

# **Lab 9 – 10**

## **Building up the Nano-Processor**

- 210302P – Kulasekara K.M.S.N
- 210306G – Kulathunga K.A.J.T

### **Lab Task :-**

- Implement 4-bit Add-Subtract unit , 3-bit Adder , 3-bit Program counter ( PC ) , Multiplexers ( 2-way 3-bit , 2-way 4-bit , 8-way 4-bit ) , Register Bank , Program ROM , Instruction Decoder in VHDL.
- Design a simple nano-processor by combining the above components.
- Simulate the components separately to ensure that we can get the expected results.
- Simulate the nano-processor and guarantee the accuracy of its outputs.
- Test on BASYS 3 board and verify the functionality of the nano-processor.

### **Assembly Program :-**

Assembly Programme	Machine code Representation
MOVI R1, 3	100010000011
MOVI R2, 1	100100000001
NEG R2	010100000000
ADD R7, R1	001110010000
ADD R1, R2	000010100000
JZR R1, 7	110010000111
JZR R0, 3	110000000011
JZR R0, 7	110000000111

# VHDL codes

## Add – Sub Unit

RCA\_4.vhd

```
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/imports/newRCA_4.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Add_Sub_Unit is
    Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
           B : in STD_LOGIC_VECTOR(3 downto 0);
           S : out STD_LOGIC_VECTOR(3 downto 0);
           M : in STD_LOGIC; -- (M=0) for addition and (M=1) for subtraction
           Overflow : out STD_LOGIC;
           Zero : out STD_LOGIC);
end Add_Sub_Unit;
architecture Behavioral of Add_Sub_Unit is
component FA
    port (
        A: in std_logic;
        B: in std_logic;
        C_in: in std_logic;
        S: out std_logic;
        C_out: out std_logic);
end component;
SIGNAL FA0_S, FA0_C, FA1_C, FA1_S, FA2_C, FA2_S, FA3_S, FA3_C : std_logic;
SIGNAL S_Copy : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL B0_xor_M, B1_xor_M, B2_xor_M, B3_xor_M : std_logic;
begin
FA_0 : FA
    port map (
        A => A(0),
        B => B0_xor_M,
        C_in => M,
        S => S_Copy(0),
        C_out => FA0_C);
FA_1 : FA
    port map (
        A => A(1),
        B => B1_xor_M,
        C_in => FA0_C,
        S => S_Copy(1),
        C_out => FA1_C);
FA_2 : FA
    port map (
        A => A(2),
        B => B2_xor_M,
        C_in => FA1_C,
        S => S_Copy(2),
        C_out => FA2_C);
FA_3 : FA
    port map (
        A => A(3),
        B => B3_xor_M,
        C_in => FA2_C,
        S => S_Copy(3),
        C_out => FA3_C);
B0_xor_M <= B(0) xor M;
B1_xor_M <= B(1) xor M;
B2_xor_M <= B(2) xor M;
B3_xor_M <= B(3) xor M;
overflow <= FA3_C xor FA2_C;
zero <= not(S_Copy(0) or S_Copy(1) or S_Copy(2) or S_Copy(3));
s<=S_Copy;
end Behavioral;
```

RCA\_4.vhd

```
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/imports/newRCA_4.vhd

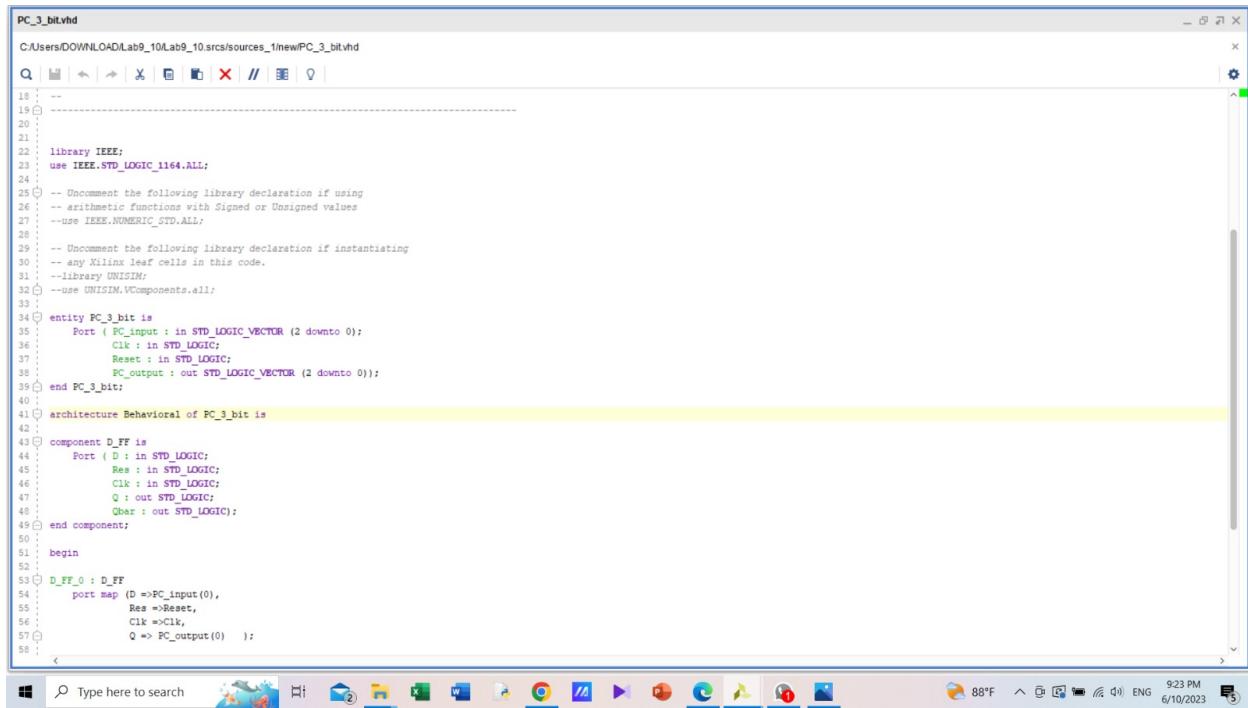
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Add_Sub_Unit is
    Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
           B : in STD_LOGIC_VECTOR(3 downto 0);
           S : out STD_LOGIC_VECTOR(3 downto 0);
           M : in STD_LOGIC; -- (M=0) for addition and (M=1) for subtraction
           Overflow : out STD_LOGIC;
           Zero : out STD_LOGIC);
end Add_Sub_Unit;
architecture Behavioral of Add_Sub_Unit is
component FA
    port (
        A: in std_logic;
        B: in std_logic;
        C_in: in std_logic;
        S: out std_logic;
        C_out: out std_logic);
end component;
SIGNAL FA0_S, FA0_C, FA1_C, FA1_S, FA2_C, FA2_S, FA3_S, FA3_C : std_logic;
SIGNAL S_Copy : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL B0_xor_M, B1_xor_M, B2_xor_M, B3_xor_M : std_logic;
begin
FA_0 : FA
    port map (
        A => A(0),
        B => B0_xor_M,
        C_in => M,
        S => S_Copy(0),
        C_out => FA0_C);
FA_1 : FA
    port map (
        A => A(1),
        B => B1_xor_M,
        C_in => FA0_C,
        S => S_Copy(1),
        C_out => FA1_C);
FA_2 : FA
    port map (
        A => A(2),
        B => B2_xor_M,
        C_in => FA1_C,
        S => S_Copy(2),
        C_out => FA2_C);
FA_3 : FA
    port map (
        A => A(3),
        B => B3_xor_M,
        C_in => FA2_C,
        S => S_Copy(3),
        C_out => FA3_C);
B0_xor_M <= B(0) xor M;
B1_xor_M <= B(1) xor M;
B2_xor_M <= B(2) xor M;
B3_xor_M <= B(3) xor M;
overflow <= FA3_C xor FA2_C;
zero <= not(S_Copy(0) or S_Copy(1) or S_Copy(2) or S_Copy(3));
s<=S_Copy;
end Behavioral;
```

## 3-bit adder

```
RCA_4.vhd (2)
C:/Users/DOWNLOADtab3/srcs/sources_1/newRCA_4.vhd
Q | ̄ | ↻ | ⌂ | X | ⌁ | // | ⌂ | ? |
18 '-----
19 -----
20 -----
21 -----
22 library IEEE;
23 use IEEE.STD.LOGIC_1164.ALL;
24 -----
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28 -----
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33 -----
34 entity RCA_3_bit is
35   Port ( A : in STD_LOGIC_VECTOR ( 2 downto 0);
36         B : in STD_LOGIC_VECTOR ( 2 downto 0);
37         S : out STD_LOGIC_VECTOR ( 2 downto 0) );
38 end RCA_3_bit;
39 -----
40 architecture Behavioral of RCA_3_bit is
41 -----
42 component FA
43   port (
44     A: in std_logic;
45     B: in std_logic;
46     C_in: in std_logic;
47     S: out std_logic;
48     C_out: out std_logic);
49 end component;
50 -----
51 SIGNAL FA0_C, FA1_C, FA2_C : std_logic;
52 -----
53 begin
54 -----
55 FA_0 : FA
56   port map (
57     A => A(0),
58     B => B(0),
59   <
```

```
RCA_4.vhd (2)
C:/Users/DOWNLOADtab3/srcs/sources_1/newRCA_4.vhd
Q | ̄ | ↻ | ⌂ | X | ⌁ | // | ⌂ | ? |
39 -----
40 architecture Behavioral of RCA_3_bit is
41 -----
42 component FA
43   port (
44     A: in std_logic;
45     B: in std_logic;
46     C_in: in std_logic;
47     S: out std_logic;
48     C_out: out std_logic);
49 end component;
50 -----
51 SIGNAL FA0_C, FA1_C, FA2_C : std_logic;
52 -----
53 begin
54 -----
55 FA_0 : FA
56   port map (
57     A => A(0),
58     B => B(0),
59     C_in => '0', -- Set to ground
60     S => S(0),
61     C_out => FA0_C);
62 FA_1 : FA
63   port map (
64     A => A(1),
65     B => B(1),
66     C_in => FA0_C,
67     S => S(1),
68     C_out => FA1_C);
69 FA_2 : FA
70   port map (
71     A => A(2),
72     B => B(2),
73     C_in => FA1_C,
74     S => S(2),
75     C_out => FA2_C);
76 -----
77 end Behavioral;
78 <
```

## 3-bit Program Counter



PC\_3\_bit.vhd

```
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/PC_3_bit.vhd

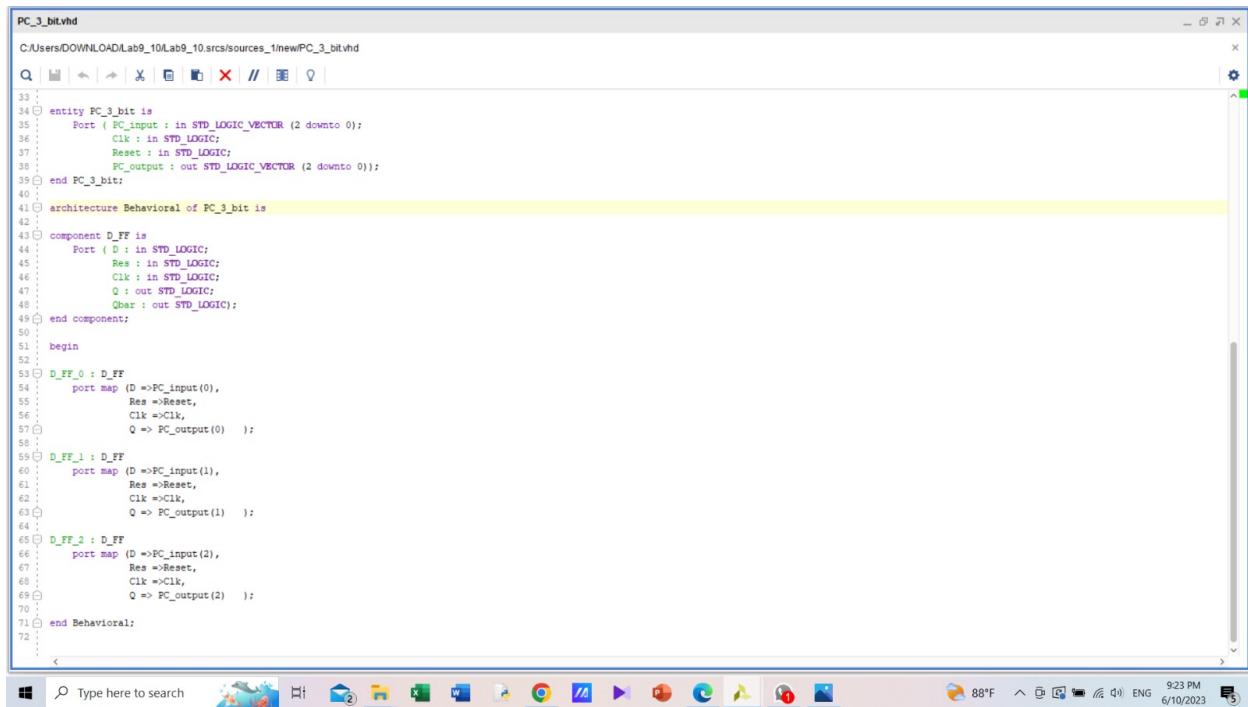
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity PC_3_bit is
    Port ( PC_input : in STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           PC_output : out STD_LOGIC_VECTOR (2 downto 0));
end PC_3_bit;

architecture Behavioral of PC_3_bit is
begin
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;
begin
    D_FF_0 : D_FF
        port map (D =>PC_input(0),
                   Res =>Reset,
                   Clk =>Clk,
                   Q => PC_output(0) );
    end;
```

Type here to search 88°F 9:23 PM 6/10/2023



PC\_3\_bit.vhd

```
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/PC_3_bit.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

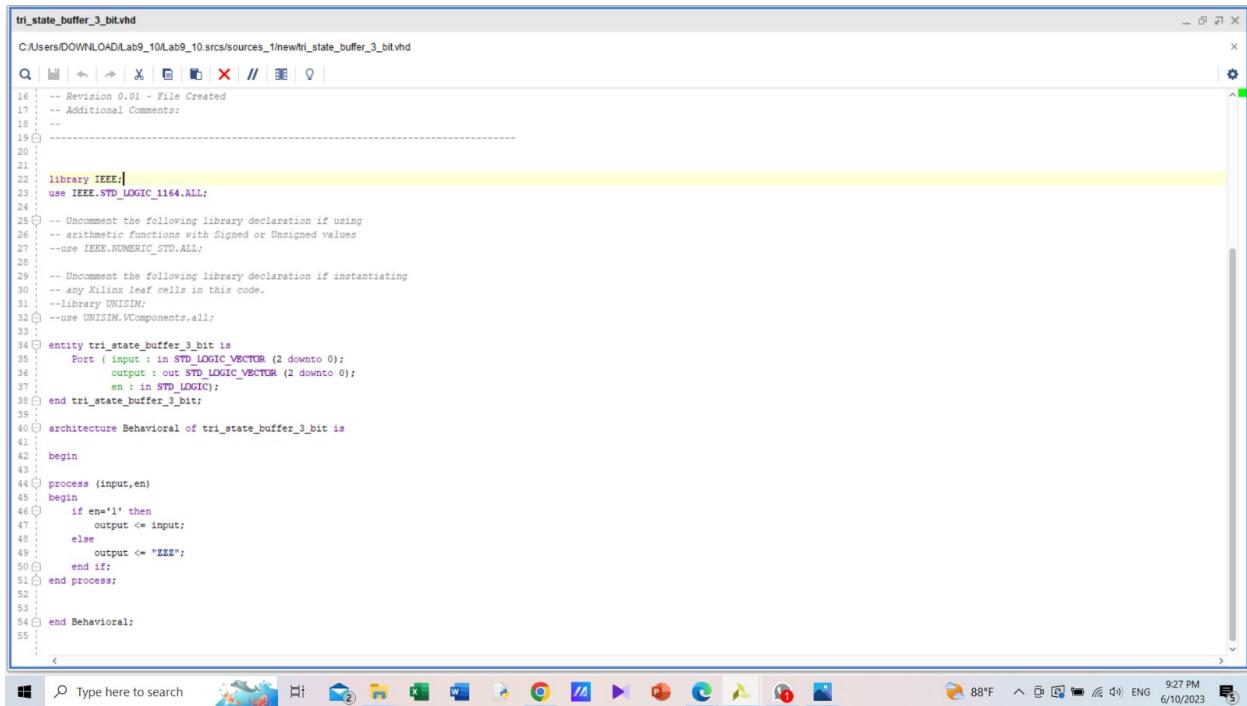
library UNISIM;
use UNISIM.VComponents.all;

entity PC_3_bit is
    Port ( PC_input : in STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           PC_output : out STD_LOGIC_VECTOR (2 downto 0));
end PC_3_bit;

architecture Behavioral of PC_3_bit is
begin
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;
begin
    begin
    D_FF_0 : D_FF
        port map (D =>PC_input(0),
                   Res =>Reset,
                   Clk =>Clk,
                   Q => PC_output(0) );
    D_FF_1 : D_FF
        port map (D =>PC_input(1),
                   Res =>Reset,
                   Clk =>Clk,
                   Q => PC_output(1) );
    D_FF_2 : D_FF
        port map (D =>PC_input(2),
                   Res =>Reset,
                   Clk =>Clk,
                   Q => PC_output(2) );
    end Behavioral;
end;
```

