Lab 09-10 - Nano processor – CS1050

K.C.K. Manawathilake - 210372D
S.M.A.N.A. Manchanayake - 210373G

Group 21

Contents

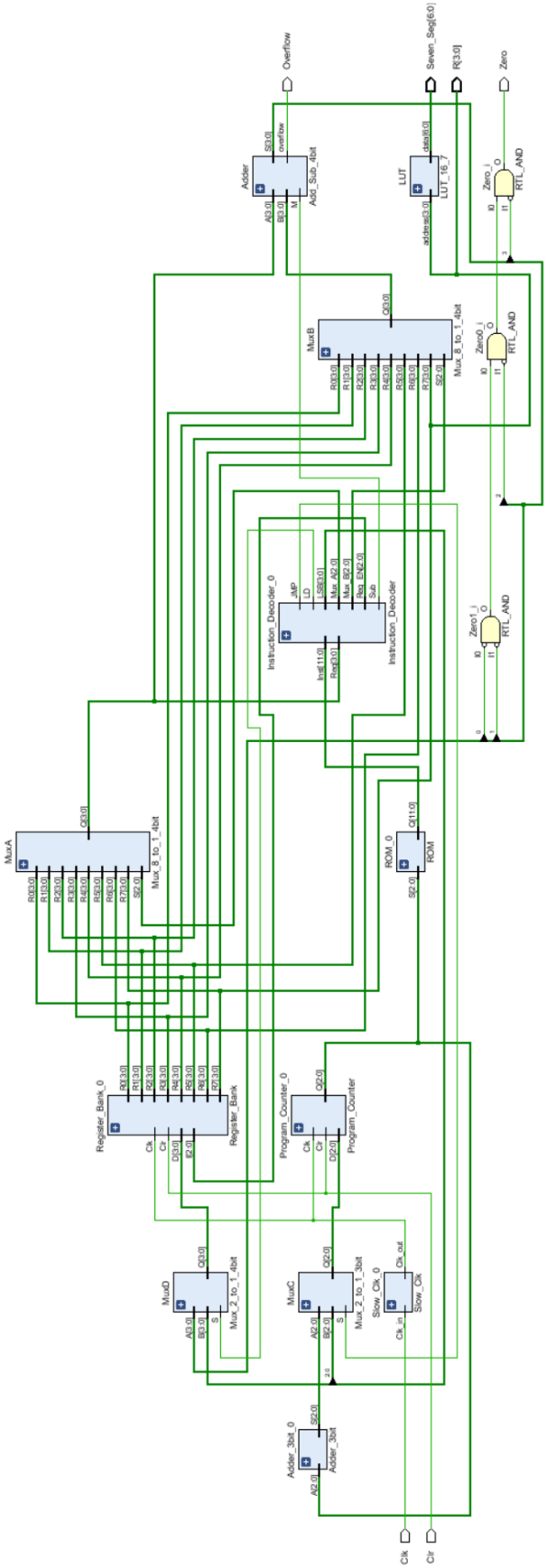
Lab Task:

The task was to design and develop a microprocessor consisting of the components from the previous labs, as well as some new components such as a Register Bank, 4-bit adder/subtractor and Instruction Decoder, using Xilinx Vivado. Then, simulate the components and the processor for a set of given instructions, and finally, implement it on the Basys 3 board after generating the bitstream.

Assembly code and Machine code:

Assembly Code	Machine code
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

Nano processor design



VHDL Codes

4-bit Add/Subtract unit

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

entity Add_Sub_4bit is
   Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          S : out STD_LOGIC_VECTOR (3 downto 0);
          M : in STD_LOGIC;
          overflow : out STD_LOGIC);
end Add_Sub_4bit;

architecture Behavioral of Add_Sub_4bit is
component FA
    port (
       A: in std_logic;
       B: in std_logic;
       C_in: in std_logic;
       S: out std_logic;
       C_out: out std_logic);
 end component;

 SIGNAL FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C,
FA3_S, FA3_C, C_out : std_logic;
 Signal B0x , B1x , B2x , B3x : std_logic;
begin
B0x <= B(0) XOR M;
B1x <= B(1) XOR M;
B2x <= B(2) XOR M;
B3x <= B(3) XOR M;

FA_0 : FA
 port map (
    A => A(0),
    B => B0x,
    C_in => M,
    S => S(0),
    C_Out => FA0_C);

FA_1 : FA
 port map (
    A => A(1),
    B => B1x,
    C_in => FA0_C,
    S => S(1),
    C_Out => FA1_C);

