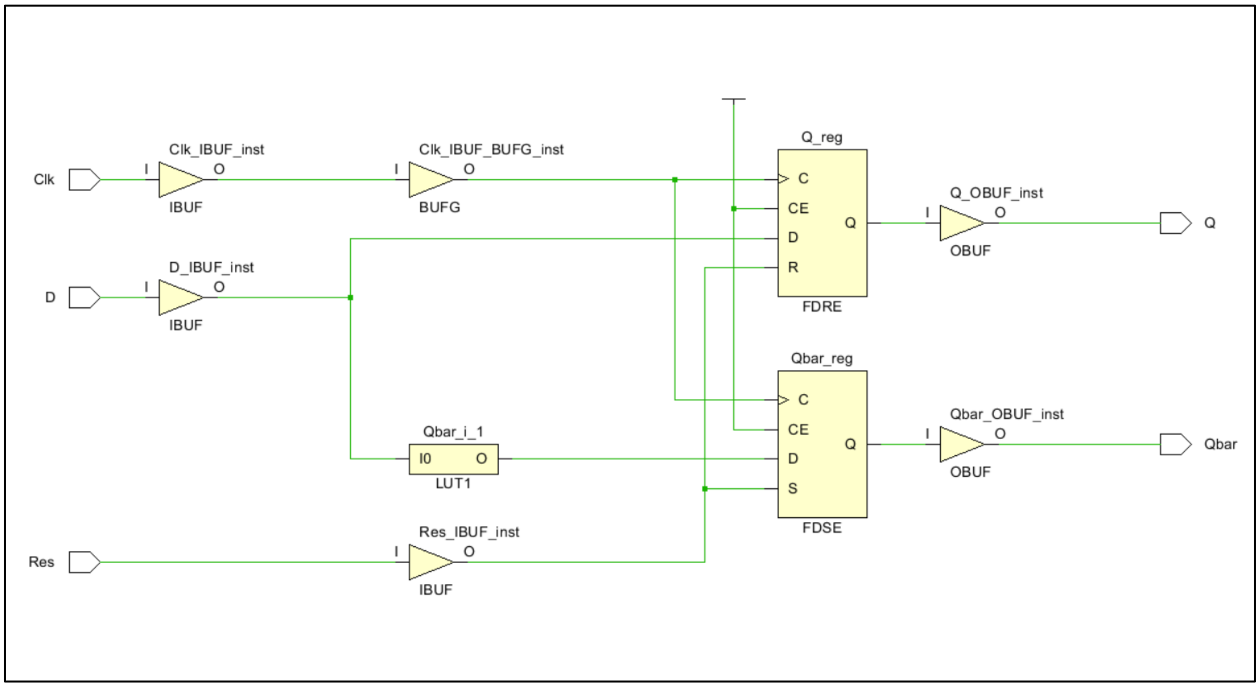
```
----------------------------------------------------------------------------
-----
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/23/2024 01:38:56 PM
-- Design Name: 
-- Module Name: TB_Register_Bank - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of TB_Register_Bank is

component Register_Bank is
    Port (Value_in: in Std_Logic_Vector (3 downto 0);
          Reg_EN: in Std_Logic_Vector (2 downto 0);
          Clk: in Std_Logic;
          Reset: in Std_Logic;
          Value_out_0: Out Std_Logic_Vector (3 downto 0);
          Value_out_1: Out Std_Logic_Vector (3 downto 0);
          Value_out_2: Out Std_Logic_Vector (3 downto 0);
          Value_out_3: Out Std_Logic_Vector (3 downto 0);
          Value_out_4: Out Std_Logic_Vector (3 downto 0);
          Value_out_5: Out Std_Logic_Vector (3 downto 0);
          Value_out_6: Out Std_Logic_Vector (3 downto 0);
          Value_out_7: Out Std_Logic_Vector (3 downto 0));
end component;

signal input : STD_LOGIC_VECTOR(3 downto 0);
signal clk,reset : STD_LOGIC := '0';
signal Reg_en : STD_LOGIC_VECTOR(2 downto 0);
signal Val_0,Val_1,Val_2,Val_3,Val_4,Val_5,Val_6,Val_7 : STD_LOGIC_VECTOR(3
downto 0);

begin

    UUT: Register_Bank PORT MAP (input, Reg_en, clk, reset, Val_0, Val_1,
Val_2, Val_3, Val_4, Val_5, Val_6, Val_7);

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

    main: process begin

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

        Reg_en <= "000";
        wait for 10 ns;     -- Testing the read only register.
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "001";
        wait for 10 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "010";
        wait for 10 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "011";
        wait for 5 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "100";
        wait for 10 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "101";
        wait for 10 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "110";
        wait for 10 ns;
        input <= "1111";

        wait for 100 ns;
        Reg_en <= "111";
        wait for 10 ns;
        input <= "1111";

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

        wait;
  end process;
end Behavioral;
```

Timing Diagram



D Flip-Flop

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/05/2024 02:04:48 PM
-- Design Name:
-- Module Name: D_FF - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

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


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of D_FF is

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


end Behavioral;
```

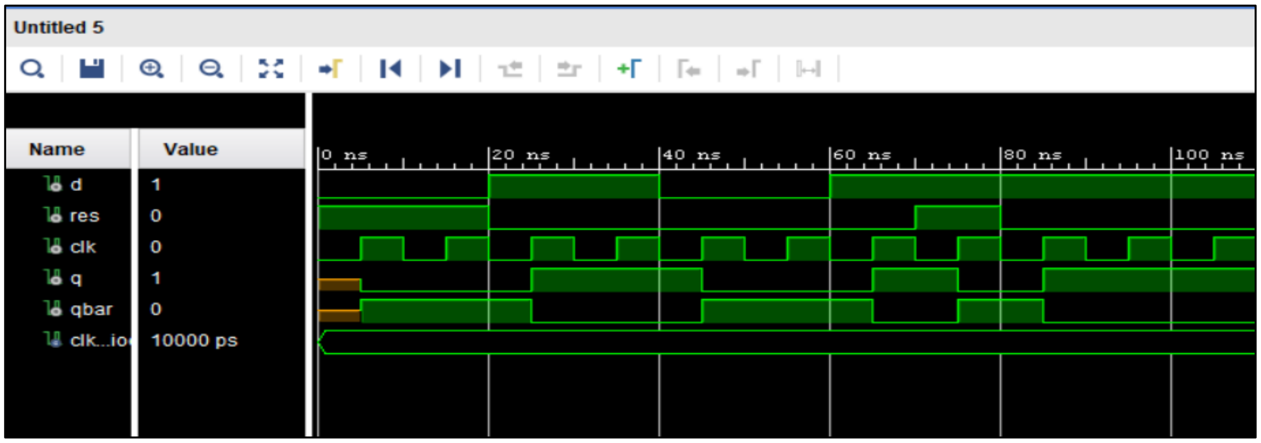
Simulation Source File

```
-----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/05/2024 02:12:26 PM
-- Design Name:
```

```vhdl
-- Module Name: D_FF_Sim - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of D_FF_Sim is
    Component D_FF
    Port (  D: in STD_LOGIC;
            Res: in STD_LOGIC;
            Clk: in STD_LOGIC;
            Q: out STD_LOGIC;
            Qbar: out STD_LOGIC );
    End Component;
    Signal d : STD_LOGIC := '0';
    Signal res: STD_LOGIC := '0';
    Signal clk: STD_LOGIC := '0';
    Signal q: STD_LOGIC;
    Signal qbar: STD_LOGIC;
    Constant clk_period: time := 10ns;      --Clock period definition

begin
    UUT: D_FF Port Map
        (   D => d,
            Res => res,
            Clk => clk,
            Q => q,
            Qbar => qbar );

    Clk_process: process
        begin
```

```
            clk <= '0';
            Wait for clk_period/2;
            clk <= '1';
            Wait for clk_period/2;
        end process;

    process
        begin
        -- Reset
        res <= '1';
        Wait for 20 ns;
        res <= '0';

        -- Apply inputs
        d <= '1';
        Wait for 20 ns;
        d <= '0';
        Wait for 20 ns;
        d <= '1';
        Wait for 10 ns;


        res <= '1';
        wait for 10 ns;
        res <= '0';

        Wait for 50 ns;
        Wait; --
    end process;

end Behavioral;
```

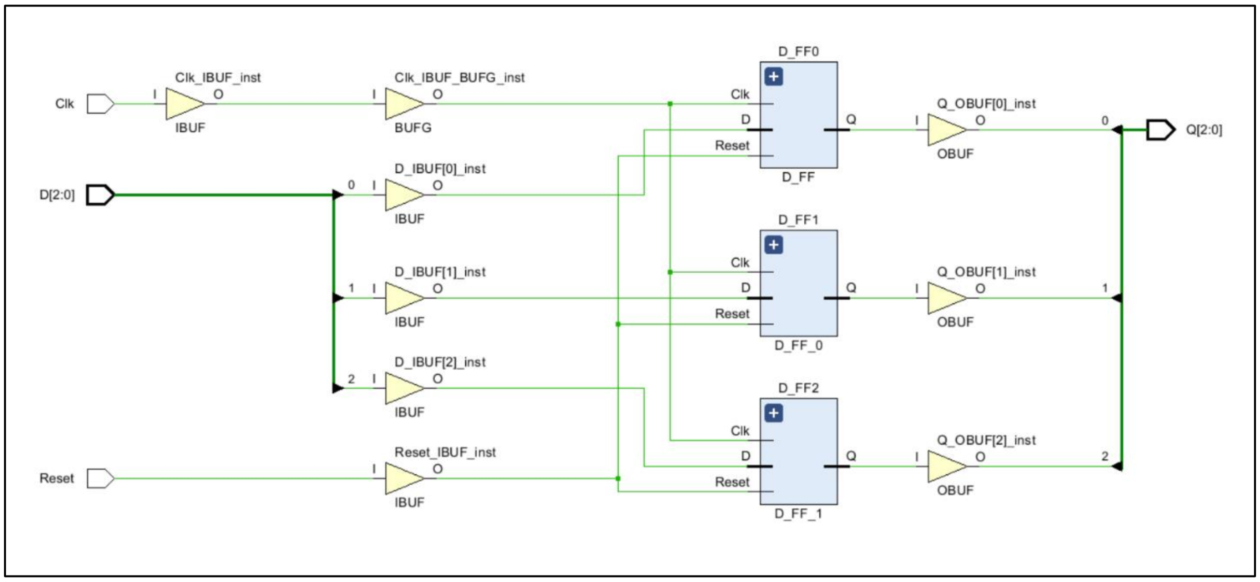
Timing Diagram



Program Counter

Program counter is implemented by using 3, D flip-flops.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/20/2024 08:10:32 PM
-- Design Name:
-- Module Name: Program_Counter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

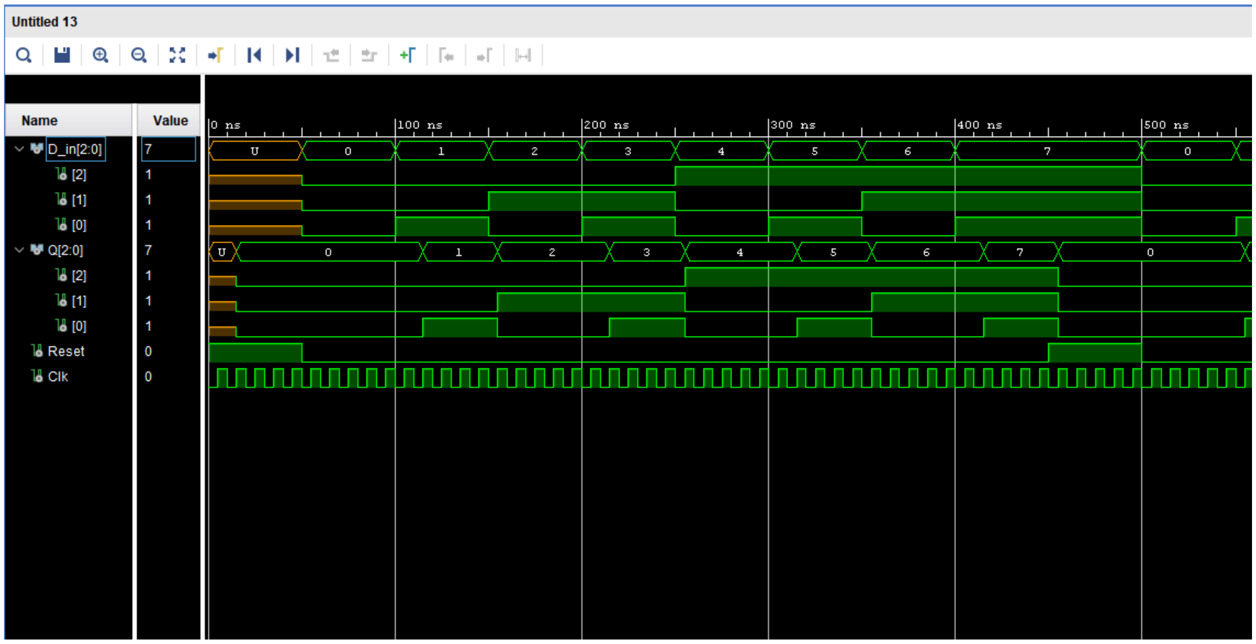
```
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of Program_Counter is
    Component D_FF
    Port ( D: in STD_LOGIC;
           Res: in STD_LOGIC;
           Clk: in STD_LOGIC;
           Q: out STD_LOGIC;
           Qbar: out STD_LOGIC);
End Component;

begin
    D_FF0: D_FF
        Port Map ( D => D(0),
                   Res => Reset,
                   Clk => Clk,
                   Q => Q(0));

    D_FF1: D_FF
        Port Map ( D => D(1),
                   Res => Reset,
                   Clk => Clk,
                   Q => Q(1));

    D_FF2: D_FF
        Port Map ( D => D(2),
                   Res => Reset,
                   Clk => Clk,
                   Q => Q(2) );

end Behavioral;
```

Simulation Source File

```vhdl
----------------------------------------------------------------------------------
-----
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/23/2024 10:19:41 AM
-- Design Name: 
-- Module Name: TB_Program_Counter - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of TB_Program_Counter is

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

SIGNAL D_in, Q : STD_LOGIC_VECTOR(2 downto 0);
SIGNAL Reset, Clk : STD_LOGIC := '0';

begin

    UUT : Program_Counter PORT MAP(D_in, Reset, Clk, Q);

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

    main: process begin

            Reset <= '1';
            wait for 50ns;   -- Reset 1
            Reset <= '0';

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

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

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

            D_in <= "011";
            wait for 50ns;

