Lab 9,10 - Nano Processor design Competition(2024) - Group 49

❖ Group Members:

➔ 220278E- Jayathunga W.M.J.S.
➔ 220267U- Jayasooriya J.A.N.S.
➔ 220282K- Jayawardhana W.S.S.
➔ 220585R- Sathsara H.M.W.C.

❖ Contents

Goto Contents

❖ Lab task

- Design a 4-bit processor capable of executing 4 instructions.

    (01) MOVI R, d - Move immediate value d to register R
    (02) ADD Ra, Rb - Add values in registers Ra and Rb and store the result in Ra
    (03) NEG R - 2's complement of register R
    (04) JZR R, d - Jump if value in register R is 0

➢ To build this circuit, we develop following components first

- 4-bit Add/Subtract unit
- 3-bit adder
- 3-bit Program Counter (PC)
- 2-way 3-bit multiplexer
- 2-way 4-bit multiplexer
- 8-way 4-bit multiplexer
- Register Bank
- Program ROM
- Instruction Decoder

★ We use 3, 4, and 12-bit buses to connect components.

Goto Contents

★ Build the necessary sub-components.

★ Test each component using simulation.

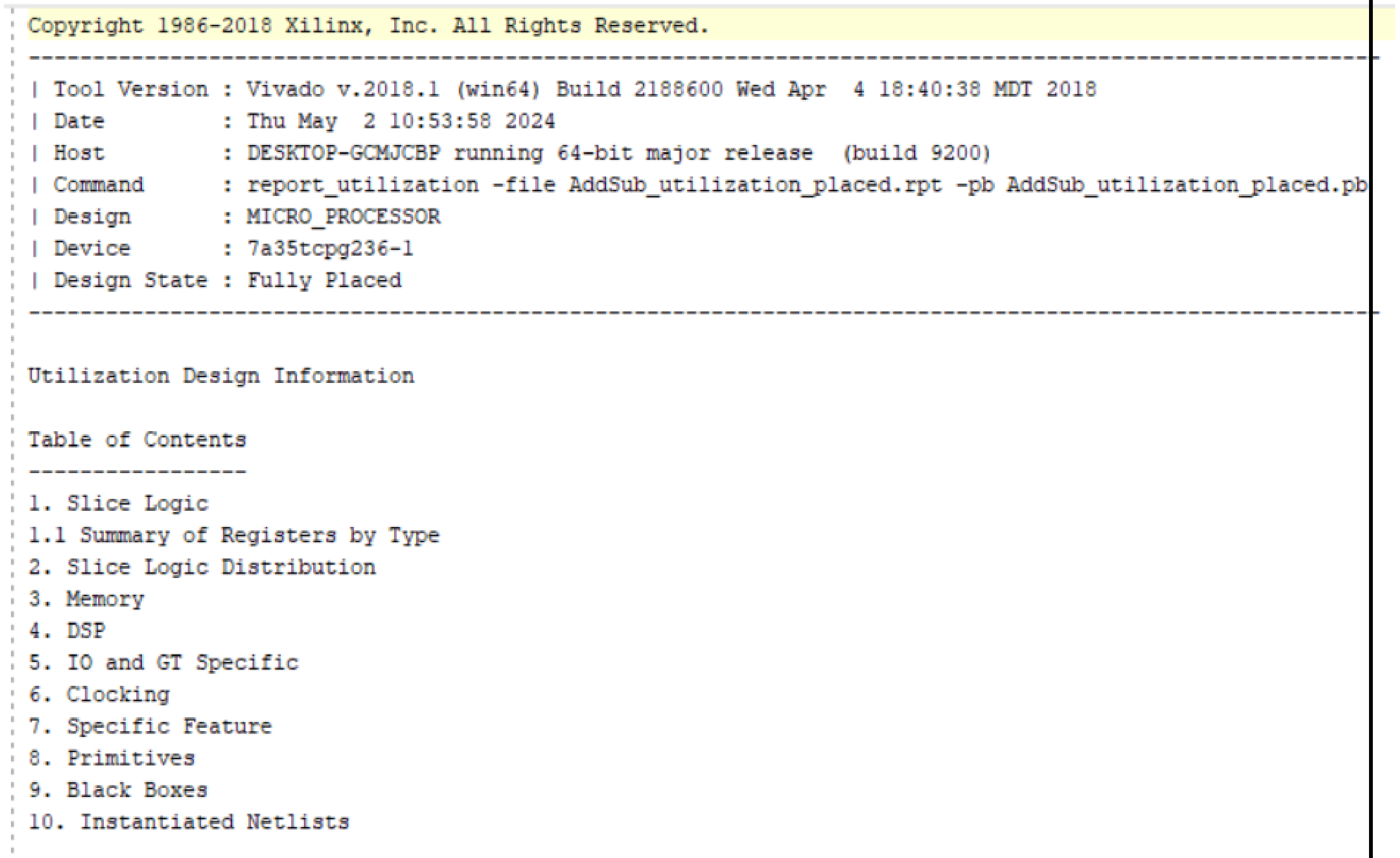
★ Build the top-level design and test using simulation.

★ Test on BASYS 3.

★ Verify the functionalities.

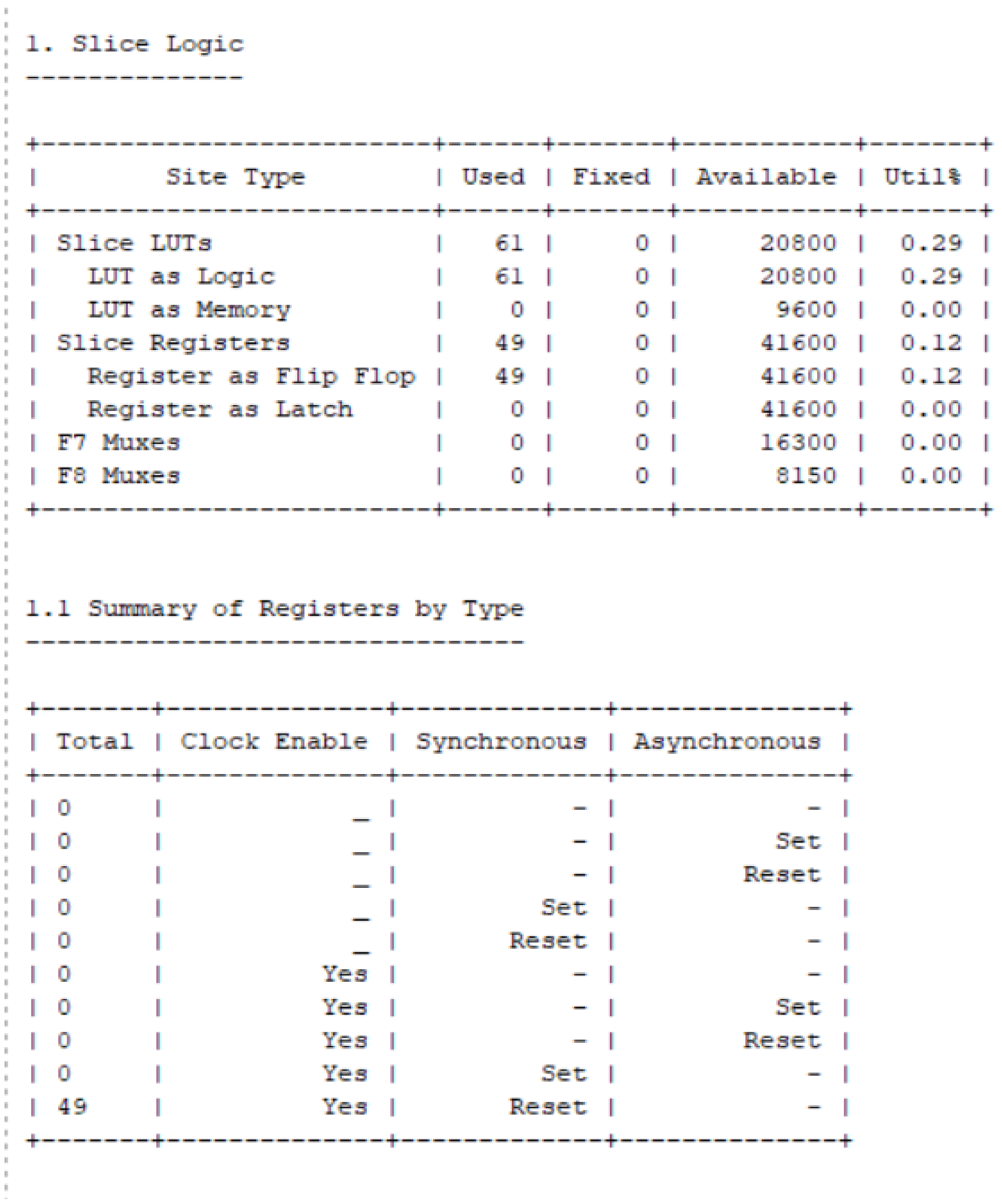
❖ Resource Utilization and Optimization

- Resource utilization design information

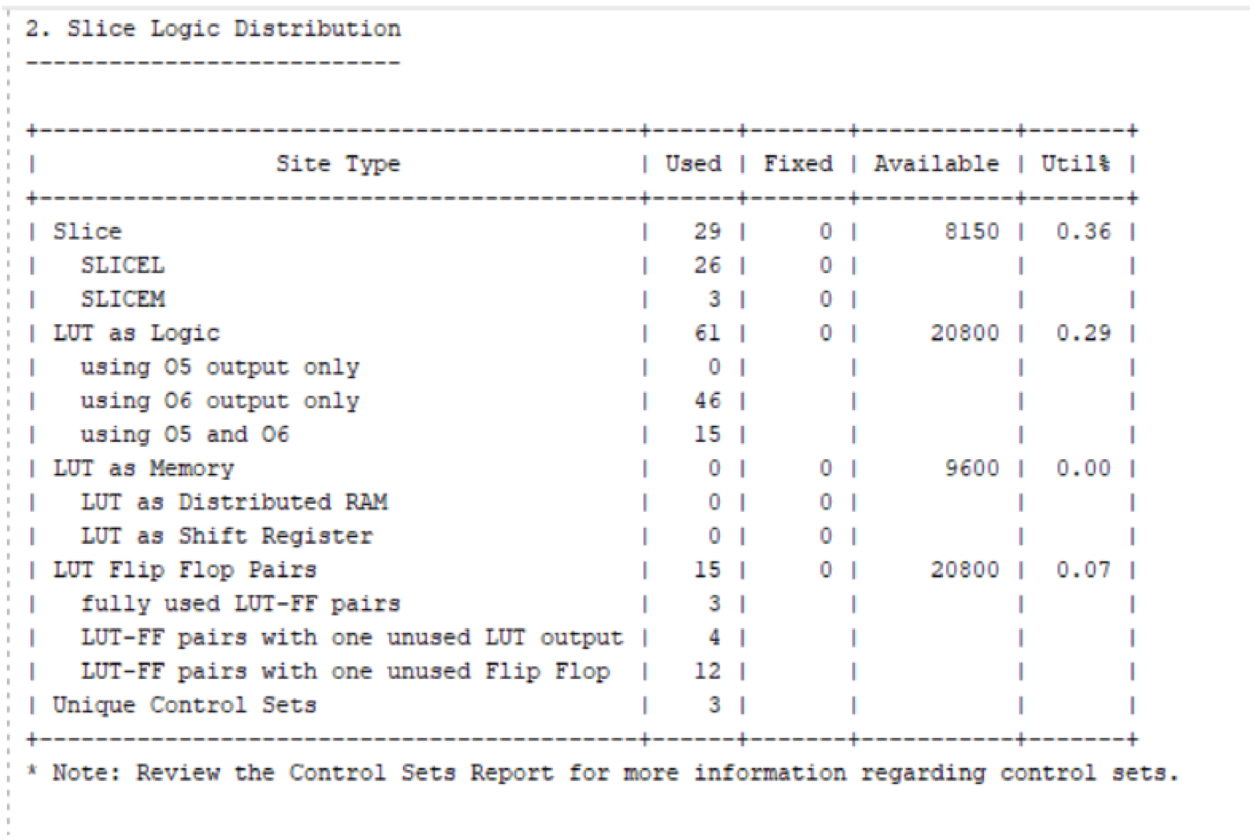


```
Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2018.1 (win64) Build 2188600 Wed Apr  4 18:40:38 MDT 2018
| Date         : Thu May  2 10:53:58 2024
| Host         : DESKTOP-GCMJCBP running 64-bit major release  (build 9200)
| Command      : report_utilization -file AddSub_utilization_placed.rpt -pb AddSub_utilization_placed.pb
| Design       : MICRO_PROCESSOR
| Device       : 7a35tcpg236-1
| Design State : Fully Placed
------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists
```

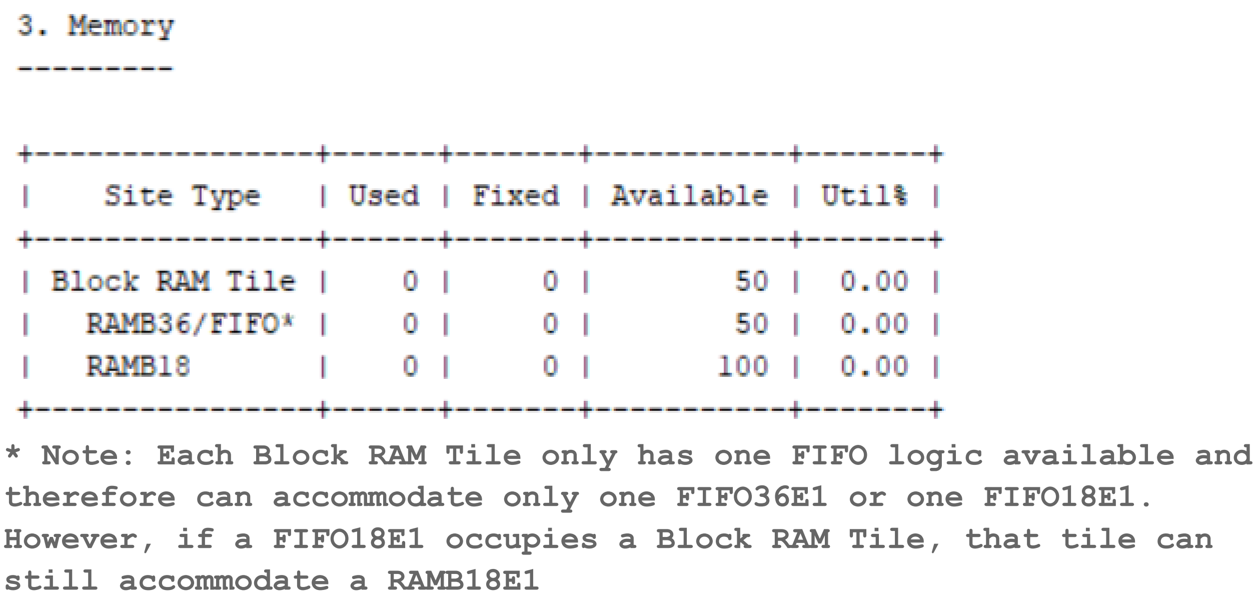
Goto Contents



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

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              |   61 |     0 |     20800 |  0.29 |
|   LUT as Logic          |   61 |     0 |     20800 |  0.29 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |   49 |     0 |     41600 |  0.12 |
|   Register as Flip Flop |   49 |     0 |     41600 |  0.12 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+


1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 49    |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+
```

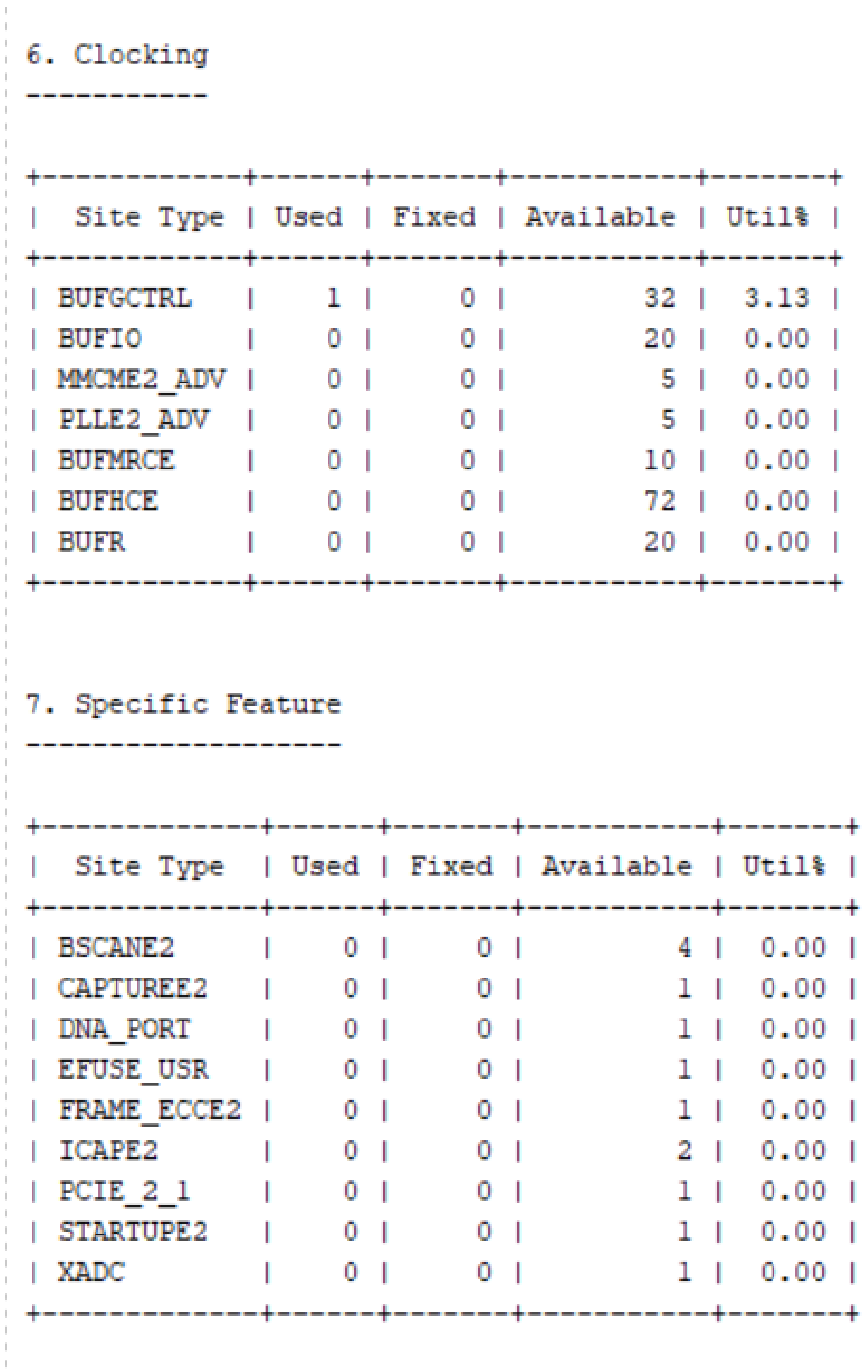


```
2. Slice Logic Distribution
---------------------------

+-------------------------------------------+------+-------+-----------+-------+
|                 Site Type                 | Used | Fixed | Available | Util% |
+-------------------------------------------+------+-------+-----------+-------+
| Slice                                     |   29 |     0 |      8150 |  0.36 |
|   SLICEL                                  |   26 |     0 |           |       |
|   SLICEM                                  |    3 |     0 |           |       |
| LUT as Logic                              |   61 |     0 |     20800 |  0.29 |
|   using O5 output only                    |    0 |       |           |       |
|   using O6 output only                    |   46 |       |           |       |
|   using O5 and O6                         |   15 |       |           |       |
| LUT as Memory                             |    0 |     0 |      9600 |  0.00 |
|   LUT as Distributed RAM                  |    0 |     0 |           |       |
|   LUT as Shift Register                   |    0 |     0 |           |       |
| LUT Flip Flop Pairs                       |   15 |     0 |     20800 |  0.07 |
|   fully used LUT-FF pairs                 |    3 |       |           |       |
|   LUT-FF pairs with one unused LUT output |    4 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop  |   12 |       |           |       |
| Unique Control Sets                       |    3 |       |           |       |
+-------------------------------------------+------+-------+-----------+-------+
* Note: Review the Control Sets Report for more information regarding control sets.
```



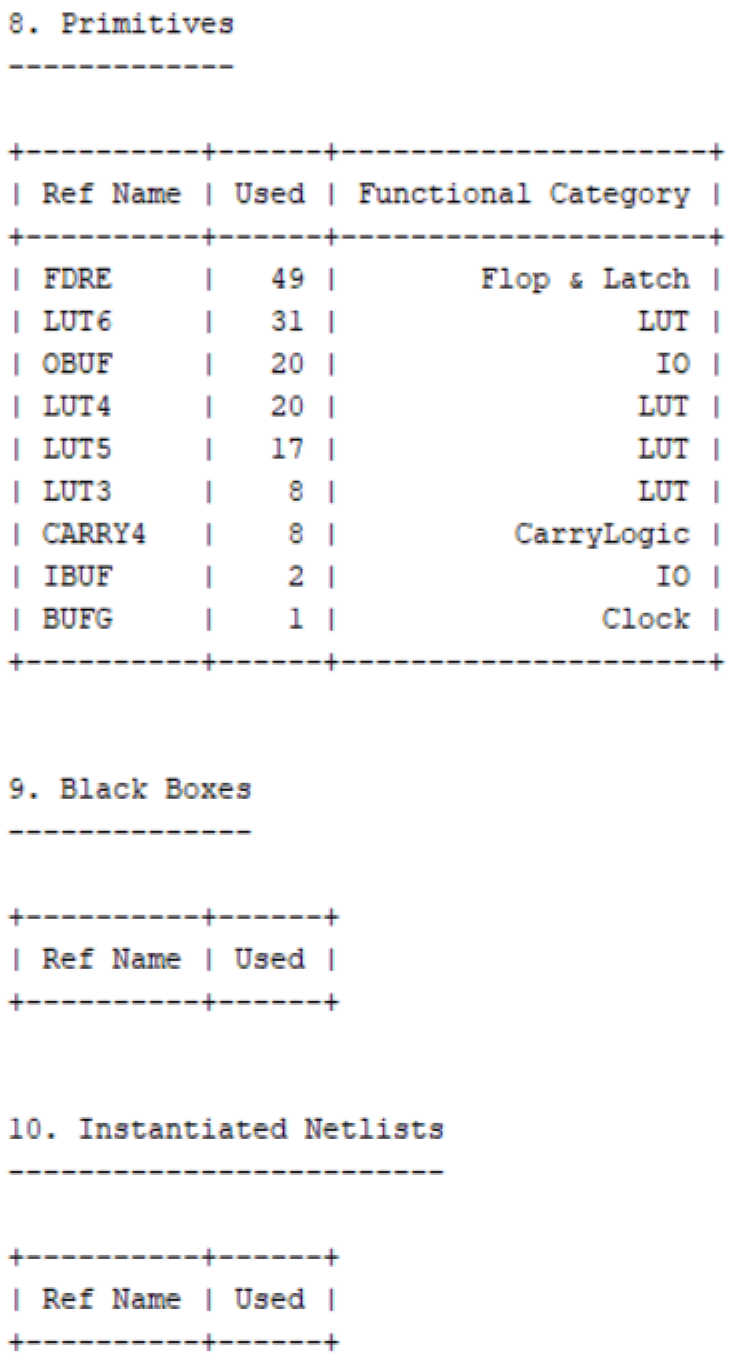
```
3. Memory
---------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| Block RAM Tile |    0 |     0 |        50 |  0.00 |
|   RAMB36/FIFO* |    0 |     0 |        50 |  0.00 |
|   RAMB18       |    0 |     0 |       100 |  0.00 |
+----------------+------+-------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and
therefore can accommodate only one FIFO36E1 or one FIFO18E1.
However, if a FIFO18E1 occupies a Block RAM Tile, that tile can
still accommodate a RAMB18E1
```