FA_2 : FA
 port map (
    A => A(2),
    B => B2x,
    C_in => FA1_C,
```

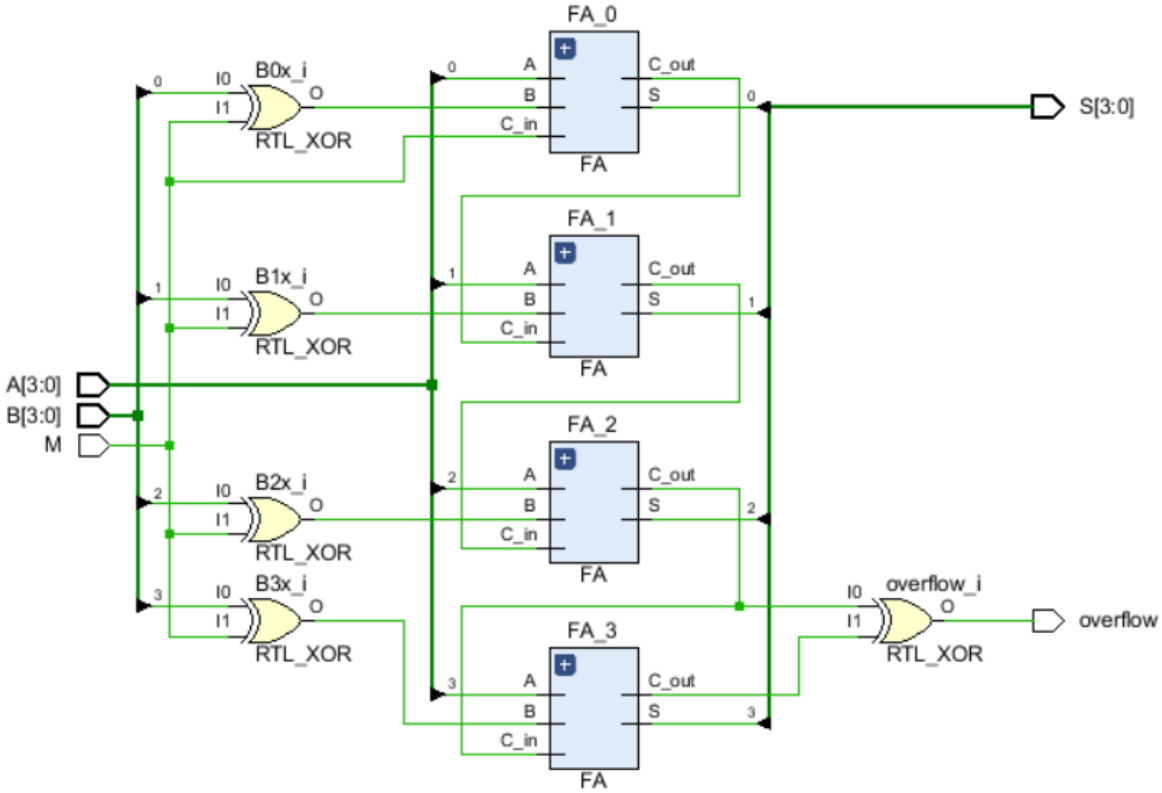
FA

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

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

architecture Behavioral of FA is
   component HA
      port (
         A: in std_logic;
         B: in std_logic;
         S: out std_logic;
         C: out std_logic);
   end component;
   SIGNAL HA0_S, HA0_C, HA1_S, HA1_C : std_logic;

begin
   HA_0 : HA
      port map (
         A => A,
         B => B,
         S => HA0_S,
         C => HA0_C);
   HA_1 : HA
      port map (
         A => HA0_S,
         B => C_in,
         S => HA1_S,
         C => HA1_C);

   S <= HA1_S;
   C_out <= HA0_C OR HA1_C;
end Behavioral;
```

```
    S => S(2),
    C_Out => FA2_C);

FA_3 : FA
 port map (
    A => A(3),
    B => B3x,
    C_in => FA2_C,
    S => S(3),
    C_Out => C_out
     );
overflow <= FA2_C xor C_out;

end Behavioral;
```