            D_in <= "100";
            wait for 50ns;

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

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

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

            Reset <= '1';
            wait for 50ns;   -- Reset 2
            Reset <= '0';

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

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

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

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

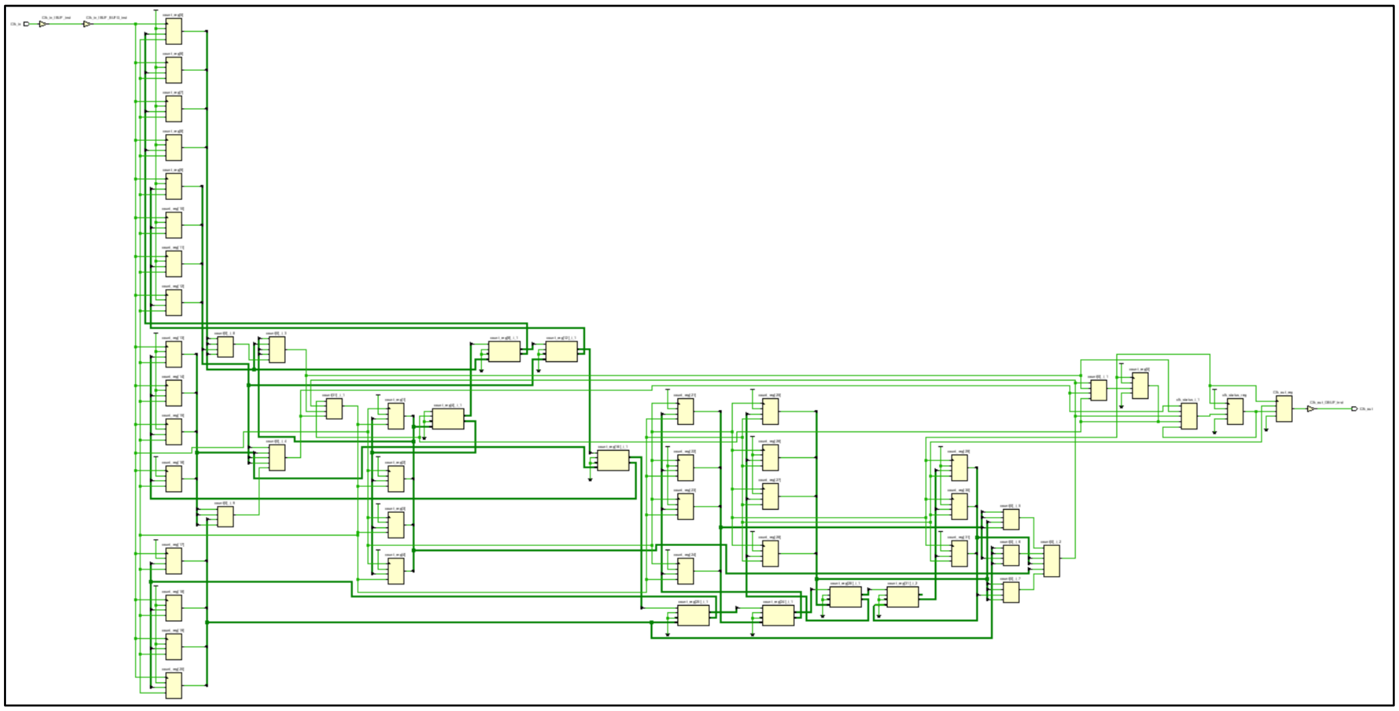
```
                D_in <= "100";
                wait for 50ns;

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

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

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

                wait;
        end process;
end Behavioral;
```

Timing Diagram



Slow Clock

Elaborated Design Schematic



Design Source File

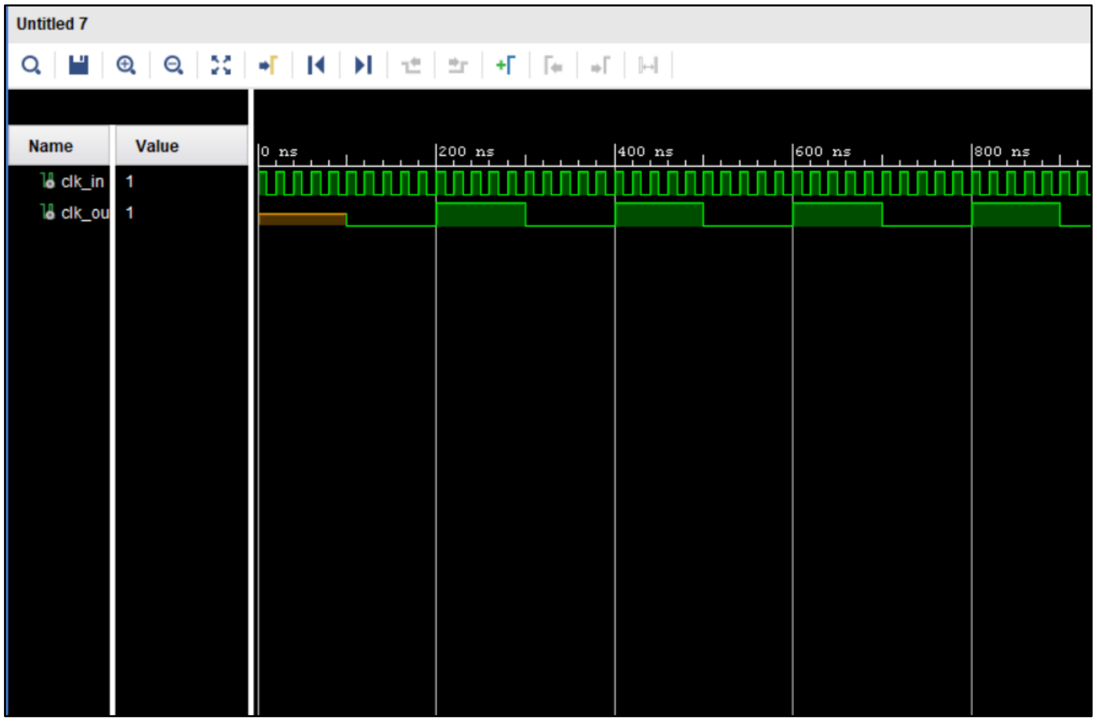
```
-------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/05/2024 03:09:56 PM
-- Design Name:
-- Module Name: Slow_Clk - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

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


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;                  -- Input clock signal
           Clk_out : out STD_LOGIC);               -- Output clock signal
end Slow_Clk;

architecture Behavioral of Slow_Clk is
    Signal count: integer := 1;                    -- Counter for dividing the
clock frequency
    Signal clk_status: STD_LOGIC := '0';           -- Initial clock status

begin

    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <= count + 1;                    --Increment counter
            if (count = 1) then             --200000000 when hardware
implementation
                clk_status <= not clk_status;   --Invert clock status
                Clk_out <= clk_status;
                count <= 1;                        --Reset counter
            end if;
        end if;
    end process;

end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 03/05/2024 03:23:28 PM
-- Design Name:
-- Module Name: Slow_Clk_Sim - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of Slow_Clk_Sim is
    Component Slow_Clk
    Port ( Clk_in: in STD_LOGIC;
           Clk_out: out STD_LOGIC);
    End Component;
    Signal clk_in: STD_LOGIC;
    Signal clk_out: STD_LOGIC;
begin
    UUT: Slow_Clk Port Map
        ( Clk_in => clk_in,
          Clk_out => clk_out );
    Process Begin
        clk_in <= '1';
        Wait for 10 ns;
        clk_in <= '0';
        Wait for 10 ns;
    End process;

end Behavioral;
```

Timing Diagram



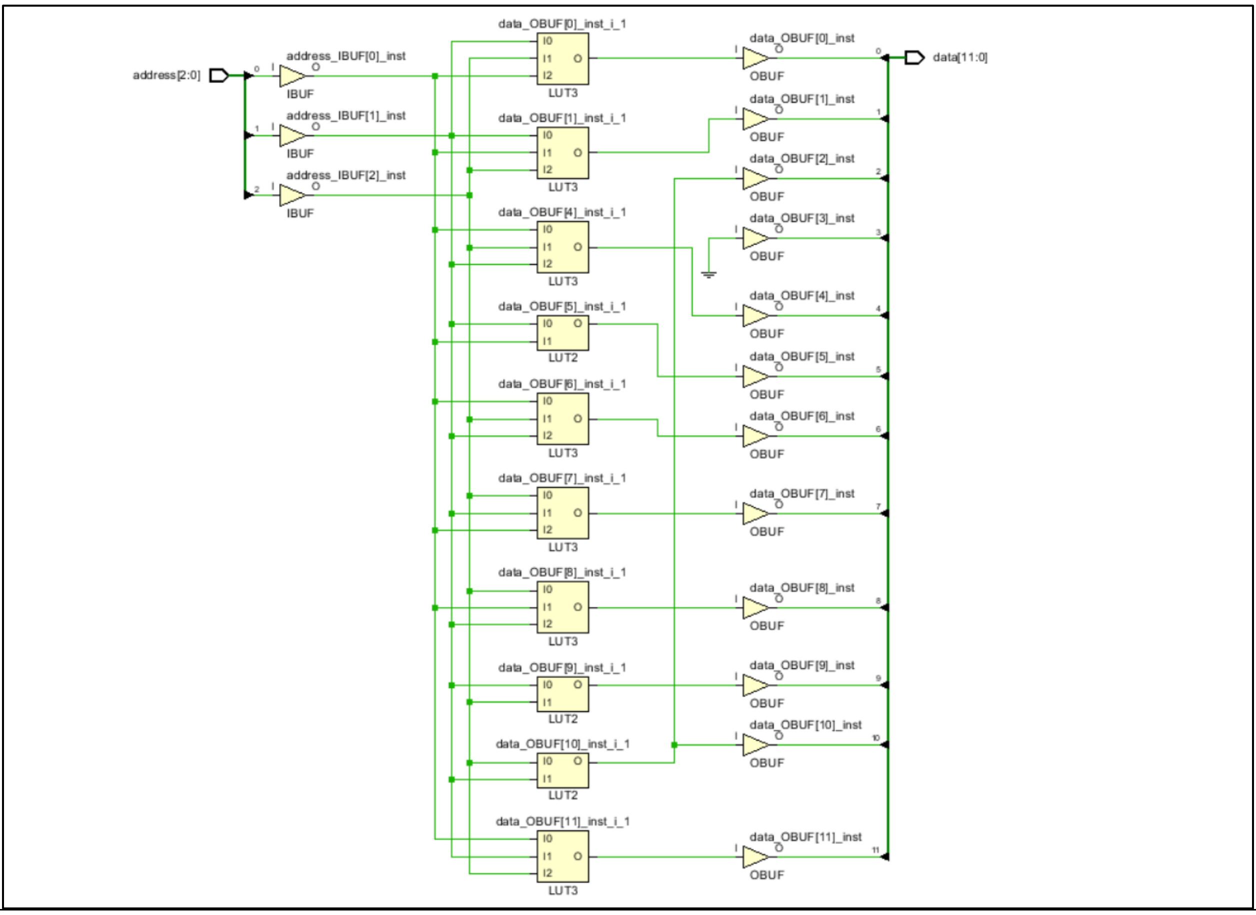
Program ROM

We've designed this program ROM to accommodate seven instructions.

1. MOVI R7, 1 - Move immediate value 1 to register R7.
2. MOVI R6, 2 - Move immediate value 2 to register R6.
3. MOVI R5, 3 - Move immediate value 3 to register R5.
4. ADD R7, R6 - Add values in the registers R7 and R6 and store the result in R7.
5. ADD R7, R5 - Add values in the registers R7 and R5 and store the result in R7.
6. MOVI R0, 0 - Move immediate value 1 to register R7.
7. JZR R0, 6 – Jump to instruction 6 if the value in the R0 is zero.

The final two instructions are intended to ensure that value in the R7 register remains constant.

Elaborated Design Schematic



Design Source File

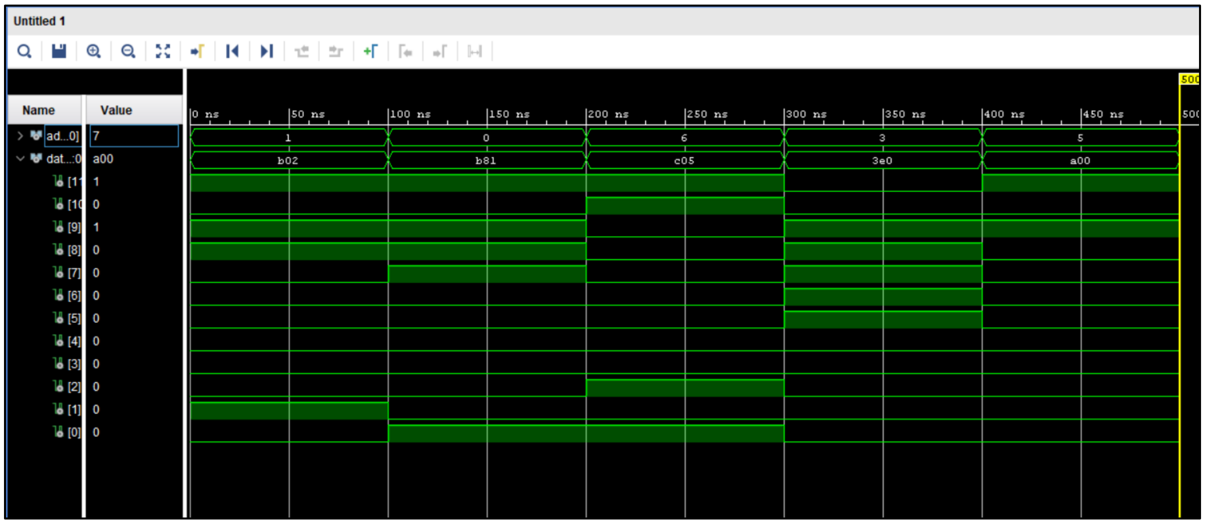
```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/18/2024 06:03:31 PM
-- Design Name:
-- Module Name: LUT - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


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

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

entity LUT is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);     -- Address input
for the LUT
           data : out STD_LOGIC_VECTOR (11 downto 0));    -- Output data from
the LUT
end LUT;

architecture Behavioral of LUT is
    -- Type declaration for ROM data
    type rom_type is array (0 to 6) of std_logic_vector(11 downto 0);

    -- Initialization of instruction ROM data
    signal instruction_ROM : rom_type := (
        "101110000001", -- MOVI R7, 1
        "101100000010", -- MOVI R6, 2
        "101010000011", -- MOVI R5, 3
        "001111100000", -- ADD R7, R6
        "001111010000", -- ADD R7, R5
        "101000000000", -- MOVI R0, 0
        "110000000101"  -- JZR R0, 6
    );

begin
    -- Output data based on address input
    data <= instruction_ROM(to_integer(unsigned(address)));
end Behavioral;
```