```
4. DSP
------

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs      |    0 |     0 |        90 |  0.00 |
+-----------+------+-------+-----------+-------+


5. IO and GT Specific
---------------------

+-----------------------------+------+-------+-----------+-------+
|          Site Type          | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
| Bonded IOB                  |   22 |    22 |       106 | 20.75 |
|   IOB Master Pads           |    8 |       |           |       |
|   IOB Slave Pads            |   13 |       |           |       |
| Bonded IPADs                |    0 |     0 |        10 |  0.00 |
| Bonded OPADs                |    0 |     0 |         4 |  0.00 |
| PHY_CONTROL                 |    0 |     0 |         5 |  0.00 |
| PHASER_REF                  |    0 |     0 |         5 |  0.00 |
| OUT_FIFO                    |    0 |     0 |        20 |  0.00 |
| IN_FIFO                     |    0 |     0 |        20 |  0.00 |
| IDELAYCTRL                  |    0 |     0 |         5 |  0.00 |
| IBUFDS                      |    0 |     0 |       104 |  0.00 |
| GTPE2_CHANNEL               |    0 |     0 |         2 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY   |    0 |     0 |        20 |  0.00 |
| PHASER_IN/PHASER_IN_PHY     |    0 |     0 |        20 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |       250 |  0.00 |
| IBUFDS_GTE2                 |    0 |     0 |         2 |  0.00 |
| ILOGIC                      |    0 |     0 |       106 |  0.00 |
| OLOGIC                      |    0 |     0 |       106 |  0.00 |
+-----------------------------+------+-------+-----------+-------+
```



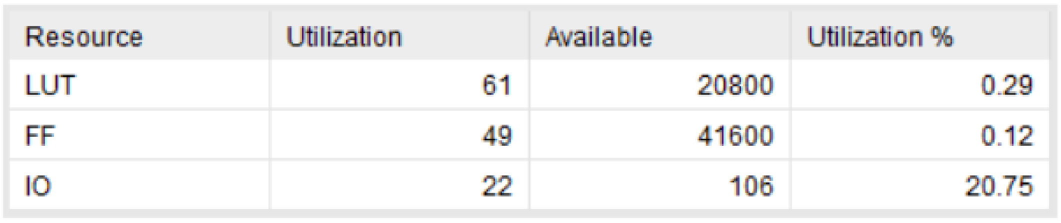
```
6. Clocking
-----------

+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BUFGCTRL    |    1 |     0 |        32 |  3.13 |
| BUFIO       |    0 |     0 |        20 |  0.00 |
| MMCME2_ADV  |    0 |     0 |         5 |  0.00 |
| PLLE2_ADV   |    0 |     0 |         5 |  0.00 |
| BUFMRCE     |    0 |     0 |        10 |  0.00 |
| BUFHCE      |    0 |     0 |        72 |  0.00 |
| BUFR        |    0 |     0 |        20 |  0.00 |
+-------------+------+-------+-----------+-------+


7. Specific Feature
-------------------

+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BSCANE2     |    0 |     0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |         2 |  0.00 |
| PCIE_2_1    |    0 |     0 |         1 |  0.00 |
| STARTUPE2   |    0 |     0 |         1 |  0.00 |
| XADC        |    0 |     0 |         1 |  0.00 |
+-------------+------+-------+-----------+-------+
```

Goto Contents

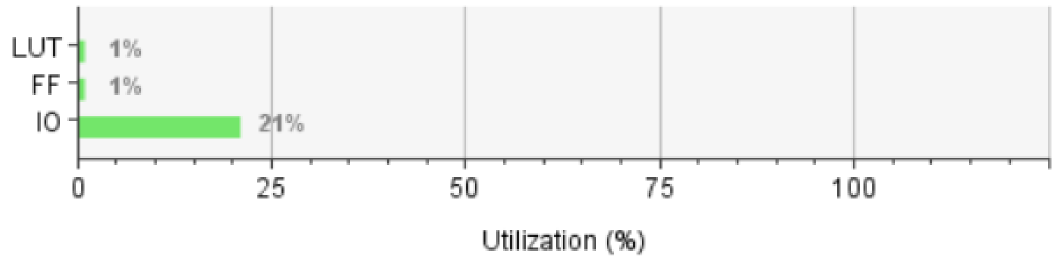


```
8. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| FDRE     |   49 |        Flop & Latch |
| LUT6     |   31 |                 LUT |
| OBUF     |   20 |                  IO |
| LUT4     |   20 |                 LUT |
| LUT5     |   17 |                 LUT |
| LUT3     |    8 |                 LUT |
| CARRY4   |    8 |          CarryLogic |
| IBUF     |    2 |                  IO |
| BUFG     |    1 |               Clock |
+----------+------+---------------------+


9. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


10. Instantiated Netlists
-------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```

- Resource utilization diagrams



Resource	Utilization	Available	Utilization %
LUT	61	20800	0.29
FF	49	41600	0.12
IO	22	106	20.75



- Strategies used to optimized resource consumption

  ❖ Using multiplexers instead of buffers. While multiplexers can often reduce resource utilization by optimizing logic sharing, buffers are essential for signal integrity and driving signals across the FPGA.

  ❖ In the 4-bit Add/Subtract unit we used 4 bit ripple carry adder because of its simplicity, low logic utilization, and minimal routing utilization. It's efficient and suitable for small adders when compared with carry lookahead adder which has unnecessary complexity and resource overhead for a small 4-bit adder.

❖ Components of Basic Design

• 4-bit Add/Subtract unit

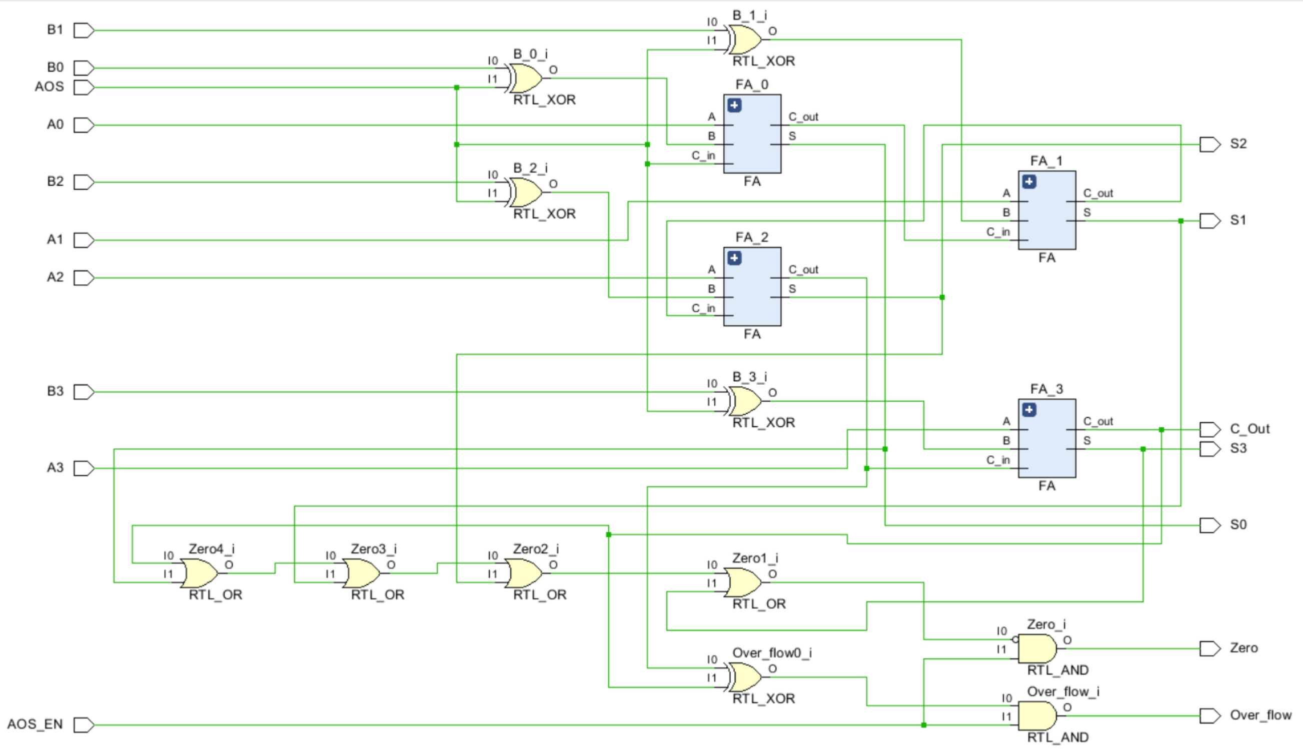
❖ This unit is capable of doing add and subtract operations. It is created using full address and half adders. Subtract operation is implemented using two's complement. It flip data value and add with other value.

❖ When subtract using for add AOS signal become 1,Then ,

B_0<=B0 XOR AOS;
B_1<=B1 XOR AOS;
B_2<=B2 XOR AOS;
B_3<=B3 XOR AOS;

- If AOS =1, B values are change to its negation.
- Also AOS_EN use to off Zero, Over flow when component not working.

Elaborated design



VHDL code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/02/2024 01:45:49 PM
-- Design Name:
-- Module Name: AddSub - 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 AddSub is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           --C_in : in STD_LOGIC;
           AOS: in STD_LOGIC;
           AOS_EN : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_Out : out STD_LOGIC;
           Zero : out STD_LOGIC;
           Over_flow : out STD_LOGIC);
end AddSub;

architecture Behavioral of AddSub is
component FA
 port (
 A: in std_logic;
 B: in std_logic;
 C_in: in std_logic;
 S: out std_logic;
 C_out: out std_logic);
 end component;

 SIGNAL  FA0_C,  FA1_C, FA2_C, C : std_logic;
 SIGNAL B_1,B_2,B_0,B_3:STD_LOGIC;
 SIGNAL S_1,S_2,S_0,S_3:STD_LOGIC;

begin
--AS<=AOS;
B_0<=B0 XOR AOS;
B_1<=B1 XOR AOS;
B_2<=B2 XOR AOS;
B_3<=B3 XOR AOS;

S0<=S_0;
S1<=S_1;
S2<=S_2;
S3<=S_3;



FA_0 : FA
 port map (
 A => A0,
 B => B_0,
 C_in => AOS, -- Set to ground
 S => S_0,
 C_Out => FA0_C);

FA_1 : FA
  port map (
  A => A1,
  B => B_1,
  C_in => FA0_C,
  S => S_1,
  C_Out => FA1_C);

  FA_2 : FA
```

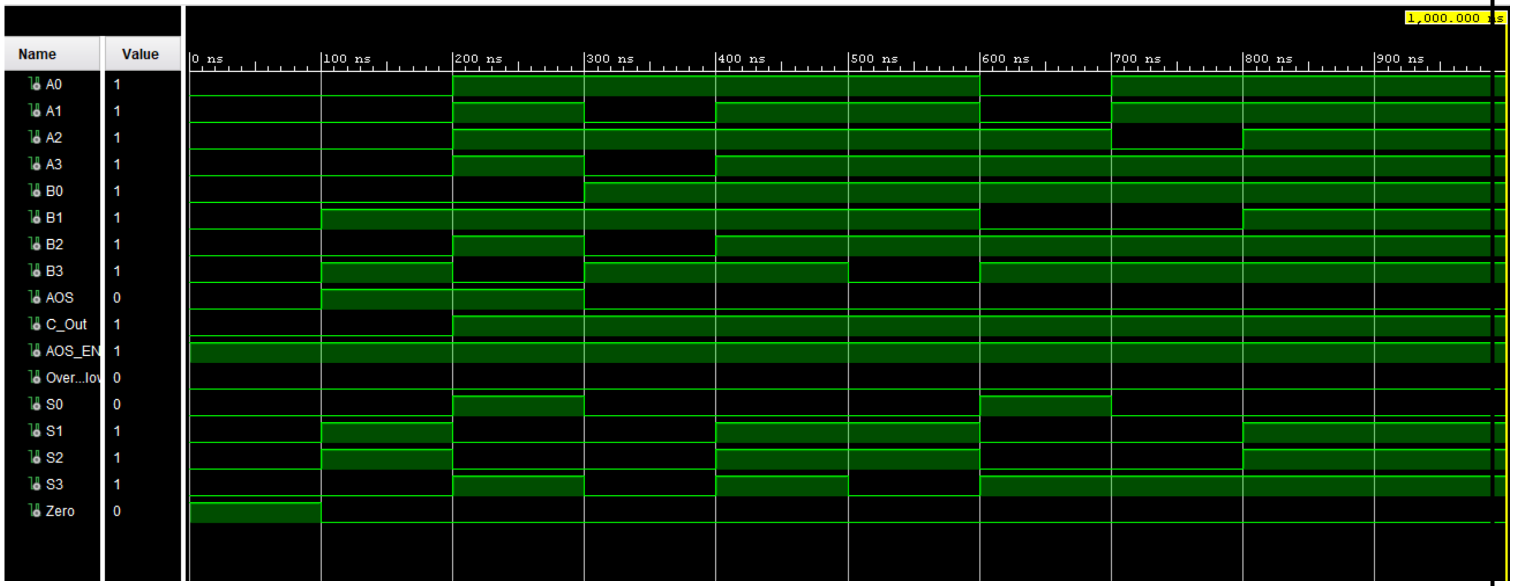
```
   port map (
   A => A2,
   B => B_2,
   C_in => FA1_C,
   S => S_2,
   C_Out => FA2_C);

   FA_3 : FA
    port map (
    A => A3,
    B => B_3,
    C_in => FA2_C,
    S => S_3,
    C_Out => C);
C_out<=C;
Over_flow<=((FA2_C XOR C) AND AOS_EN);
Zero<=(Not( C OR S_0 OR S_1 OR S_2 OR S_3) AND AOS_EN);

end Behavioral;
```

AddSub_Tb.vhd

```
----------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 02/13/2024 06:24:58 PM
-- Design Name:
-- Module Name: TB_RCA_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 AddSub_TB is
--  Port ( );
end AddSub_TB;

architecture Behavioral of AddSub_TB is
COMPONENT AddSub
PORT( A0,A1,A2,A3,B0,B1,B2,B3,AOS,AOS_EN : IN STD_LOGIC;
      S0,S1,S2,S3,Over_flow,C_Out,Zero : OUT STD_LOGIC);
END COMPONENT;

SIGNAL A0,A1,A2,A3,B0,B1,B2,B3,AOS,C_Out,AOS_EN : STD_LOGIC;
SIGNAL Over_flow,S0,S1,S2,S3,Zero : STD_LOGIC;

begin
UUT : AddSub PORT MAP(
   A0=>A0,
   A1 => A1,
   A2 => A2,
   A3 => A3,
   AOS=>AOS,
   AOS_EN=>AOS_EN,
   B0=>B0,
   B1 => B1,
   B2 => B2,
   B3 => B3,
  -- C_in =>C_in,
   S0=>S0,
   S1 => S1,
   S2 => S2,
   S3 => S3,
   C_Out=>C_Out,
   Zero=>Zero,
   Over_flow => Over_flow);
PROCESS
      BEGIN
      AOS_EN<='1';
      AOS<='0';
        A0 <= '0';
        A1 <= '0';
        A2 <= '0';
        A3 <= '0';
        B0 <= '0';
        B1 <='0';
        B2<='0';
        B3<='0';
       -- C_in <= '0';

        WAIT FOR 100 ns;
        AOS<='1';
        A0<='0';
        A1<='0';
        A2<='0';
        A3<='0';
        B0 <='0';
        B1 <='1';
        B2 <='0';
        B3 <='1';

        WAIT FOR 100 ns;
           AOS<='1';
           A0<='1';
           A1<='1';
           A2<='1';
           A3<='1';
           B0 <='0';
           B1 <='1';
           B2 <='1';
           B3 <='0';

        WAIT FOR 100 ns;
        AOS<='0';
               A0<='1';
               A1<='0';
               A2<='1';
               A3<='0';
               B0 <='1';
               B1 <='1';
               B2 <='0';
               B3 <='1';

        WAIT FOR 100 ns;
        AOS<='0';
               A0<='1';
               A1<='1';
               A2<='1';
               A3<='1';
               B0 <='1';
               B1 <='1';
               B2 <='1';
               B3 <='1';

        WAIT FOR 100ns;
        AOS<='0';

               A0<='1';
               A1<='1';
               A2<='1';
               A3<='1';
               B0 <='1';
               B1 <='1';
               B2 <='1';
               B3 <='0';

        WAIT FOR 100 ns;
        AOS<='0';
               A0<='0';
```

```
              A1<='0';
              A2<='1';
              A3<='1';
              B0 <='1';
              B1 <='0';
              B2 <='1';
              B3 <='1';
       WAIT FOR 100 ns;
       AOS<='0';
                A0<='1';
                A1<='1';
                A2<='0';
                A3<='1';
                B0 <='1';
                B1 <='0';
                B2 <='1';
                B3 <='1';
        WAIT FOR 100 ns;
        AOS<='0';
              A0<='1';
              A1<='1';
              A2<='1';
              A3<='1';
              B0 <='1';
              B1 <='1';
              B2 <='1';
              B3 <='1';

       WAIT;
       END PROCESS;

end Behavioral;
```

Timing Diagram



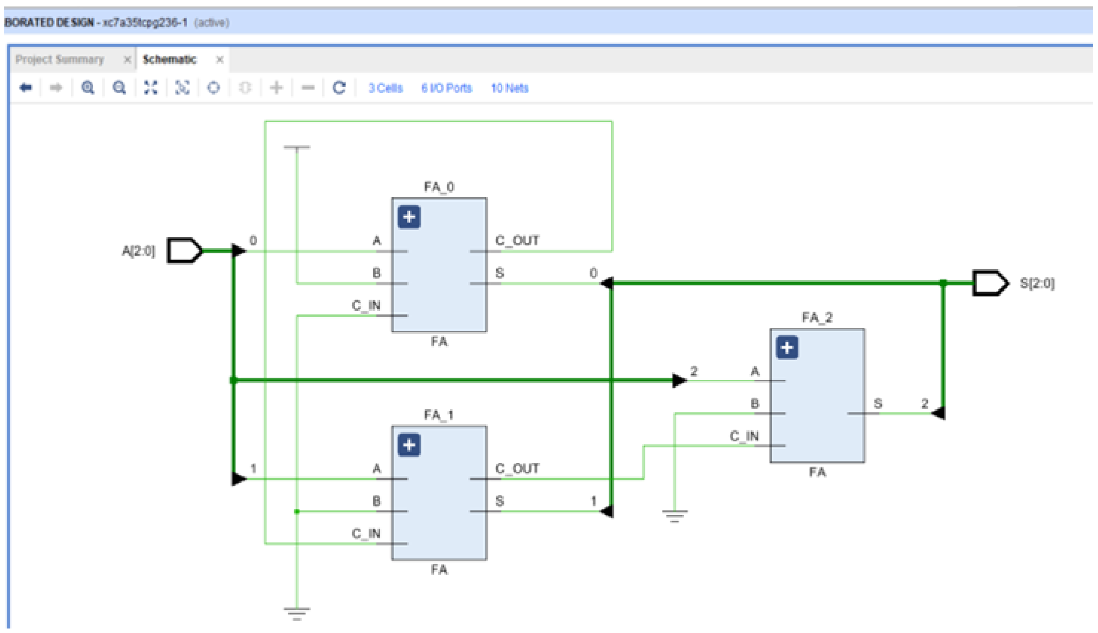
Problems when creating 4bit adder and subtractor and solution for them

Problems	Solutions
Zore and Overflow Show unnecessary outputs when the component is not in use.	Add a buffer to output . “AOS_EN” signal controls Zero and Overflow flags .

3-bit adder

❖ 3-bit adder is a pivotal component designed to increment the Program Counter (PC) . By adapting the 4-bit Ripple Carry Adder (RCA) from Lab 3, one can create a streamlined 3-bit version tailored for this purpose. The adder’s function is to accurately adjust the PC, ensuring it points to the next instruction in ROM.

Elaborated design

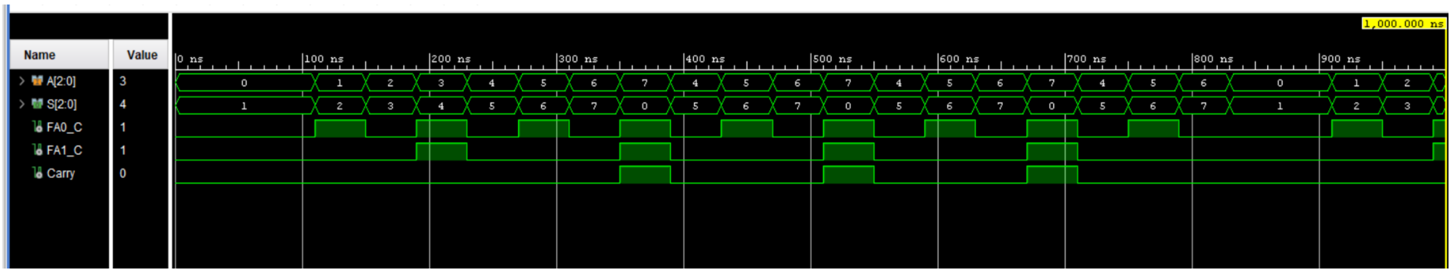


VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/02/2024 02:16:38 PM
-- Design Name:
-- Module Name: Adder_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 Adder_3_bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0));
end Adder_3_bit;

architecture Behavioral of Adder_3_bit is
component FA
    port (
    A: in std_logic;
    B: in std_logic;
    C_in: in std_logic;
    S: out std_logic;
    C_out: out std_logic);
end component;
SIGNAL FA0_C,  FA1_C,Carry : std_logic;

begin
    FA_0 : FA
    port map (
        A => A(0),
        B => '1',
        C_in => '0', -- Set to ground
        S => S(0),
        C_Out => FA0_C);

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

FA_2 : FA
    port map (
        A => A(2),
        B => '0',
        C_in => FA1_C,
        S => S(2),
        C_Out => Carry);

end Behavioral;
```

Timing Diagram



3-Bit Adder test bench file

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/18/2024 04:15:16 PM
-- Design Name:
-- Module Name: Adder_3_bit_TB - 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_3_bit_TB is
--  Port ( );
end Adder_3_bit_TB;

architecture Behavioral of Adder_3_bit_TB is
component Adder_3_bit
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal A,S : STD_LOGIC_VECTOR (2 downto 0);
begin
UUT : Adder_3_bit
port map(
    A => A,
    S => S);

process
begin

    A <= "000";
    wait for 100ns;
```

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

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

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

    end Behavioral;
```

● 3-bit Program Counter (PC)

❖ Used to keep track of memory address of next instruction to be executed.

❖ Use 3 D flipflops with a common clock.

❖ All bits stored in the register at the same clock signal.