HA

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

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

architecture Behavioral of HA is

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

end Behavioral;
```

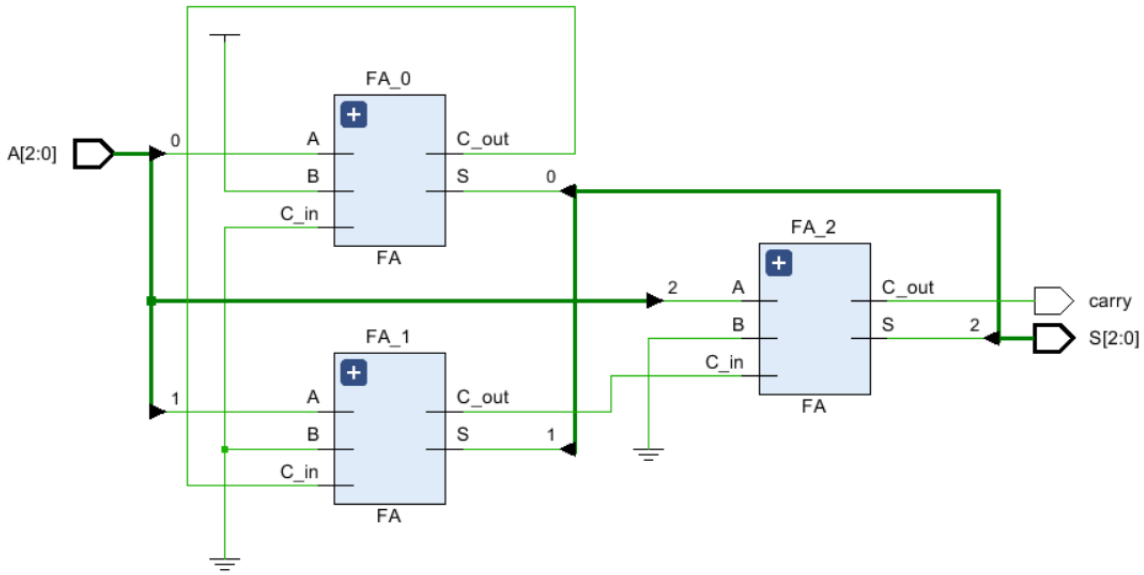


4-bit Add/Subtract unit simulation

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

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

architecture Behavioral of TB_Add_Sub_4bit is
COMPONENT Add_Sub_4bit
    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;
           overflow : out STD_LOGIC );
END COMPONENT;


SIGNAL A,B,S : std_logic_vector (3 downto 0);
SIGNAL M, overflow : std_logic;

begin
UUT: Add_Sub_4bit PORT MAP(
   M => M,
   A => A,
   B=>B,
   S => S,
   overflow => overflow
   );

process
 begin
    A <= "0101";
    B <= "1100";
    M <= '0';

 WAIT FOR 100 ns; -- after 100 ns change inputs
    A <="0010";
    B <= "1110";
    M <= '1';
 WAIT FOR 100 ns; --change again
    A <="1010";
    B <= "1010";
    M <= '1';
 WAIT FOR 100 ns; --change again
    A <= "1110";
    B <= "0001";
    M <= '1';
 WAIT FOR 100 ns;
    A <= "0110";
    B <= "1010";
    M <= '1';
 WAIT; -- will wait forever
end process;
end Behavioral;
```

3-bit adder

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

entity Adder_3bit is
   Port (  A : in STD_LOGIC_VECTOR (2 downto 0);
           S : out STD_LOGIC_VECTOR(2 downto 0);
           carry : out STD_LOGIC);
end Adder_3bit;

architecture Behavioral of Adder_3bit is
   component FA
      port (
         A: in std_logic;
         B: in std_logic;
         C_in: in std_logic;
         S: out std_logic;
         C_out: out std_logic);
      end component;