Type here to search 88°F 9:23 PM 6/10/2023

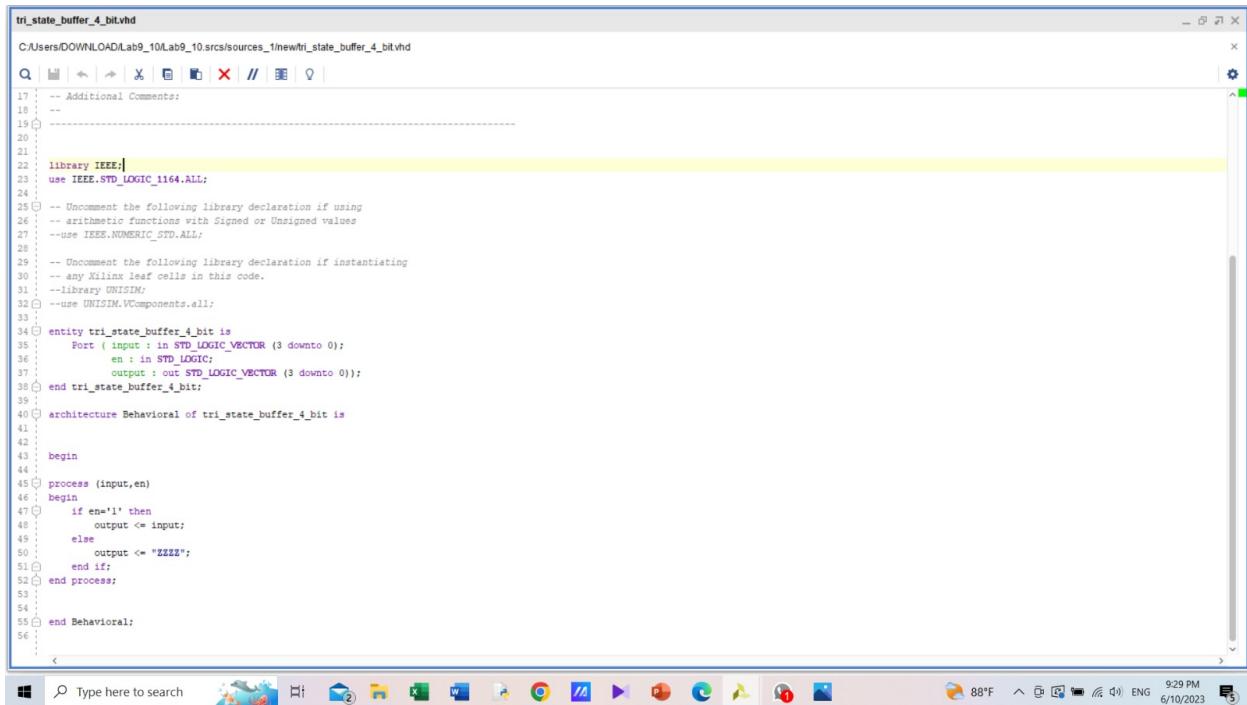
## Tri state 3-bit buffer



```
tri_state_buffer_3_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10/srcs/sources_1/new/tri_state_buffer_3_bit.vhd

16 : -- Revision 0.01 - File Created
17 : -- Additional Comments:
18 :
19 -----
20 :
21 :
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 :
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28 :
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33 :
34 entity tri_state_buffer_3_bit is
35     Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
36             output : out STD_LOGIC_VECTOR (2 downto 0);
37             en : in STD_LOGIC);
38 end tri_state_buffer_3_bit;
39 :
40 architecture Behavioral of tri_state_buffer_3_bit is
41 :
42 begin
43 :
44 process (input,en)
45 begin
46     if en='1' then
47         output <= input;
48     else
49         output <= "ZZZZ";
50     end if;
51 end process;
52 :
53 :
54 end Behavioral;
55 :
56 :
```

## Tri state 4-bit buffer

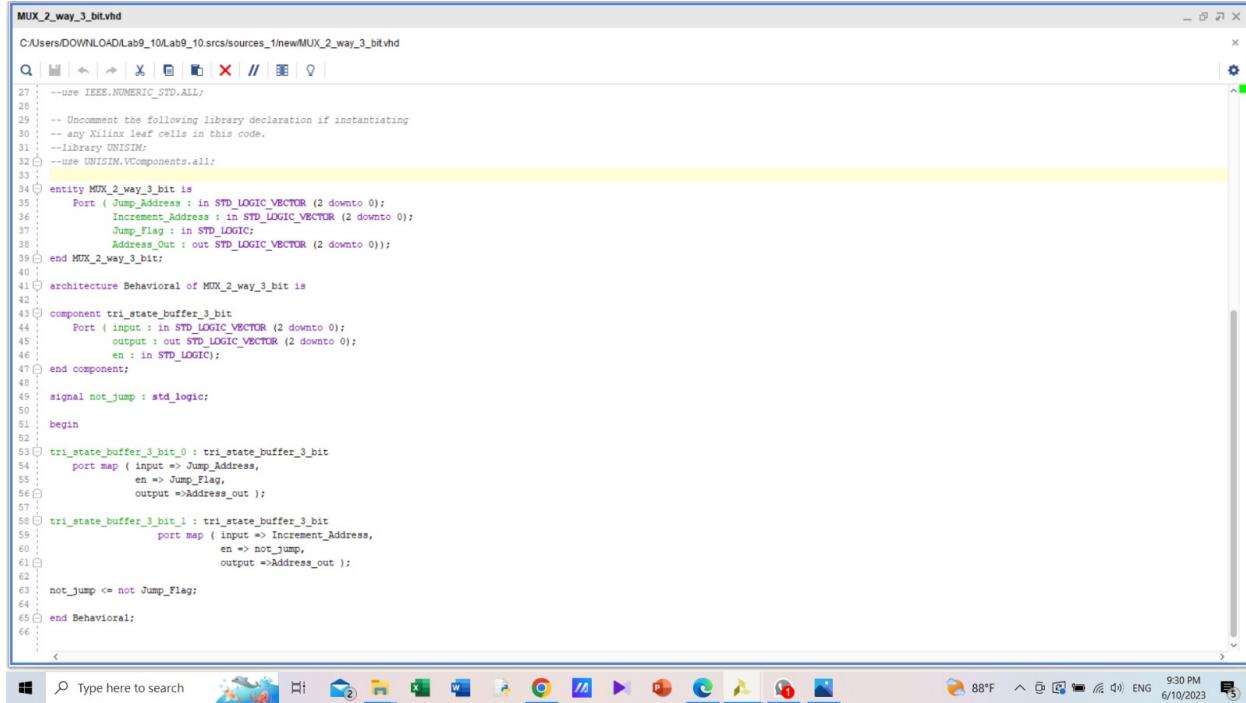


```
tri_state_buffer_4_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10/srcs/sources_1/new/tri_state_buffer_4_bit.vhd

17 : -- Additional Comments:
18 :
19 -----
20 :
21 :
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 :
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28 :
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33 :
34 entity tri_state_buffer_4_bit is
35     Port ( input : in STD_LOGIC_VECTOR (3 downto 0);
36             en : in STD_LOGIC;
37             output : out STD_LOGIC_VECTOR (3 downto 0));
38 end tri_state_buffer_4_bit;
39 :
40 architecture Behavioral of tri_state_buffer_4_bit is
41 :
42 begin
43 :
44 process (input,en)
45 begin
46     if en='1' then
47         output <= input;
48     else
49         output <= "ZZZZ";
50     end if;
51 end process;
52 :
53 :
54 end Behavioral;
55 :
56 :
```

# Multiplexers

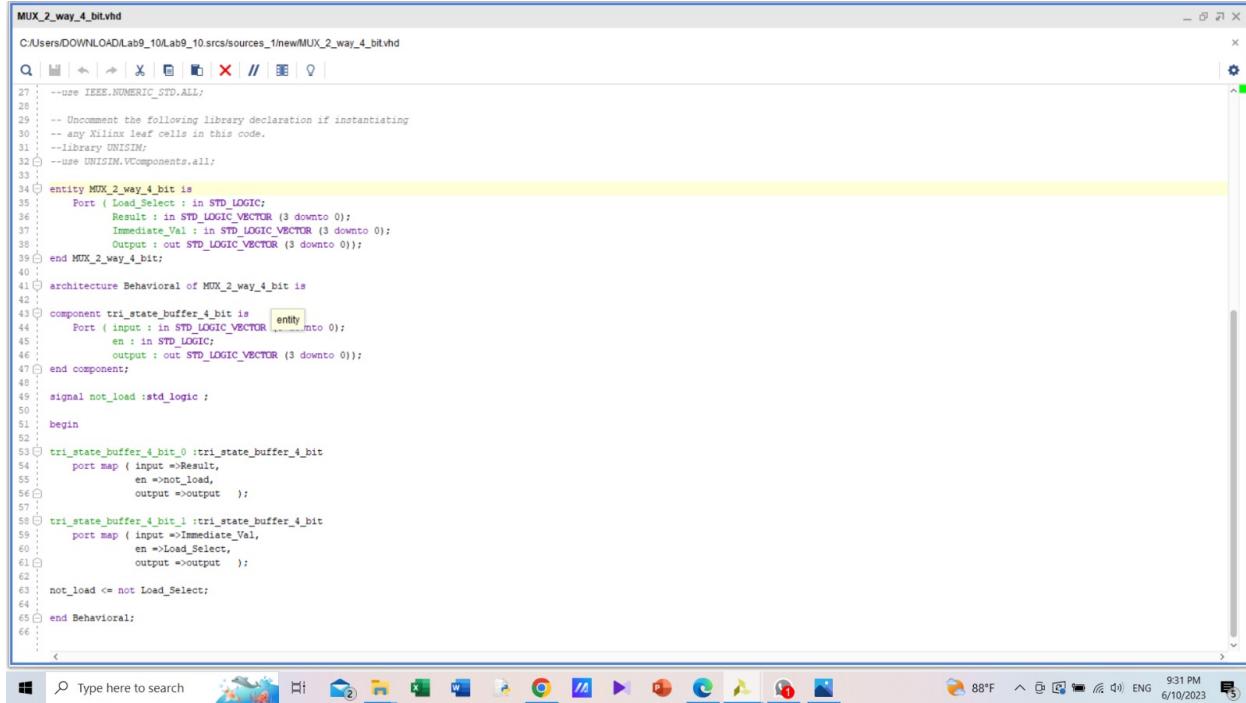
## 2 way 3-bit



```
MUX_2_way_3_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srcs/sources_1/new/MUX_2_way_3_bit.vhd

27 : --use IEEE.NUMERIC_STD.ALL;
28 :
29 : -- Uncomment the following library declaration if instantiating
30 : -- any Xilinx leaf cells in this code.
31 : --library UNISIM;
32 : --use UNISIM.VComponents.all;
33 :
34 : entity MUX_2_way_3_bit is
35 :     Port ( Jump_Address : in STD_LOGIC_VECTOR (2 downto 0);
36 :            Increment_Address : in STD_LOGIC_VECTOR (2 downto 0);
37 :            Jump_Flag : in STD_LOGIC;
38 :            Address_Out : out STD_LOGIC_VECTOR (2 downto 0));
39 : end MUX_2_way_3_bit;
40 :
41 : architecture Behavioral of MUX_2_way_3_bit is
42 :
43 : component tri_state_buffer_3_bit
44 :     Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
45 :            output : out STD_LOGIC_VECTOR (2 downto 0);
46 :            en : in STD_LOGIC);
47 : end component;
48 :
49 : signal not_jump : std_logic;
50 :
51 : begin
52 :
53 :     tri_state_buffer_3_bit_0 : tri_state_buffer_3_bit
54 :         port map ( input => Jump_Address,
55 :                    en => Jump_Flag,
56 :                    output =>Address_out );
57 :
58 :     tri_state_buffer_3_bit_1 : tri_state_buffer_3_bit
59 :         port map ( input => Increment_Address,
60 :                    en => not_jump,
61 :                    output =>Address_out );
62 :
63 :     not_jump <= not Jump_Flag;
64 :
65 : end Behavioral;
66 :
67 :
```

## 2 way 4-bit



```
MUX_2_way_4_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srcs/sources_1/new/MUX_2_way_4_bit.vhd

27 : --use IEEE.NUMERIC_STD.ALL;
28 :
29 : -- Uncomment the following library declaration if instantiating
30 : -- any Xilinx leaf cells in this code.
31 : --library UNISIM;
32 : --use UNISIM.VComponents.all;
33 :
34 : entity MUX_2_way_4_bit is
35 :     Port ( Load_Select : in STD_LOGIC;
36 :            Result : in STD_LOGIC_VECTOR (3 downto 0);
37 :            Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
38 :            Output : out STD_LOGIC_VECTOR (3 downto 0));
39 : end MUX_2_way_4_bit;
40 :
41 : architecture Behavioral of MUX_2_way_4_bit is
42 :
43 : component tri_state_buffer_4_bit is
44 :     Port ( input : in STD_LOGIC_VECTOR (3 downto 0);
45 :            en : in STD_LOGIC;
46 :            output : out STD_LOGIC_VECTOR (3 downto 0));
47 : end component;
48 :
49 : signal not_load :std_logic ;
50 :
51 : begin
52 :
53 :     tri_state_buffer_4_bit_0 :tri_state_buffer_4_bit
54 :         port map ( input =>Result,
55 :                    en =>not_load,
56 :                    output =>output );
57 :
58 :     tri_state_buffer_4_bit_1 :tri_state_buffer_4_bit
59 :         port map ( input =>Immediate_Val,
60 :                    en =>Load_Select,
61 :                    output =>output );
62 :
63 :     not_load <= not Load_Select;
64 :
65 : end Behavioral;
66 :
67 :
```

## 8 way 4-bit

```
MUX_8_way_4_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srcs/sources_1/new/MUX_8_way_4_bit.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library UNISIM;
use UNISIM.VComponents.all;
entity MUX_8_way_4_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD_LOGIC_VECTOR (3 downto 0);
           R2 : in STD_LOGIC_VECTOR (3 downto 0);
           R3 : in STD_LOGIC_VECTOR (3 downto 0);
           R4 : in STD_LOGIC_VECTOR (3 downto 0);
           R5 : in STD_LOGIC_VECTOR (3 downto 0);
           R6 : in STD_LOGIC_VECTOR (3 downto 0);
           R7 : in STD_LOGIC_VECTOR (3 downto 0);
           Reg_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           output_reg : out STD_LOGIC_VECTOR (3 downto 0));
end MUX_8_way_4_bit;
architecture Behavioral of MUX_8_way_4_bit is
component tri_state_buffer_4_bit is
    Port ( input : in STD_LOGIC_VECTOR (3 downto 0);
           en : in STD_LOGIC;
           output : 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;
begin
end;
```

```
MUX_8_way_4_bit.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srcs/sources_1/new/MUX_8_way_4_bit.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library UNISIM;
use UNISIM.VComponents.all;
architecture Behavioral of MUX_8_way_4_bit is
component tri_state_buffer_4_bit is
    Port ( input : in STD_LOGIC_VECTOR (3 downto 0);
           en : in STD_LOGIC;
           output : 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 Register_Select : std_logic_vector(7 downto 0);
begin
Decoder_3_to_8_0 : Decoder_3_to_8
    port map ( I => Reg_Sel,
               en => '1',
               Y => Register_Select );
tri_state_buffer_4_bit_0 : tri_state_buffer_4_bit
    port map ( input => R0,
               en => Register_Select(0),
               output => output_reg );
tri_state_buffer_4_bit_1 : tri_state_buffer_4_bit
    port map ( input => R1,
               en => Register_Select(1),
               output => output_reg );
tri_state_buffer_4_bit_2 : tri_state_buffer_4_bit
    port map ( input => R2,
               en => Register_Select(2),
               output => output_reg );
tri_state_buffer_4_bit_3 : tri_state_buffer_4_bit
    port map ( input => R3,
               en => Register_Select(3),
               output => output_reg );
```