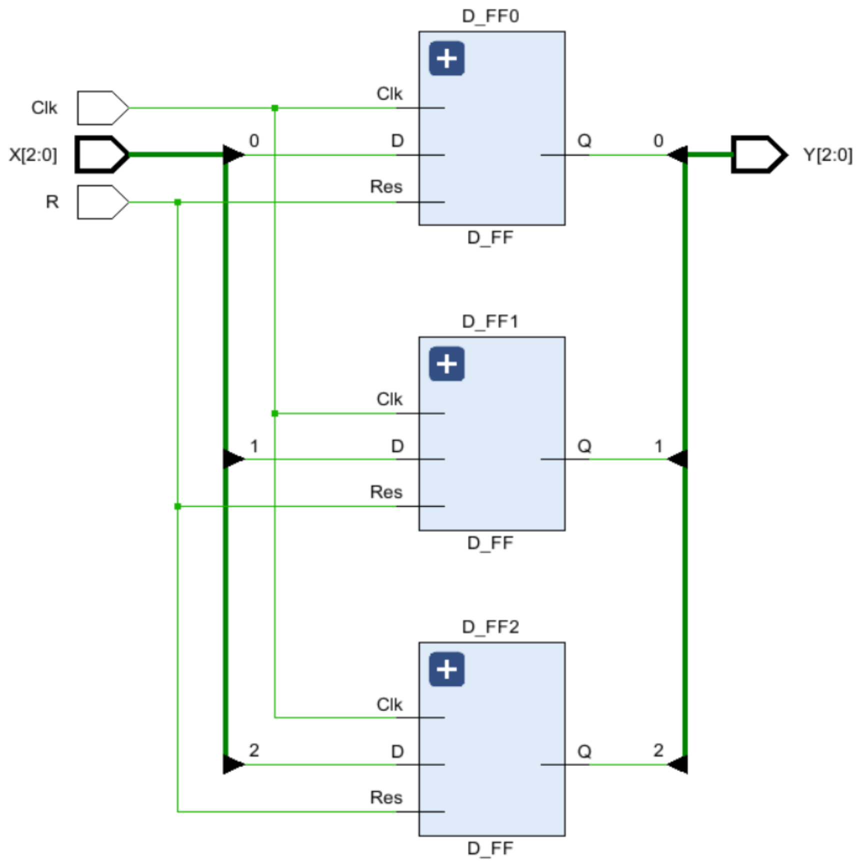
❖ The next instruction address coming from 2-way 3-bit Mux as a 3 bit Bus and it store in PC.

❖ For next clock signal PC output the next instruction address and it goes to Program ROM and 3-bit Adder

❖ When Reset button push, the PC goes to the start of the program.

Elaborated design

Goto Contents



VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 03:28:46 PM
-- Design Name:
-- Module Name: PC - 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 PC is
    Port ( X : in STD_LOGIC_VECTOR (2 downto 0);
           R : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end PC;

architecture Behavioral of PC is
component D_FF
   port (
       D : in STD_LOGIC;
       Res: in STD_LOGIC;
       Clk : in STD_LOGIC;
       Q : out STD_LOGIC;
      Qbar : out STD_LOGIC
       );
   end component;

begin

     D_FF0 : D_FF
          port map (
          D => X(0),
          Res=> R,
          Clk => Clk,
          Q => Y(0));

     D_FF1 : D_FF
        port map (
        D => X(1),
        Res=> R,
        Clk => Clk,
        Q => Y(1));

    D_FF2 : D_FF
      port map (
      D => X(2),
      Res=> R,
      Clk => Clk,
      Q => Y(2));

end Behavioral;
```

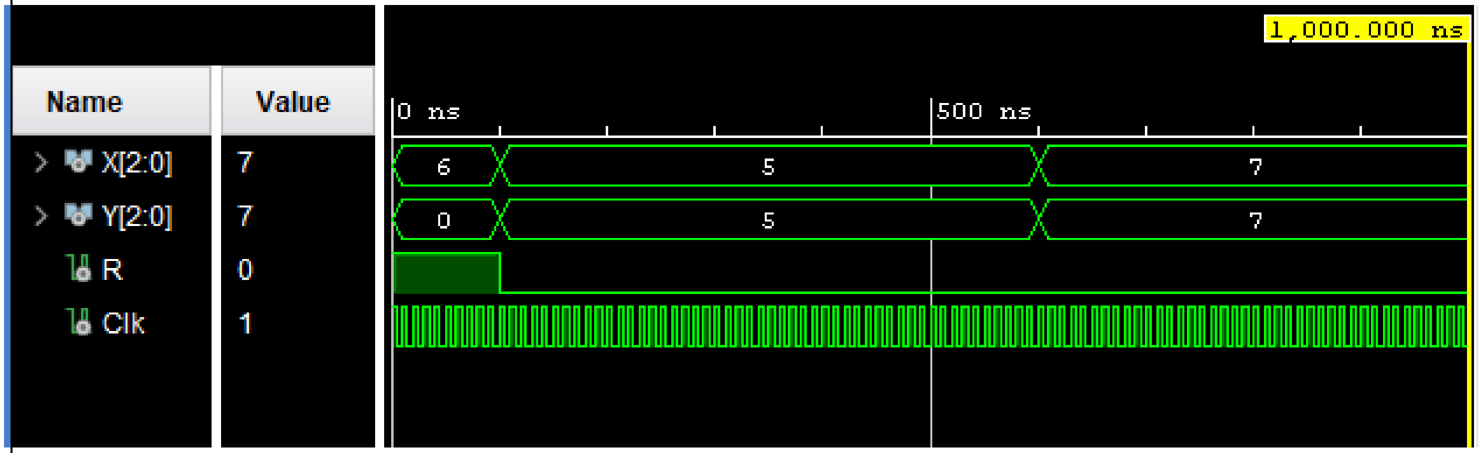
Program counter test bench file

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 03:55:48 PM
-- Design Name:
-- Module Name: PC_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 PC_SIM is
--  Port ( );
end PC_SIM;

architecture Behavioral of PC_SIM is
component  PC port(
     X : in STD_LOGIC_VECTOR (2 downto 0);
     R : in STD_LOGIC;
     Clk : in STD_LOGIC;
     Y : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal X,Y : STD_LOGIC_VECTOR (2 downto 0);
signal R,Clk : STD_LOGIC;
begin
UUT : PC port map(
        X => X,
        R => R,
        Clk => Clk,
        Y => Y);
Clock : process
            begin
              clk <= '1';
              wait for 5ns;
              clk <= '0';
              wait for 5ns;
           end process;
        process
```

```
begin
    R <= '1';
    X <= "110";
    wait for 100ns;
   R <= '0';
   X <= "101";
   wait for 500ns;
    R <= '0';
    X <= "111";
    wait for 500ns;
    end process;


end Behavioral;
```

Timing Diagram



• 2-way 3-bit multiplexer

❖ Use 3 , 2 to 1 multiplexers.

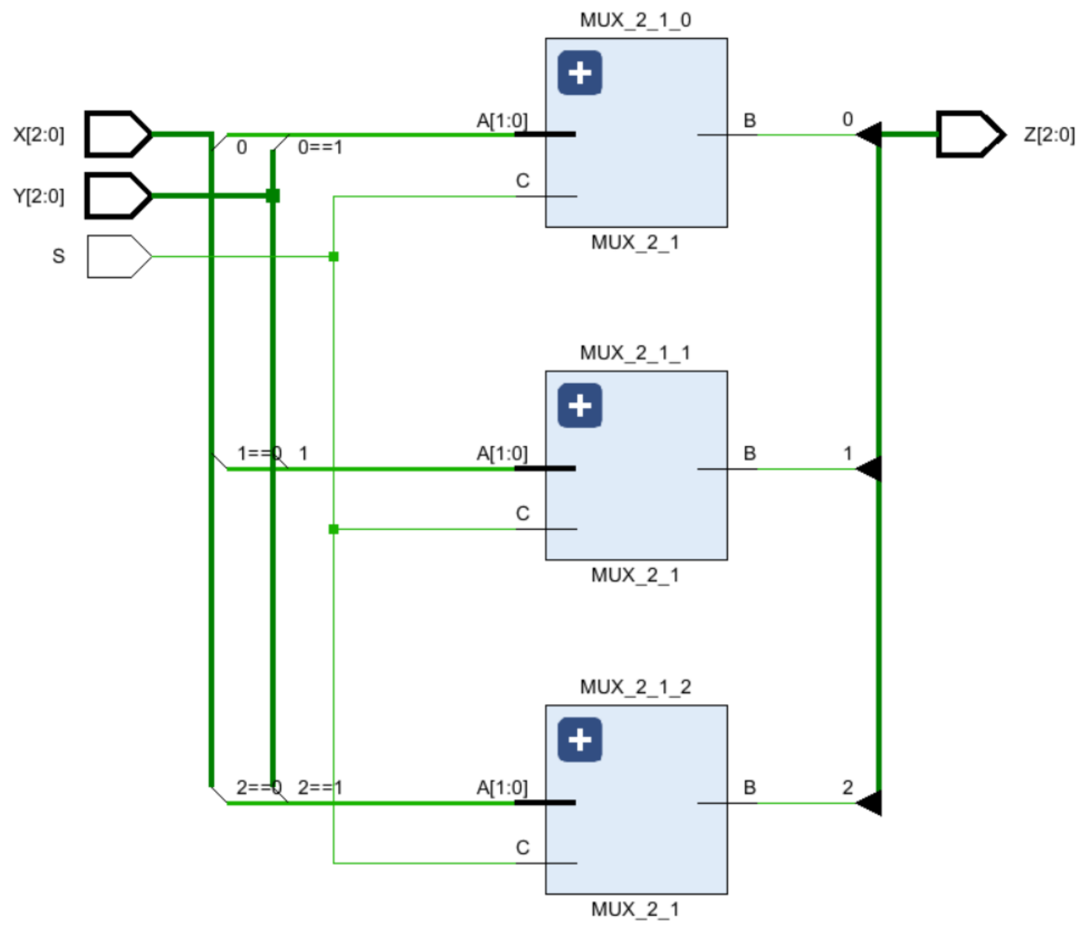
❖ If jump flag is set to 1, output the jumping register address
Else
Output the next instruction address that come from 3-bit Adder.

- Truth Table of 2 to 1 multiplexer

A(0)	A(1)	C	B
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

❖ B <= ((NOT C) AND A(0)) OR (C AND A(1))

Elaborated design



VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/02/2024 03:03:28 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 ( X : in STD_LOGIC_VECTOR (2 downto 0);
           Y : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (2 downto 0));
end MUX_2_WAY_3_Bit;

architecture Behavioral of MUX_2_WAY_3_Bit is
component MUX_2_1
   port (
       A : in STD_LOGIC_VECTOR (1 downto 0);
       B : out STD_LOGIC;
       C : in STD_LOGIC);
   end component;

begin
 MUX_2_1_0 :  MUX_2_1
          port map (
          A(0)=>X(0),
          A(1)=>Y(0),
          B=>Z(0),
          C => S);
 MUX_2_1_1 :  MUX_2_1
```

```
           port map (
           A(0)=>X(1),
           A(1)=>Y(1),
           B=>Z(1),
           C => S);
 MUX_2_1_2 :  MUX_2_1
             port map (
             A(0)=>X(2),
             A(1)=>Y(2),
             B=>Z(2),
             C => S);

end Behavioral;
```

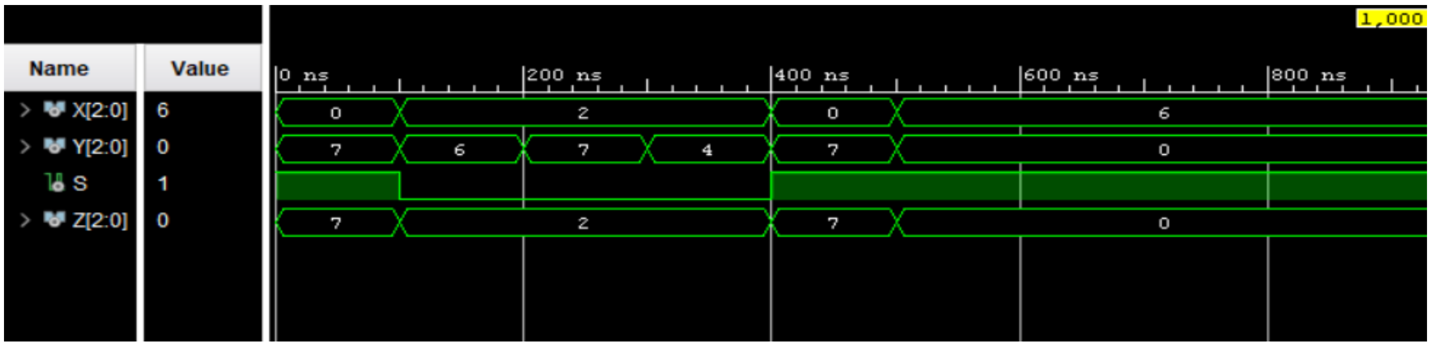
Behavioral simulation source code for 2 - way 3 bit multiplexer

```
----------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 01:59:29 PM
-- Design Name:
-- Module Name: MUX_2_WAY_3_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 MUX_2_WAY_3_SIM is
--  Port ( );
end MUX_2_WAY_3_SIM;

architecture Behavioral of MUX_2_WAY_3_SIM is
component MUX_2_WAY_3_Bit
port(
          X : in STD_LOGIC_VECTOR (2 downto 0);
          Y : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (2 downto 0)
);
end component;
signal X :  STD_LOGIC_VECTOR (2 downto 0);
signal Y :  STD_LOGIC_VECTOR (2 downto 0);
signal S :  STD_LOGIC;
signal Z :  STD_LOGIC_VECTOR (2 downto 0);
begin
UUT: MUX_2_WAY_3_Bit
port map(
    X => X,
    Z => Z,
    S => S,
    Y => Y
);
process

begin
X <= "000";
Y <= "111";
S <= '1';
```

```
wait for 100ns;
X <= "010";
Y <= "110";
S<='0';
wait for 100ns;
X <= "010";
Y <= "111";
S <= '0';
wait for 100ns;
X <= "010";
Y <= "100";
S <= '0';
wait for 100ns;
X <= "000";
Y <= "111";
S <= '1';
wait for 100ns;
X <= "110";
Y <= "000";
S <= '1';

wait;
  end process;
end Behavioral;
```

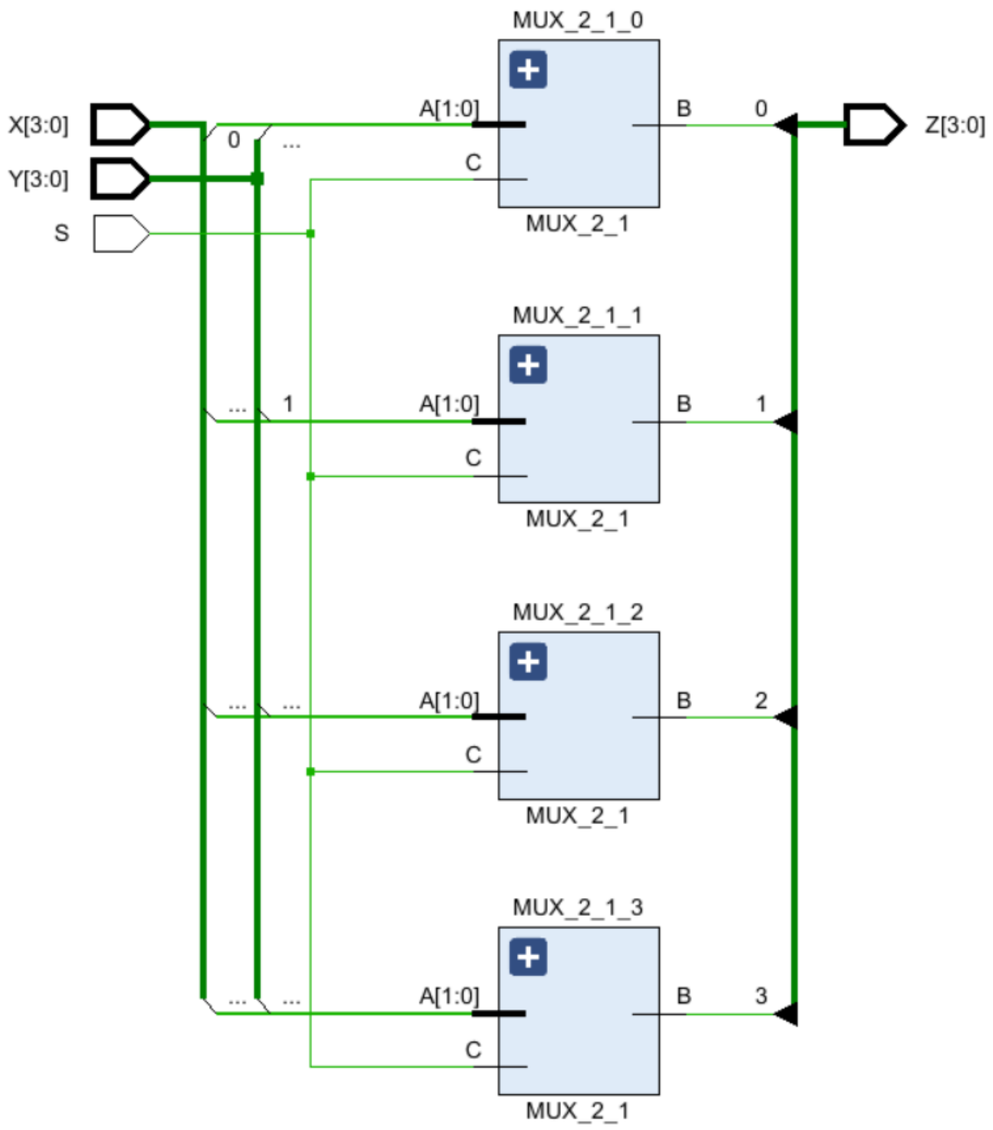
Timing Diagram



- **2-way 4-bit multiplexer**

- ❖ Use 4 , 2 to 1 multiplexers.
- ❖ If Load Selector is set to 1, get the immediate value from Instruction Decoder.
  Else get the out put value of 4-bit Add/Sub Unit as a input.
- ❖ Store the output of 2-way 4-bit multiplexer in a Register.

Elaborated design



VHDL code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 02:34:13 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 ( X : in STD_LOGIC_VECTOR (3 downto 0);
           Y : in STD_LOGIC_VECTOR (3 downto 0);
           Z : out STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC);
end MUX_2_WAY_4_BIT;
```

```
architecture Behavioral of MUX_2_WAY_4_BIT is
component MUX_2_1
   port (
       A : in STD_LOGIC_VECTOR (1 downto 0);
       B : out STD_LOGIC;
       C : in STD_LOGIC);
   end component;

begin
 MUX_2_1_0 :  MUX_2_1
          port map (
          A(0)=>X(0),
          A(1)=>Y(0),
          B=>Z(0),
          C => S);
 MUX_2_1_1 :  MUX_2_1
            port map (
            A(0)=>X(1),
            A(1)=>Y(1),
            B=>Z(1),
            C => S);
 MUX_2_1_2 :  MUX_2_1
              port map (
              A(0)=>X(2),
              A(1)=>Y(2),
              B=>Z(2),
              C => S);
MUX_2_1_3 :  MUX_2_1
            port map (
            A(0)=>X(3),
            A(1)=>Y(3),
            B=>Z(3),
            C => S);

end Behavioral;
```

Behavioral simulation source code for 2 - way 4 bit multiplexer

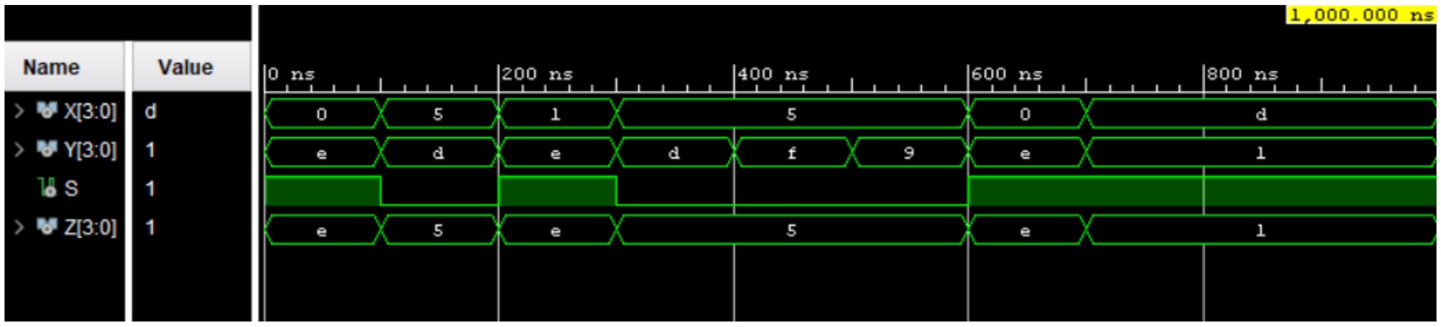
Goto Contents

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 02:40:55 PM
-- Design Name:
-- Module Name: MUX_2_WAY_4_BIT_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 MUX_2_WAY_4_BIT_SIM is
--  Port ( );
end MUX_2_WAY_4_BIT_SIM;

architecture Behavioral of MUX_2_WAY_4_BIT_SIM is
component MUX_2_WAY_4_BIT
```

```
port(
          X : in STD_LOGIC_VECTOR (3 downto 0);
          Y : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (3 downto 0)
);
end component;
signal X :  STD_LOGIC_VECTOR (3 downto 0);
signal Y :  STD_LOGIC_VECTOR (3 downto 0);
signal S :  STD_LOGIC;
signal Z :  STD_LOGIC_VECTOR (3 downto 0);
begin
UUT: MUX_2_WAY_4_BIT
port map(
    X => X,
    Z => Z,
    S => S,
    Y => Y
);
process

begin
X <= "0000";
Y <= "1110";
S <= '1';
wait for 100ns;
X <= "0101";
Y <= "1101";
S<='0';
wait for 100ns;
X <= "0001";
Y <= "1110";
S <= '1';
wait for 100ns;
X <= "0101";
Y <= "1101";
S<='0';
wait for 100ns;
X <= "0101";
Y <= "1111";
S <= '0';
```