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

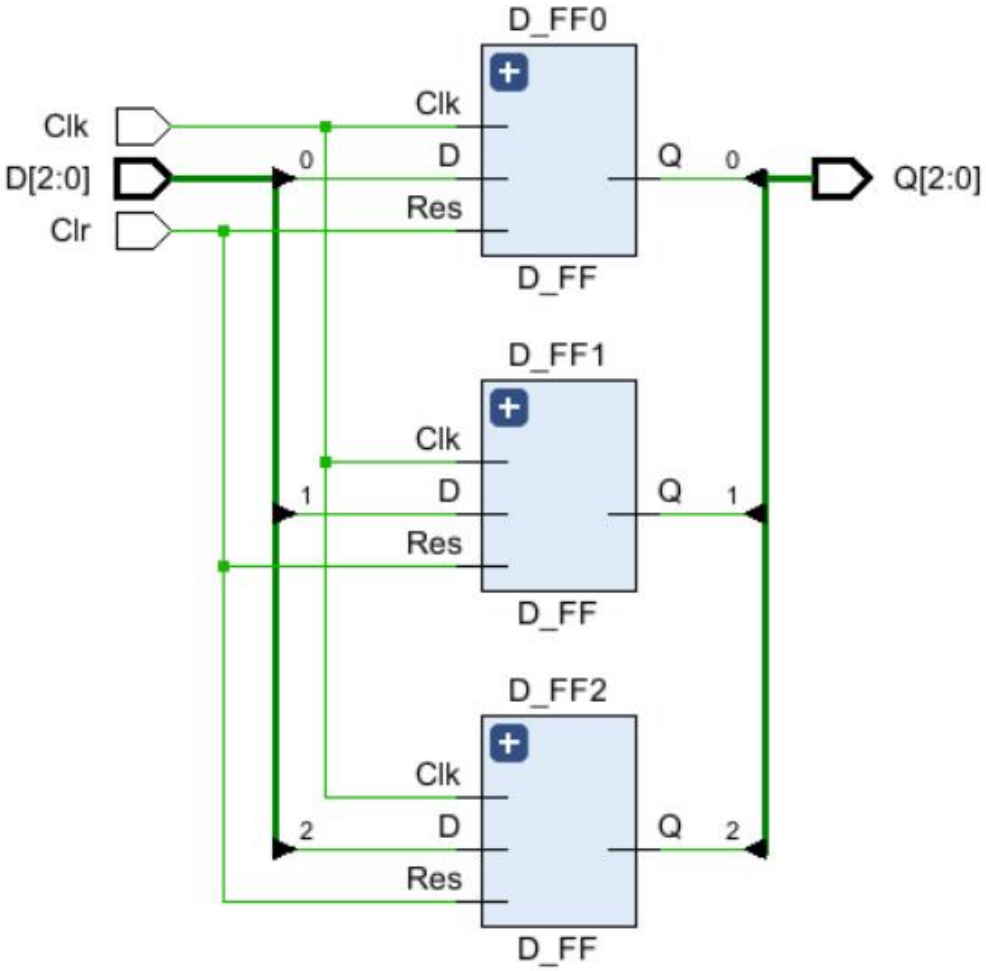

A[2:0]
0
FA_0
A
B
C_in
C_out
S
0
FA
FA_1
1
A
B
C_in
C_out
S
1
FA
FA_2
2
A
B
C_in
C_out
S
2
FA
carry
S[2:0]

3-bit Adder simulation

```
entity TB_Adder_3bit is
--  Port ( );
end TB_Adder_3bit;

architecture Behavioral of TB_Adder_3bit is
COMPONENT RCA_3
 PORT(  A : IN STD_LOGIC_VECTOR (2 downto 0);
       S: OUT STD_LOGIC_VECTOR (2 downto 0);
       carry : out STD_LOGIC );
END COMPONENT;

SIGNAL A : std_logic_vector (2 downto 0);
SIGNAL S: std_logic_vector (2 downto 0);
SIGNAL carry : std_logic;
begin
UUT: RCA_3 PORT MAP(
   A(0) => A(0),
   A(1) => A(1),
   A(2)=> A(2),
   S(0)=>S(0), S(1)=>S(1), S(2)=>S(2),
   carry  => carry
   );

process
 begin
    A(0) <= '1';
    A(1) <= '0';
    A(2) <= '1';
 WAIT FOR 100 ns; -- after 100 ns change inputs
    A(0) <= '1';
    A(1) <= '1';
    A(2) <= '1';
 WAIT FOR 100 ns; --change again
   A(0) <= '0';
   A(1) <= '0';
   A(2) <= '1';
 WAIT FOR 100 ns; --change again
   A(0) <= '0';
   A(1) <= '1';
   A(2) <= '1';
 WAIT ;

end process;
end Behavioral;
```

3-bit Program Counter (PC)

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

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

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

begin
   D_FF0 : D_FF
      port map (
      D => D(0),
      Q => Q(0),
      Res =>Clr,
      Clk => Clk);