## Register bank

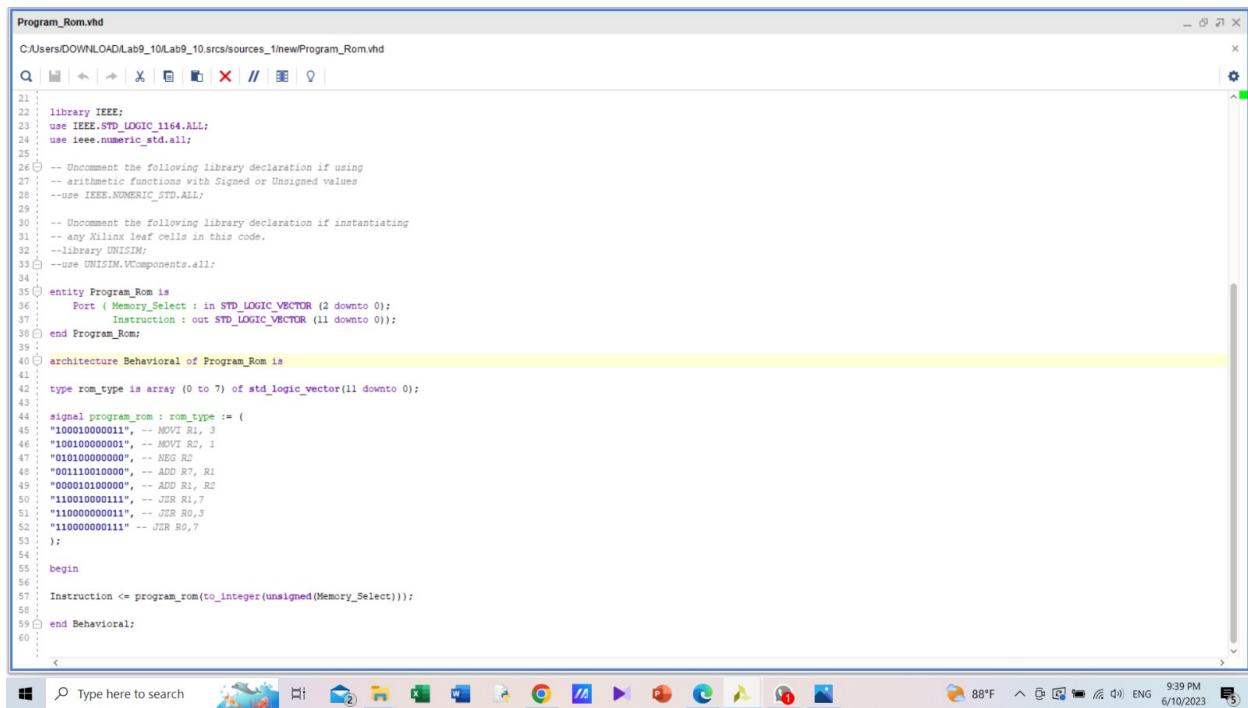
```
Register_Bank.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Register_Bank.vhd

65 end component;
66
67 component Reg_new is
68   Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
69         En : in STD_LOGIC;
70         Clk : in STD_LOGIC;
71         Res : in STD_LOGIC;
72         Q : out STD_LOGIC_VECTOR (3 downto 0));
73 end component;
74
75 signal Reg_en : std_logic_vector(7 downto 0);
76 begin
77
78 Decoder_3_to_8_0 : Decoder_3_to_8
79   port map ( I => Register_enable,
80             EN =>'1',
81             Y => Reg_en );
82
83 Reg_0 : Reg_new
84   port map ( D => Data,
85             En =>Reg_en(0),
86             Clk => Clk,
87             Res => Reset,
88             Q => R0 );
89
90 Reg_1 : Reg_new
91   port map ( D => Data,
92             En =>Reg_en(1),
93             Clk => Clk,
94             Res => Reset,
95             Q => R1 );
96
97 Reg_2 : Reg_new
98   port map ( D => Data,
99             En =>Reg_en(2),
100            Clk => Clk,
101            Res => Reset,
102            Q => R2 );
103
104 Reg_3 : Reg_new
105   port map ( D => Data,
106             En =>Reg_en(3),
107             Clk => Clk,
108             Res => Reset,
109             Q => R3 );
110
111 Reg_4 : Reg_new
112   port map ( D => Data,
113             En =>Reg_en(4),
114             Clk => Clk,
115             Res => Reset,
116             Q => R4 );
117
118 Reg_5 : Reg_new
119   port map ( D => Data,
120             En =>Reg_en(5),
121             Clk => Clk,
122             Res => Reset,
123             Q => R5 );
124
125 Reg_6 : Reg_new
126   port map ( D => Data,
127             En =>Reg_en(6),
128             Clk => Clk,
129             Res => Reset,
130             Q => R6 );
131
132 Reg_7 : Reg_new
133   port map ( D => Data,
134             En =>Reg_en(7),
135             Clk => Clk,
136             Res => Reset,
137             Q => R7 );
138
139 end Behavioral;
```

```
Register_Bank.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Register_Bank.vhd

81 Res => Reset,
82 Q => R2 );
83
84 Reg_3 : Reg_new
85   port map ( D => Data,
86             En =>Reg_en(3),
87             Clk => Clk,
88             Res => Reset,
89             Q => R3 );
90
91 Reg_4 : Reg_new
92   port map ( D => Data,
93             En =>Reg_en(4),
94             Clk => Clk,
95             Res => Reset,
96             Q => R4 );
97
98 Reg_5 : Reg_new
99   port map ( D => Data,
100             En =>Reg_en(5),
101             Clk => Clk,
102             Res => Reset,
103             Q => R5 );
104
105 Reg_6 : Reg_new
106   port map ( D => Data,
107             En =>Reg_en(6),
108             Clk => Clk,
109             Res => Reset,
110             Q => R6 );
111
112 Reg_7 : Reg_new
113   port map ( D => Data,
114             En =>Reg_en(7),
115             Clk => Clk,
116             Res => Reset,
117             Q => R7 );
118
119 end Behavioral;
```

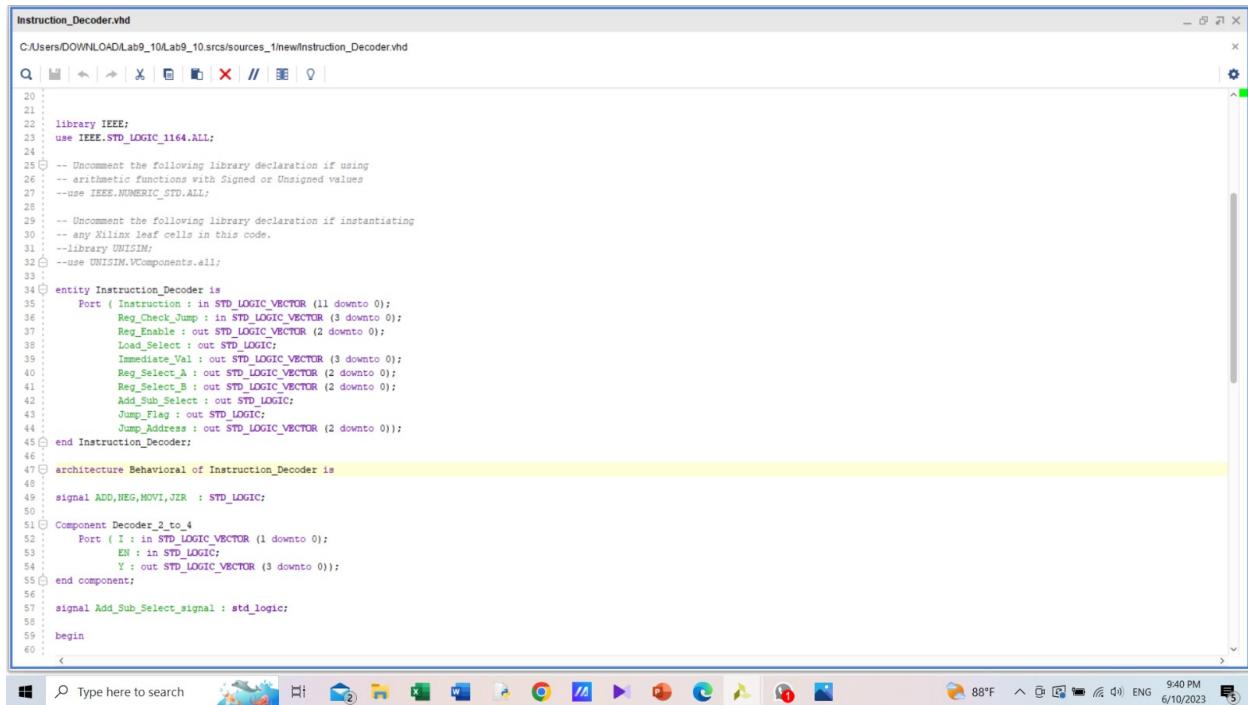
## Program ROM



```
Program_Rom.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Program_Rom.vhd

21 :
22 : library IEEE;
23 : use IEEE.STD_LOGIC_1164.ALL;
24 : use ieee.numeric_std.all;
25 :
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 --use IEEE.NUMERIC_STD.ALL;
29 :
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34 :
35 entity Program_Rom is
36   Port ( Memory_Select : in STD_LOGIC_VECTOR (2 downto 0);
37         Instruction : out STD_LOGIC_VECTOR (11 downto 0));
38 end Program_Rom;
39 :
40 architecture Behavioral of Program_Rom is
41 :
42 type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);
43 :
44 signal program_rom : rom_type := (
45   "100010000011", -- MOVI R1, 3
46   "100100000001", -- MOVI R2, 1
47   "010100000000", -- NEG R2
48   "001110010000", -- ADD R7, R1
49   "000010100000", -- ADD R1, R2
50   "110010000011", -- JZR R1, 7
51   "110000000011", -- JZR R0, 3
52   "110000000011" -- JZR R0, 7
53 );
54 :
55 begin
56 :
57   Instruction <= program_rom(to_integer(unsigned(Memory_Select)));
58 :
59 end Behavioral;
60 :
61 
```

## Instruction decoder



```
Instruction_Decoder.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Instruction_Decoder.vhd

20 :
21 :
22 : library IEEE;
23 : use IEEE.STD_LOGIC_1164.ALL;
24 :
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28 :
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33 :
34 entity Instruction_Decoder is
35   Port ( Instruction : in STD_LOGIC_VECTOR (11 downto 0);
36         Reg_Check_Jump : in STD_LOGIC_VECTOR (3 downto 0);
37         Reg_Enable : out STD_LOGIC_VECTOR (2 downto 0);
38         Load_Select : out STD_LOGIC;
39         Immediate_Val : out STD_LOGIC_VECTOR (3 downto 0);
40         Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
41         Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
42         Add_Sub_Select : out STD_LOGIC;
43         Jump_Flag : out STD_LOGIC;
44         Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
45 end Instruction_Decoder;
46 :
47 architecture Behavioral of Instruction_Decoder is
48 :
49 signal ADD,NEG,MOVI,JZR : STD_LOGIC;
50 :
51 component Decoder_2_to_4
52   Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
53         EN : in STD_LOGIC;
54         Y : out STD_LOGIC_VECTOR (3 downto 0));
55 end component;
56 :
57 signal Add_Sub_Select_signal : std_logic;
58 :
59 begin
60   <
```

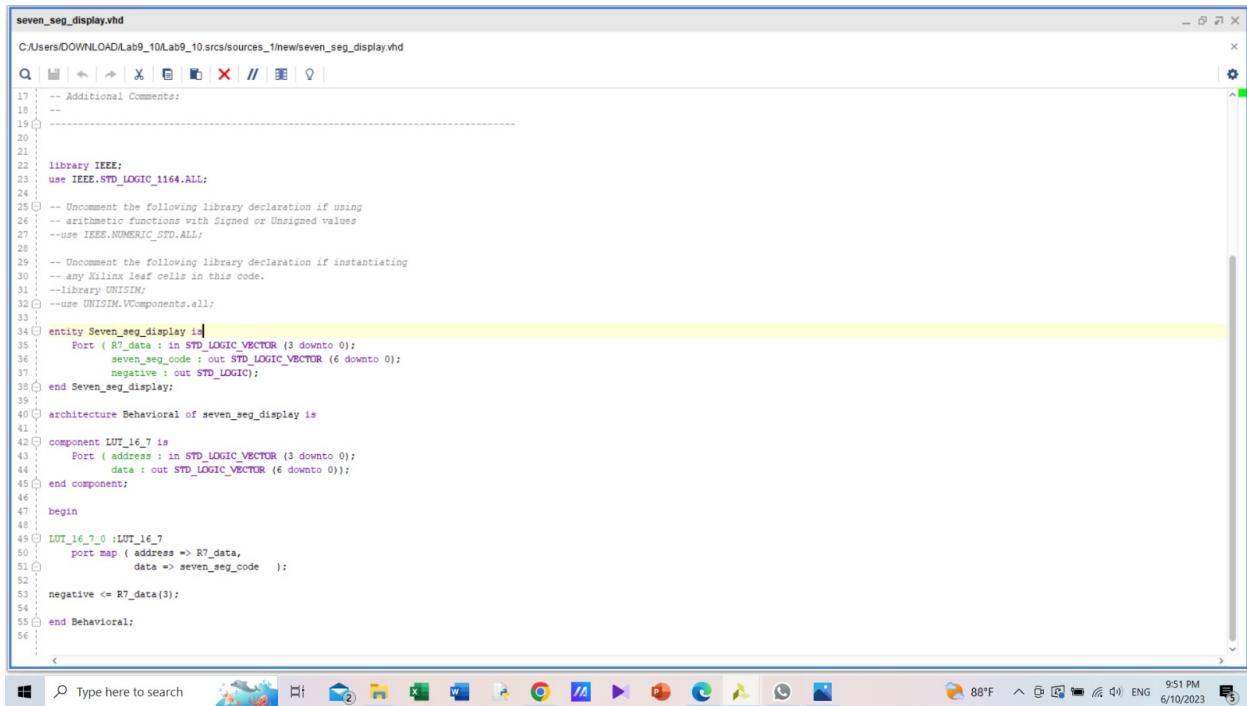
```
Instruction_Decoder.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Instruction_Decoder.vhd

begin
Decoder_2_to_4_0 : Decoder_2_to_4
port map (I => Instruction (11 downto 10),
           EN => '1',
           Y(0) => ADD,
           Y(1) => NEG,
           Y(2) => MOVI,
           Y(3) => JZR      );
end;
process (ADD,NEG,MOVI,JZR,Instruction,Reg_Check_Jump)
begin
if ADD='1' Then
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= "0000";
    Load_Select <= '0';
    Reg_Select_A <= Instruction(9 downto 7) ;
    Reg_Select_B <= Instruction (6 downto 4) ;
    Add_Sub_Select <= '0';
    Jump_Flag <='0';
    Jump_Address <="0000";
elsif NEG='1' Then
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= "0000";
    Load_Select <= '0';
    Reg_Select_A <= "000" ;
    Reg_Select_B <= Instruction(9 downto 7) ;
    Add_Sub_Select <= '1';
    Jump_Flag <='0';
    Jump_Address <="0000";
elsif MOVI ='1' THEN
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= Instruction ( 3 downto 0);
    Load_Select <= '1';
    Reg_Select_B <="000";
    Reg_Select_A <="000";
    Add_Sub_Select <= '0';
    Jump_Flag <='0';
    Jump_Address <="0000";
else
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= "0000";
    Load_Select <= '0';
    Reg_Select_A <= "000" ;
    Reg_Select_B <= Instruction(9 downto 7) ;
    Add_Sub_Select <= '1';
    Jump_Flag <='0';
    Jump_Address <="0000";
end if;
end process;
end Behavioral;
```

```
Instruction_Decoder.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srsc/sources_1/new/Instruction_Decoder.vhd

Load_Select <= '0';
Reg_Select_A <= Instruction(9 downto 7) ;
Reg_Select_B <= Instruction (6 downto 4) ;
Add_Sub_Select <= '0';
Jump_Flag <='0';
Jump_Address <="0000";
elsif NEG='1' Then
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= "0000";
    Load_Select <= '0';
    Reg_Select_A <= "000" ;
    Reg_Select_B <= Instruction(9 downto 7) ;
    Add_Sub_Select <= '1';
    Jump_Flag <='0';
    Jump_Address <="0000";
elsif MOVI ='1' THEN
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= Instruction ( 3 downto 0);
    Load_Select <= '1';
    Reg_Select_A <= "000";
    Reg_Select_B <="000";
    Add_Sub_Select <= '0';
    Jump_Flag <='0';
    Jump_Address <="0000";
else
    Reg_Enable <= Instruction (9 downto 7);
    Immediate_Val <= "0000";
    Load_Select <= '0';
    Reg_Select_A <= Instruction(9 downto 7);
    Reg_Select_B <="000";
    Add_Sub_Select <= '0';
    Jump_Flag <='0';
    Jump_Address <= Instruction( 2 downto 0);
end if;
end process;
end Behavioral;
```