```
wait for 100ns;
X <= "0101";
Y <= "1001";
S <= '0';
wait for 100ns;
X <= "0000";
Y <= "1110";
S <= '1';
wait for 100ns;
X <= "1101";
Y <= "0001";
S <= '1';

wait;
  end process;

end Behavioral;
```

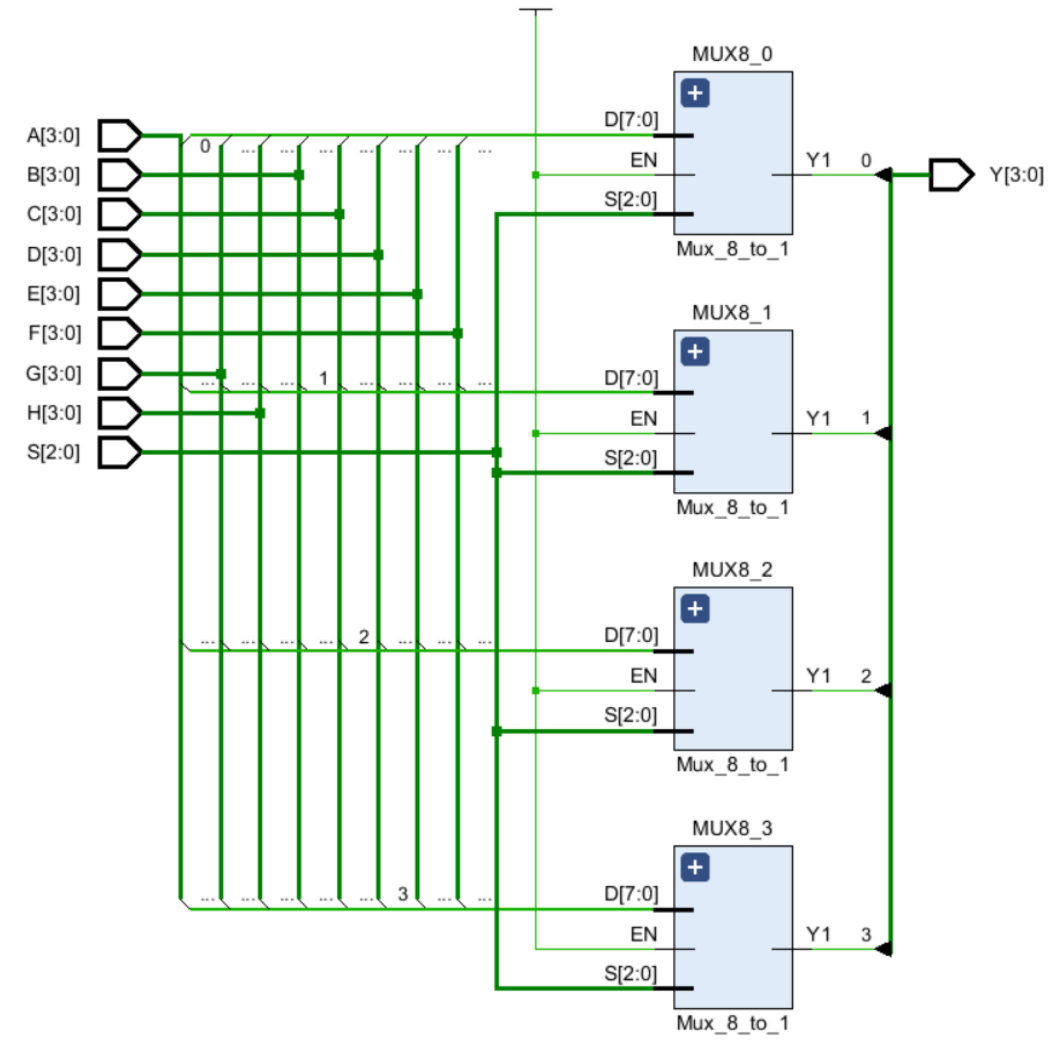
Timing Diagram



8-way 4-bit multiplexer

❖ This component has been made by using 4 8 to 1 multiplexer.
❖ Selectors select the same wire of every 4MUXs.
❖ Each of these MUX enable is always set to 1.

Elaborated design



VHDL code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/02/2024 02:46:44 PM
-- Design Name:
-- Module Name: Mux_8_way_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 Mux_8_way_4 is
    Port (A : in STD_LOGIC_VECTOR (3 downto 0);--register0 out
           B : in STD_LOGIC_VECTOR (3 downto 0);--register1 out
           C : in STD_LOGIC_VECTOR (3 downto 0);--register2 out
           D : in STD_LOGIC_VECTOR (3 downto 0);--register3 out
           E : in STD_LOGIC_VECTOR (3 downto 0);--register4 out
           F : in STD_LOGIC_VECTOR (3 downto 0);--register5 out
           G : in STD_LOGIC_VECTOR (3 downto 0);--register6 out
           H : in STD_LOGIC_VECTOR (3 downto 0);--register7 out
           S : in STD_LOGIC_VECTOR (2 downto 0);-- selector
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end Mux_8_way_4;
```

```
architecture Behavioral of Mux_8_way_4 is
component Mux_8_to_1 is
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           D : in STD_LOGIC_VECTOR (7 downto 0);
           EN : in STD_LOGIC;
           Y1 : out STD_LOGIC);
end component;
begin
MUX8_0: Mux_8_to_1
PORT MAP(
EN=>'1',
S=>S,
D(0)=>A(0),
D(1)=>B(0),
D(2)=>C(0),
D(3)=>D(0),
D(4)=>E(0),
D(5)=>F(0),
D(6)=>G(0),
D(7)=>H(0),
Y1=>Y(0));

MUX8_1: Mux_8_to_1
PORT MAP(
EN=>'1',
S=>S,
D(0)=>A(1),
D(1)=>B(1),
D(2)=>C(1),
D(3)=>D(1),
D(4)=>E(1),
D(5)=>F(1),
D(6)=>G(1),
D(7)=>H(1),
Y1=>Y(1));

MUX8_2: Mux_8_to_1
PORT MAP(
EN=>'1',
S=>S,
D(0)=>A(2),
D(1)=>B(2),
D(2)=>C(2),
D(3)=>D(2),
D(4)=>E(2),
D(5)=>F(2),
D(6)=>G(2),
D(7)=>H(2),
Y1=>Y(2));

MUX8_3: Mux_8_to_1
PORT MAP(
EN=>'1',
S=>S,
D(0)=>A(3),
D(1)=>B(3),
D(2)=>C(3),
D(3)=>D(3),
D(4)=>E(3),
D(5)=>F(3),
D(6)=>G(3),
D(7)=>H(3),
Y1=>Y(3));

end Behavioral;
```

8 way 4 bit Multiplexer test bench file

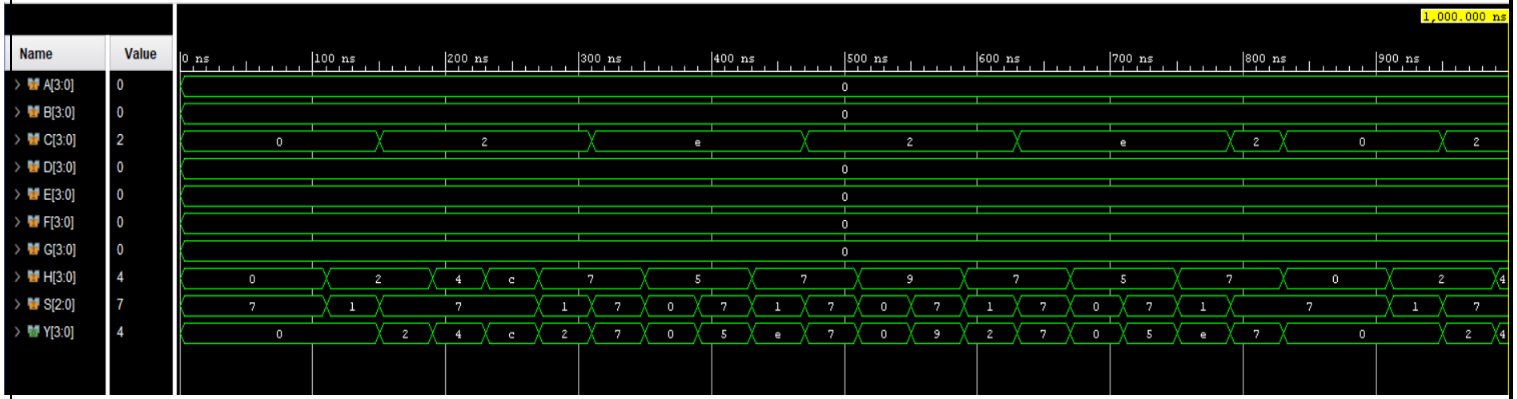
```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/02/2024 08:51:28 PM
-- Design Name:
-- Module Name: MUX8_WAY_4-SIM - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
```

```
--
-- 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 MUX8_WAY_4_SIM is
--  Port ( );
end MUX8_WAY_4_SIM;

architecture Behavioral of MUX8_WAY_4_SIM is
COMPONENT Mux_8_way_4 is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           C : in STD_LOGIC_VECTOR (3 downto 0);
           D : in STD_LOGIC_VECTOR (3 downto 0);
           E : in STD_LOGIC_VECTOR (3 downto 0);
           F : in STD_LOGIC_VECTOR (3 downto 0);
           G : in STD_LOGIC_VECTOR (3 downto 0);
           H : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end COMPONENT;
SIGNAL A,B,C,D,E,F,G,H,Y : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL S: STD_LOGIC_VECTOR(2 downto 0);
begin
UUT:Mux_8_way_4
port map(
A=>A,
B=>B,
C=>C,
D=>D,
E=>E,
F=>F,
G=>G,
H=>H,
S=>S,
Y=>Y
);
process begin
A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="000";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="001";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
```

```
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="010";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="011";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="100";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="101";
WAIT FOR 100NS;
```

```
A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="110";
WAIT FOR 100NS;

A<="1100";
B<="1010";
C<="0000";
D<="0001";
E<="0010";
F<="0100";
G<="1000";
H<="1111";
S<="111";
WAIT FOR 100NS;
WAIT;
end process;

end Behavioral;
```

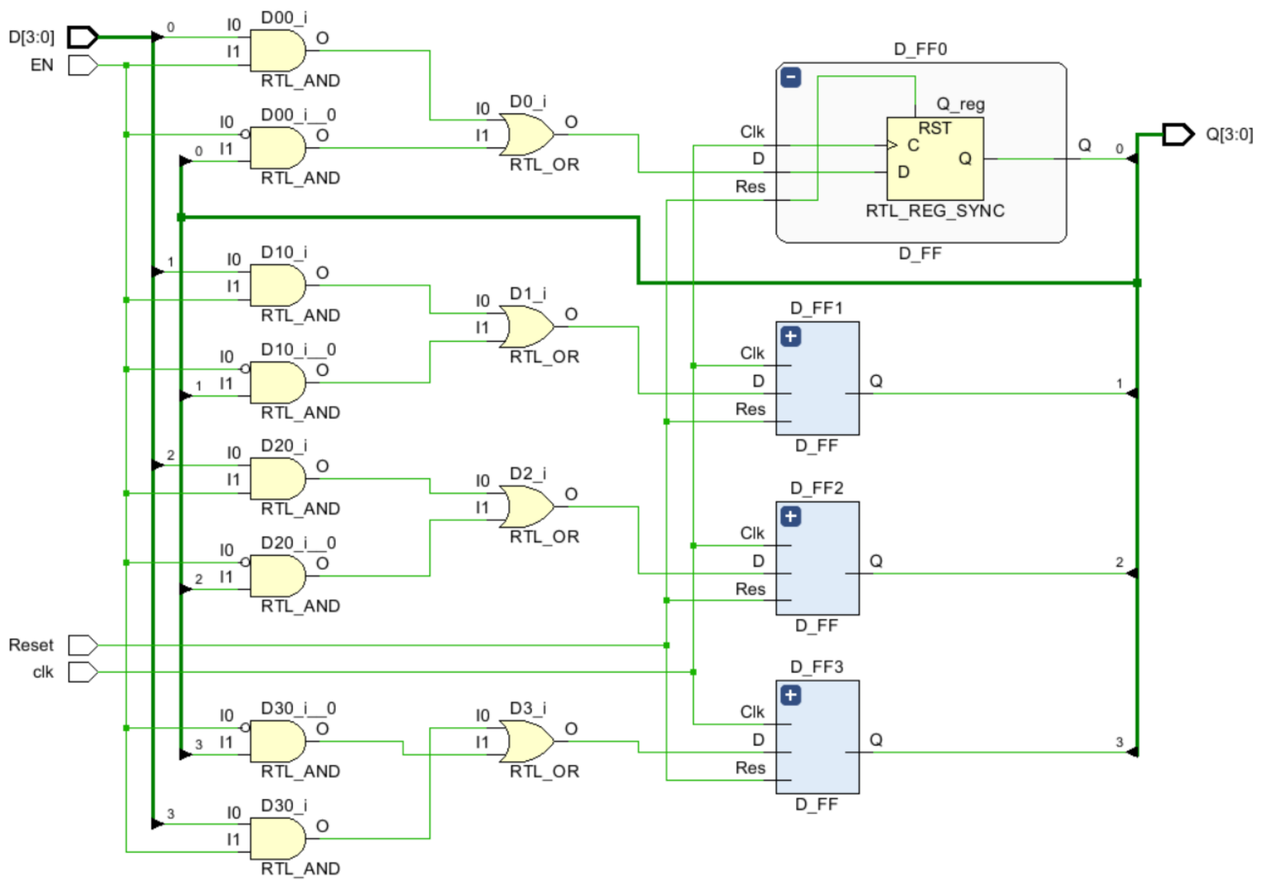
Timing Diagra



- **Register Bank**

- It has 2 components
    - 4-bit Register
    - 3 to 8 decoder
- Register
    - One register has 4 D flip flops
    - The Initial value of the register is set to 0.
    - When the register is enabled, Data can be written.
    - Although it is enabled, if it has current value before enabling it remains there until reset or overwrite register.
    - There are 8 registers in register bank.
    - Register0 cannot overwrite. Its value is always 0.
    - Content of Register 7 can be display by using Seven segment.

Elaborated design of Register



VHDL code of Re

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 02:46:58 PM
-- Design Name:
-- Module Name: Register - 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_D_FF is
    Port ( D : in STD_LOGIC_vector(3 downto 0);
           clk : in STD_LOGIC;
           Reset:in std_logic;
           EN:in std_logic;
           Q : out STD_LOGIC_vector(3 downto 0);
          Qbar : out STD_LOGIC_vector(3 downto 0)
          );

end Reg_D_FF;

architecture Behavioral of Reg_D_FF is
component 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 component;
signal D0,D1,D2,D3,Q0,Q1,Q2,Q3: STD_LOGIC;
```

```
begin
FF0:D_FF
port map(
D=>D0,
Clk=>Clk,
Res=>Reset,
Q=>Q0,
Qbar=>Qbar(0)
);

FF1:D_FF
port map(
D=>D1,
Clk=>Clk,
Res=>Reset,
Q=>Q1,
Qbar=>Qbar(0)
);

FF2:D_FF
port map(
D=>D2,
Clk=>Clk,
Res=>Reset,
Q=>Q2,
Qbar=>Qbar(0)
);

FF3:D_FF
port map(
D=>D3,
Clk=>Clk,
Res=>Reset,
Q=>Q3,
Qbar=>Qbar(3)
);

Q(0)<=Q0;
Q(1)<=Q1;
Q(2)<=Q2;
Q(3)<=Q3;
```

Goto Contents

```
D0<=(D(0) AND EN) Or (not EN AND Q0);
D1<=(D(1) AND EN) Or (not EN AND Q1);
D2<=(D(2) AND EN) Or (not EN AND Q2);
D3<=(D(3) AND EN) Or (not EN AND Q3);

end Behavioral;
```

- 3 to 8 Decoder
  - The decoder enables the register which data should be written.

3-to-8 Decoder VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 02/18/2024 09:52:41 AM
-- 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);
           Y : out STD_LOGIC_VECTOR (7 downto 0);
           EN : in STD_LOGIC);
end Decoder_3_to_8;

architecture Behavioral of Decoder_3_to_8 is
component Decoder_2_to_4
port(
I: in STD_LOGIC_VECTOR;
EN: in STD_LOGIC;
Y: out STD_LOGIC_VECTOR );
end component;
signal I0,I1 : STD_LOGIC_VECTOR (1 downto 0);
signal Y0,Y1 : STD_LOGIC_VECTOR (3 downto 0);
signal en0,en1, I2 : STD_LOGIC;
begin
Decoder_2_to_4_0 : Decoder_2_to_4
port map(
I => I0,
EN => en0,
Y => Y0 );
Decoder_2_to_4_1 : Decoder_2_to_4
port map(
I => I1,
EN => en1,
Y => Y1 );
en0 <= NOT(I(2)) AND EN;
en1 <= I(2) AND EN;
I0 <= I(1 downto 0);
I1 <= I(1 downto 0);
I2 <= I(2);

Y(3 downto 0) <= Y0;
```

```
Y(7 downto 4) <= Y1;

end Behavioral;
```

Register bank Test bench file

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 04:40:00 PM
-- Design Name:
-- Module Name: Register_Bank_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;
```

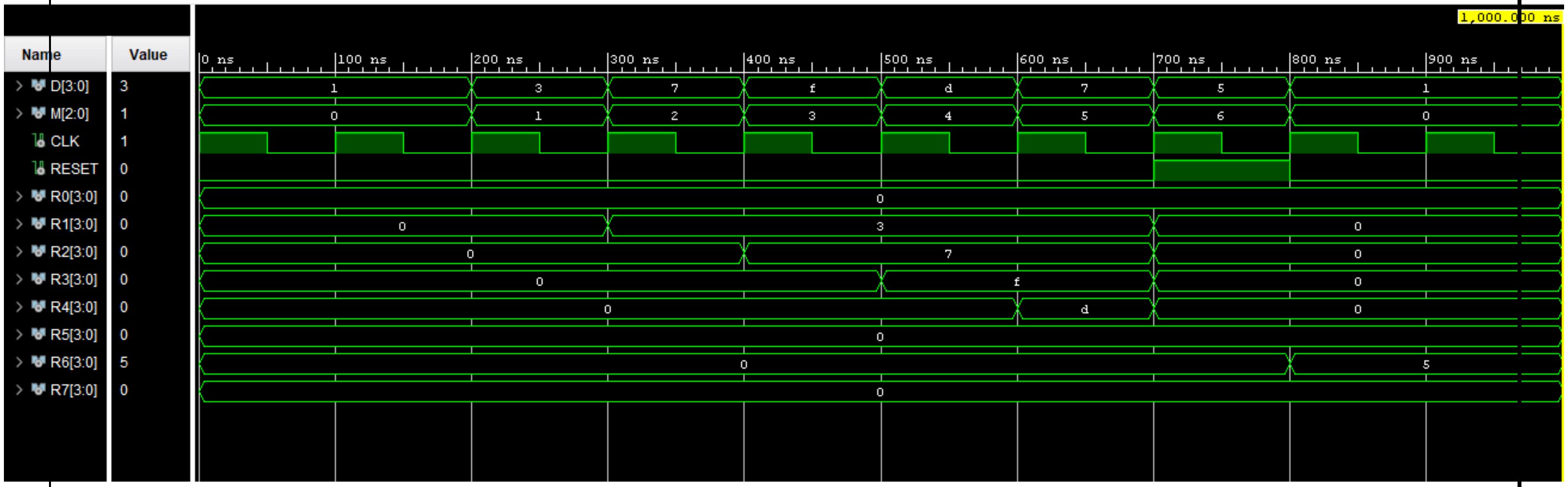
```
entity Register_Bank_Sim is
--  Port ( );
end Register_Bank_Sim;

architecture Behavioral of Register_Bank_Sim is
component REGISTER_BANK
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           CLK : in STD_LOGIC;
           Reset:in std_logic;
           M : in STD_LOGIC_VECTOR (2 downto 0);
           R0  : out STD_LOGIC_VECTOR (3 downto 0);
           R1  : out STD_LOGIC_VECTOR (3 downto 0);
           R2  : out STD_LOGIC_VECTOR (3 downto 0);
           R3  : out STD_LOGIC_VECTOR (3 downto 0);
           R4  : out STD_LOGIC_VECTOR (3 downto 0);
           R5  : out STD_LOGIC_VECTOR (3 downto 0);
           R6  : out STD_LOGIC_VECTOR (3 downto 0);
           R7  : out STD_LOGIC_VECTOR (3 downto 0));
end component;
SIGNAL D: STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL M: STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL CLK:STD_LOGIC:='1';
SIGNAL RESET:STD_LOGIC;
SIGNAL R0,R1,R2,R3,R4,R5,R6,R7 : STD_LOGIC_VECTOR (3 downto 0);
begin
UUT:REGISTER_BANK
port map(
D=> D,
CLK=>CLK,
RESET=>RESET,
M=>M,
R0=>R0,
R1=>R1,
R2=>R2,
R3=>R3,
R4=>R4,
R5=>R5,
R6=>R6,
R7=>R7);
PROCESS BEGIN
```

```
wait for 50ns;
clk<=not clk;
END PROCESS;

process begin
--wait for 100ns;
RESET<='1';
D<="0001";
Reset<='0';
M<="000";

wait for 100ns;
RESET<='0';
D<="0001";
Reset<='0';
M<="000";

wait for 100ns;
D<="0011";
Reset<='0';
m<="001";

wait for 100ns;
D<="0111";
Reset<='0';
m<="010";

wait for 100ns;
D<="1111";
Reset<='0';
m<="011";

wait for 100ns;
D<="1101";
Reset<='0';
M<="100";

wait for 100ns;
D<="0111";
Reset<='0';
M<="101";
```

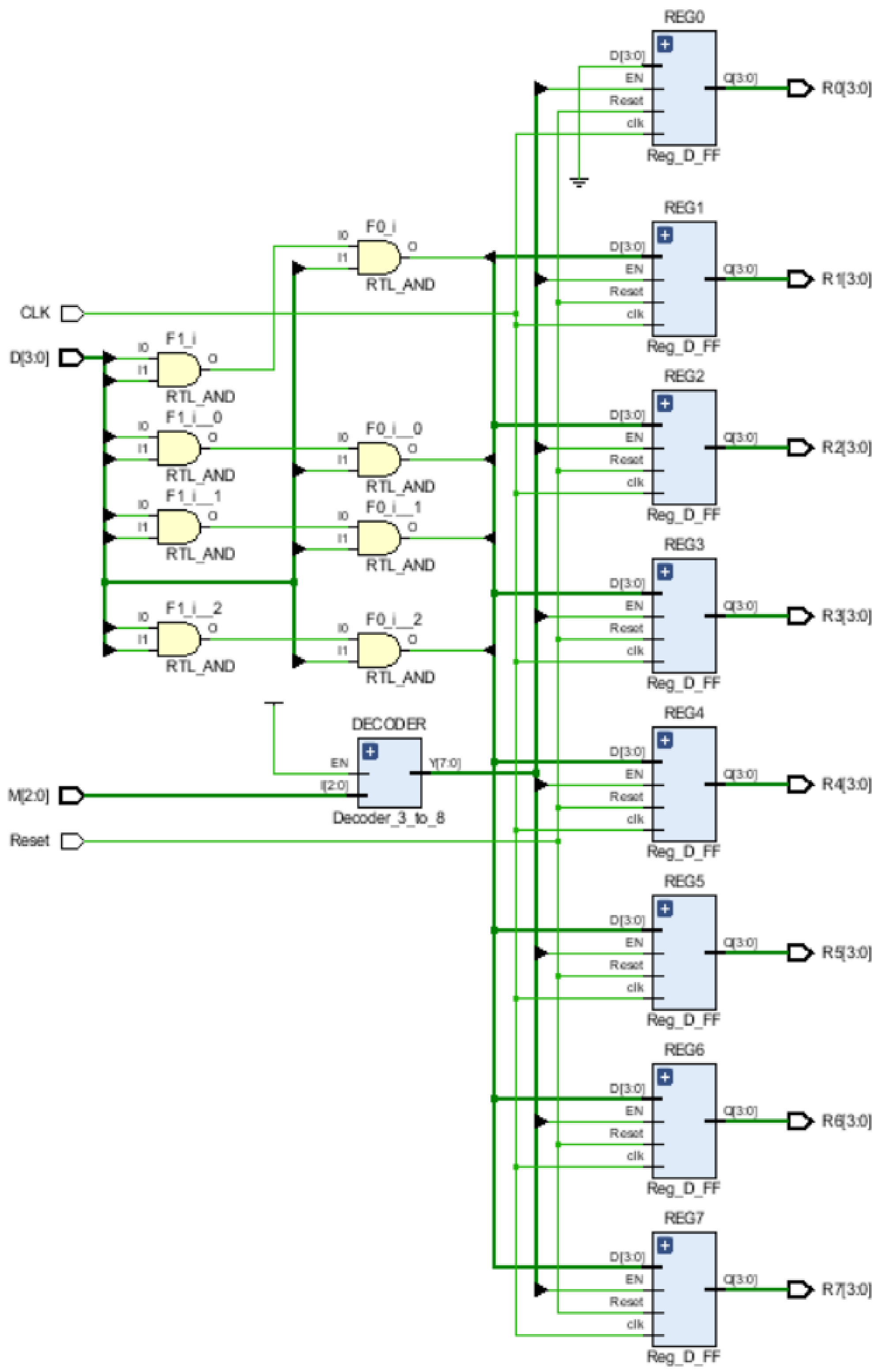
Goto Contents

```
wait for 100ns;
D<="0101";
Reset<='1';
M<="110";

wait for 100ns;
D<="1001";
Reset<='0';
M<="111";
end process;
end Behavioral;
```

Timing Diagram



Elaborated design of Register Bank



Register Bank VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 01:48:18 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 ( D : in STD_LOGIC_VECTOR (3 downto 0);--Data in
           CLK : in STD_LOGIC;
           Reset:in std_logic;
           M : in STD_LOGIC_VECTOR (2 downto 0);--2 way 3 bit
Mux out
```