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

   D_FF2 : D_FF
      port map (
      D => D(2),
      Q => Q(2),
      Res =>Clr,
      Clk => Clk);
end Behavioral;
```


D_FF0
Clk
D[2:0]
Clr
Clk
D
Res
0
Q
0
Q[2:0]
D_FF
D_FF1
Clk
D
Res
1
Q
1
D_FF
D_FF2
Clk
D
Res
2
Q
2
D_FF

3-bit PC simulation

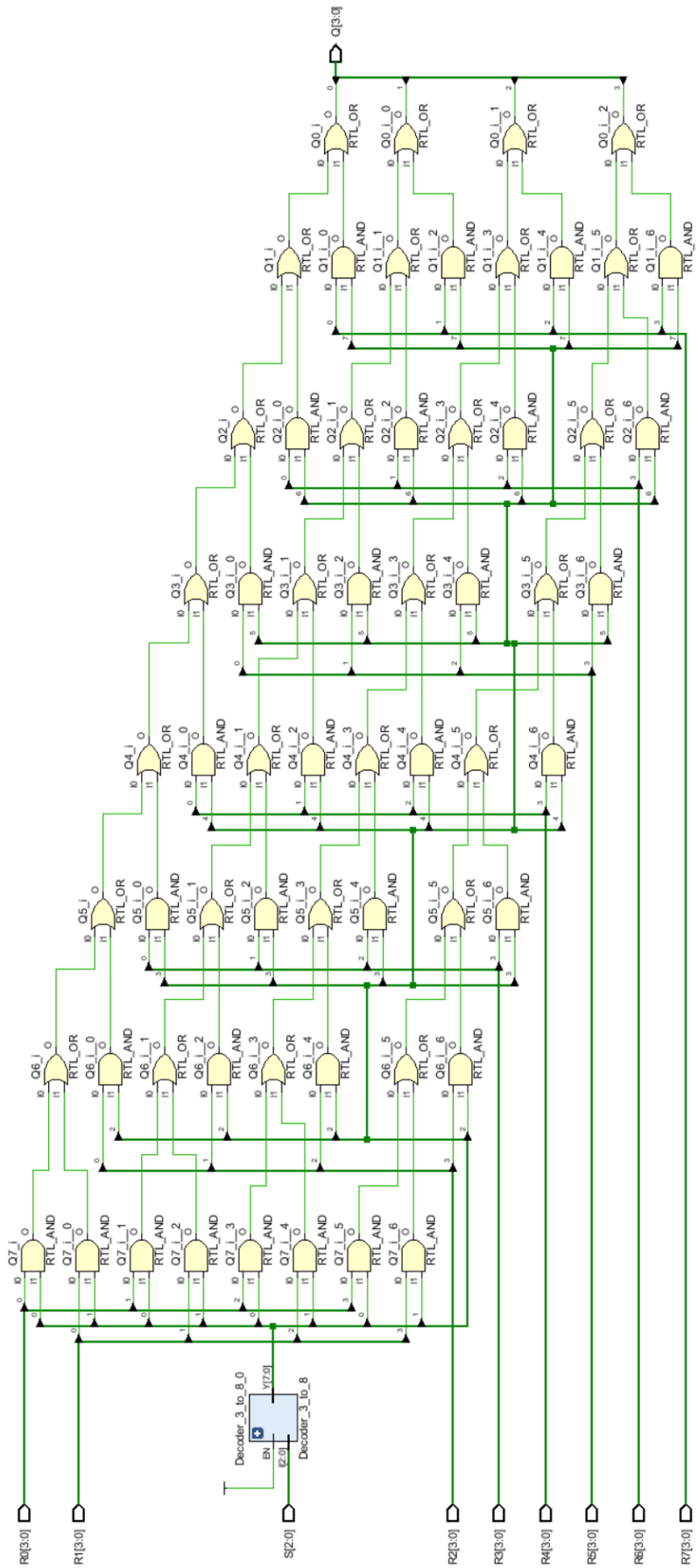
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

architecture Behavioral of TB_Program_Counter is
   component Program_Counter is
      Port (  D : in std_logic_vector;
            Clr : in std_logic;
            Clk : in std_logic;
            Q : out std_logic_vector);
   end component;
   signal D, Q : std_logic_vector (2 downto 0);
   signal Clr, Clk : std_logic;

begin
   uut: Program_Counter port map(
      D => D,
      Clk => Clk,
      Clr => Clr,
      Q => Q);
   process
   begin
      Clk <= '0';
      wait for 50ns;
      Clk <= '1';
      wait for 50ns;
   end process;
   process
   begin
      Clr <= '0';
      wait for 10ns;
      D <= "000";
      wait for 100ns;
      D <= "001";
      wait for 100ns;
      D <= "010";
      wait for 100ns;
      D <= "011";
      wait for 50ns;
      Clr <= '1';
      wait for 50ns;
      D <= "100";
      wait for 100ns;
      D <= "101";
      wait for 100ns;
      D <= "110";
      wait for 100ns;
      D <= "111";
      wait;
   end process;
end Behavioral;
```

Multiplexer 8 to 1_ 4bit

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

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

architecture Behavioral of Mux_8_to_1_4bit is
   component Decoder_3_to_8
      port(
         I : in std_logic_vector;
         EN : in std_logic;
         Y : out std_logic_vector);
   end component;
signal I0 : std_logic_vector (2 downto 0);
signal EN0 : std_logic;
signal X: std_logic_vector (7 downto 0);

begin
   Decoder_3_to_8_0 : Decoder_3_to_8
      port map(
         I => I0,
         EN => EN0,
         Y => X);
   EN0 <= '1';
   I0 <= S;
   Q(0) <= (R0(0) AND X(0)) OR (R1(0) AND X(1)) OR (R2(0) AND X(2)) OR
(R3(0) AND X(3)) OR (R4(0) AND X(4)) OR (R5(0) AND X(5)) OR (R6(0) AND
X(6)) OR (R7(0) AND X(7));
   Q(1) <= (R0(1) AND X(0)) OR (R1(1) AND X(1)) OR (R2(1) AND X(2)) OR
(R3(1) AND X(3)) OR (R4(1) AND X(4)) OR (R5(1) AND X(5)) OR (R6(1) AND
X(6)) OR (R7(1) AND X(7));
   Q(2) <= (R0(2) AND X(0)) OR (R1(2) AND X(1)) OR (R2(2) AND X(2)) OR
(R3(2) AND X(3)) OR (R4(2) AND X(4)) OR (R5(2) AND X(5)) OR (R6(2) AND
X(6)) OR (R7(2) AND X(7));
   Q(3) <= (R0(3) AND X(0)) OR (R1(3) AND X(1)) OR (R2(3) AND X(2)) OR
(R3(3) AND X(3)) OR (R4(3) AND X(4)) OR (R5(3) AND X(5)) OR (R6(3) AND
X(6)) OR (R7(3) AND X(7));

end Behavioral;
```

Decoder_3_to_8

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

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

architecture Behavioral of Decoder_3_to_8 is
   component 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;
```


Q[3:0]
RTL_OR
RTL_AND
Decoder_3_to_8_0
Decoder_3_to_8
EN
I[2:0]
Y[7:0]
R0[3:0]
R1[3:0]
S[2:0]
R2[3:0]
R3[3:0]
R4[3:0]
R5[3:0]
R6[3:0]
R7[3:0]