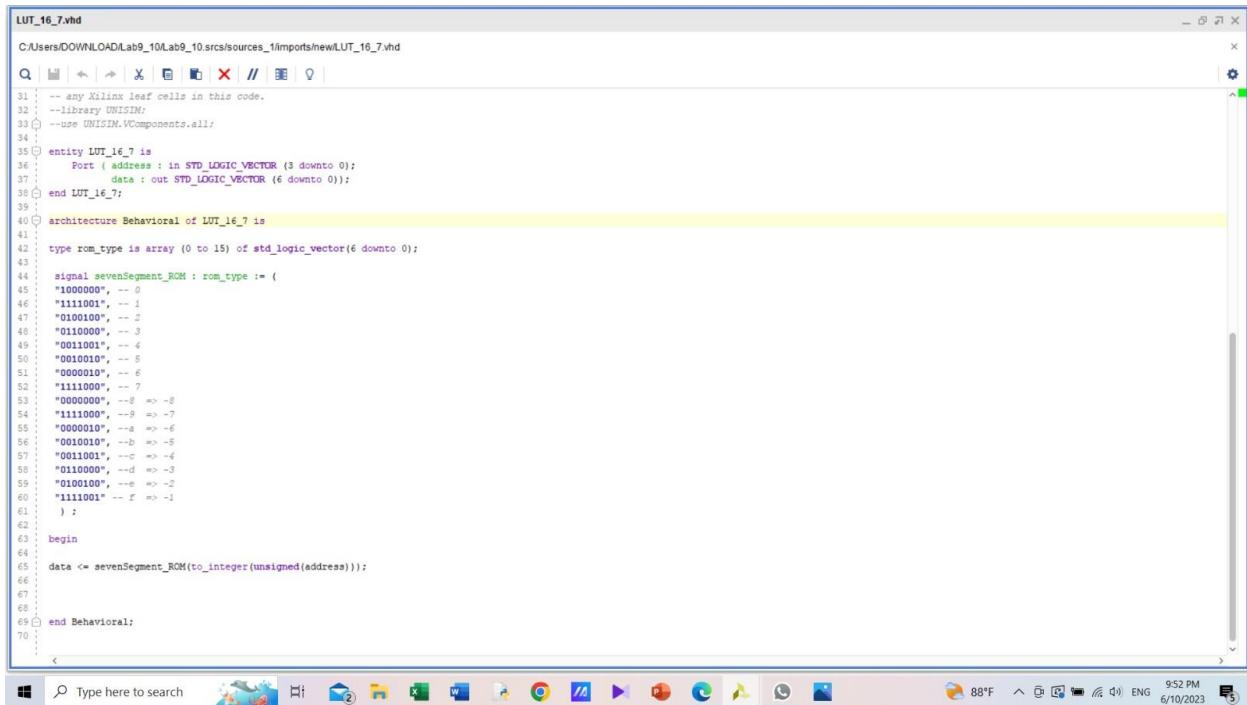
# Seven segment display



```
seven_seg_display.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srzs/sources_1/new/seven_seg_display.vhd

17'-- Additional Comments:
18'--
19'-----
20'-
21'-
22'library IEEE;
23'use IEEE.STD_LOGIC_1164.ALL;
24'
25'-- Uncomment the following library declaration if using
26'-- arithmetic functions with Signed or Unsigned values
27'--use IEEE.NUMERIC_STD.ALL;
28'
29'-- Uncomment the following library declaration if instantiating
30'-- any Xilinx leaf cells in this code.
31'--library UNISIM;
32'--use UNISIM.VComponents.all;
33'
34'entity Seven_seg_display is
35'    Port ( R7_data : in STD_LOGIC_VECTOR (3 downto 0);
36'          seven_seg_code : out STD_LOGIC_VECTOR (6 downto 0);
37'          negative : out STD_LOGIC);
38'end Seven_seg_display;
39'
40'architecture Behavioral of seven_seg_display is
41'
42'component LUT_16_7 is
43'    Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
44'          data : out STD_LOGIC_VECTOR (6 downto 0));
45'end component;
46'
47'begin
48'
49'LUT_16_7_0 :LUT_16_7
50'    port map ( address => R7_data,
51'                data => seven_seg_code );
52'
53'    negative <= R7_data(3);
54'
55'end Behavioral;
56'
57'<
```

## Lut\_16\_7

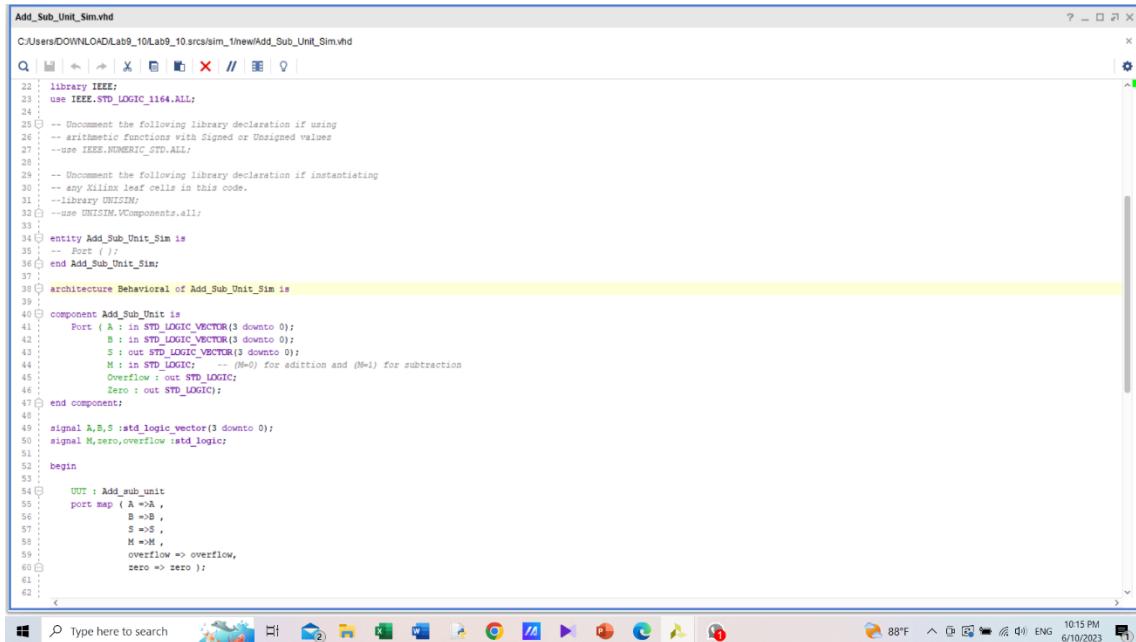


```
LUT_16_7.vhd
C:/Users/DOWNLOAD/Lab9_10/Lab9_10.srzs/sources_1/imports/new/LUT_16_7.vhd

31'-- any Xilinx leaf cells in this code.
32'--library UNISIM;
33'--use UNISIM.VComponents.all;
34'
35'entity LUT_16_7 is
36'    Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
37'          data : out STD_LOGIC_VECTOR (6 downto 0));
38'end LUT_16_7;
39'
40'architecture Behavioral of LUT_16_7 is
41'
42'type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);
43'
44'signal sevenSegment_ROM : rom_type := (
45'    "1000000",
46'    "1111001", -- 1
47'    "0100100", -- 2
48'    "0110010", -- 3
49'    "0011001", -- 4
50'    "0010010", -- 5
51'    "0000010", -- 6
52'    "1111000", -- 7
53'    "0000000", -- 8
54'    "1111000", -- 9
55'    "0000010", -- 0
56'    "0010010", -- b
57'    "0011001", -- c
58'    "0110000", -- d
59'    "0110010", -- e
60'    "1111001" -- f
61'    );
62'
63'begin
64'
65'data <= sevenSegment_ROM(to_integer(unsigned(address)));
66'
67'
68'end Behavioral;
69'
70'<
```

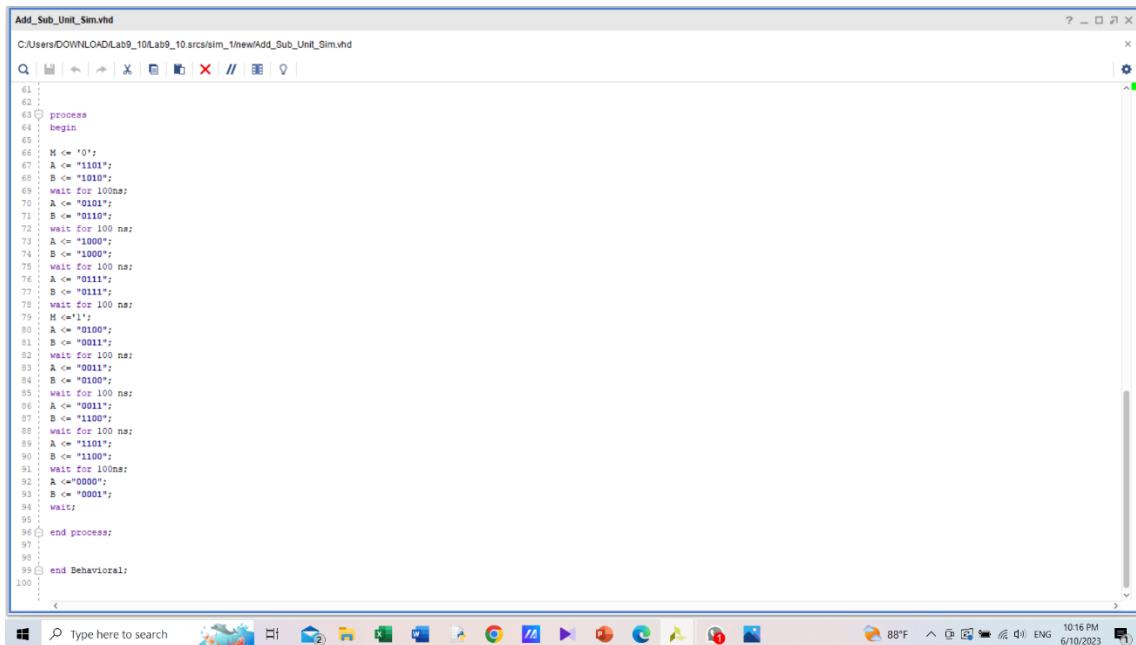
# Test Bench Files

## Add/Sub Unit



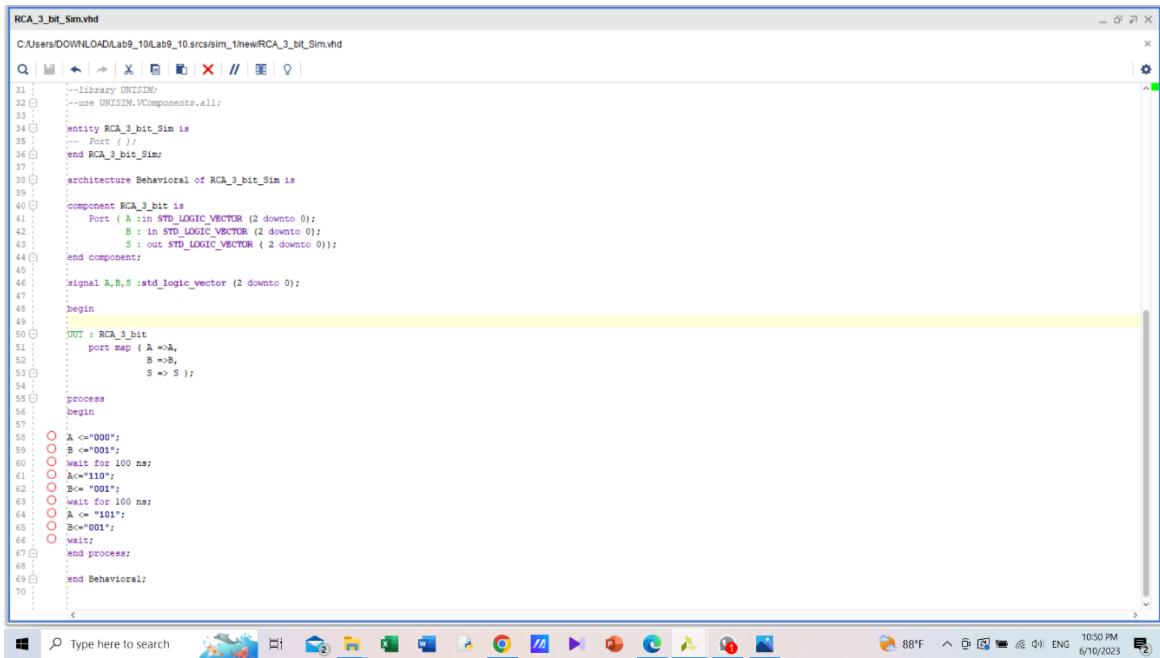
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Add_Sub_Unit_Sim is
  -- Port ();
end Add_Sub_Unit_Sim;
architecture Behavioral of Add_Sub_Unit_Sim is
component Add_Sub_Unit is
  Port ( A : in STD_LOGIC_VECTOR(3 downto 0);
         B : in STD_LOGIC_VECTOR(3 downto 0);
         S : out STD_LOGIC_VECTOR(3 downto 0);
         M : in STD_LOGIC; -- (M=0) for addition and (M=1) for subtraction
         Overflow : out STD_LOGIC;
         Zero : out STD_LOGIC);
end component;
signal A,B,S :std_logic_vector(3 downto 0);
signal M_zero,overflow :std_logic;
begin
begin
  UUT : Add_sub_unit
    port map ( A =>A ,
               B =>B ,
               S =>S ,
               M=>M ,
               overflow => overflow,
               zero => zero );
end;

```



```
process
begin
  M <= '0';
  A <= "1101";
  B <= "1010";
  wait for 100ns;
  A <= "0101";
  B <= "0110";
  wait for 100 ns;
  A <= "1000";
  B <= "0100";
  wait for 100 ns;
  A <= "0111";
  B <= "0111";
  wait for 100 ns;
  M <='1';
  A <= "0100";
  B <= "0011";
  wait for 100 ns;
  A <= "0011";
  B <= "0100";
  wait for 100 ns;
  A <= "0011";
  B <= "1100";
  wait for 100 ns;
  A <= "1101";
  B <= "1100";
  wait for 100 ns;
  A <= "1101";
  B <= "1100";
  wait for 100 ns;
  A <= "0000";
  B <= "0001";
  wait;
end process;
end Behavioral;
```

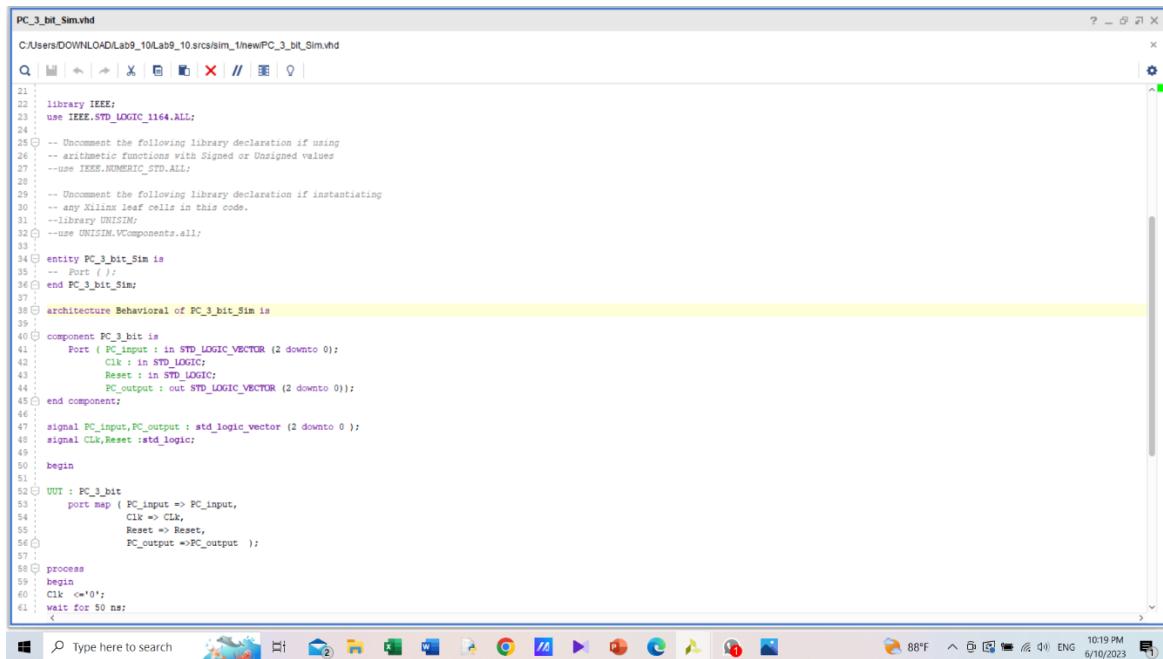
## 3-bit Adder



RCA\_3\_bit\_Sim.vhd