Simulation Source File

```
------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/25/2024 10:41:17 AM
-- Design Name:
-- Module Name: TB_LUT - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

```
architecture Behavioral of TB_LUT is
    COMPONENT LUT
        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
        PORT MAP (Address => address,
                  Data => data);

    Process Begin
        address <= "001";     Wait for 100ns;
        address <= "000";     Wait for 100ns;
        address <= "110";     Wait for 100ns;
        address <= "011";     Wait for 100ns;
        address <= "101";     Wait for 100ns;
        address <= "111";     Wait for 100ns;
    End Process;

end Behavioral;
```

Timing Diagram

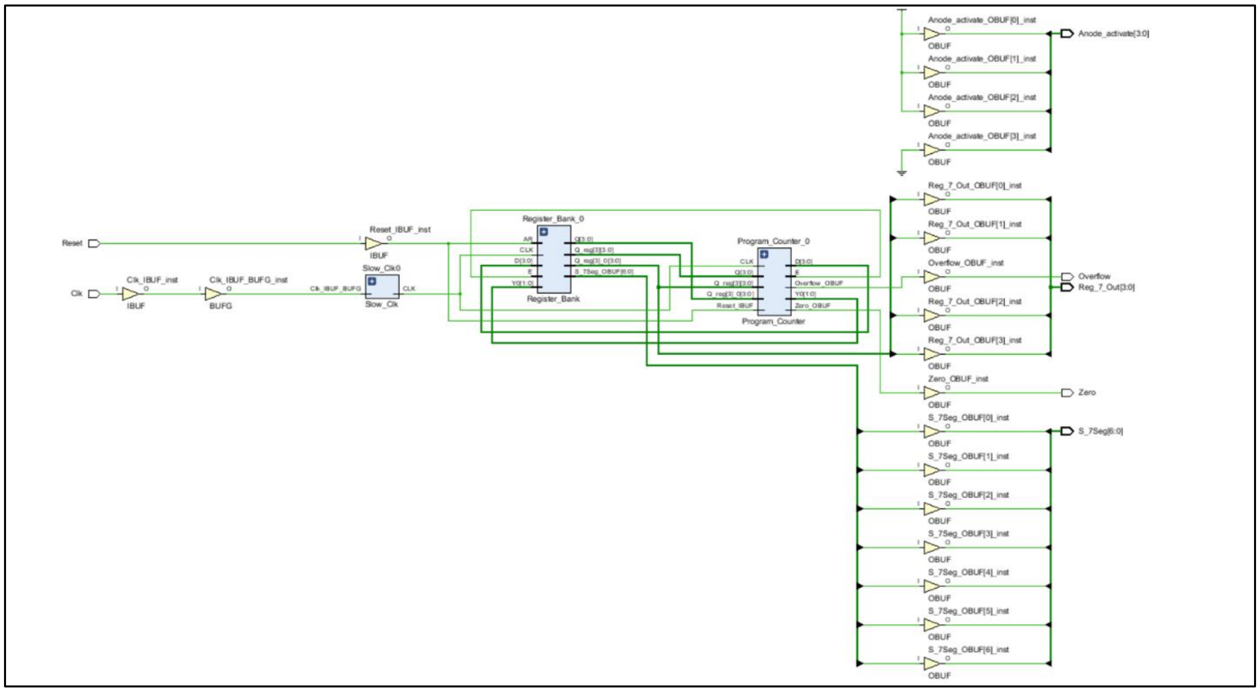


Instruction Decoder

An instruction consists of 12 bits, with initial 2 bits indicating the operation. This instruction decoder can accommodate up to 4 instructions.

- MOVI R, d - Move immediate value d to register R.
- ADD Ra, Rb – Add the values stored in the Ra, Rb registers and store the result in the Ra register.
- NEG Ra – 2’s complement of the value stored in R register.
- JZR R, d - Jump to the d th step if the value in register R is 0.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/17/2024 08:03:12 PM
-- Design Name:
-- Module Name: Instruction_Decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Instruction_Decoder is
    Port ( Instruction : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC;
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub_Select : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is
----------------------------------------------
---------- **  Internal Signals  ** ----------
```

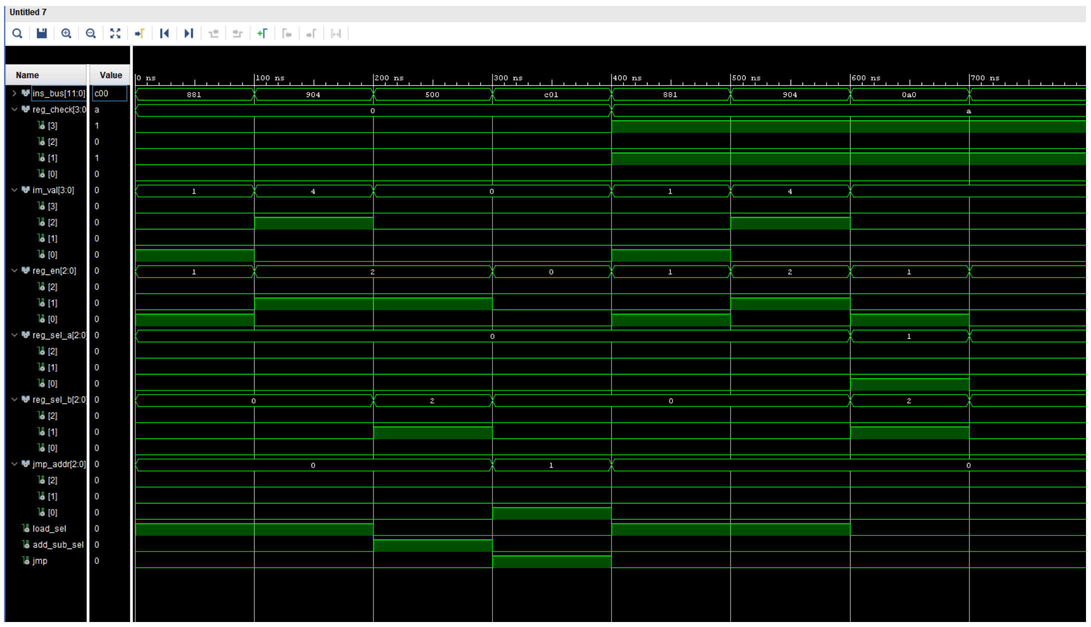
```
---------------------------------------------
-- SIGNAL(s) - Inputs
signal I_Instruction : STD_LOGIC_VECTOR (1 downto 0);
signal I_Ra : STD_LOGIC_VECTOR (2 downto 0);
signal I_Rb : STD_LOGIC_VECTOR (2 downto 0);
signal I_D : STD_LOGIC_VECTOR (3 downto 0);

begin
    -- Extracting values from the instruction vector for clear and readable
code
    I_Instruction <= Instruction(11 downto 10);
    I_Ra <= Instruction(9 downto 7);
    I_Rb <= Instruction(6 downto 4);
    I_D <= Instruction(3 downto 0);

    process(I_Instruction, I_Ra, I_Rb, I_D) begin
    -- When I_Instruction, I_Ra, I_Rb or I_D gets changed, this process will
run
        case I_Instruction is
        -- Instructions begin,
            when "10" =>
            -- Instruction 01 - MOVI
                Reg_Enable <= I_Ra;           -- Enable the given Ra register
                Load_Select <= '1';           -- For immidiate values, 1
                Immediate_Val <= I_D;         -- Instruction, pass D
                Reg_Select_A <= "000";        -- Prevent undefined
                Reg_Select_B <= "000";        -- Prevent undefined
                Add_Sub_Select <= '0';        -- Prevent undefined
                Jump_Flag <= '0';             -- Default, 0
                Jump_Address <= "000";        -- Prevent undefined

            when "00" =>
            -- Instruction 02 - ADD
                Reg_Enable <= I_Ra;           -- Enable the given Ra register
                Load_Select <= '0';           -- Default, To connect
Adder/Substaracter, 0
                Immediate_Val <= "0000";      -- Prevent undefined
                Reg_Select_A <= I_Ra;         -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;         -- Instruction, pass Rb
                Add_Sub_Select <= '0';        -- Default, Addition, 0
                Jump_Flag <= '0';             -- Default, 0
                Jump_Address <= "000";        -- Prevent undefined

            when "01" =>
            -- Instruction 03 - NEG
                -- Perform substraction (0 - Ra) to calc -R,
                Reg_Enable <= I_Ra;           -- Enable the given Ra register
                Load_Select <= '0';           -- Default, To connect
Adder/Substaracter, 0
                Immediate_Val <= "0000";      -- Prevent undefined
                Reg_Select_A <= I_Rb;         -- Instruction, pass Rb, 0
                Reg_Select_B <= I_Ra;         -- Instruction, pass Ra, input R
                Add_Sub_Select <= '1';        -- Substraction, 1
                Jump_Flag <= '0';             -- Default, 0
                Jump_Address <= "000";        -- Prevent undefined

            when "11" =>
            -- Instruction 04 - JZR
                Reg_Enable <= "000";                    -- Prevent undefined
                Load_Select <= '0';                     -- Prevent undefined
                Immediate_Val <= "0000";                -- Prevent undefined
                Reg_Select_A <= I_Ra;                   -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;                   -- Prevent undefined
                Add_Sub_Select <= '0';                  -- Prevent undefined

                if Reg_Check = "0000" then
                    Jump_Flag <= '1';                   -- Jump, 1
                    Jump_Address <= I_D(2 downto 0);    -- Instruction, pass D
                else
                    Jump_Flag <= '0';                   -- Default, 0
                    Jump_Address <= "000";              -- Prevent undefined
                end if;

            when others =>
                Reg_Enable <= "000";           -- Prevent undefined
                Load_Select <= '0';            -- Prevent undefined
                Immediate_Val <= "0000";       -- Prevent undefined
                Reg_Select_A <= "000";         -- Prevent undefined
                Reg_Select_B <= "000";         -- Prevent undefined
                Add_Sub_Select <= '0';         -- Prevent undefined
                Jump_Flag <= '0';              -- Prevent undefined
                Jump_Address <= "000";         -- Prevent undefined
        end case;
    end process;
end Behavioral;
```

Simulation Source File

```
-------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/22/2024 11:45:18 AM
-- Design Name:
-- Module Name: TB_Instruction_Decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
```

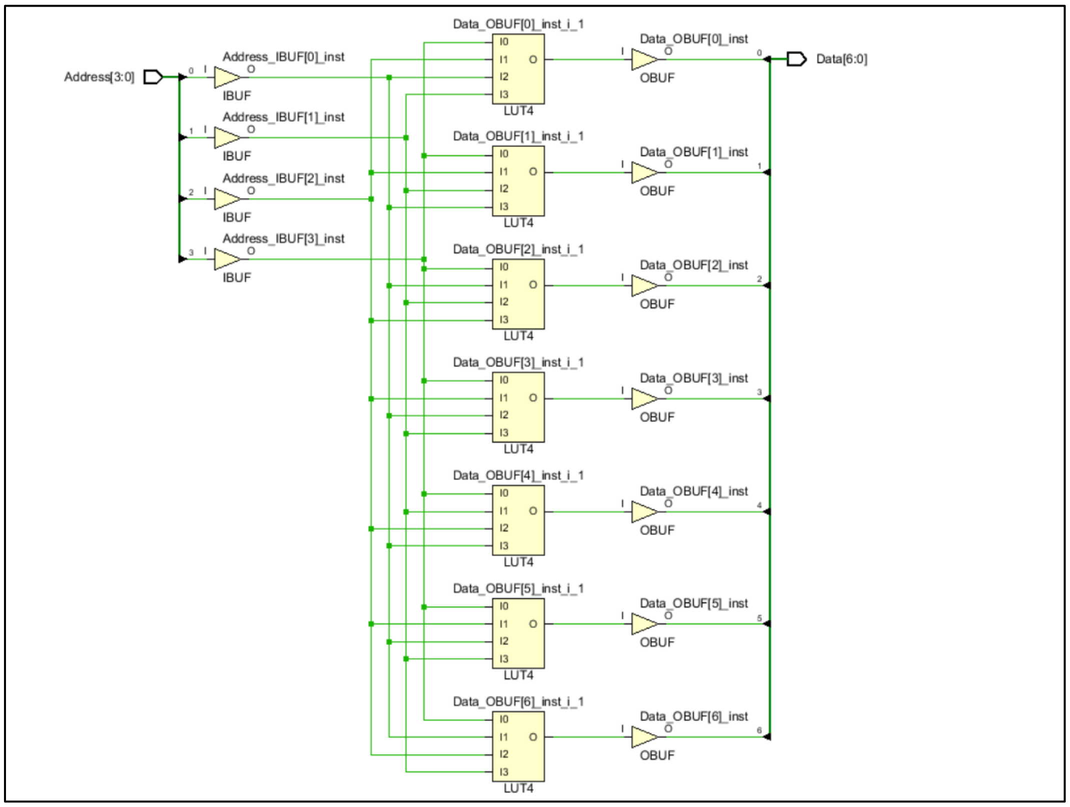
```
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of TB_Instruction_Decoder is

component Instruction_Decoder is
    Port ( Instruction : in STD_LOGIC_VECTOR (11 downto 0);
           Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC;
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub_Select : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
end component;

    SIGNAL ins_bus : STD_LOGIC_VECTOR(11 downto 0);
    SIGNAL reg_check, im_val : STD_LOGIC_VECTOR(3 downto 0);
    SIGNAL reg_en, reg_sel_a, reg_sel_b, jmp_addr : STD_LOGIC_VECTOR(2 downto
0);
    SIGNAL load_sel, add_sub_sel, jmp : STD_LOGIC;

begin
    UUT : Instruction_decoder PORT MAP( Instruction => ins_bus,
                                        Reg_Check => reg_check,
                                        Reg_Enable => reg_en,
                                        Load_Select => load_sel,
                                        Immediate_Val => im_val,
                                        Reg_Select_A => reg_sel_a,
                                        Reg_Select_B => reg_sel_b,
                                        Add_Sub_Select => add_sub_sel,
                                        Jump_Flag => jmp,
                                        Jump_Address => jmp_addr );

    main: process begin

        Reg_Check <= "0000";
        -- MOVI R1, 1
        ins_bus <= "100010000001";
        wait for 100ns;

        -- MOVI R2, 4
        ins_bus <= "100100000100";
        wait for 100ns;

        -- NEG R2
        ins_bus <= "010100000000";
        wait for 100ns;

        -- JZR 1
        ins_bus <= "110000000001";
        wait for 100ns;

        ----------------------------------------

        Reg_Check <= "1010";

        -- MOVI R1, 1
        ins_bus <= "100010000001";
        wait for 100ns;

        -- MOVI R2, 4
        ins_bus <= "100100000100";
        wait for 100ns;

        -- ADD R1, R2
        ins_bus <= "000010100000";
        wait for 100ns;

        -- JZR 0
        ins_bus <= "110000000000";
        wait;

    end process;
end Behavioral;
```