Multiplexer 8 to 1_4bit simulation

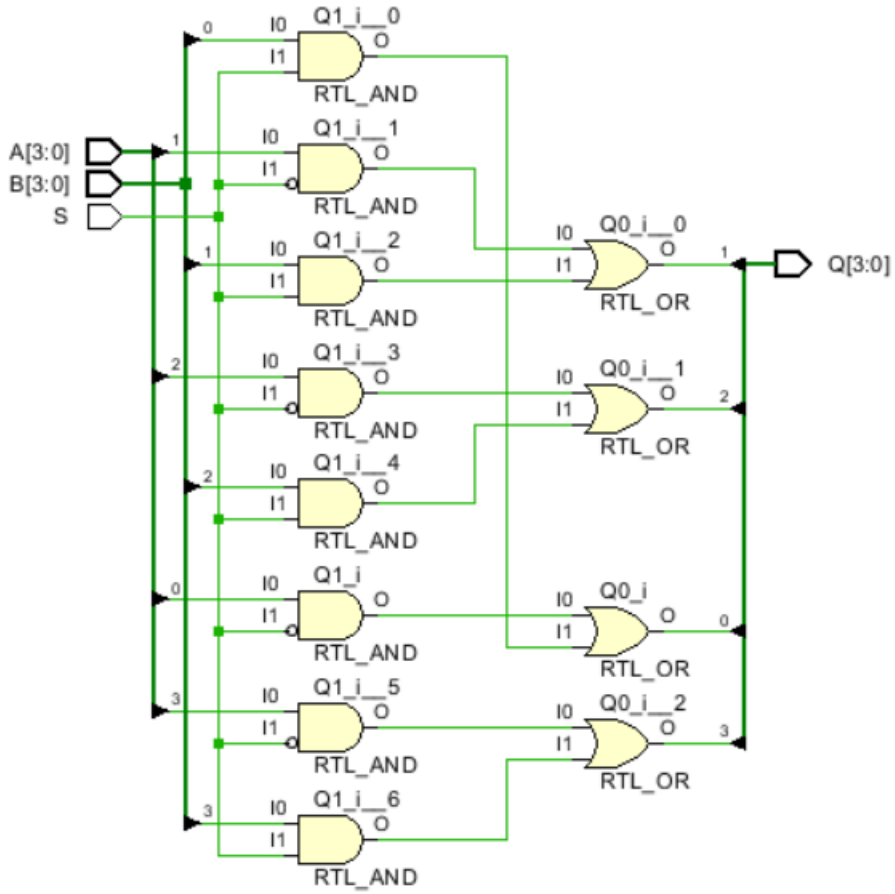
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.Numeric_std.all;
entity TB_Mux_8_to_1_4bit is
--  Port ( );
end TB_Mux_8_to_1_4bit;

architecture Behavioral of TB_Mux_8_to_1_4bit is
   component Mux_8_to_1_4bit is
      Port ( S : in STD_LOGIC_VECTOR;
           R0 : in STD_LOGIC_VECTOR;
           R1 : in STD_LOGIC_VECTOR;
           R2 : in STD_LOGIC_VECTOR;
           R3 : in STD_LOGIC_VECTOR;
           R4 : in STD_LOGIC_VECTOR;
           R5 : in STD_LOGIC_VECTOR;
           R6 : in STD_LOGIC_VECTOR;
           R7 : in STD_LOGIC_VECTOR;
           Q : out STD_LOGIC_VECTOR);
   end component;
   signal S : std_logic_vector(2 downto 0);
   signal Q,R0,R1,R2,R3,R4,R5,R6,R7 : std_logic_vector (3 downto 0);

begin
   uut: Mux_8_to_1_4bit port map(
      S => S,
      R0 => R0,
      R1 => R1,
      R2 => R2,
      R3 => R3,
      R4 => R4,
      R5 => R5,
      R6 => R6,
      R7 => R7,
      Q => Q);
   process
   begin
      R0 <= "0000";
      R1 <= "0001";
      R2 <= "0010";
      R3 <= "0011";
      R4 <= "0100";
      R5 <= "0101";
      R6 <= "0110";
      R7 <= "0111";
      for i in 0 to 8 loop
         S <= std_logic_vector(to_unsigned(i,3));
         wait for 100ns;
      end loop;
      wait;
   end process;
end Behavioral;
```

Decoder_2_to_4

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

entity Decoder_2_to_4 is
   Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
          EN : in STD_LOGIC;
          Y : out STD_LOGIC_VECTOR (3 downto 0));
end Decoder_2_to_4;

architecture Behavioral of Decoder_2_to_4 is

begin
   Y(0) <= (not I(0)) and (not I(1)) and EN;
   Y(1) <= I(0)  and (not I(1)) and EN;
   Y(2) <= I(1) and (not I(0)) and EN;
   Y(3) <= I(0) and I(1) and EN;
end Behavioral;
```