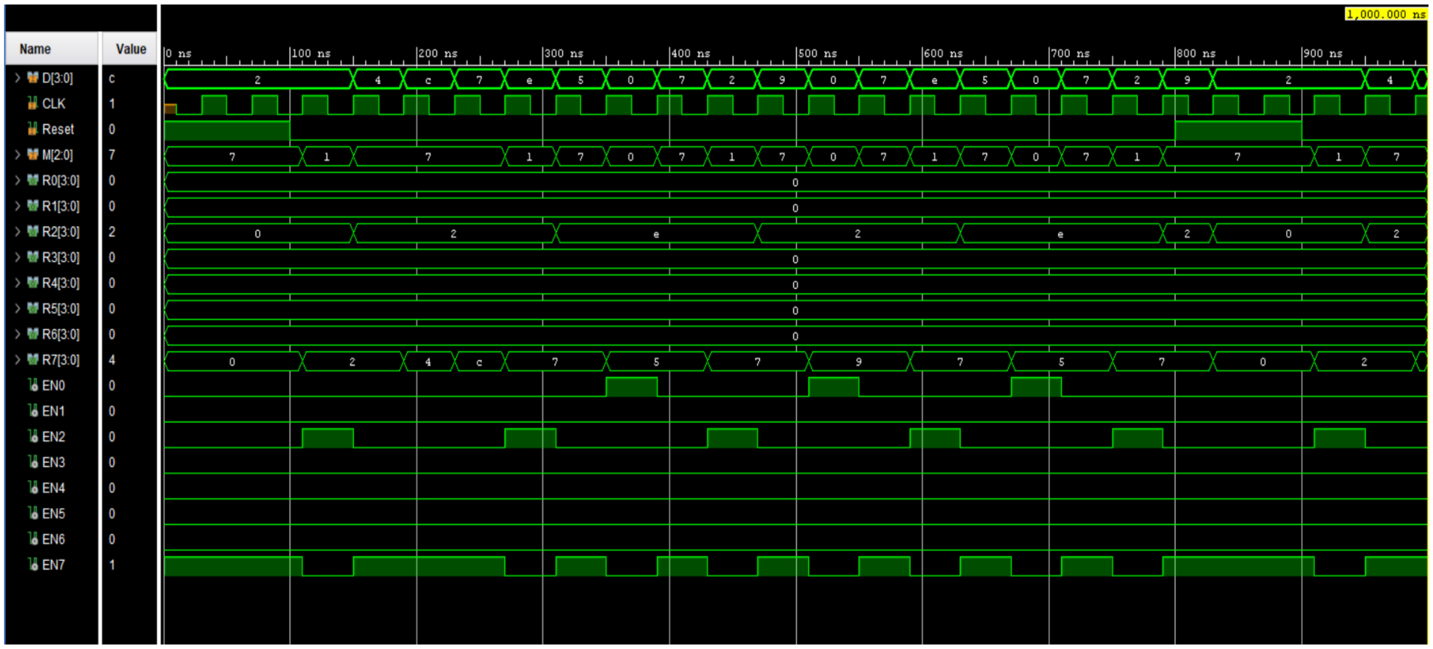
```
           R0 : out STD_LOGIC_VECTOR (3 downto 0); --Register0
out
           R1 : out STD_LOGIC_VECTOR (3 downto 0);--register1
out
           R2 : out STD_LOGIC_VECTOR (3 downto 0);--register2
out
           R3 : out STD_LOGIC_VECTOR (3 downto 0);--register3
out
           R4 : out STD_LOGIC_VECTOR (3 downto 0);--register4
out
           R5 : out STD_LOGIC_VECTOR (3 downto 0);--register5
out
           R6 : out STD_LOGIC_VECTOR (3 downto 0);--register6
out
           R7 : out STD_LOGIC_VECTOR (3 downto 0));--register7
out
end REGISTER_BANK;

architecture Behavioral of REGISTER_BANK is
component Reg_D_FF is
    Port ( D : in STD_LOGIC_vector(3 downto 0);
           clk : in STD_LOGIC;
           EN:in std_logic;
           Reset:in std_logic;
           Q : out STD_LOGIC_vector(3 downto 0);
           Qbar : out STD_LOGIC_vector(3 downto 0)
           );

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

--signal F:std_logic_vector(3 downto 0);
SIGNAL EN0,EN1,EN2,EN3,EN4,EN5,EN6,EN7: STD_LOGIC;
begin

F<=D;
```

```
REG0: Reg_D_FF
port map(
D=>"0000",
EN=>EN0,
Clk=>CLK,
Reset=>Reset,
Q=>R0
);

REG1: Reg_D_FF
port map(
D=>D,
EN=>EN1,
Clk=>CLK,
Reset=>Reset,
Q=>R1
);

REG2: Reg_D_FF
port map(
D=>D,
EN=>EN2,
Clk=>CLK,
Reset=>Reset,
Q=>R2
);

REG3: Reg_D_FF
port map(
D=>D,
EN=>EN3,
Clk=>CLK,
Reset=>Reset,
Q=>R3
);

REG4: Reg_D_FF
port map(
D=>D,
EN=>EN4,
Clk=>CLK,
Reset=>Reset,
Q=>R4
);

REG5: Reg_D_FF
port map(
D=>D,
EN=>EN5,
Clk=>CLK,
Reset=>Reset,
Q=>R5
);

REG6: Reg_D_FF
port map(
D=>D,
EN=>EN6,
Clk=>CLK,
Reset=>Reset,
Q=>R6
);

REG7: Reg_D_FF
port map(
D=>D,
EN=>EN7,
Clk=>CLK,
Reset=>Reset,
Q=>R7
);
DECODER:Decoder_3_to_8
PORT MAP(
I=>M,
EN=>'1',
Y(0)=>EN0,
Y(1)=>EN1,
Y(2)=>EN2,
Y(3)=>EN3,
Y(4)=>EN4,
Y(5)=>EN5,
Y(6)=>EN6,
```

Goto Contents

```
Y(7)=>EN7);

end Behavioral;
```

Timing Diagram



Problems when creating Register bank and solution for them

Problem Occurred	Solution
After writing to a register , when again writing another register the data of previously written registers erased and became 0.	Added logic if register was not enabled it can store furthermore the previous value until reset or overwritten.

- **Program ROM**
  - ❖ ROM stores all instructions in its memory.
  - ❖ The main instructions are,

- Move a value to a register
- Add the values of two registers
- Get negation of number containing in a register
- Jump

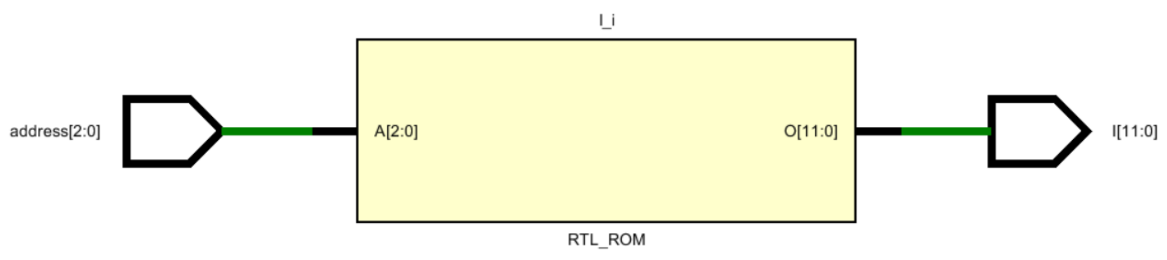
Assembly Program and Machine code representation

Machine Code Representation	Assembly Program
101110000001	MOVE 1 TO REG 7
100010000010	MOV 2 TO REG 1
001110010000	ADD REG 7 VALUE AND REG 1 VALUE
011110000000	NEGATION OF REG 7
101110000011	MOV 3 TO REG 7
010010000000	NEGATE REG 1 VALUE
001110010000	ADD REG 7 VALUE AND REG 1 VALUE
110000000100	JUMP TO LINE 4 IF REG 0 VALUE IS 0

- **Look Up Table for Program ROM**

Line No. of Assembly Code	ROM 0	ROM 1	ROM 2	ROM 3	ROM 4	ROM 5	ROM 6	ROM 7	ROM 8	ROM 9	ROM 10	ROM 11
0	1	0	1	1	1	0	0	0	0	0	0	1
1	1	0	0	0	1	0	0	0	0	0	1	0
2	0	0	1	1	1	0	0	1	0	0	0	0
3	0	1	1	1	1	0	0	0	0	0	0	0
4	1	0	1	1	1	0	0	0	0	0	1	1
5	0	1	0	0	1	0	0	0	0	0	0	0
6	0	0	1	1	1	0	0	1	0	0	0	0
7	1	1	0	0	0	0	0	0	0	1	0	0

Elaborated design



Goto Contents

VHDL code

```
----------------------------------------------------------------------------------
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/04/2024 01:43:38 PM
-- Design Name: 
-- Module Name: ROM - 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 ROM is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           I : out STD_LOGIC_VECTOR (11 downto 0));
end ROM;
```

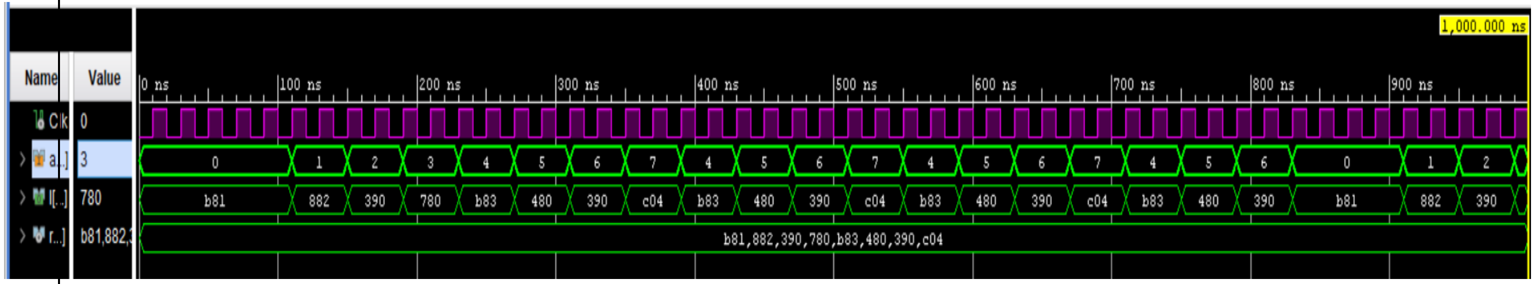
```
architecture Behavioral of ROM is

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

 "101110000001", -- MOVE 1 TO REG 7
 "100010000010",--MOVE 2 TO REG  1
 "001110010000",--ADD REG 7 VALUE AND REG 1 VALUE
 "011110000000",--NEGATION OF REG 7

 "101110000011", -- MOVE 3 TO REG 7
 "010010000000",-- NEGATE REG 1 VALUE
 "001110010000",--ADD REG 7 VALUE AND REG 1 VALUE
 "110000000100"--JUMP TO LINE 4 IF REG 0 VALUE IS 0


);

begin
I <= rom(to_integer(unsigned(address)));

end Behavioral;
```

Timing Diagram

"101110000001", -- MOVE 1 TO REG 7
"100010000010",--MOVE 2 TO REG 1
"001110010000",--ADD REG 7 VALUE AND REG 1 VALUE
"011110000000",--NEGATION OF REG 7

"101110000011", -- MOVE 3 TO REG 7
"010010000000",-- NEGATE REG 1 VALUE
"001110010000",--ADD REG 7 VALUE AND REG 1 VALUE
"110000000100"--JUMP TO LINE 4 IF REG 0 VALUE IS 0

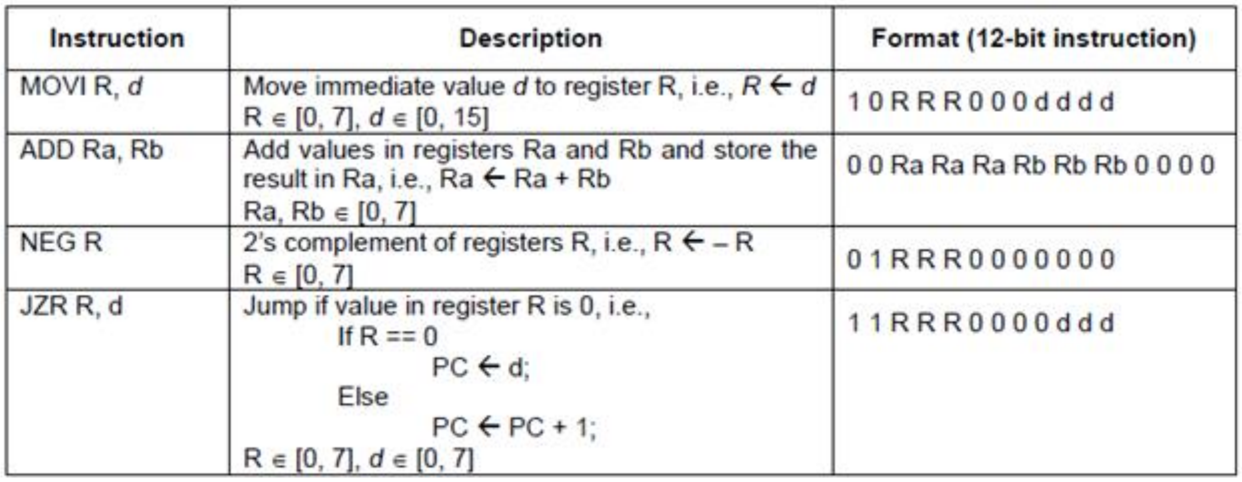
Goto Contents



- **Instruction Decoder**

❖ Instruction decoder is the main control unite of the processer. It decodes the instruction which is given by Rom and send control and data signals to relevant components.

- Main instructions,



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

❖ According to above instructions it send data and control signals.

- **Ports**
- I =>Rom instruction. (12 bit instruction line).
- RE=> Register enable (choose enable register to store data).
- D => Immediate value.(last 4-bit of instruction).

- RS_1 and RS_2=> Register select 1 and 2(Choose the register which give values to operations these signals are connected to 8 way 4-bit MUX).
- AOS => Select Add or Subtract .
- AOSE=>Enable Add/Sub unite.
- Ch_FJ=>To execute jump instruction we have to check a register whether it is 0 . This input line to get that register value.
- JF=> Jump flag.
- Ad_TJ => give number of the line what should jump.
- **Truth Tables**

I(11) I(10)	LS	AOS	AOSE
0 0	0	0	1
0 1	0	1	1
1 0	1	0	0
1 1	0	0	0

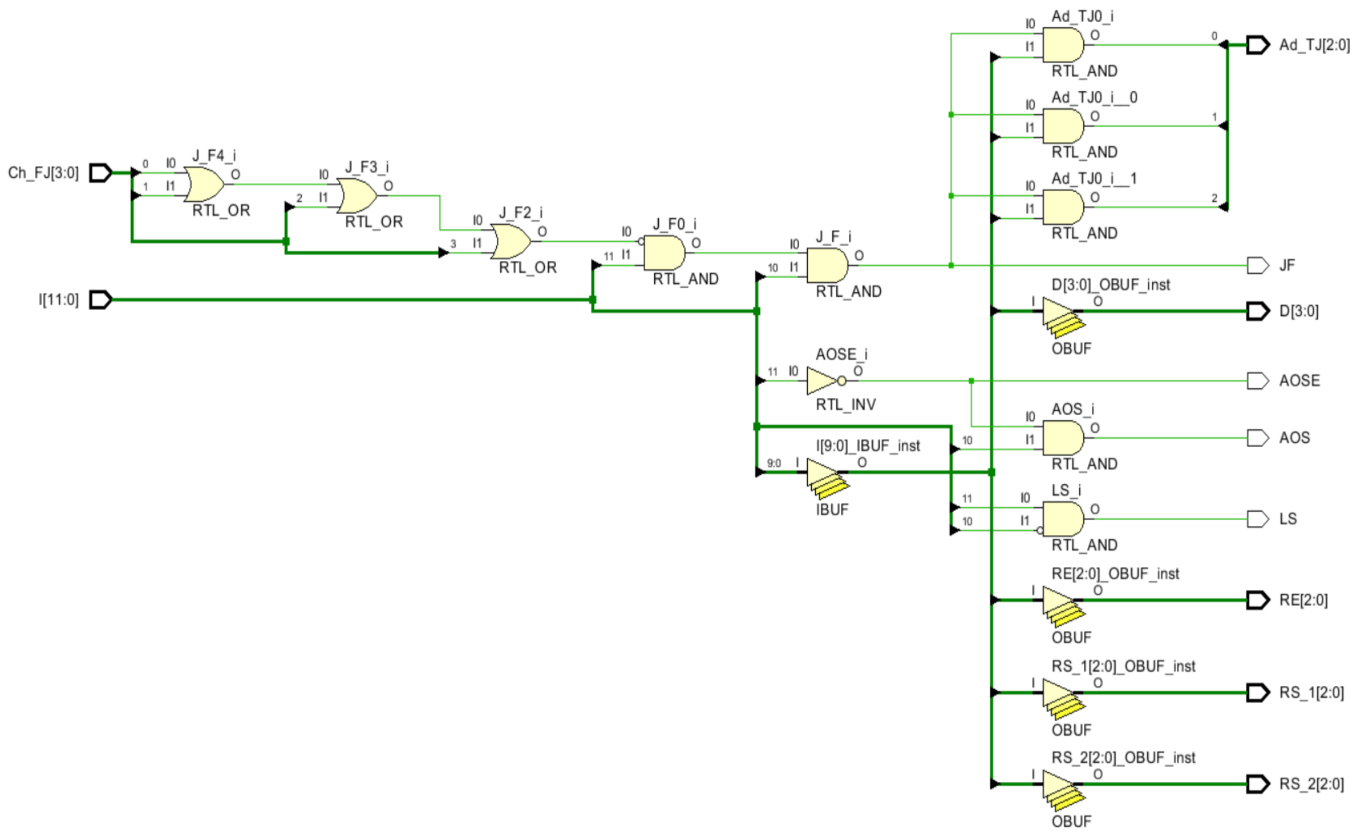
- LS<=I(11) AND NOT I(10);
- AOS<=NOT I(11) AND I(10);
- AOSE<=NOT I(11);

❖ If Ch_FJ =0000, I(11) = 1 and I(10) =1 ,

Then JF =1;

- JF<=NOT(Ch_FJ(0) OR Ch_FJ(1) OR Ch_FJ(2) OR Ch_FJ(3)) AND I(11) AND I(10)**;**

Elaborated design



VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 04:34:22 PM
-- Design Name:
-- Module Name: Inst_Decoder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
```

Goto Contents

```
-- 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 Inst_Decoder is
    Port (  I : in STD_LOGIC_VECTOR (11 downto 0);--instruction
         RE : out STD_LOGIC_VECTOR (2 downto 0);--register
enable
         LS : out STD_LOGIC;  --load select
         D : out STD_LOGIC_VECTOR (3 downto 0);--immediate value
         RS_1 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 1
         RS_2 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 2
         AOS,AOSE : out STD_LOGIC;--add or substerct change
         Ch_FJ : in STD_LOGIC_VECTOR (3 downto 0);--register
check for jump
         JF : out STD_LOGIC;--jump flag
         Ad_TJ : out STD_LOGIC_VECTOR (2 downto 0));--address to
jump
end Inst_Decoder;

architecture Behavioral of Inst_Decoder is

SIGNAL J_F: STD_LOGIC;
```

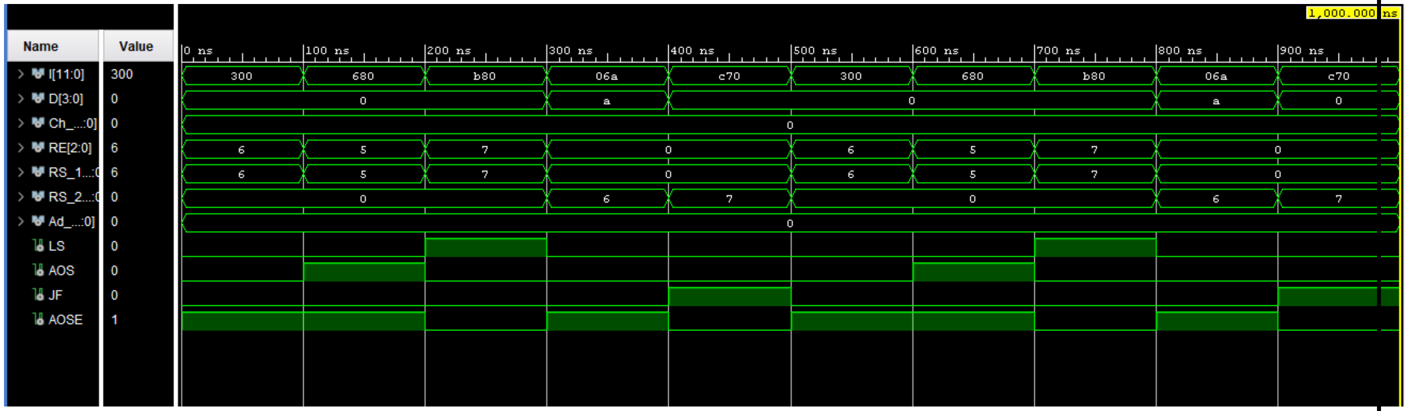
```
begin

J_F<=NOT(Ch_FJ(0) OR Ch_FJ(1) OR Ch_FJ(2) OR Ch_FJ(3)) AND I(11)
AND I(10);
JF<=J_F;
RS_1<=I(9 DOWNTO 7);
RS_2<=I(6 DOWNTO 4);
D<=I(3 DOWNTO 0);
RE<=I(9 DOWNTO 7);

Ad_TJ(2)<=J_F AND I(2);
Ad_TJ(1)<=J_F AND I(1);
Ad_TJ(0)<=J_F AND I(0);

LS<=I(11) AND NOT I(10);
AOS<=NOT I(11) AND I(10);
AOSE<=NOT I(11);


end Behavioral;
```

Instruction Decoder Test bench

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 05:21:17 PM
-- Design Name:
-- Module Name: InsDeco_TB - 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 InsDeco_TB is
--  Port ( );
end InsDeco_TB;

architecture Behavioral of InsDeco_TB is
COMPONENT Inst_Decoder
 Port (   I : in STD_LOGIC_VECTOR (11 downto 0);--instruction
RE : out STD_LOGIC_VECTOR (2 downto 0);--register enable
        LS : out STD_LOGIC;  --load select
D : out STD_LOGIC_VECTOR (3 downto 0);--immediate value
RS_1 : out STD_LOGIC_VECTOR (2 downto 0);--register select 1
RS_2 : out STD_LOGIC_VECTOR (2 downto 0);--register select 2
AOS,AOSE : out STD_LOGIC;--add or substerct change
Ch_FJ : in STD_LOGIC_VECTOR (3 downto 0);--register check for
jump
JF : out STD_LOGIC;--jump flag
Ad_TJ : out STD_LOGIC_VECTOR (2 downto 0));--address to jump
 END COMPONENT;

SIGNAL I:STD_LOGIC_VECTOR (11 downto 0);
SIGNAL D,Ch_FJ:STD_LOGIC_VECTOR (3 downto 0);
SIGNAL RE,RS_1,RS_2,Ad_TJ:STD_LOGIC_VECTOR (2 downto 0);
SIGNAL LS,AOS,JF,AOSE:STD_LOGIC;
```

```
begin

UUT:Inst_Decoder
PORT MAP
(
I=>I,
RE=>RE,
LS=>LS,
D=>D,
RS_1=>RS_1,
RS_2=>RS_2,
AOS=>AOS,
AOSE=>AOSE,
Ch_FJ=>Ch_FJ,
JF=>JF,
Ad_TJ=>Ad_TJ
);

PROCESS BEGIN
I<="001100000000";
Ch_FJ<="0000";
WAIT FOR 100 ns;
I<="011010000000";
WAIT FOR 100 ns;

I<="101110000000";
WAIT FOR 100 ns;

I<="000001101010";
WAIT FOR 100 ns;

I<="110001110000";
WAIT FOR 100 ns;


END PROCESS;

end Behavioral;
```