Timing Diagram



<u>Lookup Table for the 7 Segment Display</u>

Elaborated Design Schematic



Design Source File

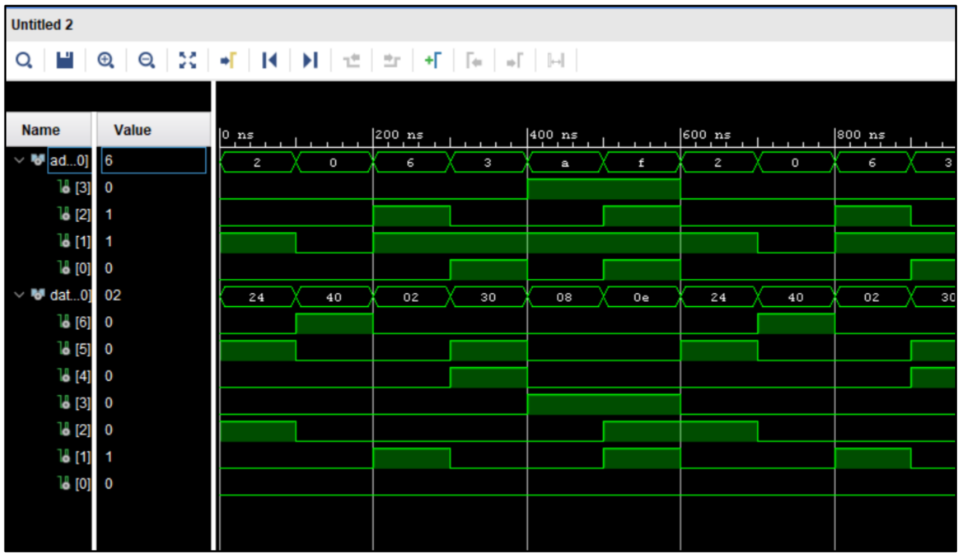
```vhdl
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/19/2024 02:24:53 PM
-- Design Name:
-- Module Name: LUT_16_7 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


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

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

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

entity LUT_16_7 is
    Port ( Address : in STD_LOGIC_VECTOR (3 downto 0);     -- Address input
           Data : out STD_LOGIC_VECTOR (6 downto 0));     -- Data output
end LUT_16_7;

architecture Behavioral of LUT_16_7 is

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

    -- Initialization of seven-segment display values
    signal sevenSegment_ROM : rom_type := (
        "1000000", -- 0
        "1111001", -- 1
        "0100100", -- 2
        "0110000", -- 3
        "0011001", -- 4
        "0010010", -- 5
        "0000010", -- 6
        "1111000", -- 7
        "0000000", -- 8
        "0010000", -- 9
        "0001000", -- A
        "0000011", -- B
        "1000110", -- C
        "0100001", -- D
        "0000110", -- E
        "0001110"  -- F
    );

begin
    -- Output data based on address input
    Data <= sevenSegment_ROM(to_integer(unsigned(Address)));

end Behavioral;
```

Simulation Source File

```
-----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 03/19/2024 02:51:37 PM
-- Design Name:
-- Module Name: TB_LUT_16_7 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of TB_LUT_16_7 is
    COMPONENT LUT_16_7
        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
     --Give 2,0,6,3,a,f as inputs
        address <= "0010";     Wait for 100ns;
        address <= "0000";     Wait for 100ns;
        address <= "0110";     Wait for 100ns;
        address <= "0011";     Wait for 100ns;
        address <= "1010";     Wait for 100ns;
        address <= "1111";     Wait for 100ns;
    End Process;

end Behavioral;
```

Timing Diagram



<u>Nanoprocessor</u>

This Nanoprocessor is assembled by integrating,

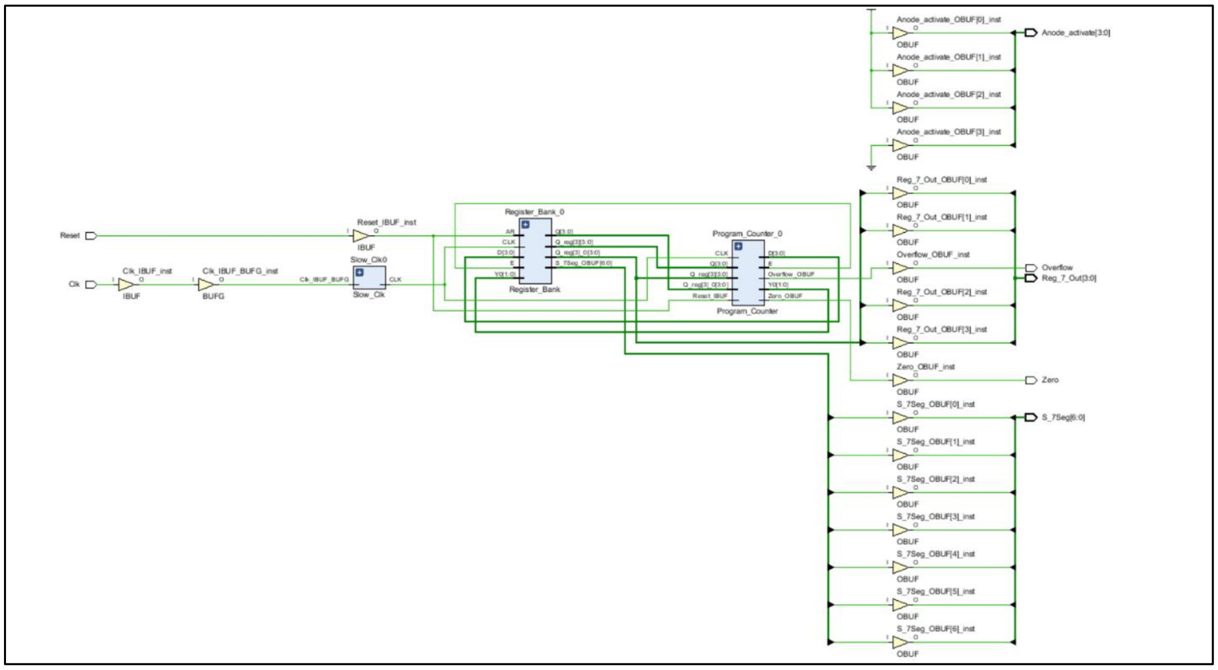
- 1, four-bit add/sub unit
- 2, eight-way four-bit multiplexers
- 1, two-way four-bit multiplexer
- 1, two-way three-bit multiplexer
- 1, three-bit adder
- 1 instruction decoder
- 1 register bank
- 1 slow clock
- 1 program ROM.

This nanoprocessor can execute up to 4 instructions.

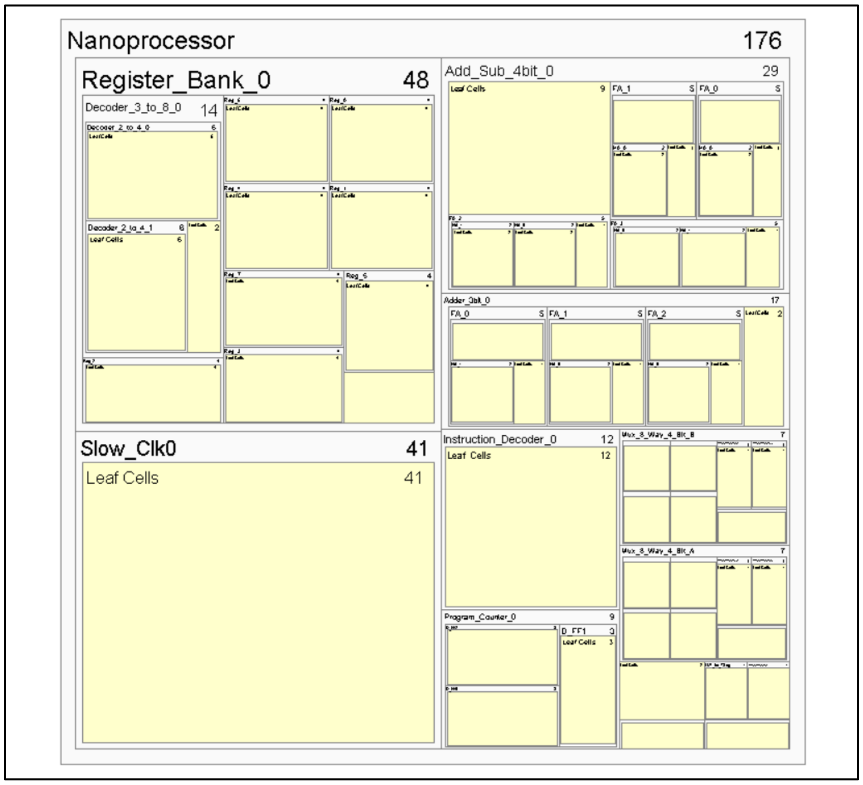
- MOVI R, d - Move immediate value d to register R.
- ADD Ra, Rb - Add values in registers Ra and Rb and store the result in Ra.
- NEG Ra, d - 2's complement of the value stored in register R.

- JZR R, d - Jump if value in register R is 0.

Elaborated Design Schematic



Hierarchy



Design Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/19/2024 12:51:42 AM
-- Design Name:
-- Module Name: Nanoprocessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- 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 Nanoprocessor is
  Port (Clk: in Std_logic;
        Reset: in Std_Logic;
        Overflow: Out Std_Logic;
        Zero: Out Std_Logic;
        Reg_7_Out: Out Std_Logic_Vector (3 downto 0);
        S_7Seg:  Out Std_Logic_Vector (6 downto 0);
        Anode_activate: Out Std_Logic_Vector (3 downto 0)
        );
end Nanoprocessor;

architecture Behavioral of Nanoprocessor is
    Component Add_Sub_4bit
    Port (A: in Std_Logic_Vector (3 downto 0);
          B: in Std_Logic_Vector (3 downto 0);
          S: out Std_Logic_Vector (3 downto 0);
```

```
            Add_Sub_Sel : in STD_LOGIC;
            Carry : out STD_LOGIC;
            Zero: out Std_Logic);
      End Component;

      Component Adder_3bit
      Port (A: in Std_Logic_Vector (2 downto 0);
            B: in Std_Logic_Vector (2 downto 0);
            S: out Std_Logic_Vector (2 downto 0);
            C_in : in STD_LOGIC;
            C_out : out STD_LOGIC);
      End Component;

      Component Program_Counter
      Port (D : in STD_LOGIC_Vector (2 downto 0);
            Reset : in STD_LOGIC;
            Clk : in STD_LOGIC;
            Q : out STD_LOGIC_VECTOR (2 downto 0));
      End Component;

      Component Mux_8_Way_4_Bit
      Port (Data_0, Data_1, Data_2, Data_3, Data_4, Data_5, Data_6, Data_7  :
  in STD_LOGIC_VECTOR (3 downto 0);
                RegSel : in STD_LOGIC_VECTOR (2 downto 0);
                Output : out STD_LOGIC_VECTOR (3 downto 0));
      End Component;

      Component Mux_2_Way_4_Bit
      Port (A_in: in Std_Logic_Vector (3 downto 0);
            B_in: in Std_Logic_Vector (3 downto 0);
            S: in Std_Logic;
            C_out: out Std_Logic_Vector (3 downto 0));
      End Component;

      Component Mux_2_Way_3_Bit
      Port (A_in : in STD_LOGIC_VECTOR (2 downto 0);
            B_in : in STD_LOGIC_VECTOR (2 downto 0);
            S : in STD_LOGIC;
            C_out : out STD_LOGIC_VECTOR (2 downto 0));
      End Component;

      Component Register_Bank
      Port (Value_in: in Std_Logic_Vector (3 downto 0);
            Reg_EN: in Std_Logic_Vector (2 downto 0);
            Clk: in Std_Logic;
            Reset: in Std_Logic;
            Value_out_0: Out Std_Logic_Vector (3 downto 0);
            Value_out_1: Out Std_Logic_Vector (3 downto 0);
            Value_out_2: Out Std_Logic_Vector (3 downto 0);
            Value_out_3: Out Std_Logic_Vector (3 downto 0);
            Value_out_4: Out Std_Logic_Vector (3 downto 0);
            Value_out_5: Out Std_Logic_Vector (3 downto 0);
            Value_out_6: Out Std_Logic_Vector (3 downto 0);
            Value_out_7: Out Std_Logic_Vector (3 downto 0));
      End Component;

      Component LUT
```

```
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
    End Component;

    Component Instruction_Decoder
    Port (Instruction : in STD_LOGIC_VECTOR (11 downto 0);
          Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
          Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
          Load_Select : out STD_LOGIC;
          Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
          Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
          Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
          Add_Sub_Select : out STD_LOGIC;
          Jump_Flag : out STD_LOGIC;
          Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
    End Component;

    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              --Have to create the slow clock
        Port ( Clk_in: in STD_LOGIC;
               Clk_out: out STD_LOGIC);
    End Component;

    Signal instruction: Std_Logic_Vector (11 downto 0);
    Signal value_in, value_out_0, value_out_1, value_out_2, value_out_3,
value_out_4, value_out_5, value_out_6, value_out_7: Std_Logic_Vector (3
downto 0);
    Signal mux_out_A, mux_out_B, sum, immediate_val: Std_Logic_Vector (3
downto 0);
    Signal reg_en, reg_sel_A, reg_sel_B, jump_address, adder_in, adder_out,
counter_in, memory_sel: Std_Logic_Vector (2 downto 0);
    Signal add_sub_sel, load_sel, jump: Std_Logic;
    Signal Clk_slow: STD_LOGIC; --Internal Clock

begin

    --Turn on ony one of the seven segment displays
    Anode_activate <= "0111";

    Slow_Clk0: Slow_Clk
        Port Map ( Clk_in => Clk,
                   Clk_out => Clk_slow);