```
31: library UNISIM;
32: use UNISIM.VComponents.all;
33:
34: entity RCA_3_bit_Sim is
35:   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
36:          B : in STD_LOGIC_VECTOR (2 downto 0);
37:          S : out STD_LOGIC_VECTOR ( 2 downto 0));
38: end entity;
39:
40: architecture Behavioral of RCA_3_bit_Sim is
41: component RCA_3_bit is
42:   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
43:          B : in STD_LOGIC_VECTOR (2 downto 0);
44:          S : out STD_LOGIC_VECTOR ( 2 downto 0));
45: end component;
46:
47: signal A,B,S :std_logic_vector (2 downto 0);
48:
49: begin
50:   UUT : RCA_3_bit
51:     port map ( A =>A,
52:                B =>B,
53:                S => S );
54:
55:   process
56:   begin
57:
58:     A <="000";
59:     B <="001";
60:     wait for 100 ns;
61:     A<="110";
62:     B<="001";
63:     wait for 100 ns;
64:     A <="101";
65:     B<="001";
66:     wait;
67:   end process;
68:
69: end Behavioral;
70:
```

## 3-bit Program Counter



PC\_3\_bit\_Sim.vhd

```
21: library IEEE;
22: use IEEE.STD_LOGIC_1164.ALL;
23:
24: -- Uncomment the following library declaration if using
25: -- arithmetic functions with Signed or Unsigned values
26: --use IEEE.NUMERIC_STD.ALL;
27:
28: -- Uncomment the following library declaration if instantiating
29: -- any Xilinx leaf cells in this code.
30: --library UNISIM;
31: --use UNISIM.VComponents.all;
32:
33: entity PC_3_bit_Sim is
34:   -- Port ();
35: end PC_3_bit_Sim;
36:
37:
38: architecture Behavioral of PC_3_bit_Sim is
39:
40: component PC_3_bit is
41:   Port ( PC_input : in STD_LOGIC_VECTOR (2 downto 0);
42:          Clk : in STD_LOGIC;
43:          Reset : in STD_LOGIC;
44:          PC_output : out STD_LOGIC_VECTOR (2 downto 0));
45: end component;
46:
47: signal PC_input,PC_output : std_logic_vector (2 downto 0);
48: signal CLK,Reset :std_logic;
49:
50: begin
51:
52:   UUT : PC_3_bit
53:     port map ( PC_input => PC_input,
54:                Clk => CLK,
55:                Reset => Reset,
56:                PC_output =>PC_output );
57:
58:   process
59:   begin
60:     Clk <="0";
61:     wait for 50 ns;
62:
```

```
PC_3_bit_Sim.vhd
C:\Users\DDOWNLOAD\Lab9_10\Lab9_10\srcs\sim_1\newPC_3_bit_Sim.vhd

Q | ━ | ← | → | X | ☰ | ━ | X | // | ━ | ? |
45 end component;
46
47 signal PC_input,PC_output : std_logic_vector ( 2 downto 0 );
48 signal CLK,Reset :std_logic;
49 begin
50 begin
51 UUT : PC_3_bit
52 port map ( PC_input => PC_input,
53 CLK => CLK,
54 Reset => Reset,
55 PC_output =>PC_output );
56
57 process
58 begin
59 Clk <='0';
60 wait for 50 ns;
61 Clk <= not Clk;
62 wait for 50 ns;
63 end process;
64
65 process
66 begin
67
68 PC_input <="111";
69 Reset <='0';
70 wait for 100 ns;
71 PC_input <="110";
72 wait for 100 ns;
73 PC_input <="010";
74 wait for 100 ns;
75 Reset <='1';
76 wait for 100 ns;
77 PC_input <="111";
78 Reset <='1';
79 wait;
80 end process;
81
82 end Behavioral;
83
84
```

Type here to search               88°F 10:19 PM ENG 6/10/2023

## Tri state 3-bit buffer

```
tri_state_buffer_3_bit_sim.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10.srcs\sim_1\newfri_state_buffer_3_bit_sim.vhd

36 end tri_state_buffer_3_bit_sim;
37
38 architecture Behavioral of tri_state_buffer_3_bit_sim is
39
40 component tri_state_buffer_3_bit is
41   Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
42          output : out STD_LOGIC_VECTOR (2 downto 0);
43          en : in STD_LOGIC);
44 end component;
45
46 signal input,output :std_logic_vector (2 downto 0);
47 signal en : std_logic;
48
49 begin
50
51 UUT : tri_state_buffer_3_bit
52   port map ( input => input,
53              output => output,
54              en => en );
55
56 process
57 begin
58
59  input <="11";
60  wait for 100 ns;
61  input <="000";
62  wait for 100 ns;
63  input <="110";
64  wait for 100 ns;
65  input <="01";
66  wait for 100 ns;
67  input <="111";
68  wait for 100 ns;
69  input <="110";
70
71 end process;
72
73
74 end Behavioral;
75
```

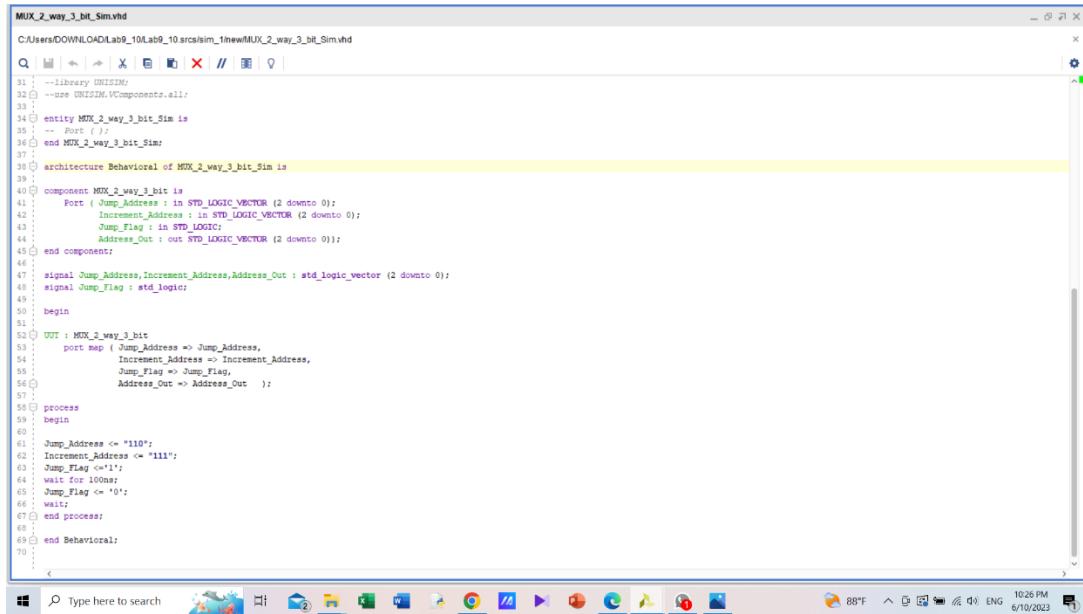
## Tri state 4-bit buffer

```
tri_state_buffer_4_bit_Sim.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10.srcs\sim_1\newfri_state_buffer_4_bit_Sim.vhd

27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity tri_state_buffer_4_bit_Sim is
35   -- Port ()
36 end tri_state_buffer_4_bit_Sim;
37
38 architecture Behavioral of tri_state_buffer_4_bit_Sim is
39
40 component tri_state_buffer_4_bit is
41   Port ( input : in STD_LOGIC_VECTOR (3 downto 0);
42          en : in STD_LOGIC;
43          output : out STD_LOGIC_VECTOR (3 downto 0));
44 end component;
45
46 signal input,output : std_logic_vector (3 downto 0);
47 signal en : std_logic;
48
49 begin
50
51 UUT : tri_state_buffer_4_bit
52   port map ( input => input,
53              output => output,
54              en => en );
55
56 process
57 begin
58
59  input <="111";
60  input <="1111";
61  wait for 100ns;
62  en <='0';
63
64 end process;
65
66 end Behavioral;
67
```

# Multiplexers

## 2 way 3-bit



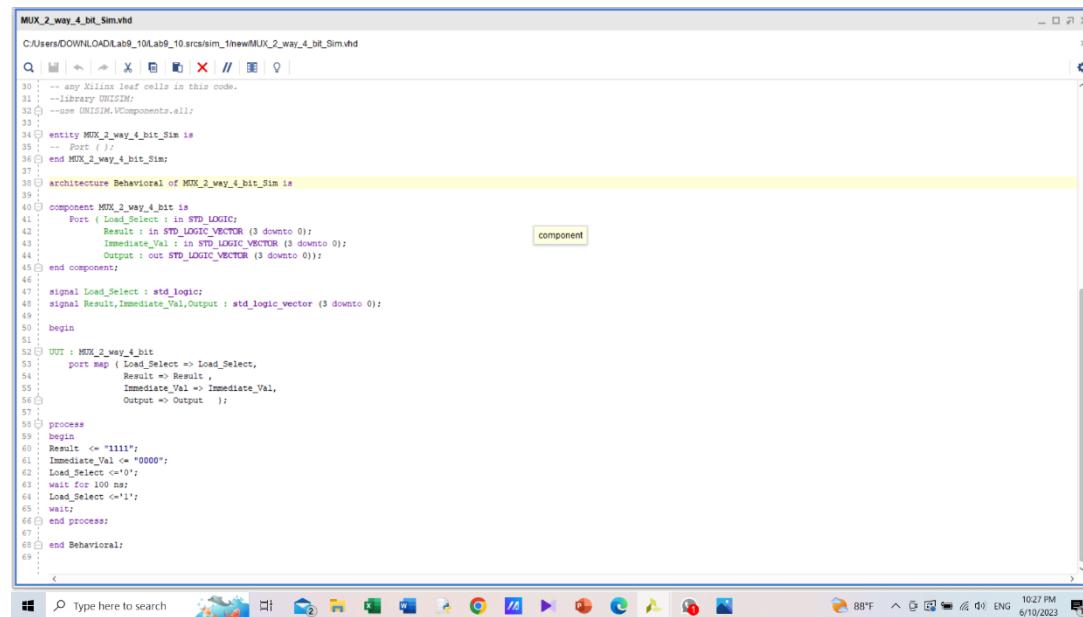
```
MUX_2_way_3_bit_Sim.vhd
C:/Users/DOWNLOADLab9_10Lab9_10/srcs/sim_1newMUX_2_way_3_bit_Sim.vhd

library UNISIM;
use UNISIM.VComponents.all;
entity MUX_2_way_3_bit_Sim is
    -- Port
end MUX_2_way_3_bit_Sim;

architecture Behavioral of MUX_2_way_3_bit_Sim is
component MUX_2_way_3_bit is
    Port ( Jump_Address : in STD_LOGIC_VECTOR (2 downto 0);
           Increment_Address : in STD_LOGIC_VECTOR (2 downto 0);
           Jump_Flag : in STD_LOGIC;
           Address_Out : out STD_LOGIC_VECTOR (2 downto 0));
end component;
begin
    UUT : MUX_2_way_3_bit
        port map ( Jump_Address => Jump_Address,
                   Increment_Address => Increment_Address,
                   Jump_Flag => Jump_Flag,
                   Address_Out => Address_Out );
    process
    begin
        Jump_Address <= "110";
        Increment_Address <= "111";
        Jump_Flag <= '1';
        wait for 10ns;
        Jump_Flag <= '0';
        wait;
    end process;
end Behavioral;

```

## 2 way 4-bit



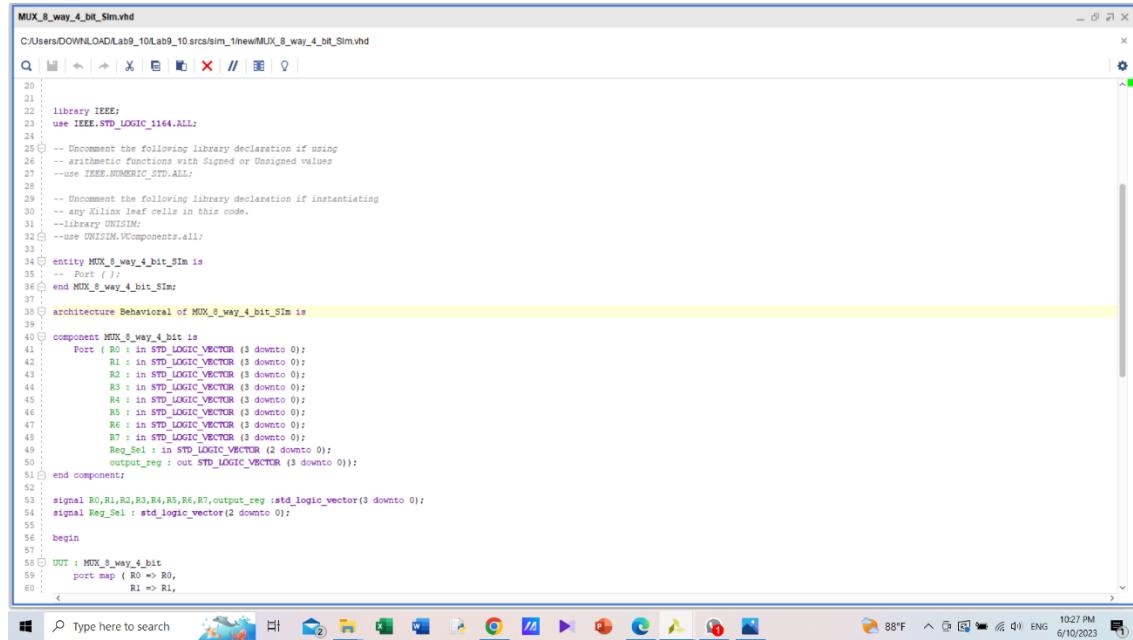
```
MUX_2_way_4_bit_Sim.vhd
C:/Users/DOWNLOADLab9_10Lab9_10/srcs/sim_1newMUX_2_way_4_bit_Sim.vhd