Timing Diagram



- **Slow Clock**

VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 03/05/2024 02:58:12 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;
```

Goto Contents

```
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of Slow_Clk is
signal count:integer:=1;
signal Clk_status :std_logic:='0';

begin

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

end Behavioral;
```

- **Seven segment Display**

VHDL code

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

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

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

entity Seven_Seg is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          -- Clk : in STD_LOGIC;

           S_LED : out STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           anode:out STD_LOGIC_VECTOR (3 downto 0)
           );
end Seven_Seg;

architecture Behavioral of Seven_Seg 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 S_0 : STD_LOGIC_VECTOR (3 downto 0);

begin
S_0<=A;

LUT_16_7_0 : LUT_16_7
    port map(
        address => S_0,
        data => S_7Seg);

S_LED <= S_0;
anode<="1110";
end Behavioral;
```

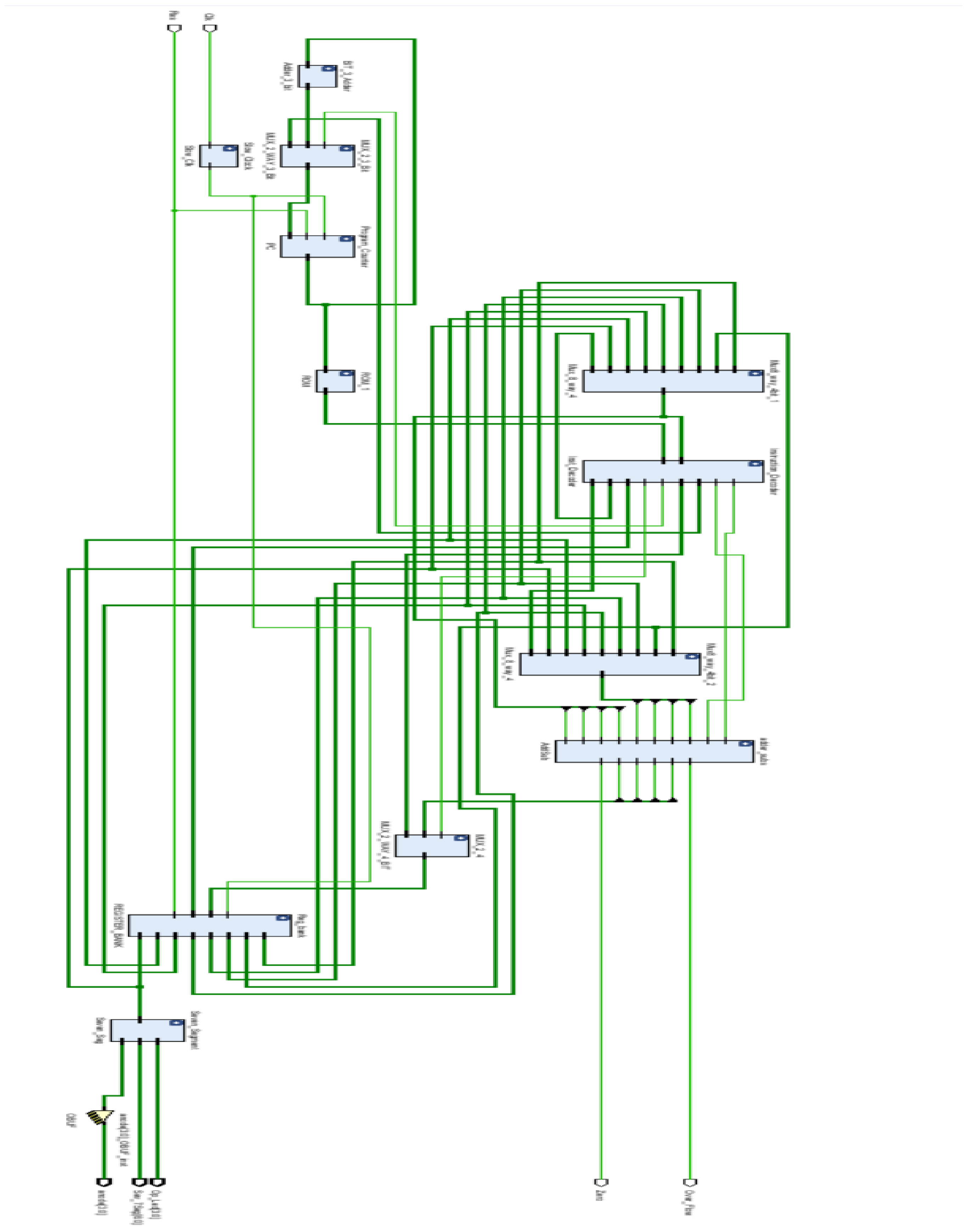
- **LUT - Seven segment display**

VHDL code

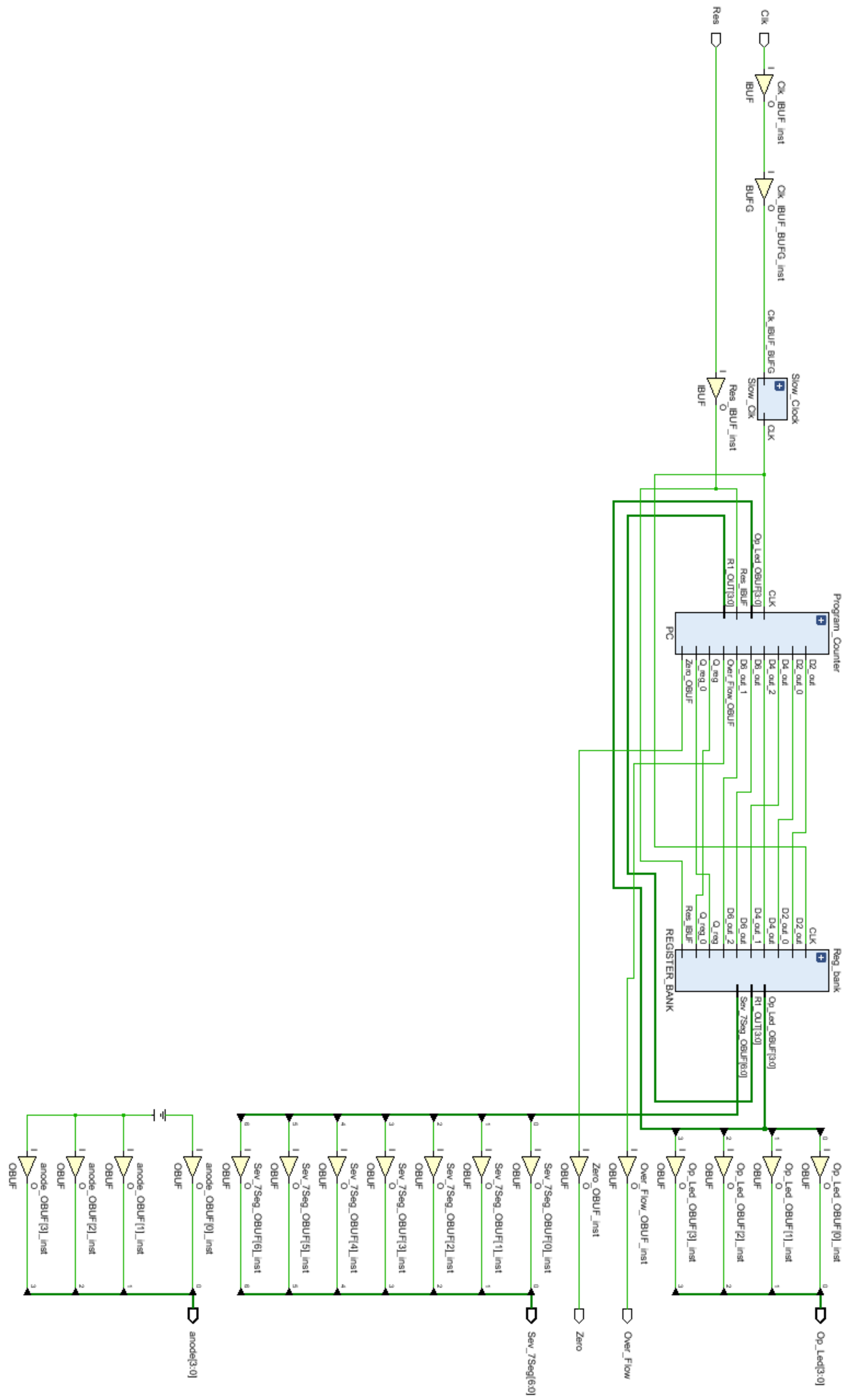
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

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

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

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

architecture Behavioral of LUT_16_7 is
type rom_type is array (0 to 15) of std_logic_vector(6 downto
0);
signal sevenSegment_ROM : rom_type := (
"1000000", -- 0
"1111001",
"0100100",
"0110000",
"0011001",
"0010010",
"0000010",
"1111000",
"0000000",
"0010000",
"0001000", -- a
"0000011",
"1000110",
"0100001",
"0000110",
"0001110" -- f
);

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

end Behavioral;
```

- **Nanoprocessor**

Elaborated design



Schematic Design



Goto Contents

VHDL code

```
----------------------------------------------------------------
------------------
-- Company: 
-- Engineer: 
-- 
-- Create Date: 04/16/2024 08:54:15 PM
-- Design Name: 
-- Module Name: M_P - 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 M_P is
    Port ( Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           Op_Led : out STD_LOGIC_VECTOR (3 downto 0);
           Over_Flow : out STD_LOGIC;
           Zero : out STD_LOGIC;
           anode:out STD_LOGIC_VECTOR (3 downto 0);
           Sev_7Seg :out STD_LOGIC_VECTOR (6 downto 0));
end M_P;



architecture Behavioral of M_P is

component REGISTER_BANK
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           CLK : in STD_LOGIC;
           Reset:in std_logic;
           M : in STD_LOGIC_VECTOR (2 downto 0);
           R0 : out STD_LOGIC_VECTOR (3 downto 0);
           R1 : out STD_LOGIC_VECTOR (3 downto 0);
           R2 : out STD_LOGIC_VECTOR (3 downto 0);
           R3 : out STD_LOGIC_VECTOR (3 downto 0);
           R4 : out STD_LOGIC_VECTOR (3 downto 0);
           R5 : out STD_LOGIC_VECTOR (3 downto 0);
           R6 : out STD_LOGIC_VECTOR (3 downto 0);
           R7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

COMPONENT AddSub
Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
          -- C_in : in STD_LOGIC;
           AOS,AOS_EN : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_Out : out STD_LOGIC;
           Zero : out STD_LOGIC;
           Over_flow : out STD_LOGIC);
END COMPONENT;

COMPONENT MUX_2_WAY_4_BIT
 Port ( X : in STD_LOGIC_VECTOR (3 downto 0);
          Y : in STD_LOGIC_VECTOR (3 downto 0);
          Z : out STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC);

END COMPONENT;


COMPONENT Mux_8_way_4
 Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          C : in STD_LOGIC_VECTOR (3 downto 0);
          D : in STD_LOGIC_VECTOR (3 downto 0);
          E : in STD_LOGIC_VECTOR (3 downto 0);
          F : in STD_LOGIC_VECTOR (3 downto 0);
          G : in STD_LOGIC_VECTOR (3 downto 0);
          H : in STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC_VECTOR (2 downto 0);
          Y : out STD_LOGIC_VECTOR (3 downto 0));

END COMPONENT;

COMPONENT ROM
 Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
          I : out STD_LOGIC_VECTOR (11 downto 0));

END COMPONENT;

COMPONENT Adder_3_bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

COMPONENT MUX_2_WAY_3_Bit is
    Port ( X : in STD_LOGIC_VECTOR (2 downto 0);
           Y : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
```

```
            Z : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

COMPONENT PC is
    Port ( X : in STD_LOGIC_VECTOR (2 downto 0);
           R : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

COMPONENT Seven_Seg is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          -- Clk : in STD_LOGIC;
           S_LED : out STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           anode:out STD_LOGIC_VECTOR (3 downto 0)
          );
end COMPONENT;

COMPONENT Inst_Decoder is
    Port (  I : in STD_LOGIC_VECTOR (11 downto 0);--instruction
         RE : out STD_LOGIC_VECTOR (2 downto 0);--register
enable
         LS : out STD_LOGIC;  --load select
         D : out STD_LOGIC_VECTOR (3 downto 0);--immediate value
         RS_1 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 1
         RS_2 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 2
         AOS,AOSE : out STD_LOGIC;--add or substerct change
         Ch_FJ : in STD_LOGIC_VECTOR (3 downto 0);--register
check for jump
         JF : out STD_LOGIC;--jump flag
         Ad_TJ : out STD_LOGIC_VECTOR (2 downto 0));--address to
jump
end COMPONENT;

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

```
SIGNAL I,ROM_OUT: STD_LOGIC_VECTOR (11 downto 0);
SIGNAL
RE,RegSel_1,RegSel_2,RS_2,Address_To_Jump,PC_OUT,A,Adder_3_OUT,a
ddress,A_Jump,A_out,Z_out:STD_LOGIC_VECTOR (2 downto 0);
SIGNAL
LS,Add_Or_Sub,Jump_Flag,Reset,R,RegSel,Clk_out,Load_Select,Jump_
F,AOS_EN:STD_LOGIC;
SIGNAL
ID_OUT,Ch_FJ,X,S_LED,way8_4bit_Mux1_out,way8_4bit_Mux2_out,bit_4
_out,Data_in,Data_out,Adder_OUT:STD_LOGIC_VECTOR (3 downto 0);
Signal
Mux_out,R0_OUT,R1_OUT,R2_OUT,R3_OUT,R4_OUT,R5_OUT,R6_OUT,R7_OUT
: STD_LOGIC_VECTOR (3 downto 0);
Signal Reg_enable,register_select : STD_LOGIC_VECTOR (2 downto
0);
--SIGNAL S_7Seg,Sev_7Seg:STD_LOGIC_VECTOR (6 downto 0);

begin
Instruction_Decoder:Inst_Decoder
port map(
    I=>ROM_OUT,
    RE=>Reg_enable,
    LS=>Load_Select,
    AOSE=>AOS_EN,
    D=>ID_OUT,
    RS_1=>RegSel_1,
    RS_2=>RegSel_2,
    AOS=>Add_Or_Sub,
    Ch_FJ=>way8_4bit_Mux1_out,
    JF=>Jump_Flag,
    Ad_TJ=>Address_To_Jump
);
Reg_bank: REGISTER_BANK
port map(
           D =>Data_out,
           CLK => Clk_out,
           Reset=>Res,
           M =>Reg_enable,
           R0=> R0_OUT,
           R1 =>R1_OUT,
           R2 =>R2_OUT,
           R3 =>R3_OUT,
           R4 =>R4_OUT,
           R5 =>R5_OUT,
           R6 =>R6_OUT,
           R7 =>R7_OUT);

Program_Counter: PC
PORT MAP(
   X=>Z_out,
   R=>Res,
   Clk=>Clk_out,
   Y=>PC_OUT
);

BIT_3_Adder:Adder_3_bit
PORT MAP
(
    A=>PC_OUT,
    S=>Adder_3_OUT

);
Seven_Segment:Seven_Seg
port map(
A =>R7_OUT,

anode=>anode,
S_LED => S_LED,
S_7Seg => Sev_7Seg
 );

-- Sev_7Seg<=S_7Seg;
 Mux8_way_4bit_1 : Mux_8_way_4
  port map(
             A => R0_OUT,
             B =>R1_OUT,
             C =>R2_OUT,
             D =>R3_OUT,
             E =>R4_OUT,
             F =>R5_OUT,
             G =>R6_OUT,
             H =>R7_OUT,
             S =>RegSel_1,
             Y =>way8_4bit_Mux1_out
  );
  Mux8_way_4bit_2 : Mux_8_way_4
   port map(
              A => R0_OUT,
              B =>R1_OUT,
              C =>R2_OUT,
              D =>R3_OUT,
              E =>R4_OUT,
              F =>R5_OUT,
              G =>R6_OUT,
              H =>R7_OUT,
              S =>RegSel_2,
              Y =>way8_4bit_Mux2_out
   );

ROM_1:ROM
  PORT MAP(
    address=>PC_OUT,
    I=>ROM_OUT
  );

 Slow_Clock: Slow_Clk
 port map (
      Clk_in=>Clk,
      Clk_out=>Clk_out
 );

 MUX_2_4 : MUX_2_WAY_4_BIT
 port map(
    X => Adder_OUT,
    Y => ID_OUT,
    S => Load_Select,
    Z => Data_out);

adder_subs :AddSub
      port map(
                 A0 =>way8_4bit_Mux2_out(0),
                 A1 =>way8_4bit_Mux2_out(1),
                 A2 =>way8_4bit_Mux2_out(2),
                 A3 =>way8_4bit_Mux2_out(3),
                 B0 =>way8_4bit_Mux1_out(0),
                 B1 =>way8_4bit_Mux1_out(1),
                 B2 =>way8_4bit_Mux1_out(2),
                 B3 =>way8_4bit_Mux1_out(3),
                -- C_in=> '0',
                 AOS_EN=>AOS_EN,
                 AOS =>Add_Or_Sub,
                 S0 =>Adder_OUT(0),
                 S1 =>Adder_OUT(1),
                 S2 =>Adder_OUT(2),
                 S3 =>Adder_OUT(3),
                 --C_Out => out STD_LOGIC;
                 Zero =>Zero,
                 Over_flow => Over_Flow
      );

MUX_2_3_Bit : MUX_2_WAY_3_Bit
          port map(

                 X => Adder_3_OUT,
                 Y=>Address_To_Jump,
                 S => Jump_Flag,
                 Z =>  Z_out);

 Op_Led<=  S_LED;
end Behavioral;
```

Nano Processor Test bench file

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/17/2024 07:13:29 AM
-- Design Name:
-- Module Name: M_P_TB - 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 M_P_TB is
--  Port ( );
end M_P_TB;

architecture Behavioral of M_P_TB is
COMPONENT M_P
Port ( Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           Op_Led : out STD_LOGIC_VECTOR (3 downto 0);
           Over_Flow : out STD_LOGIC;
           Zero : out STD_LOGIC;
            Sev_7Seg :out STD_LOGIC_VECTOR (6 downto 0));
END COMPONENT;
SIGNAL Clk,Res,Over_Flow,Zero:STD_LOGIC;
SIGNAL Op_Led: STD_LOGIC_VECTOR (3 downto 0);
```

```
signal Sev_7Seg : STD_LOGIC_VECTOR (6 downto 0);

begin
UUT:M_P
port map(
Clk=>Clk,
Res=>Res,
Op_Led=>Op_Led,
Over_Flow=>Over_Flow,
Zero=>Zero,
Sev_7Seg=>Sev_7Seg
);

PROCESS BEGIN
Clk<='0';
WAIT FOR 10ns;
Clk<='1';
WAIT FOR 10ns;
END PROCESS;

PROCESS BEGIN
Res<='1';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;

END PROCESS;

end Behavioral;
```

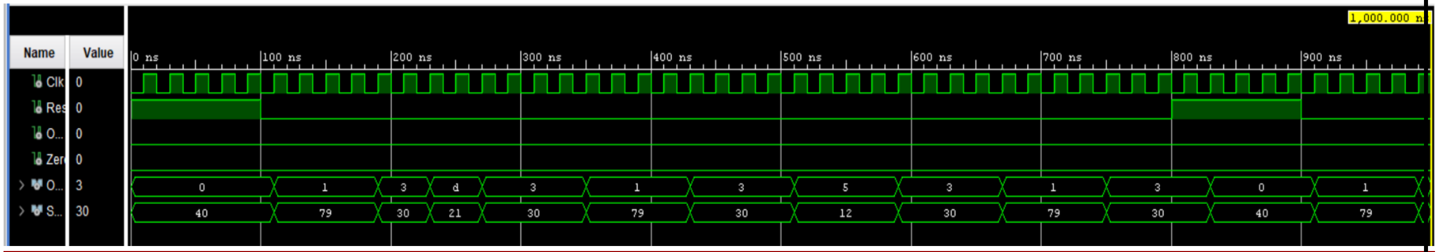
Goto Contents

Constrains file

```
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}]

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

set_property PACKAGE_PIN P1 [get_ports {Over_Flow}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Over_Flow}]
set_property PACKAGE_PIN L1 [get_ports {Zero}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]



set_property PACKAGE_PIN W7 [get_ports {Sev_7Seg[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {Sev_7Seg[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {Sev_7Seg[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {Sev_7Seg[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {Sev_7Seg[4]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {Sev_7Seg[5]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {Sev_7Seg[6]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[6]}]


set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
```

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


set_property PACKAGE_PIN U18 [get_ports Res]
    set_property IOSTANDARD LVCMOS33 [get_ports Res]
```

Timing Diagram



❖ Components of improved design

❖ We add new component - comparator.

❖ We added the Subtract Instruction.

❖ For this purpose we update

- Instruction decoder
- Rom
- Micro processor

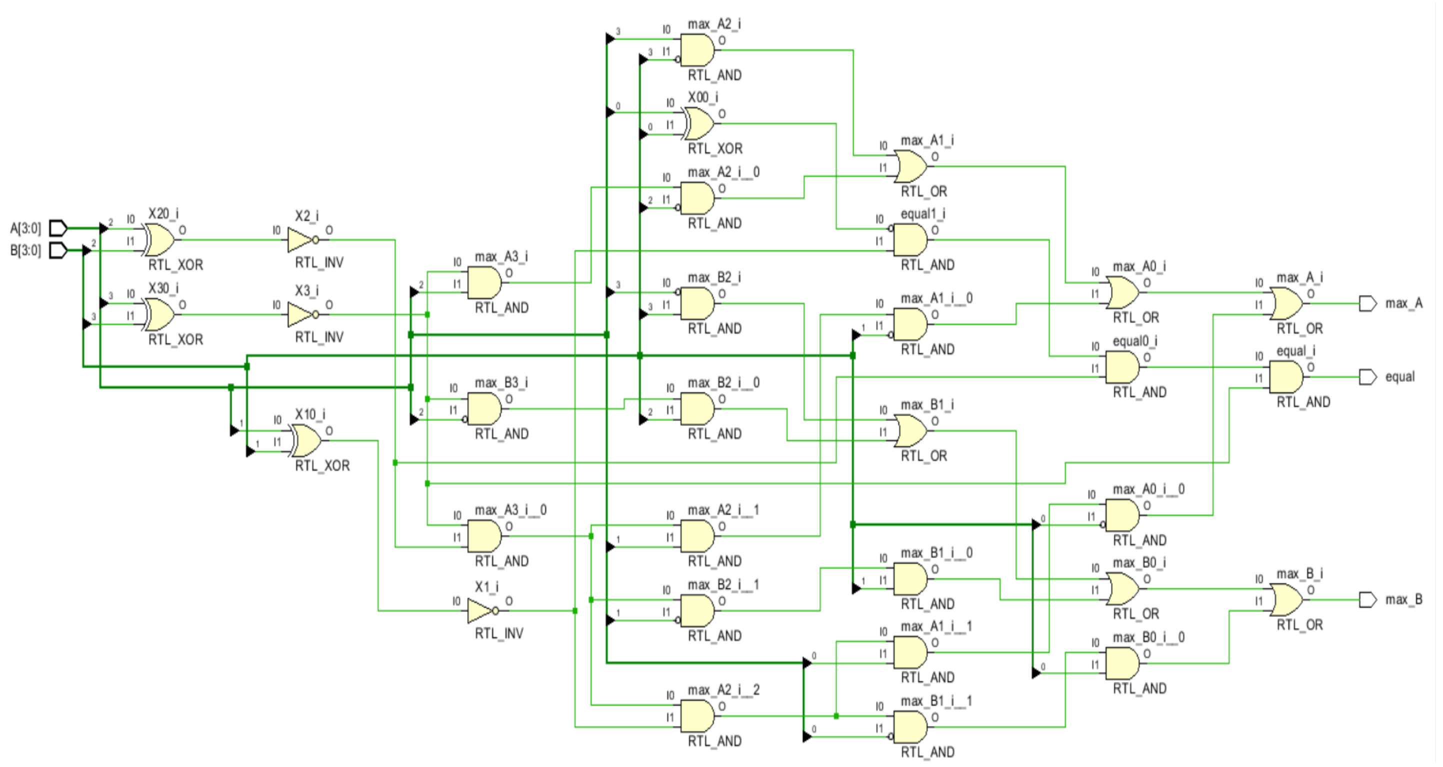
● Comparator

❖ The 4-bit comparator is comparing two 4-bit binary numbers, A and B. Its purpose is to determine if the inputs are equal (equal output) and which input is greater (max_A and max_B outputs). Using XOR gates and logical operations, the comparator efficiently evaluates bit-by-bit relationships between A and B, offering valuable insights into their relative magnitudes and equality.