    Register_Bank_0: Register_Bank
    Port Map (Value_in => value_in,
              Reg_EN => reg_en,
              Clk => Clk_slow,
              Reset => Reset,
              Value_out_0 => value_out_0,
              Value_out_1 => value_out_1,
              Value_out_2 => value_out_2,
              Value_out_3 => value_out_3,
              Value_out_4 => value_out_4,
```

```vhdl
          Value_out_5 => value_out_5,
          Value_out_6 => value_out_6,
          Value_out_7 => value_out_7);

 Reg_7_Out <= value_out_7;
 Mux_8_Way_4_Bit_A: Mux_8_Way_4_Bit
 Port Map (Data_0 => value_out_0,
           Data_1 => value_out_1,
           Data_2 => value_out_2,
           Data_3 => value_out_3,
           Data_4 => value_out_4,
           Data_5 => value_out_5,
           Data_6 => value_out_6,
           Data_7 => value_out_7,
           RegSel => reg_sel_A,
           Output => mux_out_A);

Mux_8_Way_4_Bit_B: Mux_8_Way_4_Bit
Port Map (Data_0 => value_out_0,
          Data_1 => value_out_1,
          Data_2 => value_out_2,
          Data_3 => value_out_3,
          Data_4 => value_out_4,
          Data_5 => value_out_5,
          Data_6 => value_out_6,
          Data_7 => value_out_7,
          RegSel => reg_sel_B,
          Output => mux_out_B);

Mux_2_Way_4_Bit_0: Mux_2_Way_4_Bit
Port Map (A_in => sum,
          B_in => immediate_val,
          S => load_sel,
          C_out => value_in);

Add_Sub_4bit_0: Add_Sub_4bit
Port Map (A => mux_out_A,
          B => mux_out_B,
          S => sum,
          Add_Sub_Sel => add_sub_sel,
          Carry => Overflow,    --To Led
          Zero => Zero);        --To Led

Instruction_Decoder_0: Instruction_Decoder
Port Map (Instruction => instruction,
          Reg_Check => mux_out_A,
          Reg_Enable => reg_en,
          Load_Select => load_Sel,
          Immediate_Val => immediate_val,
          Reg_Select_A => reg_sel_A,
          Reg_Select_B => reg_sel_B,
          Add_Sub_Select => add_sub_sel,
          Jump_Flag => jump,
          Jump_Address => jump_address);

Adder_3bit_0: Adder_3bit
Port Map ( A => adder_in,
                B => "001",
                S => adder_out,
                C_in => '0');

    Mux_2_Way_3_Bit_0: Mux_2_Way_3_Bit
    Port Map (A_in => adder_out,
              B_in => jump_address,
              S => jump,
              C_out => counter_in);

    Program_Counter_0: Program_Counter
    Port Map (D => counter_in,
              Reset => Reset,
              Clk => Clk_slow,
              Q => memory_sel);

    adder_in <= memory_sel;

    Program_ROM: LUT
    Port Map ( address => memory_sel,
               data => instruction);

    LUT_for_7Seg: LUT_16_7
    Port Map (Address => value_out_7,
              Data => S_7Seg);

end Behavioral;
```

Simulation Source File

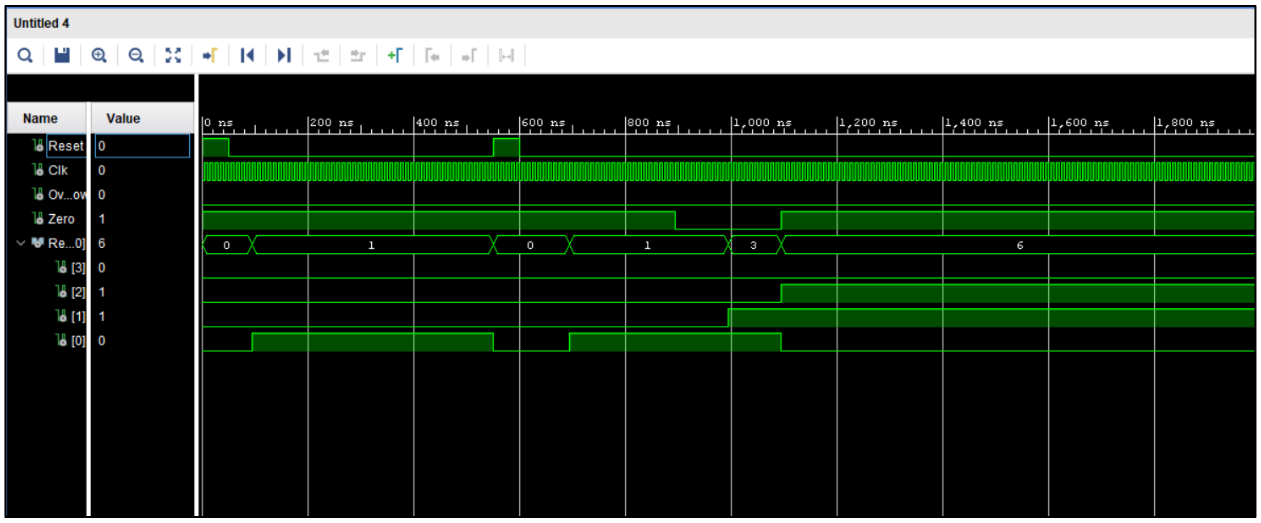
```
-----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/23/2024 10:05:18 PM
-- Design Name:
-- Module Name: TB_Nanoprocessor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-----------------------------------------------------------------------------
-----
```

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

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

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

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

architecture Behavioral of TB_Nanoprocessor is

component Nanoprocessor is
  Port (Clk: in Std_logic;
        Reset: in Std_Logic;
        Overflow: Out Std_Logic;
        Zero: Out Std_Logic;
        Reg_7_Out: Out Std_Logic_Vector (3 downto 0));
end component;

SIGNAL Reset, Clk, Overflow, Zero : STD_LOGIC;
SIGNAL Reg_7_Out : STD_LOGIC_VECTOR(3 downto 0);

begin

    UTT : Nanoprocessor port map (Clk, Reset, Overflow, Zero, Reg_7_Out);

    clock : process
        begin
            Clk <= '0';
            wait for 5ns;
            Clk <= '1';
            wait for 5ns;
    end process;

    main: process begin
            Reset <= '1';
            wait for 50ns;
            Reset <= '0';

            wait for 500ns;

            Reset <= '1';
            wait for 50ns;
            Reset <= '0';
            wait;
    end process;
end Behavioral;
```

Constraints

```
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_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 P1 [get_ports {Zero}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]
set_property PACKAGE_PIN L1 [get_ports {Overflow}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Overflow}]

##7 segment display
set_property PACKAGE_PIN W7 [get_ports {S_7Seg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {S_7Seg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {S_7Seg[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {S_7Seg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {S_7Seg[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {S_7Seg[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {S_7Seg[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S_7Seg[6]}]

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

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

Timing Diagram



<u>**Additional Features**</u>

We have extended our nanoprocessor to accommodate the execution of up to 12 instructions inclusive of the initial 4 instructions.

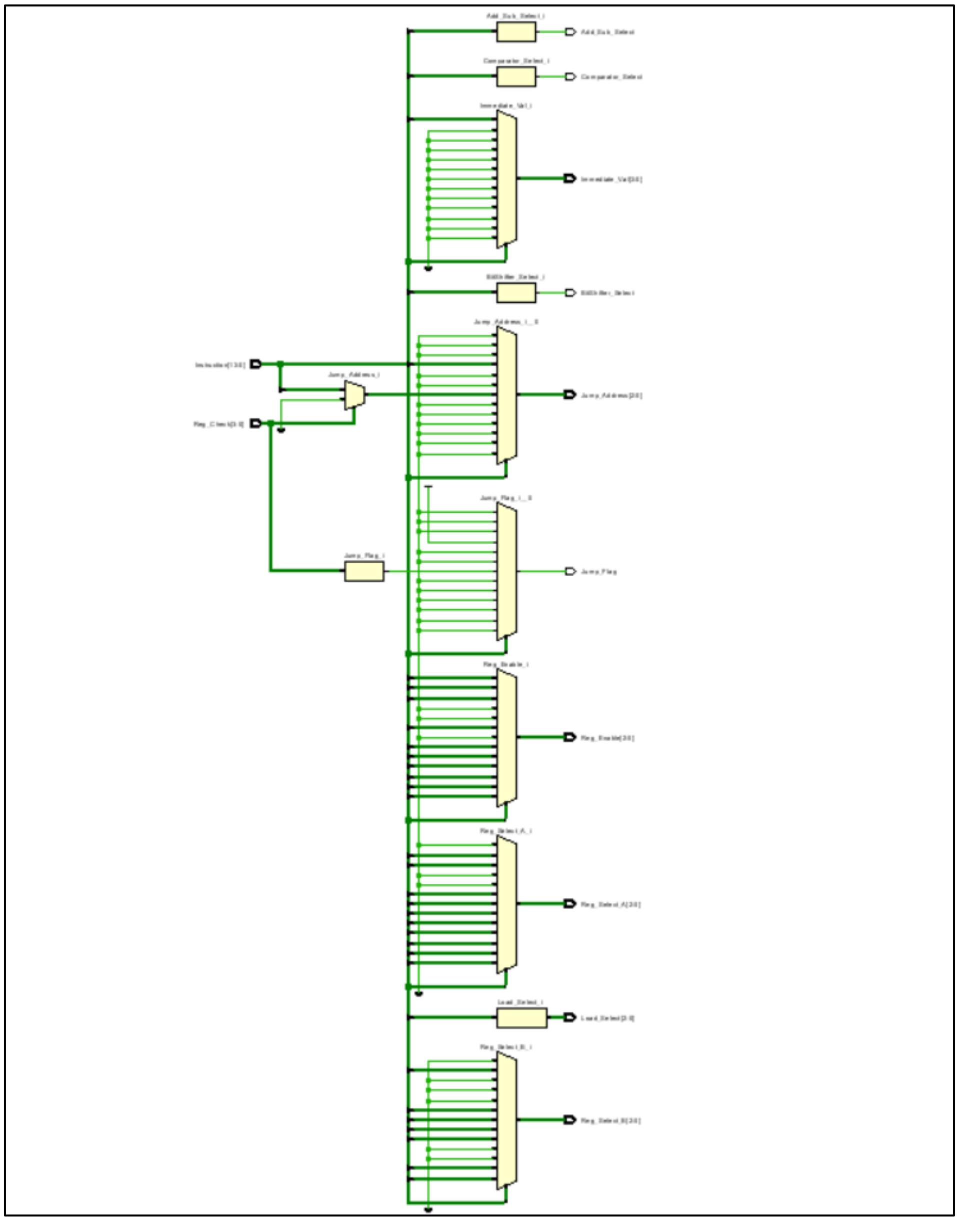
- SUB Ra, Rb - Subtract the value of Rb from Ra and store the result in Ra.
  - Executed by the 4-bit Add/Sub Unit.
- MUL Ra, Rb - Multiply Rb by Ra and store the result in Ra. (Only for two bits)
  - Executed by the newly added 2-bit Multiplier.
- MOV Ra, Rb - Move value of Rb to Ra.
- JMP d - Jump instructions by d steps.
- SHL Ra - Shift the value of the Ra to the left by one bit.
  - Executed by the newly added bit shifter.
- RHL Ra - Shift the value of the Ra to the right by one bit.
  - Executed by the newly added bit shifter.
- MAX Ra, Rb - Find the maximum value stored in Ra and Rb.
  - Executed by the newly added comparator.
- MIN Ra, Rb - Find the minimum value stored in Ra and Rb.
  - Executed by the newly added comparator.

Enhanced and Newly Added Components to Accommodate the Additional Features

Enhanced Instruction Decoder

In order to accommodate all the features, we extended the length of the instruction vector to 14 bits, with the initial 4 bit now indicating the instruction instead of the first 2 bits.

Elaborated Design Schematic



Design Source File

```vhdl
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/17/2024 08:03:12 PM
-- Design Name:
-- Module Name: Instruction_Decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Instruction_Decoder is
    Port ( Instruction : in STD_LOGIC_VECTOR (13 downto 0);
           Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub_Select : out STD_LOGIC;
           Comparator_Select : out STD_LOGIC;
           BitShifter_Select : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is
```

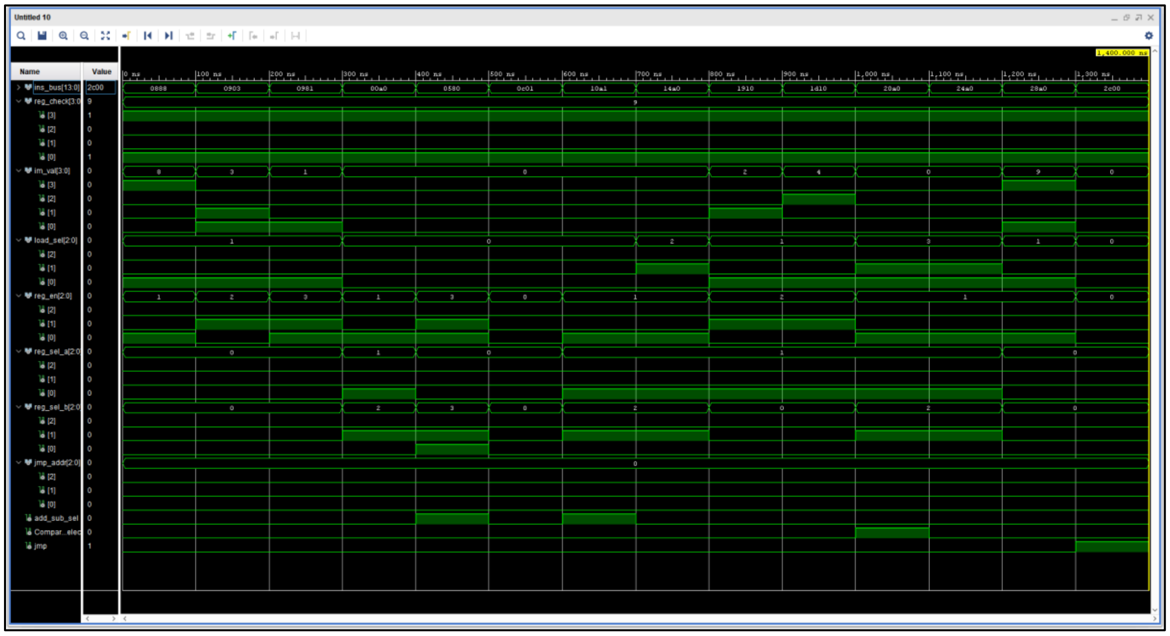
```
----------------------------------------------
---------- **  Internal Signals  ** ----------
----------------------------------------------
-- SIGNAL(s) - Inputs
signal I_Instruction : STD_LOGIC_VECTOR (3 downto 0);
signal I_Ra : STD_LOGIC_VECTOR (2 downto 0);
signal I_Rb : STD_LOGIC_VECTOR (2 downto 0);
signal I_D : STD_LOGIC_VECTOR (3 downto 0);