Multiplexer 2 to 1_ 4bit

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

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

architecture Behavioral of Mux_2_to_1_4bit is

begin
   Q(0) <= (A(0) AND (NOT S)) OR (B(0) AND S);
   Q(1) <= (A(1) AND (NOT S)) OR (B(1) AND S);
   Q(2) <= (A(2) AND (NOT S)) OR (B(2) AND S);
   Q(3) <= (A(3) AND (NOT S)) OR (B(3) AND S);
end Behavioral;
```

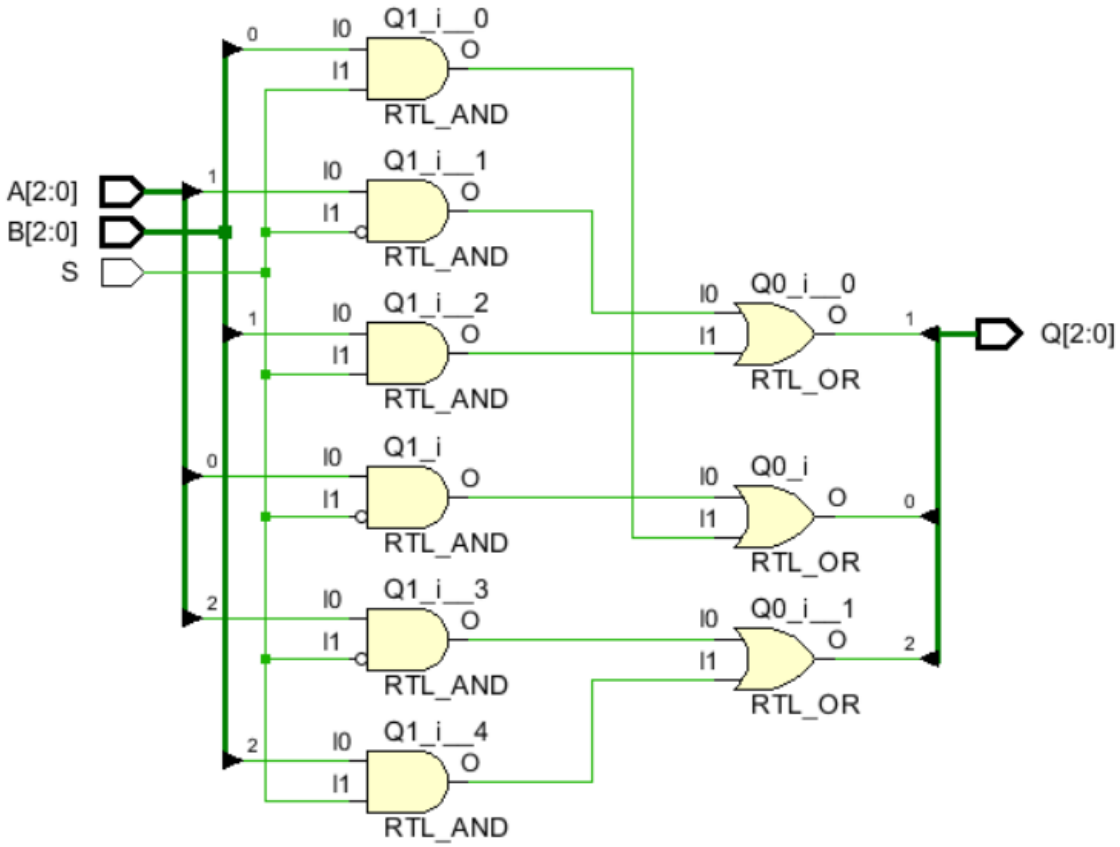


Multiplexer 2 to 1_4bit simulation

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

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

architecture Behavioral of TB_Mux_2_to_1_4bit is
  component Mux_2_to_1_4bit is
    Port (  A : in std_logic_vector;
            B : in std_logic_vector;
            S : in std_logic;
            Q : out std_logic_vector);
  end component;
  signal A,B,Q : std_logic_vector(3 downto 0);
  signal C : std_logic;

begin
  uut : Mux_2_to_1_4bit port map(
    A => A,
    B => B,
    S => C,
    Q => Q);
  process
  begin
    C <= '0';
    A <= "0000";
    B <= "0000";
 wait for 100ns;
 C <= '1';
 wait for 100ns;
 A <= "0010";
 B <= "0011";
 wait for 100ns;
 C <= '0';
 wait for 100ns;
 A <= "0100";
 B <= "0101";
 wait for 100ns;
 C <= '1';
 wait for 100ns;
 A <= "0110";
 B <= "0111";
 wait for 100ns;
 C <= '0';
 wait for 100ns;
 A <= "1000";
 B <= "1001";
 wait for 100ns;
 C <= '0';
 wait;
end process;
end Behavioral;
```