- When building the comparator, we initially had to construct a comparator as part of the adder-subtractor circuit. However, in this case, while comparing two values in the register, there is an addition operation in the adder-subtractor circuit. After we built a new component named ‘comparator’, also when the compare signal is enabled, the first value (A) in the register is also enabled.

- Therefore, we had to ensure that this was disabled when comparing two values.

Elaborated design

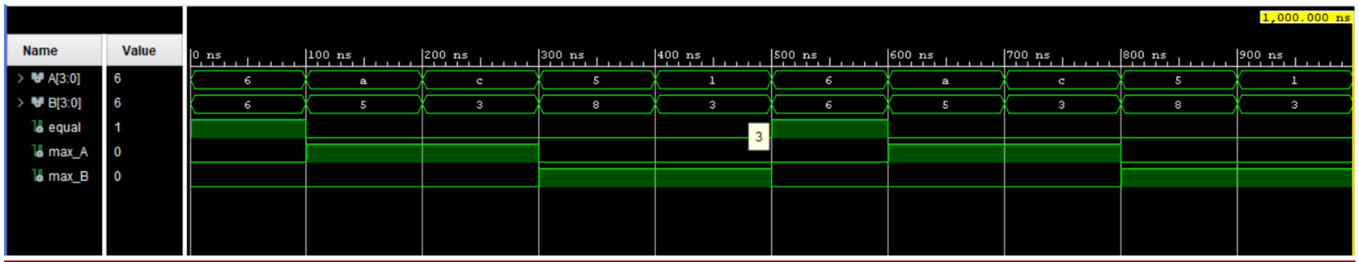


VHDL code

```
-------------------------------------------------------------------
-------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/23/2024 11:04:26 AM
-- Design Name:
-- Module Name: 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 comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           equal : out STD_LOGIC;
           max_A : out STD_LOGIC;
           max_B : out STD_LOGIC);
end comparator;

architecture Behavioral of comparator is
signal X0,X1,X2,X3 : std_logic;
begin
X0<=NOT(A(0) XOR B(0));
X1<=NOT(A(1) XOR B(1));
X2<=NOT(A(2) XOR B(2));
X3<=NOT(A(3) XOR B(3));

Equal<= X0 AND X1 AND X2 AND X3;
Max_A<= (A(3) AND NOT B(3)) OR (X3 AND A(2) AND NOT B(2)) OR (X3
AND X2 AND A(1) AND NOT B(1)) OR (X3 AND X2 AND X1 AND A(0) AND
NOT B(0));
Max_B<= (NOT A(3) AND B(3)) OR (X3 AND NOT A(2) AND B(2)) OR (X3
AND X2 AND NOT A(1) AND B(1)) OR (X3 AND X2 AND X1 AND NOT A(0)
AND B(0));

end Behavioral;
```

Timing Diagram



Comparator Test bench file

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/23/2024 11:23:25 AM
-- Design Name:
-- Module Name: comparator_TB - 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_TB is
--  Port ( );
end comparator_TB;

architecture Behavioral of comparator_TB is
COMPONENT comparator
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           equal : out STD_LOGIC;
           max_A : out STD_LOGIC;
           max_B : out STD_LOGIC);
end COMPONENT;

signal A,B : std_logic_vector(3 downto 0);
signal equal,max_A,max_B :std_logic;

begin
UUT : comparator port map(
    A => A,
    B => B,
    equal => equal,
    max_A => max_A,
    max_B => max_B);

process
begin
    A <= "0110";
    B <= "0110";
    wait for 100ns;

    A <= "1010";
    B <= "0101";
    wait for 100ns;

    A <= "1100";
```

```
        B <= "0011";
        wait for 100ns;

        A <= "0101";
        B <= "1000";
        wait for 100ns;

        A <= "0001";
        B <= "0011";
        wait for 100ns;

    end process;
    end Behavioral
```

- **Instruction decoder**

❖ As a improvement we added some extra instructions (subtract and compare).

❖ So we added an extra bit to rom instructions. According to that Instruction decoder is changed.

- Main instructions,

Instruction	Description	Format
ADD Ra ,Rb	Add values in registers Ra and Rb and store the result in Ra, i.e., Ra Ra + Rb	0 0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0
NEG R	2's complement of registers R, i.e., R – R	0 0 1 R R R 0 0 0 0 0 0 0
MOVI R, d	Move immediate value *d* to register R, i.e., *R d*	0 1 0 R R R 0 0 0 d d d d

JZR R, d	Jump if value in register R is 0, i.e., If R == 0 PC d; Else PC PC + 1;	0 1 1 R R R 0 0 0 0 d d d
SUB Rb - Ra	Subtract values in registers Ra and Rb and store the result in Ra, i.e., Ra Rb - Ra	1 0 0 Ra Ra Ra Rb Rb Rb 0 0 0 0
COMP Ra ,Rb	Compare Ra value and Rb value	1 1 0 Ra Ra Ra Rb Rb Rb 0 0 0 0

- **Truth Table**

I(12) I(11) I(10)	LS	AOS	AOS_EN	CM_EN
000	0	0	1	1
001	0	1	1	1
010	1	0	0	1
011	0	0	0	1
100	0	0	1	1
101	0	0	X	1
110	0	0	1	0
111	0	0	X	1

- LS<=I(11) AND NOT I(10) AND NOT I(12);
- AOS<=(NOT I(12) AND NOT I(11) AND I(10)) OR (I(12) AND NOT I(11) AND not I(10));
- AOS_EN<=NOT I(11);

  - When compare instruction executing CM_EN is set to 0.

- CM_EN<=NOT I(12) OR NOT I(11) OR I(10);

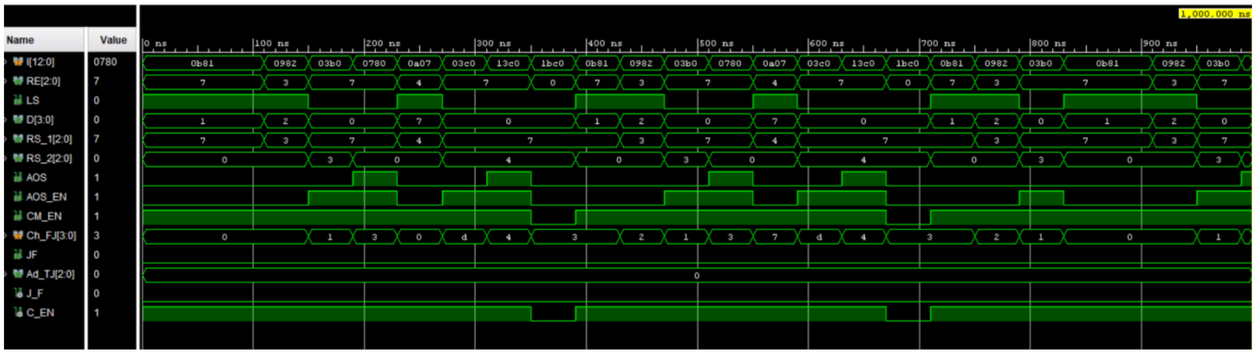
- Then we set the RE value to 0 to avoid store unnecessary values to registers.

VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 04:34:22 PM
-- Design Name:
-- Module Name: Inst_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 INS_DEC is
    Port (  I : in STD_LOGIC_VECTOR (12 downto 0);--instruction
         RE : out STD_LOGIC_VECTOR (2 downto 0);--register
enable
         LS : out STD_LOGIC;  --load select
         D : out STD_LOGIC_VECTOR (3 downto 0);--immediate value
         RS_1 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 1
         RS_2 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 2
         AOS,AOS_EN,CM_EN: out STD_LOGIC;--add or substerct
change

         Ch_FJ : in STD_LOGIC_VECTOR (3 downto 0);--register
check for jump
         JF : out STD_LOGIC;--jump flag
         Ad_TJ : out STD_LOGIC_VECTOR (2 downto 0));--address to
jump
end INS_DEC;

architecture Behavioral of INS_DEC is

SIGNAL J_F,C_EN: STD_LOGIC;

begin

J_F<=NOT(Ch_FJ(0) OR Ch_FJ(1) OR Ch_FJ(2) OR Ch_FJ(3)) AND I(11)
AND I(10);
JF<=J_F;
RS_1<=I(9 DOWNTO 7);
RS_2<=I(6 DOWNTO 4);
D<=I(3 DOWNTO 0);
--RE<=I(9 DOWNTO 7);
RE(2)<=I(9) AND C_EN;
RE(1)<=I(8) AND C_EN;
RE(0)<=I(7) AND C_EN;

Ad_TJ(2)<=J_F AND I(2);
Ad_TJ(1)<=J_F AND I(1);
Ad_TJ(0)<=J_F AND I(0);

LS<=I(11) AND NOT I(10) AND NOT I(12);
```

```
AOS<=(NOT I(12) AND NOT I(11) AND I(10)) OR ( I(12) AND NOT
I(11) AND not I(10));
AOS_EN<=NOT I(11);
C_EN<=NOT I(12) OR NOT I(11) OR I(10);
CM_EN<=C_EN;


end Behavioral;
```

Timing Diagram



- **Rom**

  - Assembly Program and Machine code representation

Machine Code Representation	Assembly Program
0101110000001	MOV 1 TO REG 7
0100010000010	MOV 2 TO REG 1
0001110010000	ADD REG 7 VALUE AND REG 6 VALUE
0011110000000	GET NEG OF REG 7 VALUE

0101110000001	MOV 1 TO REG 7
1001110010000	SUB REG 7 VALUE FROM REG 1 VALUE
110111001000	COMPARE VALUES OF REG7 AND REG 1
0110000000100	JUMP TO 5TH INSTRUCTION

- Look Up Table for Program ROM

Line No. of Assembly Code	ROM 0	ROM 1	RO M2	RO M3	RO M4	RO M5	RO M6	RO M7	RO M8	ROM 9	RO M10	ROM 11	ROM 12
0	0	1	0	1	1	1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0	0	0	0	0	1	0
2	0	0	0	1	1	1	0	0	1	0	0	0	0
3	0	0	1	1	1	1	0	0	0	0	0	0	0
4	0	1	0	1	1	1	0	0	0	0	0	0	1
5	1	0	0	1	1	1	0	0	1	0	0	0	0
6	1	1	0	1	1	1	0	0	1	0	0	0	0
7	0	1	1	0	0	0	0	0	0	0	1	0	0

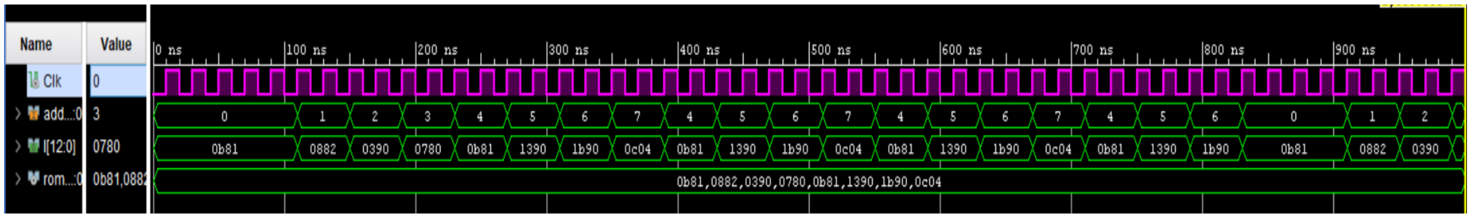
Goto Contents

VHDL code

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2024 01:43:38 PM
-- Design Name:
-- Module Name: ROM - 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 ROM12 is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           I : out STD_LOGIC_VECTOR (12 downto 0));
end ROM12;
```

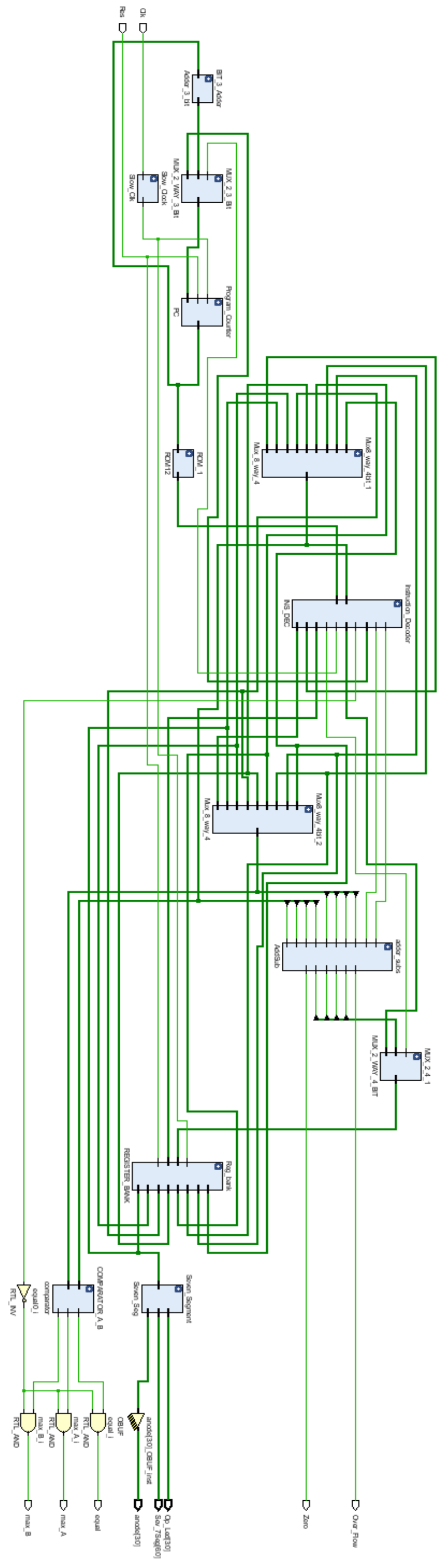
```
architecture Behavioral of ROM12 is

type rom_type is array (0 to 7) of std_logic_vector(12 downto
0);
signal rom : rom_type := (
"0101110000001", --move 1 to reg7
"0100010000010",--move 2 to reg1
"0001110010000",--reg7<--reg7+reg1
"0011110000000",--negation of value in reg7

"0101110000001", --move 1 to reg7
"1001110010000",--sub reg 7 value from reg 1 value
"1101110010000",--COMPARE REG 7 VALUE AND REG 1 VALUE
"0110000000100"--jump to 5th instruction
);

begin
I <= rom(to_integer(unsigned(address)));