begin

    -- Extracting values from the instruction vector for clear and readable
code
    I_Instruction <= Instruction(13 downto 10);
    I_Ra <= Instruction(9 downto 7);
    I_Rb <= Instruction(6 downto 4);
    I_D <= Instruction(3 downto 0);

    process(I_Instruction, I_Ra, I_Rb, I_D) begin
    -- When I_Instruction, I_Ra, I_Rb or I_D gets changed, this process will
run
        case I_Instruction is
        -- Instructions begin,
            when "0010" =>
            -- Instruction 01 - MOVI
                Reg_Enable <= I_Ra;          -- Enable the given Ra register
                Load_Select <= "001";        -- For immidiate values, 001
                Immediate_Val <= I_D;        -- Instruction, pass D
                Reg_Select_A <= "000";       -- Prevent undefined
                Reg_Select_B <= "000";       -- Prevent undefined
                Add_Sub_Select <= '0';       -- Prevent undefined
                Comparator_Select <= '0';    -- Prevent undefined
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '0';            -- Default, 0
                Jump_Address <= "000";       -- Prevent undefined

            when "0000" =>
            -- Instruction 02 - ADD
                Reg_Enable <= I_Ra;          -- Enable the given Ra register
                Load_Select <= "000";        -- Default, To connect
Adder/Substaracter, 000
                Immediate_Val <= "0000";     -- Prevent undefined
                Reg_Select_A <= I_Ra;        -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;        -- Instruction, pass Rb
                Add_Sub_Select <= '0';       -- Default, Addition, 0
                Comparator_Select <= '0';    -- Prevent undefined
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '0';            -- Default, 0
                Jump_Address <= "000";       -- Prevent undefined

            when "0001" =>
            -- Instruction 03 - NEG
                -- Perform substraction (0 - Ra) to calc -R,
                Reg_Enable <= I_Ra;          -- Enable the given Ra register
                Load_Select <= "000";        -- Default, To connect
Adder/Substaracter, 000
```

```
                Immediate_Val <= "0000";    -- Prevent undefined
                Reg_Select_A <= I_Rb;       -- Instruction, pass Rb, 0
                Reg_Select_B <= I_Ra;       -- Instruction, pass Ra, input R
                Add_Sub_Select <= '1';      -- Substraction, 1
                Comparator_Select <= '0';   -- Prevent undefined
                BitShifter_Select <= '0';   -- Prevent undefined
                Jump_Flag <= '0';           -- Default, 0
                Jump_Address <= "000";      -- Prevent undefined

            when "0011" =>
            -- Instruction 04 - JZR
                Reg_Enable <= "000";                 -- Prevent undefined
                Load_Select <= "000";                -- Prevent undefined
                Immediate_Val <= "0000";             -- Prevent undefined
                Reg_Select_A <= I_Ra;                -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;                -- Prevent undefined
                Add_Sub_Select <= '0';               -- Prevent undefined
                Comparator_Select <= '0';            -- Prevent undefined
                BitShifter_Select <= '0';   -- Prevent undefined

                if Reg_Check = "0000" then
                    Jump_Flag <= '1';                -- Jump, 1
                    Jump_Address <= I_D(2 downto 0); -- Instruction, pass D
                else
                    Jump_Flag <= '0';                -- Default, 0
                    Jump_Address <= "000";           -- Prevent undefined
                end if;

            when "0100" =>
                -- Instruction 05 - SUB
                Reg_Enable <= I_Ra;         -- Enable the given Ra register
                Load_Select <= "000";       -- Default, To connect
    Adder/Substaracter, 000
                Immediate_Val <= "0000";    -- Prevent undefined
                Reg_Select_A <= I_Ra;       -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;       -- Instruction, pass Rb
                Add_Sub_Select <= '1';      -- Default, Substraction, 1
                Comparator_Select <= '0';   -- Prevent undefined
                BitShifter_Select <= '0';   -- Prevent undefined
                Jump_Flag <= '0';           -- Default, 0
                Jump_Address <= "000";      -- Prevent undefined

            when "0101" =>
                -- Instruction 06 - Multiply (2 Bit)
                Reg_Enable <= I_Ra;         -- Enable the given Ra register
                Load_Select <= "010";       -- To connect Multiplier, 010
                Immediate_Val <= "0000";    -- Prevent undefined
                Reg_Select_A <= I_Ra;       -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;       -- Instruction, pass Rb
                Add_Sub_Select <= '0';      -- Prevent undefined
                Comparator_Select <= '0';   -- Prevent undefined
                BitShifter_Select <= '0';   -- Prevent undefined
                Jump_Flag <= '0';           -- Default, 0
                Jump_Address <= "000";      -- Prevent undefined

            when "0110" =>
                -- Instruction 07 - Left BitShift
                Reg_Enable <= I_Ra;                                 -- Enable the 
given Ra register
                Load_Select <= "100";                               -- For Bit 
Shifter values, 100
                Immediate_Val <= "0000";                            -- Prevent 
undefined
                Reg_Select_A <= I_Ra;                               -- 
Instruction, pass Ra
                Reg_Select_B <= "000";                              -- Prevent 
undefined
                Add_Sub_Select <= '0';                              -- Prevent 
undefined
                Comparator_Select <= '0';                           -- Prevent 
undefined
                BitShifter_Select <= '0';                           -- For left, 
0
                Jump_Flag <= '0';                                   -- Default, 0
                Jump_Address <= "000";                              -- Prevent 
undefined

            when "0111" =>
                -- Instruction 08 - Right BitShift
                Reg_Enable <= I_Ra;                                 -- Enable the 
given Ra register
                Load_Select <= "100";                               -- For Bit 
Shifter values, 100
                Immediate_Val <= "0000";                            -- Prevent 
undefined
                Reg_Select_A <= I_Ra;                               -- 
Instruction, pass Ra
                Reg_Select_B <= "000";                              -- Prevent 
undefined
                Add_Sub_Select <= '0';                              -- Prevent 
undefined
                Comparator_Select <= '0';                           -- Prevent 
undefined
                BitShifter_Select <= '1';                           -- For right, 
1
                Jump_Flag <= '0';                                   -- Default, 0
                Jump_Address <= "000";                              -- Prevent 
undefined

            when "1000" =>
                -- Instruction 09 - MAX
                Reg_Enable <= I_Ra;         -- Enable the given Ra register
                Load_Select <= "011";       -- For Comprator, 011
                Immediate_Val <= "0000";    -- Prevent undefined
                Reg_Select_A <= I_Ra;       -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;       -- Instruction, pass Rb
                Add_Sub_Select <= '0';      -- Prevent undefined
                Comparator_Select <= '1';   -- For MAX, 1
                BitShifter_Select <= '0';   -- Prevent undefined
                Jump_Flag <= '0';           -- Default, 0
                Jump_Address <= "000";      -- Prevent undefined

            when "1001" =>
                -- Instruction 10 - MIN
                Reg_Enable <= I_Ra;          -- Enable the given Ra register
                Load_Select <= "011";        -- For Comprator, 011
                Immediate_Val <= "0000";     -- Prevent undefined
                Reg_Select_A <= I_Ra;        -- Instruction, pass Ra
                Reg_Select_B <= I_Rb;        -- Instruction, pass Rb
                Add_Sub_Select <= '0';       -- Prevent undefined
                Comparator_Select <= '0';    -- For MIN, 0
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '0';            -- Default, 0
                Jump_Address <= "000";       -- Prevent undefined

            when "1010" =>
                -- Instruction 11 - MOV
                Reg_Enable <= I_Ra;          -- Enable the given Ra register
                Load_Select <= "000";        -- Default, To connect
    Adder/Substaracter, 000
                Immediate_Val <= "0000";     -- Prevent undefined
                Reg_Select_A <= I_Rb;        -- Get Rb's value
                Reg_Select_B <= "000";       -- Prevent undefined
                Add_Sub_Select <= '0';       -- Prevent undefined
                Comparator_Select <= '0';    -- Prevent undefined
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '0';            -- Default, 0
                Jump_Address <= "000";       -- Prevent undefined

            when "1011" =>
                -- Instruction 12 - JMP
                Reg_Enable <= "000";                  -- Prevent undefined
                Load_Select <= "000";                 -- Prevent undefined
                Immediate_Val <= "0000";              -- Prevent undefined
                Reg_Select_A <= "000";                -- Prevent undefined
                Reg_Select_B <= "000";                -- Prevent undefined
                Add_Sub_Select <= '0';                -- Prevent undefined
                Comparator_Select <= '0';             -- Prevent undefined
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '1';                     -- Jump, 1
                Jump_Address <= I_D(2 downto 0);      -- Instruction, pass D


            when others =>
                Reg_Enable <= "000";         -- Prevent undefined
                Load_Select <= "000";        -- Prevent undefined
                Immediate_Val <= "0000";     -- Prevent undefined
                Reg_Select_A <= "000";       -- Prevent undefined
                Reg_Select_B <= "000";       -- Prevent undefined
                Add_Sub_Select <= '0';       -- Prevent undefined
                Comparator_Select <= '0';    -- Prevent undefined
                BitShifter_Select <= '0';    -- Prevent undefined
                Jump_Flag <= '0';            -- Prevent undefined
                Jump_Address <= "000";       -- Prevent undefined
        end case;
    end process;
end Behavioral;
```

Simulation Source File

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

architecture Behavioral of TB_Instruction_Decoder is

component Instruction_Decoder is
    Port ( Instruction : in STD_LOGIC_VECTOR (13 downto 0);
           Reg_Check : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
           Load_Select : out STD_LOGIC_VECTOR (2 downto 0);
           Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
           Add_Sub_Select : out STD_LOGIC;
           Comparator_Select : out STD_LOGIC;
           Jump_Flag : out STD_LOGIC;
           Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
end component;

    SIGNAL ins_bus : STD_LOGIC_VECTOR(13 downto 0);
    SIGNAL reg_check, im_val : STD_LOGIC_VECTOR(3 downto 0);
    SIGNAL load_sel, reg_en, reg_sel_a, reg_sel_b, jmp_addr :
STD_LOGIC_VECTOR(2 downto 0);
    SIGNAL add_sub_sel, Comparator_Select, jmp : STD_LOGIC;

begin
    UUT : Instruction_decoder PORT MAP( Instruction => ins_bus,
                                        Reg_Check => reg_check,
                                        Reg_Enable => reg_en,
                                        Load_Select => load_sel,
                                        Immediate_Val => im_val,
                                        Reg_Select_A => reg_sel_a,
                                        Reg_Select_B => reg_sel_b,
                                        Add_Sub_Select => add_sub_sel,
                                        Comparator_Select =>
Comparator_Select,
                                        Jump_Flag => jmp,
                                        Jump_Address => jmp_addr );

    main: process begin

        Reg_Check <= "1001";

        -- MOVI R1, 8
        ins_bus <= "00100010001000";
        wait for 100ns;

        -- MOVI R2, 3
        ins_bus <= "00100100000011";
        wait for 100ns;

        -- MOVI R3, 1
        ins_bus <= "00100110000001";
        wait for 100ns;

        -- ADD R1, R2
        ins_bus <= "00000010100000";
        wait for 100ns;

        -- NEG R3
        ins_bus <= "00010110000000";
        wait for 100ns;

        -- JZR 1
        ins_bus <= "00110000000001";
        wait for 100ns;

        -- SUB R2 from R1
        ins_bus <= "01000010100001";
        wait for 100ns;

        -- Mutiply R1 and R2
        ins_bus <= "01010010100000";
        wait for 100ns;

        -- Left BitShift R1 and saves to R2
        ins_bus <= "01100100010000";
        wait for 100ns;

        -- Right BitShift R1 and saves to R2
        ins_bus <= "01110100010000";
        wait for 100ns;

        -- MAX of R1 and R2
        ins_bus <= "10000010100000";
        wait for 100ns;

        -- MIN of R1 and R2
        ins_bus <= "10010010100000";
        wait for 100ns;

        -- MOV R2 to R1
        ins_bus <= "10100010100000";
        wait for 100ns;

        -- JMP to 000
        ins_bus <= "10110000000000";
        wait for 100ns;

        wait;
    end process;
end Behavioral;
```

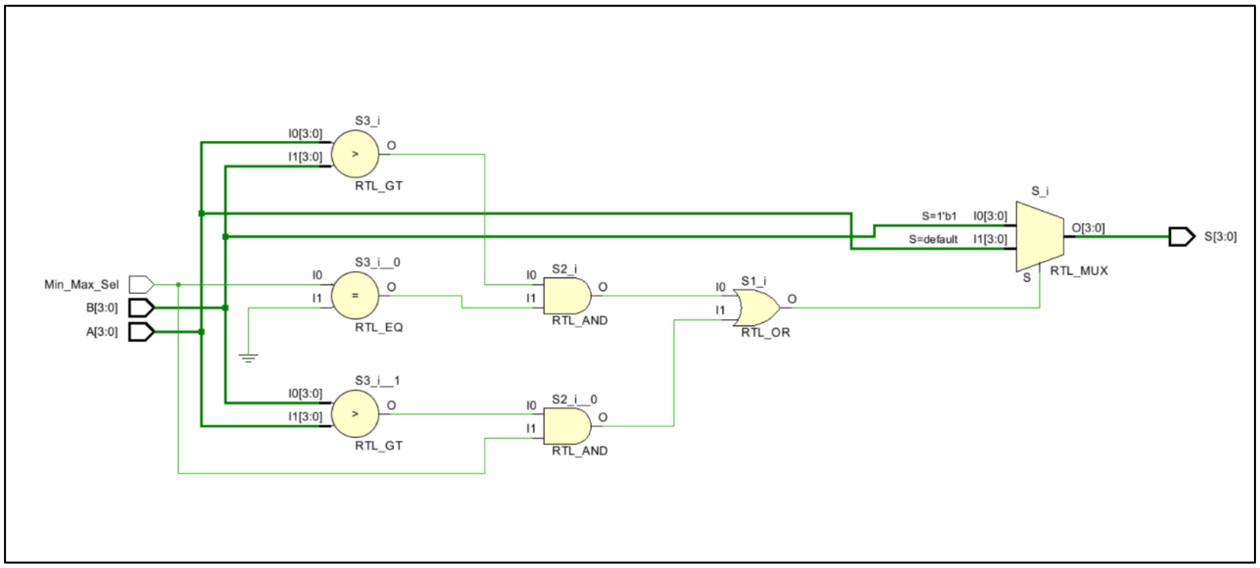
Timing Diagram



Comparator

A comparator has been incorporated to compare two numbers, enabling the execution of MIN and MAX instructions.