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 is
    Port ( Load_Select : in STD_LOGIC;
           Result : in STD_LOGIC_VECTOR (3 downto 0);
           Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
           Output : out STD_LOGIC_VECTOR (3 downto 0));
end component;
begin
    UUT : MUX_2_way_4_bit
        port map ( Load_Select => Load_Select,
                   Result => Result,
                   Immediate_Val => Immediate_Val,
                   Output => Output );
    process
    begin
        Result <= "1111";
        Immediate_Val <= "0000";
        Load_Select <= '0';
        wait for 100 ns;
        Load_Select <= '1';
        wait;
    end process;
end Behavioral;

```

## 8 way 4-bit

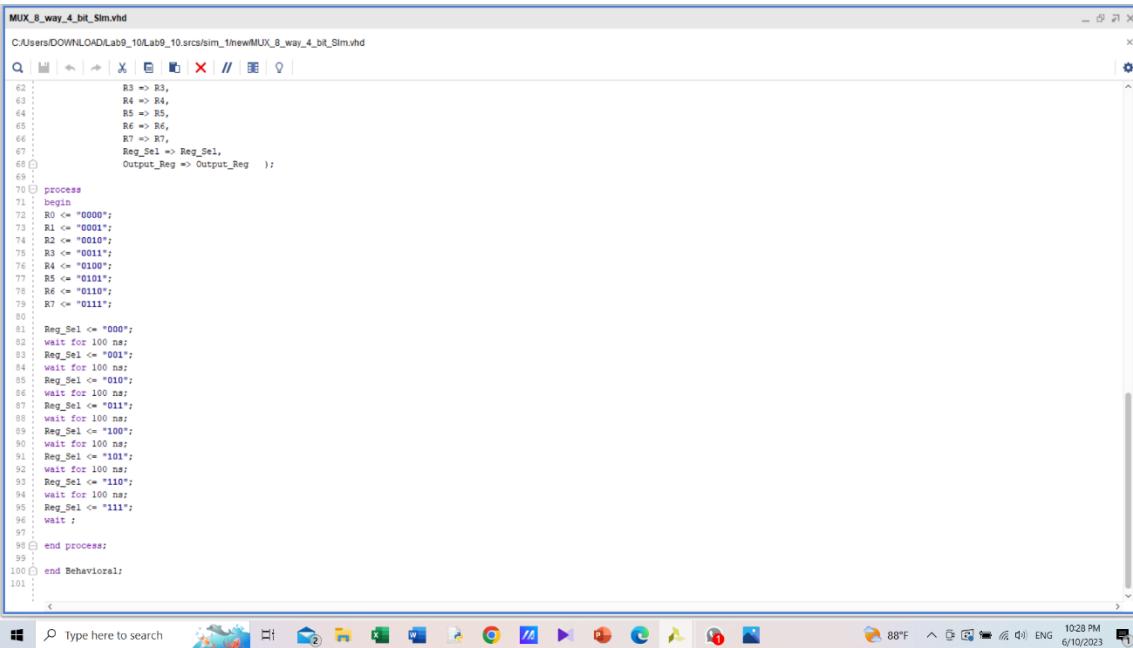


```
MUX_8_way_4_bit_Sim.vhd
C:/Users/Downloads/Lab9_10/Lab9_10/srcs/sim_1/newMUX_8_way_4_bit_Sim.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
library UNISIM;
use UNISIM.VComponents.all;
entity MUX_8_way_4_bit_Sim is
    Port ( );
end MUX_8_way_4_bit_Sim;

architecture Behavioral of MUX_8_way_4_bit_Sim is
component MUX_1way_4_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD_LOGIC_VECTOR (3 downto 0);
           R2 : in STD_LOGIC_VECTOR (3 downto 0);
           R3 : in STD.LOGIC_VECTOR (3 downto 0);
           R4 : in STD.LOGIC_VECTOR (3 downto 0);
           R5 : in STD.LOGIC_VECTOR (3 downto 0);
           R6 : in STD.LOGIC_VECTOR (3 downto 0);
           R7 : in STD.LOGIC_VECTOR (3 downto 0);
           Req_Sel : in STD.LOGIC_VECTOR (2 downto 0);
           output_reg : out STD.LOGIC_VECTOR (3 downto 0));
end component;
signal R0,R1,R2,R3,R4,R5,R6,R7,output_Reg :std_logic_vector(3 downto 0);
signal Req_Sel : std_logic_vector(2 downto 0);
begin
    UUT : MUX_8_way_4_bit
        port map ( R0 => R0,
                   R1 => R1,
                   R2 => R2,
                   R3 => R3,
                   R4 => R4,
                   R5 => R5,
                   R6 => R6,
                   R7 => R7,
                   Req_Sel => Req_Sel,
                   Output_Reg => Output_Reg );
end;

```



```
MUX_8_way_4_bit_Sim.vhd
C:/Users/Downloads/Lab9_10/Lab9_10/srcs/sim_1/newMUX_8_way_4_bit_Sim.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
library UNISIM;
use UNISIM.VComponents.all;
entity MUX_8_way_4_bit_Sim is
    Port ( );
end MUX_8_way_4_bit_Sim;

architecture Behavioral of MUX_8_way_4_bit_Sim is
component MUX_1way_4_bit is
    Port ( R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD.LOGIC_VECTOR (3 downto 0);
           R2 : in STD.LOGIC_VECTOR (3 downto 0);
           R3 : in STD.LOGIC_VECTOR (3 downto 0);
           R4 : in STD.LOGIC_VECTOR (3 downto 0);
           R5 : in STD.LOGIC_VECTOR (3 downto 0);
           R6 : in STD.LOGIC_VECTOR (3 downto 0);
           R7 : in STD.LOGIC_VECTOR (3 downto 0);
           Req_Sel : in STD.LOGIC_VECTOR (2 downto 0);
           output_Reg : out STD.LOGIC_VECTOR (3 downto 0));
end component;
signal R0,R1,R2,R3,R4,R5,R6,R7,output_Reg :std_logic_vector(3 downto 0);
signal Req_Sel : std_logic_vector(2 downto 0);
begin
    UUT : MUX_8_way_4_bit
        port map ( R0 => R0,
                   R1 => R1,
                   R2 => R2,
                   R3 => R3,
                   R4 => R4,
                   R5 => R5,
                   R6 => R6,
                   R7 => R7,
                   Req_Sel => Req_Sel,
                   Output_Reg => Output_Reg );
end;
process
begin
    R0 <= "0000";
    R1 <= "0001";
    R2 <= "0010";
    R3 <= "0011";
    R4 <= "0100";
    R5 <= "0101";
    R6 <= "0110";
    R7 <= "0111";
    Req_Sel <= "000";
    wait for 10 ns;
    Req_Sel <= "001";
    wait for 100 ns;
    Req_Sel <= "010";
    wait for 10 ns;
    Req_Sel <= "011";
    wait for 10 ns;
    Req_Sel <= "100";
    wait for 100 ns;
    Req_Sel <= "101";
    wait for 100 ns;
    Req_Sel <= "110";
    wait for 100 ns;
    Req_Sel <= "111";
    wait ;
end process;
end Behavioral;

```

## Register bank

```
Register_Bank_Sim.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10.srcs\sim_1\new\Register_Bank_Sim.vhd

11 | 
12 | 
13 | 
14 | 
15 | 
16 | 
17 | 
18 | 
19 | 
20 | 
21 | 
22 | 
23 | 
24 | 
25 | 
26 | 
27 | 
28 | 
29 | 
30 | 
31 | 
32 | 
33 | 
34 | entity Register_Bank_Sim is
35 |   -- Port ()
36 | end Register_Bank_Sim;
37 |
38 | architecture Behavioral of Register_Bank_Sim is
39 | 
40 | component Register_Bank is
41 |   Port ( Data : in STD_LOGIC_VECTOR (3 downto 0);
42 |          Clk : in STD_LOGIC;
43 |          Reset : in STD_LOGIC;
44 |          Register_enable : in STD_LOGIC_VECTOR (2 downto 0);
45 |          R0 : out STD_LOGIC_VECTOR (3 downto 0);
46 |          R1 : out STD_LOGIC_VECTOR (3 downto 0);
47 |          R2 : out STD_LOGIC_VECTOR (3 downto 0);
48 |          R3 : out STD_LOGIC_VECTOR (3 downto 0);
49 |          R4 : out STD_LOGIC_VECTOR (3 downto 0);
50 |          R5 : out STD_LOGIC_VECTOR (3 downto 0);
51 |          R6 : out STD_LOGIC_VECTOR (3 downto 0);
52 |          R7 : out STD_LOGIC_VECTOR (3 downto 0));
53 | end component;
54 |
55 | signal Data,R0,R1,R2,R3,R4,R5,R6,R7 : std_logic_vector (3 downto 0);
56 | signal Register_Enable : std_logic_vector (2 downto 0);
57 | signal Clk,Reset : std_logic;
58 |
59 | begin
60 |
61 |   UUT : Register_Bank
62 | 
63 | 
64 | 
65 | 
66 | 
67 | 
68 | 
69 | 
70 | 
71 | 
72 | 
73 | 
74 | 
75 | 
76 | 
77 | 
78 | 
79 | 
80 | 
81 | 
82 | 
83 | 
84 | 
85 | 
86 | 
87 | 
88 | 
89 | 
90 | 
91 | 
92 | 
93 | 
94 | 
95 | 
96 | 
97 | 
```

```
Register_Bank_Sim.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10.srcs\sim_1\new\Register_Bank_Sim.vhd

57 | 
58 | 
59 | 
60 | 
61 | 
62 | 
63 | 
64 | 
65 | 
66 | 
67 | 
68 | 
69 | 
70 | 
71 | 
72 | 
73 | 
74 | 
75 | 
76 | 
77 | 
78 | 
79 | 
80 | 
81 | 
82 | 
83 | 
84 | 
85 | 
86 | 
87 | 
88 | 
89 | 
90 | 
91 | 
92 | 
93 | 
94 | 
95 | 
96 | 
97 | 
```

Register\_Bank\_Sim.vhd

```

C:/Users/DOWNLOAD/Lab9_10/Lab9_10/srcs/sim_1/newRegister_Bank_Sim.vhd

87 : Reset <= '0';
88 : Register_enable <= "000";
89 : wait for 50ns;
90 : Register_enable <= "001";
91 : wait for 50ns;
92 : Register_enable <= "010";
93 : wait for 50ns;
94 : Register_enable <= "011";
95 : wait for 50ns;
96 : Register_enable <= "100";
97 : wait for 50ns;
98 : Register_enable <= "101";
99 : wait for 50ns;
100 : Register_enable <= "110";
101 : wait for 50ns;
102 : Register_enable <= "111";
103 : wait for 50ns;
104 :
105 : Reset <= '1';
106 : Register_enable <= "000";
107 : wait for 50ns;
108 : Register_enable <= "001";
109 : wait for 50ns;
110 : Register_enable <= "010";
111 : wait for 50ns;
112 : Register_enable <= "011";
113 : wait for 50ns;
114 : Register_enable <= "100";
115 : wait for 50ns;
116 : Register_enable <= "101";
117 : wait for 50ns;
118 : Register_enable <= "110";
119 : wait for 50ns;
120 : Register_enable <= "111";
121 : wait for 50ns;
122 :
123 end process;
124
125 end Behavioral;
126

```

## Program ROM

Program\_Rom\_Sim.vhd