end Behavioral;
```

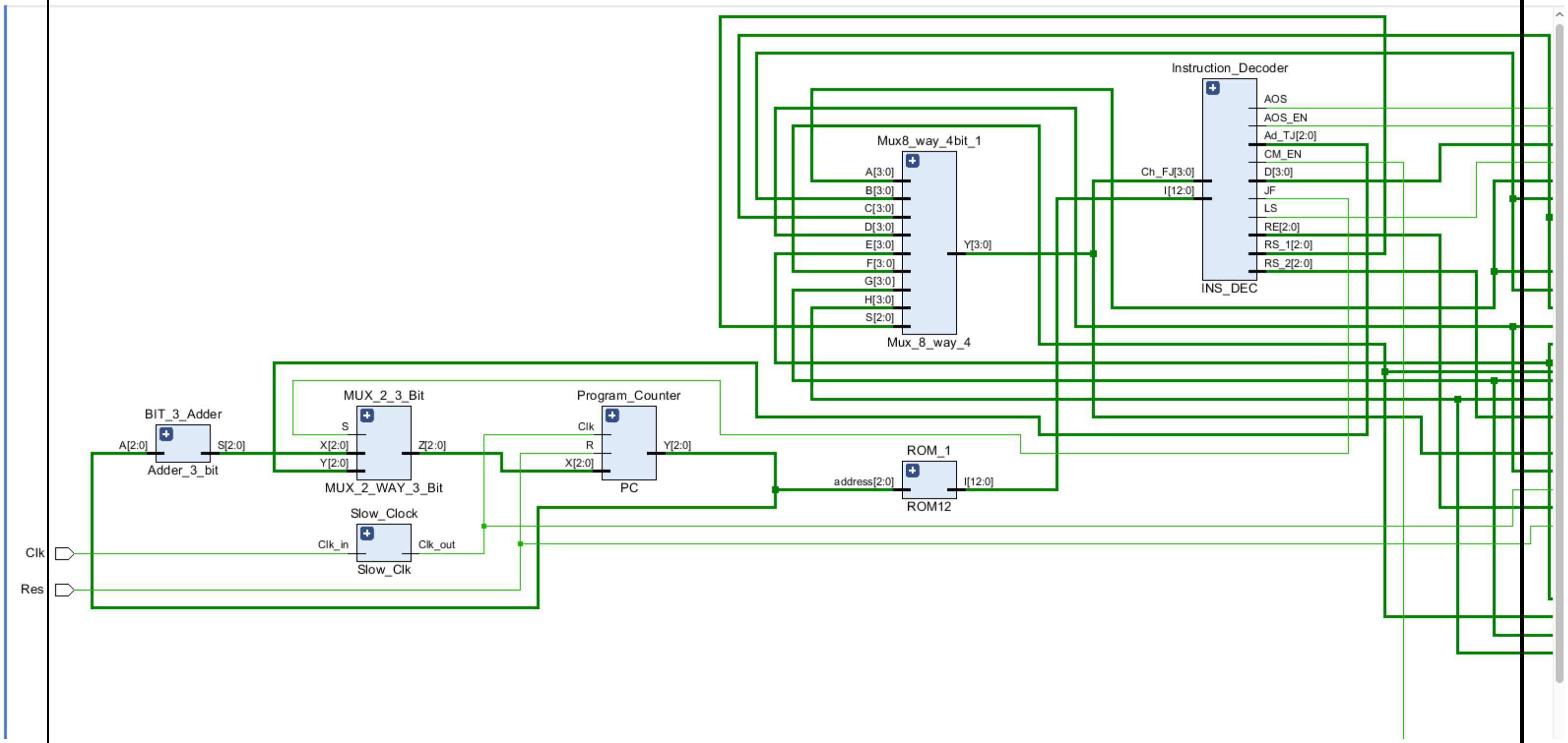
Timing Diagram

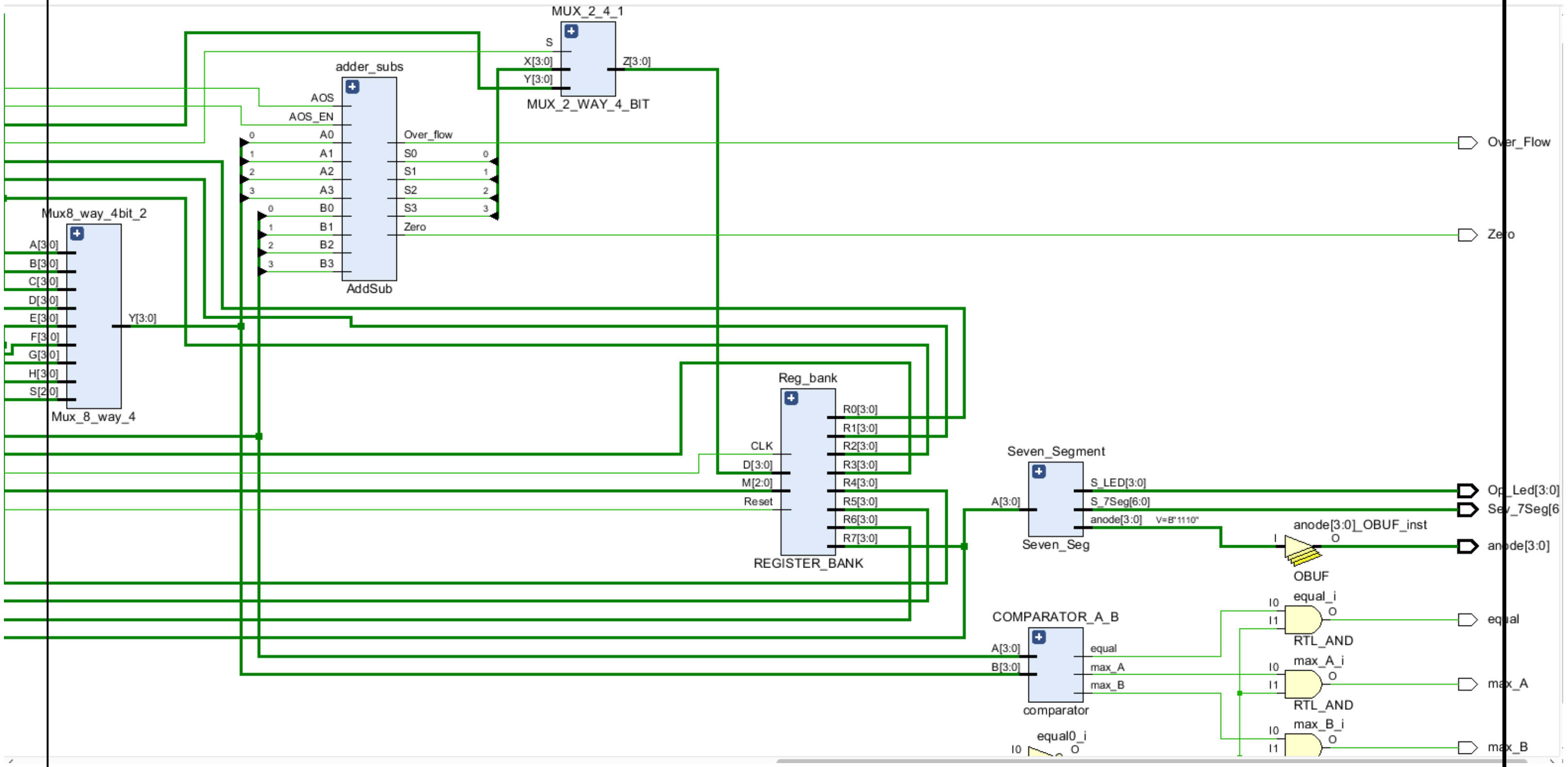


- **Nano processor (Improved)**

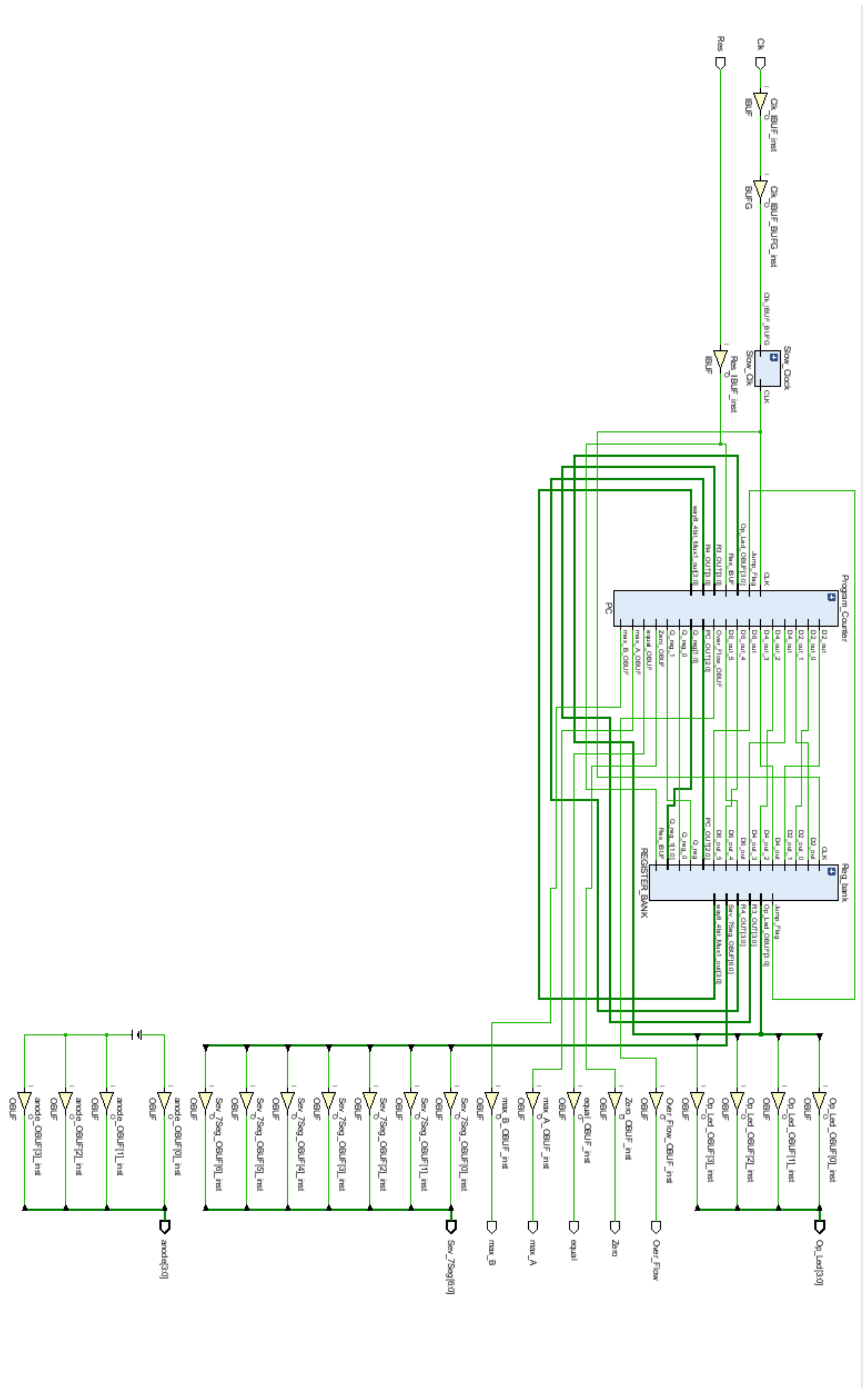
Elaborated design


Rst
Clk
BIT_3_Adder
Adder_3_bit
MUX_2_3_Bit
MUX_2_WAY_3_Bit
Slow_Clock
Slow_Clk
Program_Counter
PC
ROM_1
ROM12
Mux8_way_4bit_1
Mux_8_way_4
Instruction_Decoder
INS_DEC
Mux8_way_4bit_2
Mux_8_way_4
adder_subs
AddSub
MUX_2_4_1
MUX_2_WAY_4_BIT
Reg_bank
REGISTER_BANK
COMPARATOR_A_B
comparator
Seven_Segment
Seven_Seg
RTL_INV
RTL_AND
OBUF
max_B
max_A
equal
anode[3:0]
Sev_7Seg[6:0]
Op_Led[3:0]
Zero
Over_Flow


Instruction_Decoder
AOS
AOS_EN
Ad_TJ[2:0]
CM_EN
D[3:0]
JF
LS
RE[2:0]
RS_1[2:0]
RS_2[2:0]
Ch_FJ[3:0]
I[12:0]
INS_DEC
Mux8_way_4bit_1
A[3:0]
B[3:0]
C[3:0]
D[3:0]
E[3:0]
F[3:0]
G[3:0]
H[3:0]
S[2:0]
Y[3:0]
Mux_8_way_4
BIT_3_Adder
A[2:0]
S[2:0]
Adder_3_bit
MUX_2_3_Bit
S
X[2:0]
Y[2:0]
Z[2:0]
MUX_2_WAY_3_Bit
Program_Counter
Clk
R
X[2:0]
Y[2:0]
PC
ROM_1
address[2:0]
I[12:0]
ROM12
Slow_Clock
Clk_in
Clk_out
Slow_Clk
Clk
Res


MUX_2_4_1
S
X[3:0]
Y[3:0]
Z[3:0]
MUX_2_WAY_4_BIT
adder_subs
AOS
AOS_EN
A0
A1
A2
A3
B0
B1
B2
B3
Over_flow
S0
S1
S2
S3
Zero
AddSub
Mux8_way_4bit_2
A[3:0]
B[3:0]
C[3:0]
D[3:0]
E[3:0]
F[3:0]
G[3:0]
H[3:0]
S[2:0]
Y[3:0]
Mux_8_way_4
Over_Flow
Zero
Reg_bank
CLK
D[3:0]
M[2:0]
Reset
R0[3:0]
R1[3:0]
R2[3:0]
R3[3:0]
R4[3:0]
R5[3:0]
R6[3:0]
R7[3:0]
REGISTER_BANK
Seven_Segment
A[3:0]
S_LED[3:0]
S_7Seg[6:0]
anode[3:0]
V=B"1110"
Seven_Seg
anode[3:0]_OBUF_inst
I
O
OBUF
Op_Led[3:0]
Sev_7Seg[6
anode[3:0]
COMPARATOR_A_B
A[3:0]
B[3:0]
equal
max_A
max_B
comparator
equal_i
I0
I1
O
RTL_AND
max_A_i
RTL_AND
max_B_i
equal0_i
equal
max_A
max_B

Schematic Design



Goto Contents

VHDL code

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/16/2024 08:54:15 PM
-- Design Name:
-- Module Name: M_P - 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 MICRO_PROCESSOR is
    Port ( Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           Op_Led : out STD_LOGIC_VECTOR (3 downto 0);
           equal : out STD_LOGIC;
           max_A : out STD_LOGIC;
           max_B : out STD_LOGIC;
           Over_Flow : out STD_LOGIC;
           Zero : out STD_LOGIC;
           anode:out STD_LOGIC_VECTOR (3 downto 0);
           Sev_7Seg :out STD_LOGIC_VECTOR (6 downto 0));
end MICRO_PROCESSOR;



architecture Behavioral of MICRO_PROCESSOR is

component REGISTER_BANK
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           CLK : in STD_LOGIC;
           Reset:in std_logic;
           M : in STD_LOGIC_VECTOR (2 downto 0);
           R0 : out STD_LOGIC_VECTOR (3 downto 0);
           R1 : out STD_LOGIC_VECTOR (3 downto 0);
           R2 : out STD_LOGIC_VECTOR (3 downto 0);
           R3 : out STD_LOGIC_VECTOR (3 downto 0);
           R4 : out STD_LOGIC_VECTOR (3 downto 0);
           R5 : out STD_LOGIC_VECTOR (3 downto 0);
           R6 : out STD_LOGIC_VECTOR (3 downto 0);
           R7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

COMPONENT AddSub
Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           --C_in : in STD_LOGIC;
           AOS,AOS_EN : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_Out : out STD_LOGIC;
           Zero : out STD_LOGIC;
           Over_flow : out STD_LOGIC);
END COMPONENT;


COMPONENT MUX_2_WAY_4_BIT
 Port ( X : in STD_LOGIC_VECTOR (3 downto 0);
          Y : in STD_LOGIC_VECTOR (3 downto 0);
          Z : out STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC);

END COMPONENT;


COMPONENT Mux_8_way_4
 Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          C : in STD_LOGIC_VECTOR (3 downto 0);
          D : in STD_LOGIC_VECTOR (3 downto 0);
          E : in STD_LOGIC_VECTOR (3 downto 0);
          F : in STD_LOGIC_VECTOR (3 downto 0);
          G : in STD_LOGIC_VECTOR (3 downto 0);
          H : in STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC_VECTOR (2 downto 0);
          Y : out STD_LOGIC_VECTOR (3 downto 0));

END COMPONENT;

COMPONENT ROM12
 Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
          I : out STD_LOGIC_VECTOR (12 downto 0));

END COMPONENT;

COMPONENT Adder_3_bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;
```

```
COMPONENT MUX_2_WAY_3_Bit is
    Port ( X : in STD_LOGIC_VECTOR (2 downto 0);
           Y : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Z : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

COMPONENT PC is
    Port ( X : in STD_LOGIC_VECTOR (2 downto 0);
           R : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

COMPONENT Seven_Seg is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          -- Clk : in STD_LOGIC;
           S_LED : out STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           anode:out STD_LOGIC_VECTOR (3 downto 0)
          );
end COMPONENT;

COMPONENT INS_DEC is
    Port (  I : in STD_LOGIC_VECTOR (12 downto 0);--instruction
         RE : out STD_LOGIC_VECTOR (2 downto 0);--register
enable
         LS : out STD_LOGIC;  --load select
         D : out STD_LOGIC_VECTOR (3 downto 0);--immediate value
         RS_1 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 1
         RS_2 : out STD_LOGIC_VECTOR (2 downto 0);--register
select 2
         AOS,AOS_EN,CM_EN : out STD_LOGIC;--add or substerct
change

         Ch_FJ : in STD_LOGIC_VECTOR (3 downto 0);--register
check for jump
         JF : out STD_LOGIC;--jump flag
         Ad_TJ : out STD_LOGIC_VECTOR (2 downto 0));--address to
jump
```

```
end COMPONENT;

COMPONENT comparator is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           equal : out STD_LOGIC;
           max_A : out STD_LOGIC;
           max_B : out STD_LOGIC);
end COMPONENT;

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

SIGNAL I,ROM_OUT: STD_LOGIC_VECTOR (12 downto 0);
SIGNAL
RE,RegSel_1,RegSel_2,RS_2,Address_To_Jump,PC_OUT,A,Adder_3_OUT,a
ddress,A_Jump,A_out,Z_out:STD_LOGIC_VECTOR (2 downto 0);
SIGNAL
LS,Add_Or_Sub,Jump_Flag,Reset,R,RegSel,Clk_out,Load_Select,Jump_
F,AOS_EN,MAXA,MAXB,EQL,C_EN:STD_LOGIC;
SIGNAL
ID_OUT,Ch_FJ,X,S_LED,way8_4bit_Mux1_out,way8_4bit_Mux2_out,bit_4
_out,Data_in,Data_out,Adder_OUT:STD_LOGIC_VECTOR (3 downto 0);
Signal
Mux_out,R0_OUT,R1_OUT,R2_OUT,R3_OUT,R4_OUT,R5_OUT,R6_OUT,R7_OUT
: STD_LOGIC_VECTOR (3 downto 0);
Signal Reg_enable,register_select : STD_LOGIC_VECTOR (2 downto
0);
--SIGNAL S_7Seg,Sev_7Seg:STD_LOGIC_VECTOR (6 downto 0);

begin
Instruction_Decoder:INS_DEC
port map(
    I=>ROM_OUT,
    RE=>Reg_enable,
    LS=>Load_Select,
    D=>ID_OUT,
    RS_1=>RegSel_1,
    RS_2=>RegSel_2,
    AOS=>Add_Or_Sub,
    AOS_EN=>AOS_EN,
    CM_EN=>C_EN,
    Ch_FJ=>way8_4bit_Mux1_out,
    JF=>Jump_Flag,
    Ad_TJ=>Address_To_Jump
);
Reg_bank: REGISTER_BANK
port map(
           D =>Data_out,
           CLK => Clk_out,
           Reset=>Res,
           M =>Reg_enable,
           R0=> R0_OUT,
           R1 =>R1_OUT,
           R2 =>R2_OUT,
           R3 =>R3_OUT,
           R4 =>R4_OUT,
           R5 =>R5_OUT,
           R6 =>R6_OUT,
           R7 =>R7_OUT);

Program_Counter: PC
PORT MAP(
   X=>Z_out,
   R=>Res,
   Clk=>Clk_out,
   Y=>PC_OUT
);

BIT_3_Adder:Adder_3_bit
PORT MAP
(
    A=>PC_OUT,
    S=>Adder_3_OUT

);
Seven_Segment:Seven_Seg
port map(
A =>R7_OUT,
anode=>anode,
S_LED => S_LED,
S_7Seg => Sev_7Seg
 );

-- Sev_7Seg<=S_7Seg;
 Mux8_way_4bit_1 : Mux_8_way_4
  port map(
              A => R0_OUT,
              B =>R1_OUT,
              C =>R2_OUT,
              D =>R3_OUT,
              E =>R4_OUT,
              F =>R5_OUT,
              G =>R6_OUT,
              H =>R7_OUT,
              S =>RegSel_1,
              Y =>way8_4bit_Mux1_out
  );
  Mux8_way_4bit_2 : Mux_8_way_4
   port map(
               A => R0_OUT,
               B =>R1_OUT,
               C =>R2_OUT,
               D =>R3_OUT,
               E =>R4_OUT,
               F =>R5_OUT,
               G =>R6_OUT,
               H =>R7_OUT,
               S =>RegSel_2,
               Y =>way8_4bit_Mux2_out
   );

ROM_1:ROM12
  PORT MAP(
    address=>PC_OUT,
    I=>ROM_OUT
  );

 Slow_Clock: Slow_Clk
 port map (
      Clk_in=>Clk,
      Clk_out=>Clk_out
 );

 MUX_2_4_1 : MUX_2_WAY_4_BIT
 port map(
    X => Adder_OUT,
    Y => ID_OUT,
    S => Load_Select,
    Z => Data_out);



adder_subs :AddSub
      port map(
                 A0 =>way8_4bit_Mux2_out(0),
                 A1 =>way8_4bit_Mux2_out(1),
                 A2 =>way8_4bit_Mux2_out(2),
                 A3 =>way8_4bit_Mux2_out(3),
                 B0 =>way8_4bit_Mux1_out(0),
                 B1 =>way8_4bit_Mux1_out(1),
                 B2 =>way8_4bit_Mux1_out(2),
                 B3 =>way8_4bit_Mux1_out(3),
                 --C_in=> '0',
                 AOS =>Add_Or_Sub,
                 AOS_EN=>AOS_EN,
                 S0 =>Adder_OUT(0),
                 S1 =>Adder_OUT(1),
                 S2 =>Adder_OUT(2),
                 S3 =>Adder_OUT(3),
                 --C_Out => out STD_LOGIC;
                 Zero =>Zero,
                 Over_flow => Over_Flow
      );



MUX_2_3_Bit : MUX_2_WAY_3_Bit
          port map(

                 X => Adder_3_OUT,
                 Y=>Address_To_Jump,
                 S => Jump_Flag,
                 Z =>  Z_out);

 COMPARATOR_A_B :comparator
 PORT MAP (
                 A(0) =>way8_4bit_Mux1_out(0),
                 A(1) =>way8_4bit_Mux1_out(1),
                 A(2) =>way8_4bit_Mux1_out(2),
                 A(3) =>way8_4bit_Mux1_out(3),
                 B(0) =>way8_4bit_Mux2_out(0),
                 B(1) =>way8_4bit_Mux2_out(1),
                 B(2) =>way8_4bit_Mux2_out(2),
                 B(3) =>way8_4bit_Mux2_out(3),
                 equal=>EQL,
                 max_A =>MAXA,
                 max_B => MAXB
 );

 Op_Led<=  S_LED;
 equal<=EQL AND NOT C_EN;
 max_A<=MAXA AND NOT C_EN;
 max_B<=MAXB AND NOT C_EN;

end Behavioral;
```

Nano processor Test bench file

```
----------------------------------------------------------------
------------------
-- Company:
-- Engineer:
--
-- Create Date: 04/17/2024 07:13:29 AM
-- Design Name:
-- Module Name: M_P_TB - 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 M_P_TB is
--  Port ( );
end M_P_TB;

architecture Behavioral of M_P_TB is
COMPONENT MICRO_PROCESSOR
Port ( Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           Op_Led : out STD_LOGIC_VECTOR (3 downto 0);
           equal : out STD_LOGIC;
           max_A : out STD_LOGIC;
           max_B : out STD_LOGIC;
           Over_Flow : out STD_LOGIC;
           Zero : out STD_LOGIC;
           anode:out STD_LOGIC_VECTOR (3 downto 0);
           Sev_7Seg :out STD_LOGIC_VECTOR (6 downto 0));
END COMPONENT;
```

Goto Contents

```
SIGNAL Clk,Res,Over_Flow,Zero,equal,max_A,max_B:STD_LOGIC;
SIGNAL Op_Led: STD_LOGIC_VECTOR (3 downto 0);
signal Sev_7Seg : STD_LOGIC_VECTOR (6 downto 0);

begin
UUT:MICRO_PROCESSOR
port map(
Clk=>Clk,
Res=>Res,
equal=>equal,
max_A=>max_A,
max_B=>max_B,
Op_Led=>Op_Led,
Over_Flow=>Over_Flow,
Zero=>Zero,
Sev_7Seg=>Sev_7Seg
);

PROCESS BEGIN
Clk<='0';
WAIT FOR 10ns;
Clk<='1';
WAIT FOR 10ns;
END PROCESS;

PROCESS BEGIN
Res<='1';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
Res<='0';
WAIT FOR 100ns;
```

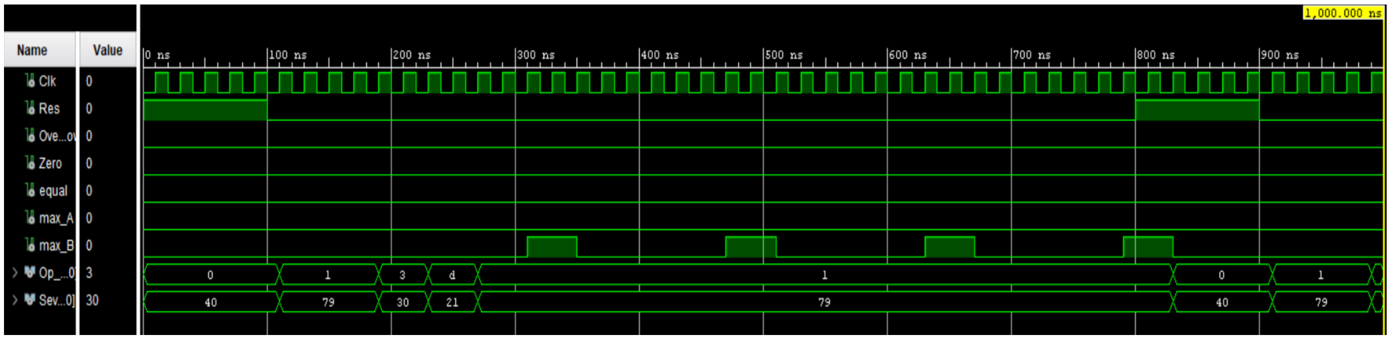
```
Res<='0';
WAIT FOR 100ns;

END PROCESS;

end Behavioral;
```

Constrain file

```
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}]
set_property PACKAGE_PIN U16 [get_ports {Op_Led[0]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Op_Led[0]}]
set_property PACKAGE_PIN E19 [get_ports {Op_Led[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Op_Led[1]}]
set_property PACKAGE_PIN U19 [get_ports {Op_Led[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Op_Led[2]}]
set_property PACKAGE_PIN V19 [get_ports {Op_Led[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Op_Led[3]}]

set_property PACKAGE_PIN U14 [get_ports {max_A}]
     set_property IOSTANDARD LVCMOS33 [get_ports {max_A}]
set_property PACKAGE_PIN V14 [get_ports {equal}]
     set_property IOSTANDARD LVCMOS33 [get_ports {equal}]
set_property PACKAGE_PIN V13 [get_ports {max_B}]
     set_property IOSTANDARD LVCMOS33 [get_ports {max_B}]

set_property PACKAGE_PIN P1 [get_ports {Over_Flow}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Over_Flow}]
set_property PACKAGE_PIN L1 [get_ports {Zero}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]


set_property PACKAGE_PIN W7 [get_ports {Sev_7Seg[0]}]
```

```
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[0]}]
set_property PACKAGE_PIN W6 [get_ports {Sev_7Seg[1]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[1]}]
set_property PACKAGE_PIN U8 [get_ports {Sev_7Seg[2]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[2]}]
set_property PACKAGE_PIN V8 [get_ports {Sev_7Seg[3]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {Sev_7Seg[4]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[4]}]
set_property PACKAGE_PIN V5 [get_ports {Sev_7Seg[5]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {Sev_7Seg[6]}]
     set_property IOSTANDARD LVCMOS33 [get_ports {Sev_7Seg[6]}]



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

set_property PACKAGE_PIN U18 [get_ports Res]
     set_property IOSTANDARD LVCMOS33 [get_ports Res]
```

Timing Diagram



❖ Conclusion

- ➢ In this lab we designed and developed a 4-bit arithmetic unit that can add and subtract signed integers.
- ➢ This nanoprocessor has main components of computer
    1) Memory
        - Primary memory-Registers
        - Secondary memory- Programm ROM
    2) CPU - Execute the instruction
    3) Input and Output- input the clock and output the status of zero flag and overflow, and results of AU.
    4) Bus-Bus carry the data , address and control signals and connect all components.
- ➢ ROM is stored with all instructions. In each clock cycle, the instructions move to Instruction decoder which decode instructions and enable relevant components to execute the program.
- ➢ We developed an 8 way 4 bit multiplexer,2 way 3 bit multiplexer and 2 way 4 bit multiplexer to ease the task of instruction decoder.
- ➢ Also we used 3 to 8 decoders which selected the register to write data.
- ➢ More than basic structure of nanoprocessor, we created comparator
- ➢ and set their outputs to leds.
- ➢ In addition we add 2 instructions to ROM for subtracting without taking negation and comparing the values of two registers.
- ➢ All these component were made separately by group members. Therefore they have used different logics to develop component. We check the functionalities individually by simulating and using baysys3 board, of each component and combine all component by port mapping.

- ➢ The challenge was to set the components because they were made by different group members. We overcome that challenge by team working and many group discussions.
- ➢ This lab helped us to get a better understanding about designing a processor and also it improved our team working skills such as communication, coordination, sharing responsibilities and integrating components developed by different team members.

❖ Contribution of each member

Index Number	Name	Designing Parts	No. of hours spend
220278E	Jayathunga W.M.J.S	8-way 4-bit multiplexer Register Register bank Nano Processor	29 hours 45 mins
220267U	Jayasooriya J.A.N.S	Add/Sub unite Instruction decoder ROM Seven Segment display Nano Processor	32 hours
220282K	Jayawardhana W.S.S.	3-bit Program Counter (PC) 2-way 3-bit multiplexer 2-way 4-bit multiplexer 2 to 1 multiplexer Nano Processor	28 hour 30min

Goto Contents

220585R	Sathsara H.M.W.C.	3-Bit Adder Comparator Nano Processor	25 hours

Goto Contents