Elaborated Design Schematic



Design Source File

```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/24/2024 09:50:12 PM
-- Design Name:
-- Module Name: Comparator_4_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Comparator_4_Bit is
  Port (
    A: in Std_Logic_Vector (3 downto 0);       -- Input A
    B: in Std_Logic_Vector (3 downto 0);       -- Input B
    S: out Std_Logic_Vector (3 downto 0);      -- Output S
    Min_Max_Sel : in STD_LOGIC                 -- Min/Max selection
  );
end Comparator_4_Bit;

architecture Behavioral of Comparator_4_Bit is

begin
  -- Comparator process
  Comparator_4_Bit : process(A, B, Min_Max_Sel) begin
    -- When A, B, or Min_Max_Sel changes, this process will run
    if (A > B AND Min_Max_Sel = '0') OR (B > A AND Min_Max_Sel = '1') then
```

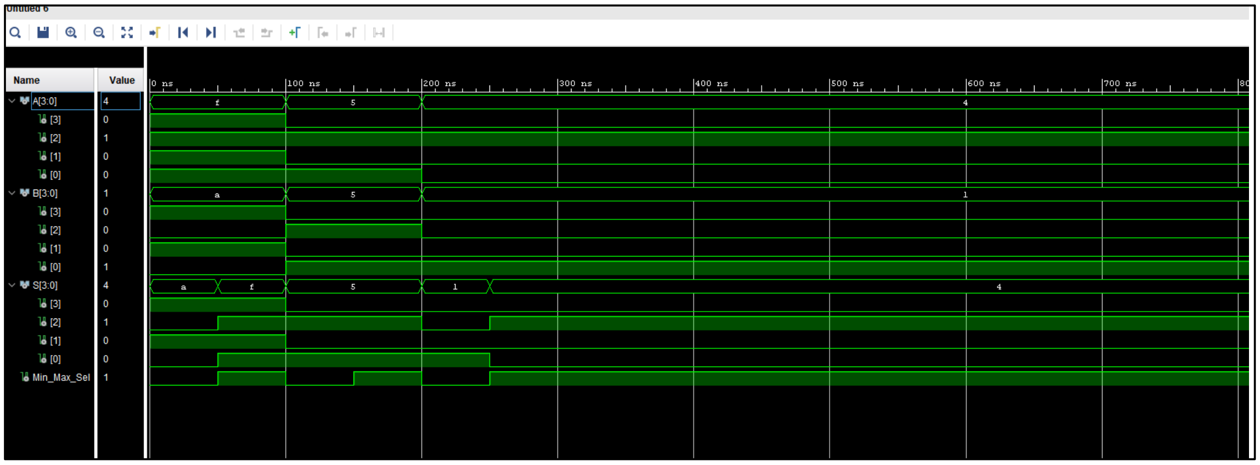
```
      S <= B;                                    -- Assign B to S if B is
greater (Min_Max_Sel = 0) or A is greater (Min_Max_Sel = 1)
    elsif (B > A AND Min_Max_Sel = '0') OR (A > B AND Min_Max_Sel = '1') then
      S <= A;                                    -- Assign A to S if A is
greater (Min_Max_Sel = 0) or B is greater (Min_Max_Sel = 1)
    else
      S <= A;                                    -- Default assignment
    end if;
  end process;
end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/25/2024 09:54:50 AM
-- Design Name:
-- Module Name: TB_Comparator - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

```vhdl
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);
          S: out Std_Logic_Vector (3 downto 0);
          Min_Max_Sel : in STD_LOGIC);
end component;

SIGNAL A, B, S : Std_Logic_Vector (3 downto 0);
SIGNAL Min_Max_Sel: Std_Logic;

begin
    UUT : Comparator PORT MAP( A => A,
                               B => B,
                               S => S,
                               Min_Max_Sel => Min_Max_Sel);


    main: process begin

        A <= "1111";
        B <= "1010";
        Min_Max_Sel <= '0';
        wait for 50ns;
        Min_Max_Sel <= '1';
        wait for 50ns;

        -- Test for equal condition
        A <= "0101";
        B <= "0101";
        Min_Max_Sel <= '0';
        wait for 50ns;
        Min_Max_Sel <= '1';
        wait for 50ns;

        A <= "0100";
        B <= "0001";
        Min_Max_Sel <= '0';
        wait for 50ns;
        Min_Max_Sel <= '1';

        wait;
    end process;
end Behavioral;
```

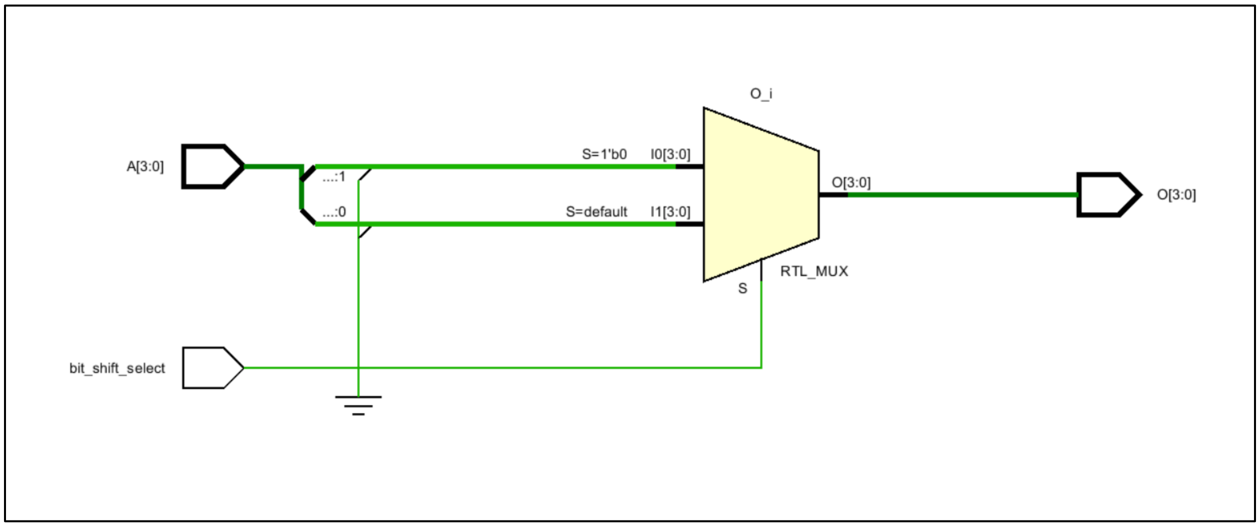
Timing Diagram



Shifter

Shift the value of the given register to the left or right by one bit.

Elaborated Design Schematic



Design Source File

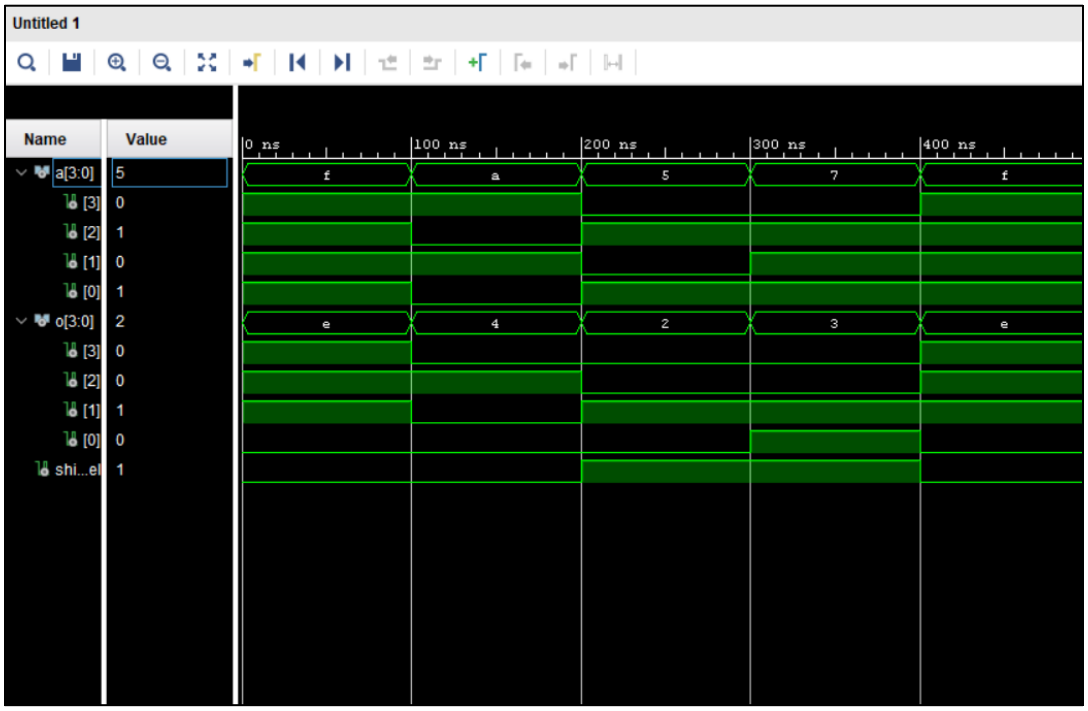
```
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 04/26/2024 03:08:05 PM
-- Design Name:
-- Module Name: Shifter_4_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Shifter_4_Bit is
  Port (
    A: in Std_Logic_Vector (3 downto 0);          -- Input A (4-bit)
    bit_shift_select: in Std_Logic;                -- Bit shift selection
    O: out Std_Logic_Vector (3 downto 0)           -- Output  (4-bit)
  );
end Shifter_4_Bit;

architecture Behavioral of Shifter_4_Bit is

begin
  -- Shifter process
  Shifter_4_Bit : process(A, bit_shift_select) begin
    -- When A or bit_shift_select changes, this process will run
    if (bit_shift_select = '0') then
      O <= A(2 downto 0) & '0';                    -- Shift A to the left by
1 bit
    else
      O <= '0' & A(3 downto 1);                    -- Shift A to the right by
1 bit
    end if;
  end process;
end Behavioral;
```

Simulation Source File

```
----------------------------------------------------------------------------
-----
-- Company: 
-- Engineer: 
-- 
-- Create Date: 05/02/2024 11:04:26 AM
-- Design Name: 
-- Module Name: TB_Shifter_4bit - Behavioral
-- Project Name: 
-- Target Devices: 
-- Tool Versions: 
-- Description: 
-- 
-- Dependencies: 
-- 
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- 
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Shifter_4bit is
--  Port ( );
end TB_Shifter_4bit;

architecture Behavioral of TB_Shifter_4bit is
```

```
Component Shifter_4_Bit
    Port (
        A: in Std_Logic_Vector (3 downto 0);          -- Input A (4-bit)
        bit_shift_select: in Std_Logic;                -- Bit shift selection
        O: out Std_Logic_Vector (3 downto 0)           -- Output  (4-bit)
      );
    End Component;

    Signal a, o: Std_Logic_Vector (3 downto 0);
    Signal shift_sel: Std_Logic;

begin
    UUT: Shifter_4_Bit
    Port Map (A => a,
              bit_shift_select => shift_sel,
              O => o);

    process begin

    shift_sel <= '0';
    a <= "1111"; Wait for 100 ns;
    a <= "1010"; Wait for 100 ns;

    shift_sel <= '1';
    a <= "0101"; Wait for 100 ns;
    a <= "0111"; Wait for 100 ns;

    end process;

end Behavioral;
```

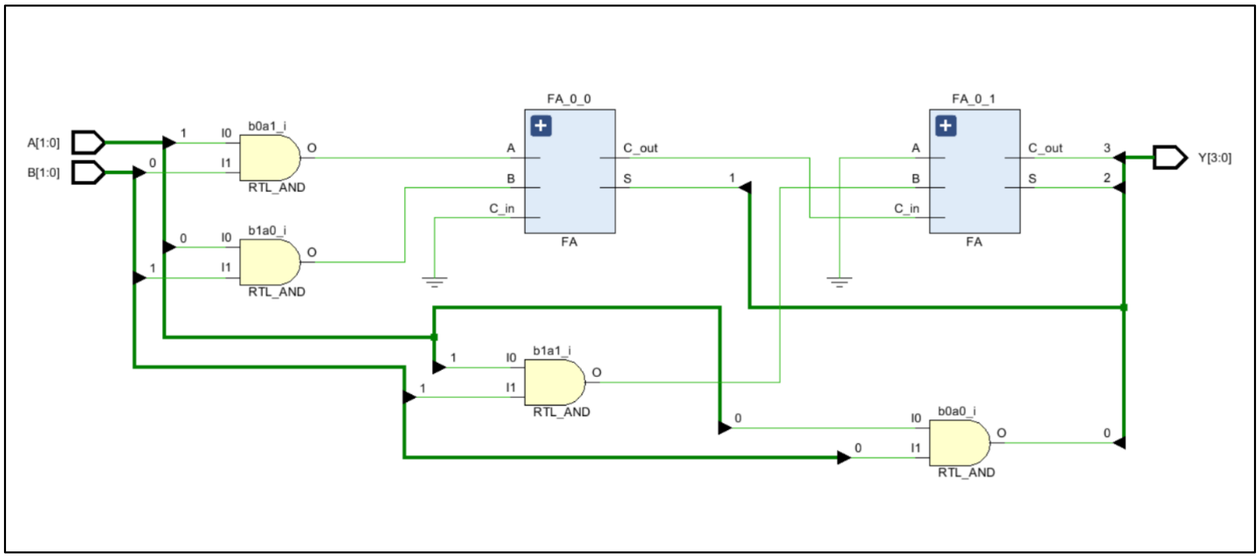
Timing Diagram



2-Bit Multiplier

The 2-bit multiplier is constructed by using 2 full adders.

Elaborated Design Schematic



Design Source File

```
-------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 02/27/2024 02:15:46 PM
-- Design Name:
-- Module Name: Multiplier_2_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
```

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


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Multiplier_2_Bit 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_Bit;

architecture Behavioral of Multiplier_2_Bit is

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

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

begin

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

FA_0_1 : FA port map (
          A => '0',
          B => b1a1,
          C_in => c_0_0,
          S => s_0_1,
          C_out => c_0_1
);


b0a0 <= A(0) AND B(0);
b0a1 <= A(1) AND B(0);
```