```

C:/Users/DOWNLOAD/Lab9_10/Lab9_10/srcs/sim_1/newProgram_Rom_Sim.vhd

38 architecture Behavioral of Program_Rom_Sim is
39 :
40 component Program_Rom is
41     Port ( Memory_Select : in STD_LOGIC_VECTOR ( 2 downto 0 );
42             Instruction : out STD_LOGIC_VECTOR ( 11 downto 0 );
43 end component;
44 :
45 signal Memory_Select : std_logic_vector ( 2 downto 0 );
46 signal Instruction : std_logic_vector ( 11 downto 0 );
47 :
48 begin
49 :
50 UUT : Program_Rom
51     port map ( Memory_Select => Memory_Select,
52                 Instruction => Instruction );
53 :
54 process
55 begin
56 :
57     Memory_Select <= "000";
58     wait for 100 ns;
59     Memory_Select <= "001";
60     wait for 100 ns;
61     Memory_Select <= "010";
62     wait for 100 ns;
63     Memory_Select <= "011";
64     wait for 100 ns;
65     Memory_Select <= "100";
66     wait for 100 ns;
67     Memory_Select <= "101";
68     wait for 100 ns;
69     Memory_Select <= "110";
70     wait for 100 ns;
71     Memory_Select <= "111";
72     wait;
73 :
74 end process;
75 :
76 end Behavioral;
77

```

## Instruction decoder

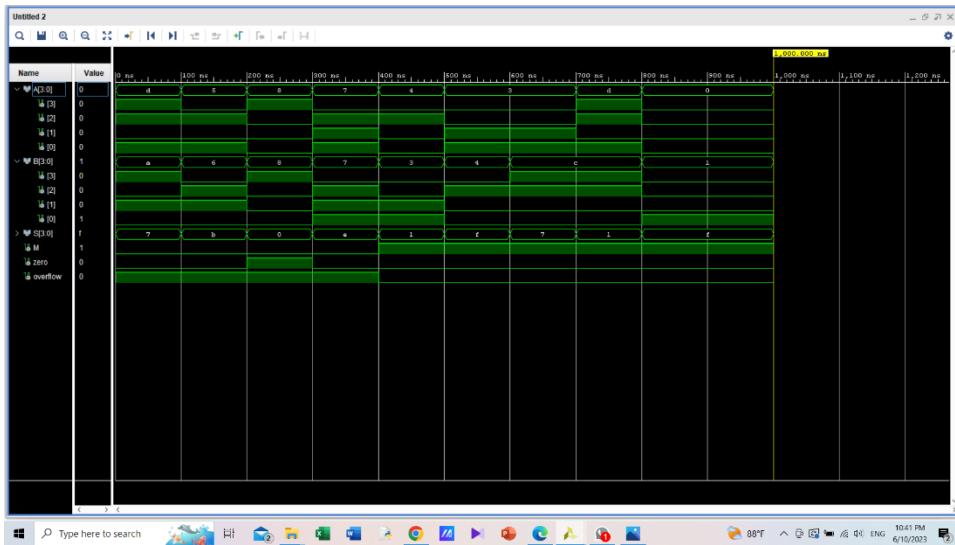
instruction\_Decoder\_Sim.vhd

C:/Users/DOWNLOAD/Lab9\_10/Lab9\_10/srcs/sim\_inthew/Instruction\_Decoder\_Sim.vhd

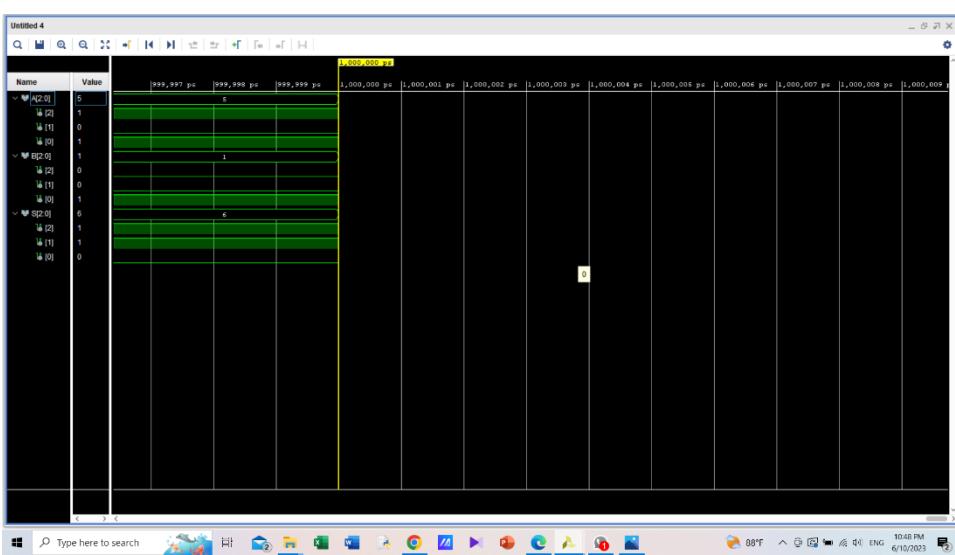
```
57 begin
58
59
60 IUT : Instruction_Decoder
61 port map ( Instruction => Instruction,
62             Reg_Check_Jump => Reg_Check_Jump,
63             Reg_Enable => Reg_Enable,
64             Load_Select => Load_Select,
65             Instruction => Instruction,
66             Reg_Select_A => Reg_Select_A,
67             Reg_Select_B => Reg_Select_B,
68             Add_Sub_Select => Add_Sub_Select,
69             Jump_Flag => Jump_Flag,
70             Jump_Address => Jump_Address );
71
72 begin
73
74
75     Reg_Check_Jump <= "0001";
76     Instruction <= "100010000011";
77     wait for 100 ns;
78     Instruction <= "100100000001";
79     wait for 100 ns;
80     Instruction <= "010100000000";
81     wait for 100 ns;
82     Instruction <= "0001100010000";
83     wait for 100 ns;
84     Instruction <= "000010100000";
85     wait for 100 ns;
86     Instruction <= "110010000111";
87     wait for 100 ns;
88     Instruction <= "110000000011";
89     wait for 100 ns;
90     Instruction <= "110000000111";
91     wait;
92
93 end process;
94
95 end Behavioral;
```

# Timing Diagrams

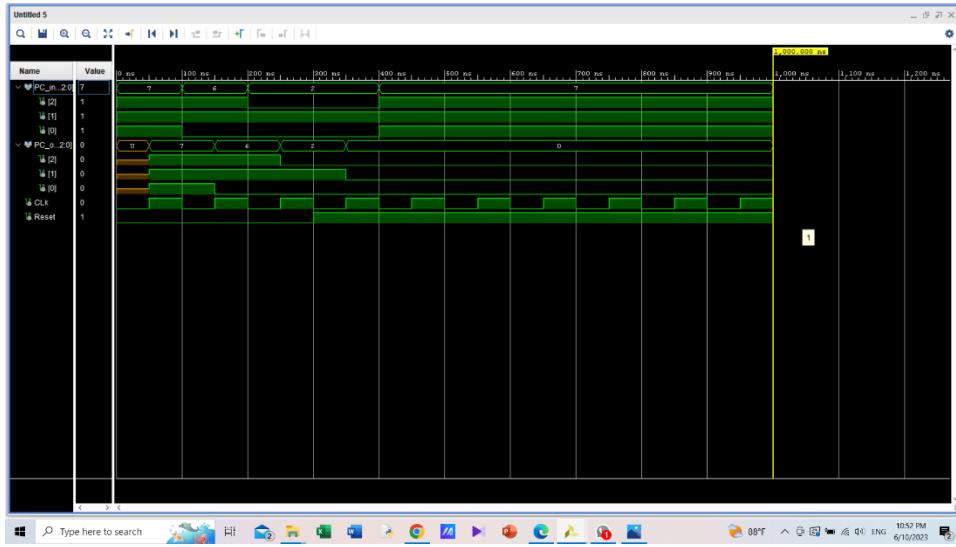
Add/sub unit



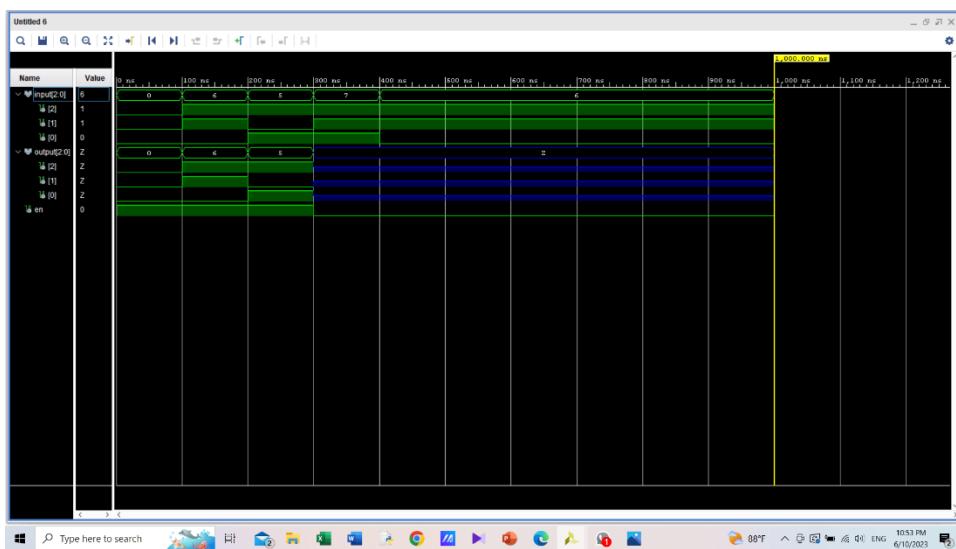
3-bit Adder



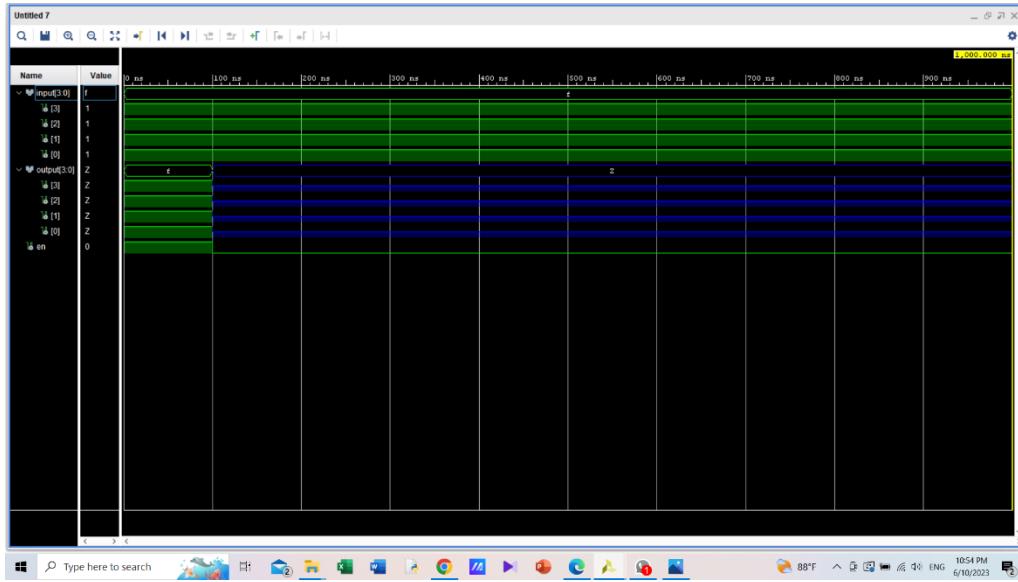
### 3-bit Program Counter



### Tri state 3-bit buffer

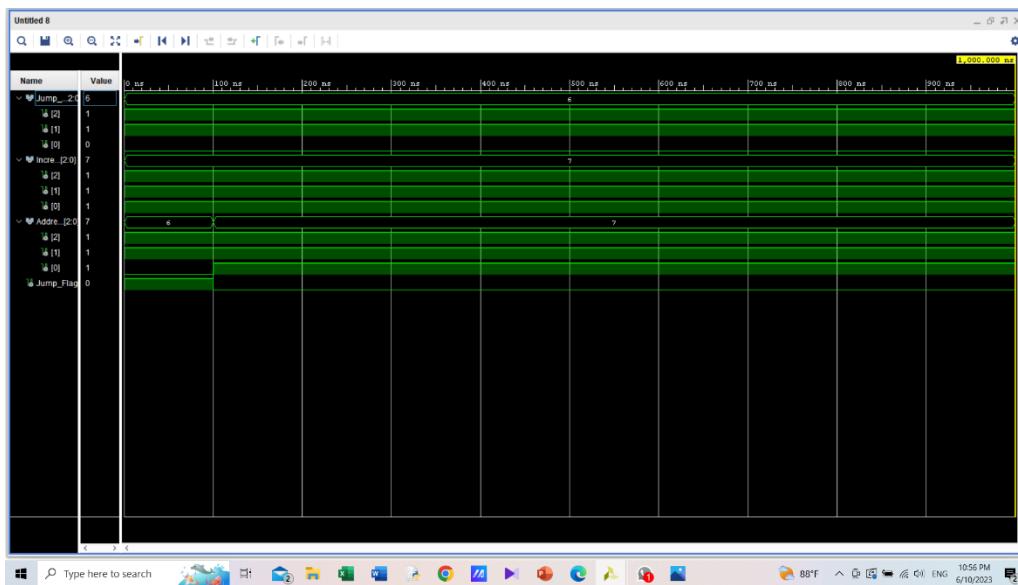


## Tri state 4-bit buffer

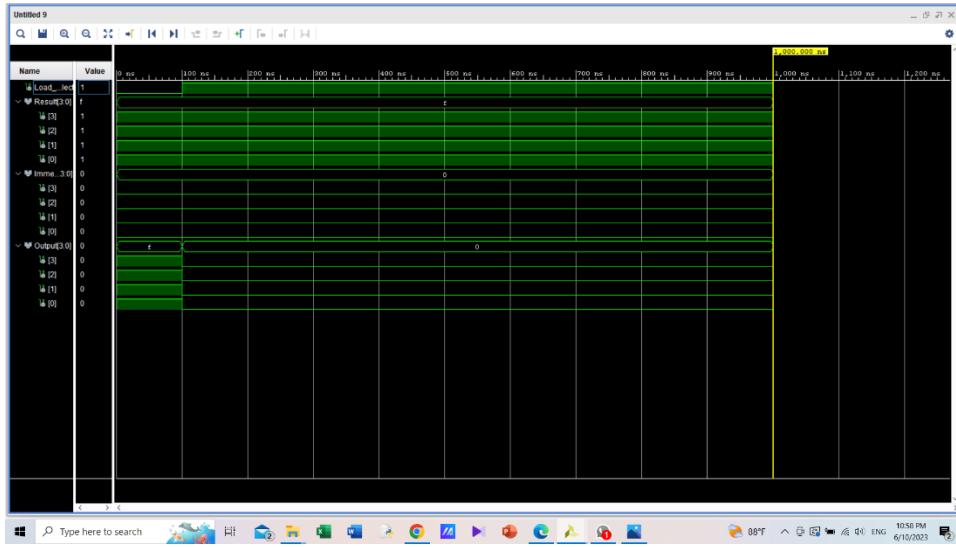


## Multiplexers

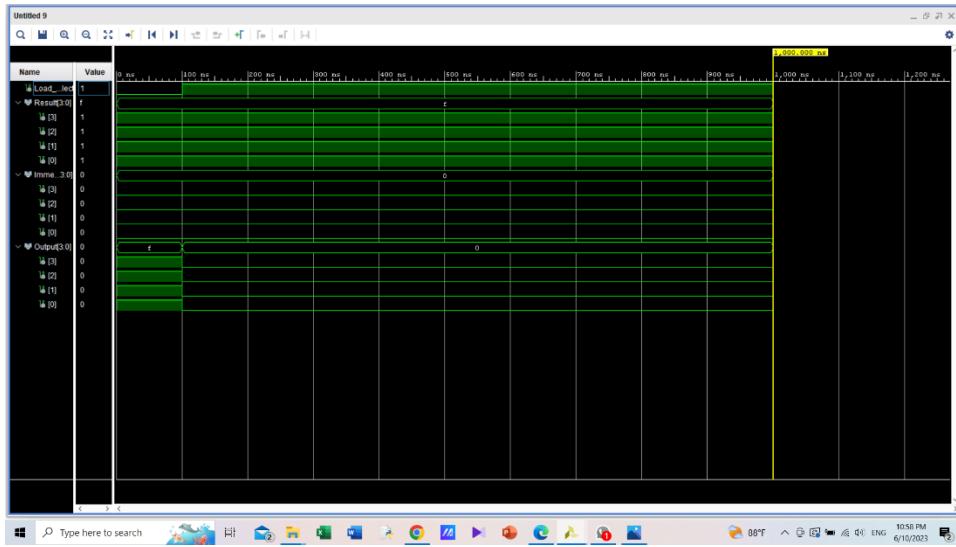
### 2 way 3-bit



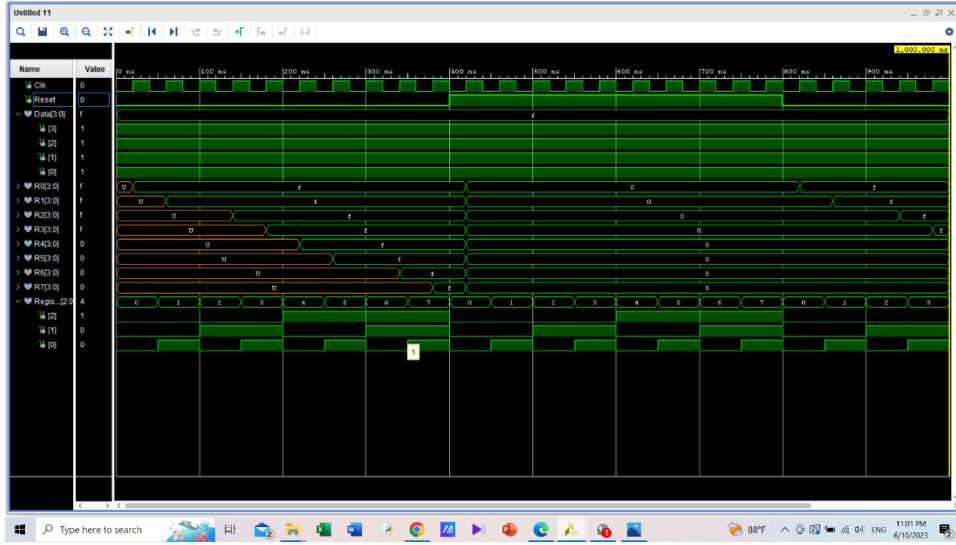
## 2 way 4-bit



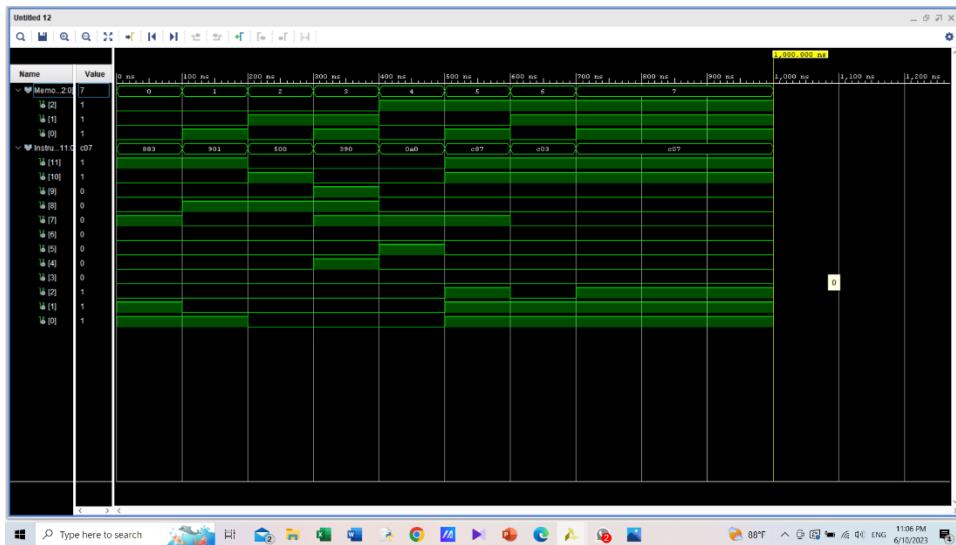
## 8 way 4-bit



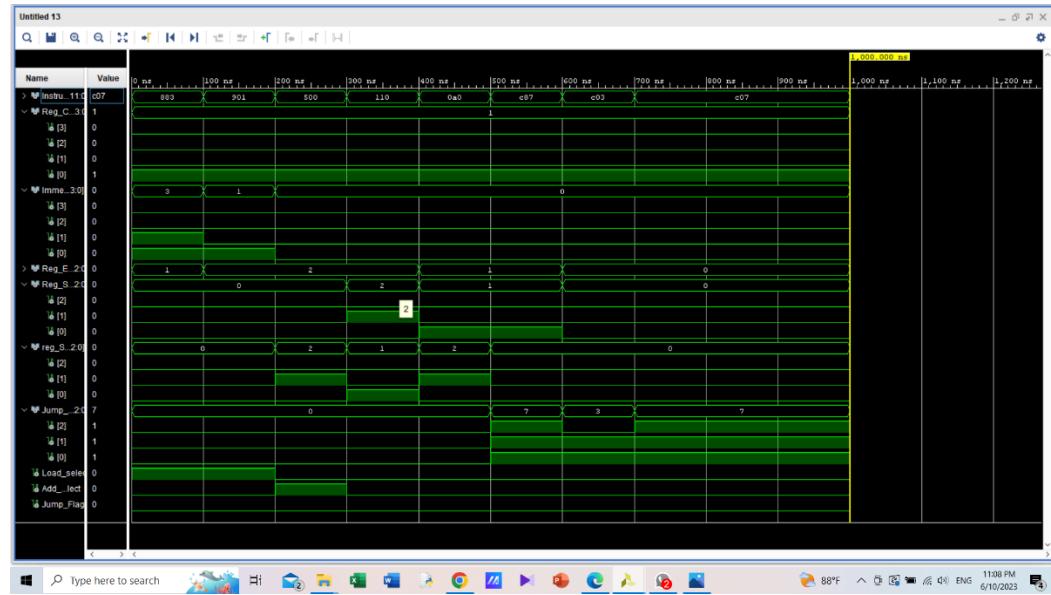
## Register bank



## Program ROM



## Instruction decoder



# Nano – Processor

# VHDL