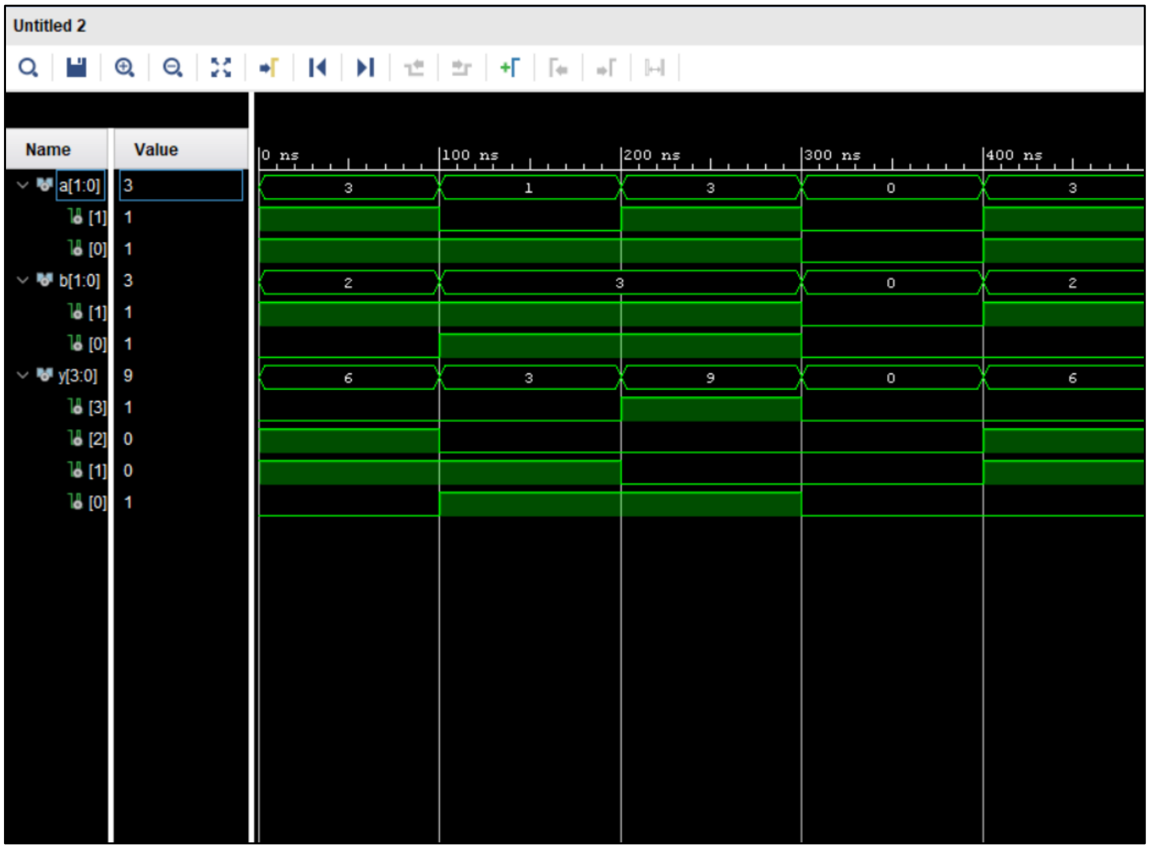
```vhdl
b1a0 <= A(0) AND B(1);
b1a1 <= A(1) AND B(1);

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

end Behavioral;
```

Simulation Source File

```vhdl
----------------------------------------------------------------------------------
-----
-- Company:
-- Engineer:
--
-- Create Date: 05/02/2024 11:30:07 AM
-- Design Name:
-- Module Name: TB_Multiplier_2_Bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity TB_Multiplier_2_Bit is
--  Port ( );
end TB_Multiplier_2_Bit;
```

```
architecture Behavioral of TB_Multiplier_2_Bit is

    Component Multiplier_2_Bit
    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_Bit
    Port Map (A => a,
              B => b,
              Y => y);

    Process begin

    a <= "11";   b <= "10"  ; Wait for 100 ns;
    a <= "01";   b <= "11"  ; Wait for 100 ns;
    a <= "11";   b <= "11"  ; Wait for 100 ns;
    a <= "00";   b <= "00"  ; Wait for 100 ns;

    End process;
end Behavioral;
```

Timing Diagram



Testing Out Newly Added Features and Instructions

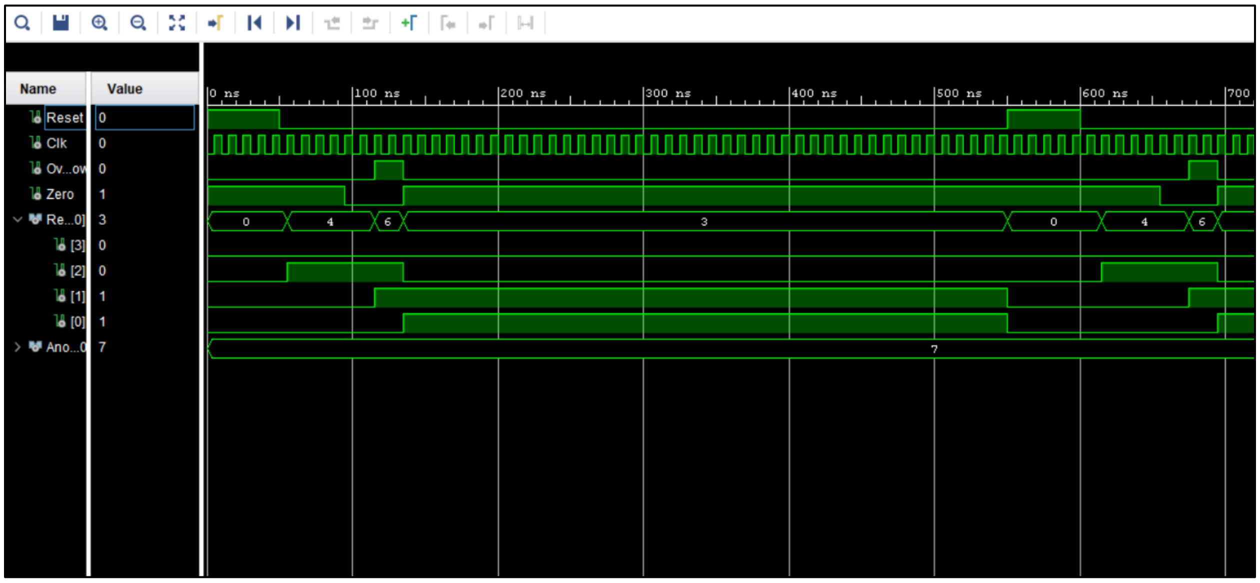
To test out the newly added features and instructions, we created 3 different LUTs and tested them out separately and got the expected results.

1. ADD, SUB, MOVI and JMP Instructions

Lookup Table

```
signal instruction_ROM : rom_type := (
    --ADD and SUB
    "00101110000001", --MOVI R7, 4
    "00101100000010", --MOVI R6, 2
    "00101010000011", --MOVI R5, 3
    "00001111100000", --ADD R7, R6
    "01001111010000", --SUB R7, R5
    "00000000000000", --Loop
    "10110000000101"
);
```

Timing Diagram



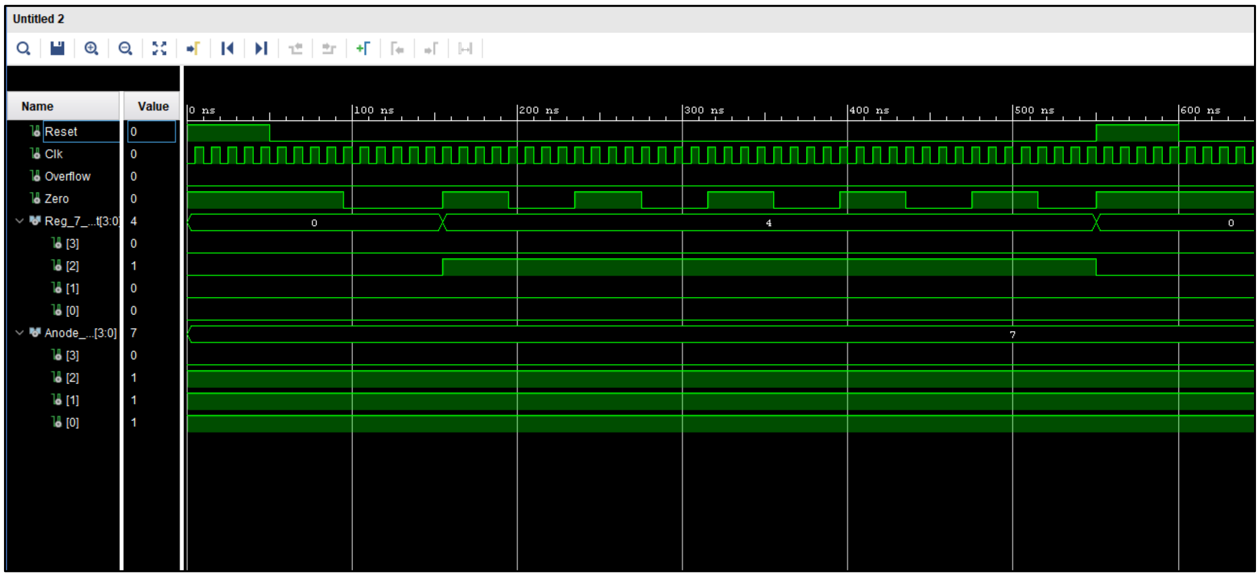
2. MAX, MIN and MOV Instructions

Max, Min instructions use the newly added comparator.

Lookup Table

```
signal instruction_ROM : rom_type := (
    --MAX and MIN
    "00101100000010", --MOVI R6, 2
    "00101010000100", --MOVI R5, 4
    "00101000001010", --MOVI R4, 10
    "10001101010000", --MAX R6, R5
    "10011101000000", --MIN R6, R4
    "10101111100000", --MOV R7, R6
    "00000000000000", --Loop
    "10110000000100"
);
```

Timing Diagram



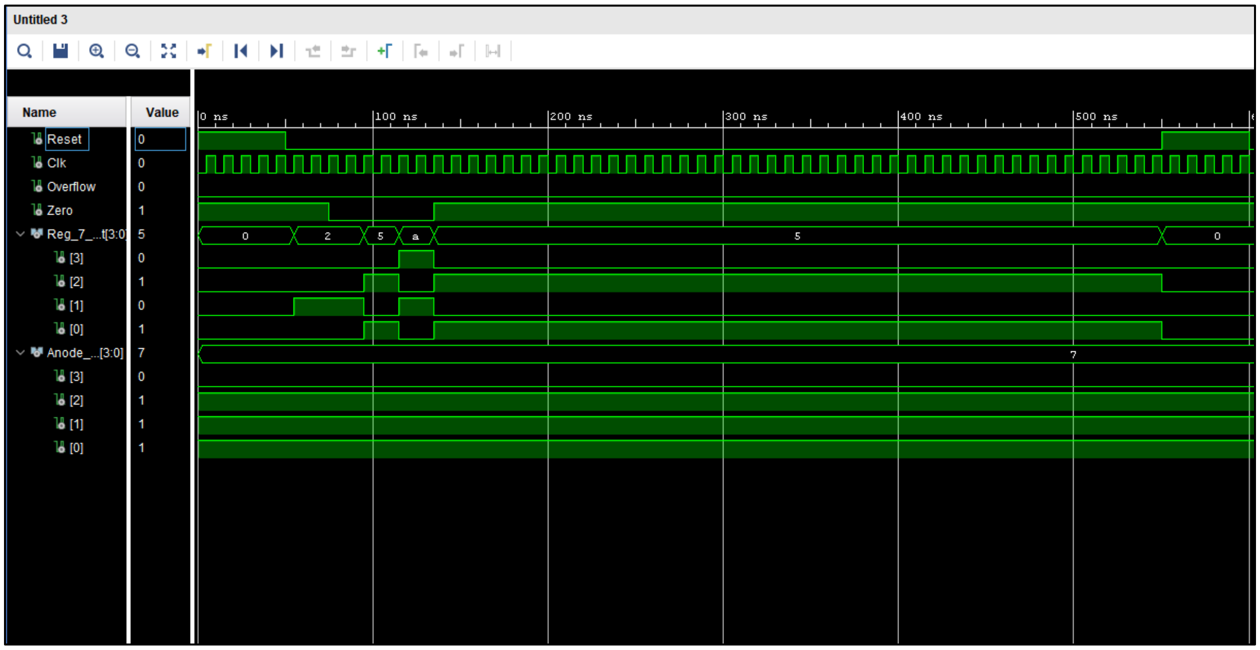
3. SHL and SHR Instructions

SHL and SHR instructions employ the newly added bit shifter.

Lookup Table

```
signal instruction_ROM : rom_type := (
    --SHL and SHR
    "00101110000010", --MOVI R7, 2
    "00101100000011", --MOVI R6, 3
    "00001111100000", --ADD R7, R6
    "01101110000000", --SHL R7
    "01111110000000", --SHR R7
    "00000000000000", --Loop
    "10110000000101"
);
```

Timing Diagram



Optimizing Resource Consumption and Resource Utilization

To enhance resource consumption and utilization, we implemented a series of measures.

- Rather than directly employing a three-to-eight decoder, we opted for its implementation using two 2-to-4 decoders.
- Furthermore, for the instruction decoder, we utilized a “switch case” approach instead of employing an “if-else-else if” block.
- Additionally, to construct an eight-way four-bit multiplexer, we employed seven two-way four-bit multiplexers, resulting in a further reduction in logic gate costs.

Conclusion

- We have successfully implemented a nanoprocessor which can execute up to 4 instructions namely, ADD, NEG, MOVI and JZR by assembling various components.
- We wrote a machine code which mirrors assembly code for summing integers between 1 to 3 and storing them in the R7 register.
- We successfully generated the bitstream files and obtained the anticipated results using the Basys 3 board.
- We extended the nanoprocessor to handle up to 12 instructions and successfully implemented them on the Basys 3 board.

Contributions

- Talagala S.A.
    - Designed the 3-bit program counter, k-way b-bit multiplexers and register bank.
    - Assembled all the components.
    - Created test bench files for some of the components.

    - Handled the documentation.

- Gunawardana S.D.M.D.
    - Designed the instruction decoder, created test bench files for most of the components.
    - Improved the nanoprocessor by adding all the additional features.
    - Handled the hardware implementation.

- Samarasinghe Y.B.P.
    - Designed the program ROM.
    - Assembled all the components.
    - Helped with the hardware implementation.

- Arachchi K.A.K.N.K.
    - Designed the 4-bit add/sub unit, 3-bit adder.
    - Helped with the hardware implementation.