```
Nanoprocessor.vhd
C:\Users\DOWNLOAD\Lab09_10\Lab09_10\src\sources_1\newNanoprocessor.vhd

--> End of component;
59
60 component MN_2_Way_2_bit is
61    Port (Jump_Address : in STD_LOGIC_VECTOR (2 downto 0);
62          Increment_Address : in STD_LOGIC_VECTOR (2 downto 0);
63          Jump_Flag : in STD_LOGIC;
64          Address_Out : out STD_LOGIC_VECTOR (2 downto 0));
65 end component;
66
67 component MN_2_Way_4_bit is
68    Port (Load_Select : in STD_LOGIC;
69          Result : out STD_LOGIC_VECTOR (3 downto 0);
70          Immediate_Val : in STD_LOGIC_VECTOR (3 downto 0);
71          Output : out STD_LOGIC_VECTOR (3 downto 0));
72 end component;
73
74 component Register_Bank is
75    Port (Data : in STD_LOGIC_VECTOR (3 downto 0);
76          CLA : in STD_LOGIC;
77          Reset : in STD_LOGIC;
78          Register : out STD_LOGIC_VECTOR (2 downto 0);
79          R0 : out STD_LOGIC_VECTOR (3 downto 0);
80          R1 : out STD_LOGIC_VECTOR (3 downto 0);
81          R2 : out STD_LOGIC_VECTOR (3 downto 0);
82          R3 : out STD_LOGIC_VECTOR (3 downto 0);
83          R4 : out STD_LOGIC_VECTOR (3 downto 0);
84          R5 : out STD_LOGIC_VECTOR (3 downto 0);
85          R6 : out STD_LOGIC_VECTOR (3 downto 0);
86          RT : out STD_LOGIC_VECTOR (3 downto 0));
87 end component;
88
89 component Add_Sub_Unit is
90    Port (A : in STD_LOGIC_VECTOR(3 downto 0);
91          B : in STD_LOGIC_VECTOR(3 downto 0);
92          S : out STD_LOGIC_VECTOR(3 downto 0);
93          M : in STD_LOGIC; --(M=0) for addition and (M=1) for subtraction
94          Overflow : out STD_LOGIC;
95          Zero : out STD_LOGIC);
96 end component;
97
98 component PC_3_bit is
99

```

```
Nanoprocessor.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10\src\sources_1\newNanoprocessor.vhd

97 component PC_3_bit is
98   Port ( PC_input : in STD_LOGIC_VECTOR (2 downto 0);
99     Clk : in STD_LOGIC;
100    Mux : out STD_LOGIC;
101   PC_output : out STD_LOGIC_VECTOR (2 downto 0));
102 end component;
103
104 component ROM_2_16is is
105   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
106     B : in STD_LOGIC_VECTOR (2 downto 0);
107     S : out STD_LOGIC_VECTOR (2 downto 0));
108 end component;
109
110 component Instruction_Decoder is
111   Port ( PC_in : in STD_LOGIC_VECTOR (11 downto 0);
112     Reg_Enable : out STD_LOGIC_VECTOR (3 downto 0);
113     Reg_Enable_b : out STD_LOGIC_VECTOR (3 downto 0);
114     Reg_Enable_o : out STD_LOGIC_VECTOR (2 downto 0);
115     Load_Select : out STD_LOGIC;
116     Immense : out STD_LOGIC_VECTOR (3 downto 0);
117     Reg_Select_A : out STD_LOGIC_VECTOR (2 downto 0);
118     Reg_Select_B : out STD_LOGIC_VECTOR (2 downto 0);
119     Add_Reg_Select : out STD_LOGIC;
120     Jump_Flag : out STD_LOGIC;
121     Jump_Address : out STD_LOGIC_VECTOR (2 downto 0));
122 end component;
123
124 component Seven_seg_Display is
125   Port ( RT_data : in STD_LOGIC_VECTOR (3 downto 0);
126     seven_seg_code : out STD_LOGIC_VECTOR (6 downto 0);
127     negative : out STD_LOGIC);
128 end component;
129
130 component Program_Rom is
131   Port ( Memory_Select : in STD_LOGIC_VECTOR (2 downto 0);
132     Address : out STD_LOGIC_VECTOR (11 downto 0));
133 end component;
134
135 component Edge_Clk is
136   Port ( Clk_in : in STD_LOGIC;
137     Clk_out : out STD_LOGIC);
138 end component;

```

Nanoprocessor.vhd

C:\Users\DOWNLOAD\Lab09\_10\Lab09\_10\sources\1newNanoprocessor.vhd

```
169 :      A => Mem_Select,
170 :      B => "001",
171 :      S => Increment_Address );
172 :
173 @ MX2_2_way_Boot_0 : MX2_2_way_1_bit
174 @    port map (
175 @      jump_Address => jump_Address,
176 @      Increment_Address => Increment_Address ,
177 @      Jump_flag => jump_flag,
178 @      Address_Out => Next_Address  );
179 :
180 @ MX2_2_way_Boot_0 : MX2_2_way_1_bit
181 @    port map (
182 @      Load_Select => Load_Select,
183 @      Result => Result,
184 @      Immediate_Val =>Immediate_val,
185 @      Output => MX2_2_way_4_bit_out );
186 :
187 @ MX2_8_to_1_A: MX2_8_way_4_bit
188 @    port map (
189 @      R0 => R0,
190 @      R1 => R1,
191 @      R2 => R2,
192 @      R3 => R3,
193 @      R4 => R4,
194 @      R5 => R5,
195 @      R6 => R6,
196 @      R7 => R7,
197 @      Reg_Sel => Reg_Select_b,
198 @      output_Reg => MX2_8_to_1_A_Out);
199 :
200 @ MX2_8_to_1_B: MX2_8_way_4_bit
201 @    port map (
202 @      R0 => R0,
203 @      R1 => R1,
204 @      R2 => R2,
205 @      R3 => R3,
206 @      R4 => R4,
207 @      R5 => R5,
208 @      R6 => R6,
209 @      R7 => R7,
```

```
Nanoprocessor.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10\src\sources_1\threeNanoprocessor.vhd
199
200  HDX_8_to_1_B: HDX_8_way_4_bit
201      port map (
202          A0 => R0,
203          R1 => R1,
204          R2 => R2,
205          R3 => R3,
206          R4 => R4,
207          R5 => R5,
208          R6 => R6,
209          R7 => R7,
210          R8 => R8,
211          Reg_Select_B,
212          output_reg => HDX_8_to_1_B_Out);
213
214  PC_3_Bits_0 : PC_3_bits
215      port map (
216          PC_Inp => Next_Address,
217          Reset => Reset,
218          Clk => Clk_out,
219          PC_Output => Mem_Select;
220
221  Program_ROM_0 : Program_ROM
222      port map (
223          Address => Mem_Select,
224          Instruction => Instruction_Bus);
225
226  RegBank_0 : Register_Bank
227      port map (
228          Reset => Reset,
229          Register_Enable => Register_Enable,
230          Clk => Clk_out,
231          Data => HDX_3_Way_4_bit_Out,
232          R0 => R0,
233          R1 => R1,
234          R2 => R2,
235          R3 => R3,
236          R4 => R4,
237          R5 => R5,
238          R6 => R6,
239          R7 => R7);
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
```

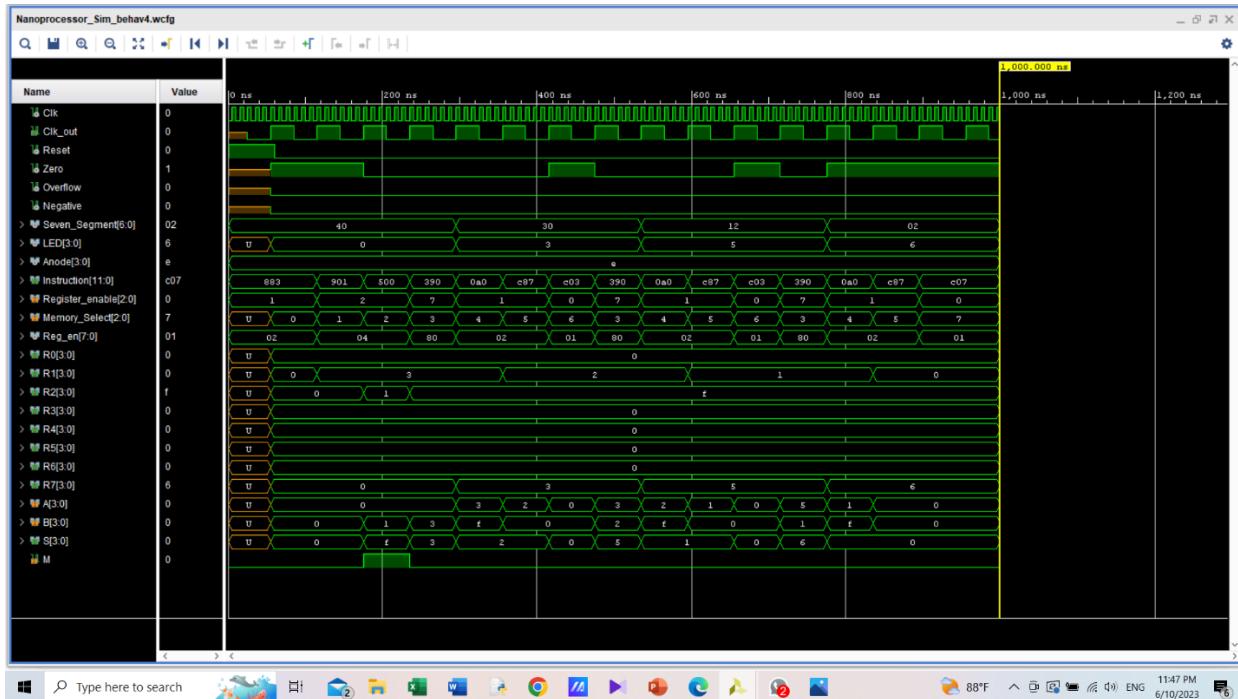
```
Nanoprocessor.vhd
C:\Users\DOWNLOAD\Lab9_10\Lab9_10\src\sources_1\threeNanoprocessor.vhd
199
200  HDX_8_to_1_B: HDX_8_way_4_bit
201      port map (
202          A0 => R0,
203          R1 => R1,
204          R2 => R2,
205          R3 => R3,
206          R4 => R4,
207          R5 => R5,
208          R6 => R6,
209          R7 => R7,
210          Reg_Select_B,
211          output_reg => HDX_8_to_1_B_Out);
212
213  PC_3_Bits_0 : PC_3_bits
214      port map (
215          PC_Inp => Next_Address,
216          Reset => Reset,
217          Clk => Clk_out,
218          PC_Output => Mem_Select;
219
220  Program_ROM_0 : Program_ROM
221      port map (
222          Address => Mem_Select,
223          Instruction => Instruction_Bus);
224
225  RegBank_0 : Register_Bank
226      port map (
227          Reset => Reset,
228          Register_Enable => Register_Enable,
229          Clk => Clk_out,
230          Data => HDX_3_Way_4_bit_Out,
231          R0 => R0,
232          R1 => R1,
233          R2 => R2,
234          R3 => R3,
235          R4 => R4,
236          R5 => R5,
237          R6 => R6,
238          R7 => R7);
239
240  Add_Sub_Unit_0 : Add_Sub_Unit
241      port map (
242          A =>HDX_8_to_1_A_Out,
243          B =>HDX_8_to_1_B_Out,
244          S =>HDX_8_to_1_S_Out,
245          M =>HDX_8_to_1_M_Out,
246          Overflow =>Overflow,
247          Zero =>Zero );
248
249  Seven_Seg_Display_0 : Seven_Seg_Display
250      port map (
251          R7_data => R7,
252          seven_seg_code =>seven_seg_code,
253          negative => Negative);
254
255  Anode <= "1110"; --selecting the rightmost seven segment
256  LED <= R7;
257  Seven_Segment <= seven_seg_code;
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
309
```

# Nano-processor test bench

```
33 :  
34     entity Nanoprocessor_Sim is  
35     -- Port (-);  
36     end Nanoprocessor_Sim;  
37 :  
38     architecture Behavioral of Nanoprocessor_Sim is  
39 :  
40     component Nanoprocessor is  
41         Port ( Clk : in STD_LOGIC;  
42             Reset : in STD_LOGIC;  
43             Zero : out STD_LOGIC;  
44             Overflow : out STD_LOGIC;  
45             Negative : out STD_LOGIC;  
46             Seven_Segment : out STD_LOGIC_VECTOR (6 downto 0);  
47             LED : out STD_LOGIC_VECTOR (3 downto 0);  
48             Anode : out STD_LOGIC_VECTOR (3 downto 0);  
49     end component;  
50 :  
51     signal Clk, Reset, Zero, Overflow, Negative :std_logic;  
52     signal Seven_Segment :std_logic_vector ( 6 downto 0);  
53     signal LED, Anode : std_logic_vector ( 3 downto 0 );  
54 :  
55     begin  
56 :  
57         UUT: nanoprocessor  
58             port map (  
59                 Clk => Clk,  
60                 Reset => Reset,  
61                 Zero => Zero,  
62                 Overflow => Overflow,  
63                 Negative => Negative,  
64                 Seven_Segment => Seven_Segment,  
65                 LED => LED,  
66                 Anode => Anode);  
67 :  
68         process  
69         begin  
70             Clk <= '0';  
71             wait for 5ns;  
72             Clk <= '1';  
73         end process;  
74 :  
75         process  
76         begin  
77             Clk <= '0';  
78             wait for 5ns;  
79             Clk <= '1';  
80             wait for 5ns;  
81         end process;  
82 :  
83         process  
84         begin  
85             Reset <= '1';  
86             wait for 60ns;  
87             Reset <= '0';  
88             wait;  
89         end process;  
90 :  
91     end Behavioral;  
92 :  
93 :<
```

```
48 :  
49     Anode : out STD_LOGIC_VECTOR (3 downto 0);  
50 :  
51     end component;  
52 :  
53     signal Clk, Reset, Zero, Overflow, Negative :std_logic;  
54     signal Seven_Segment :std_logic_vector ( 6 downto 0);  
55     signal LED, Anode : std_logic_vector ( 3 downto 0 );  
56 :  
57     begin  
58 :  
59         UUT: nanoprocessor  
60             port map (  
61                 Clk => Clk,  
62                 Reset => Reset,  
63                 Zero => Zero,  
64                 Overflow => Overflow,  
65                 Negative => Negative,  
66                 Seven_Segment => Seven_Segment,  
67                 LED => LED,  
68                 Anode => Anode);  
69 :  
70         process  
71         begin  
72             Clk <= '0';  
73             wait for 5ns;  
74             Clk <= '1';  
75             wait for 5ns;  
76         end process;  
77 :  
78         process  
79         begin  
80             Reset <= '1';  
81             wait for 60ns;  
82             Reset <= '0';  
83             wait;  
84         end process;  
85 :  
86         process  
87         begin  
88             Reset <= '1';  
89             wait for 60ns;  
90             Reset <= '0';  
91             wait;  
92         end process;  
93 :  
94     end Behavioral;  
95 :<
```

# Timing Diagram



## **Constraints :-**

## **Conclusions :-**

- We can simulate the nano-processor before implementing it on the hardware to check for logical errors in our codes. So, we don't have to use hardware components for testing purposes.
  - We can approach a bigger problem by dividing it into smaller sub problems.
  - There can be differences between our simulation and how the actual circuit behaves in hardware.
  - The slow clock must be adjusted to a suitable value for visualizing the output in a realistic way.
  - This simulation process allows for optimization and improvement of the design, leading to a more efficient and reliable nano-processor.
  - Working as a team will not only allow for the sharing of responsibilities and workload but also foster effective communication and coordination among team members.
  - Each team member can bring their unique skills and perspectives to the project, leading to a more comprehensive and well-rounded nano-processor design.

## **Contributions :-**

Kulasekara K M S N

Designed :-

- Instruction decoder.
  - Program ROM
  - 7 segment display
  - 3-bit adder
  - Program counter

20 hours

Kulathunga K A J T

Designed :-

- Combination of the components.
  - Tri state buffers and Multiplexers.
  - Add-Sub Unit
  - Register bank