Multiplexer 2 to 1_ 3bit

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

entity Mux_2_to_1_3bit is
   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
          B : in STD_LOGIC_VECTOR (2 downto 0);
          S : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (2 downto 0));
end Mux_2_to_1_3bit;

architecture Behavioral of Mux_2_to_1_3bit is

begin
   Q(0) <= (A(0) AND NOT(S)) OR (B(0) AND S);
   Q(1) <= (A(1) AND NOT(S)) OR (B(1) AND S);
   Q(2) <= (A(2) AND NOT(S)) OR (B(2) AND S);
end Behavioral;
```

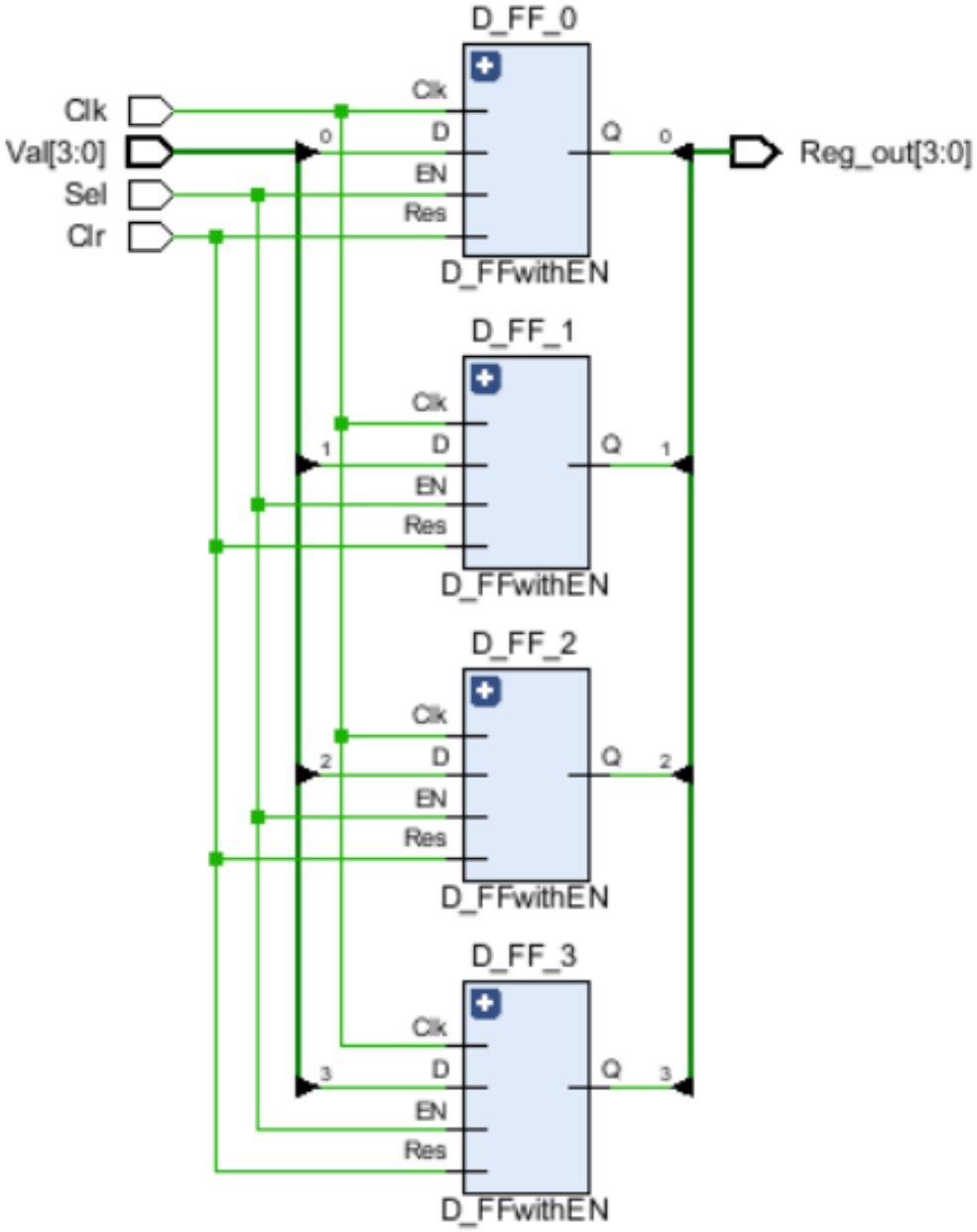


Multiplexer 2 to 1_3bit simulation

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

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

architecture Behavioral of TB_Mux_2_to_1_3bit is
   component Mux_2_to_1_3bit is
      Port (  A : in std_logic_vector;
              B : in std_logic_vector;
              s : in std_logic;
              Q : out std_logic_vector);
   end component;
   signal A,B,Q : std_logic_vector(2 downto 0);
   signal C : std_logic;

begin
   uut : Mux_2_to_1_3bit port map(
      A => A,
      B => B,
      s => C,
      Q => Q);
   process
   begin
      C <= '0';
      A <= "000";
      B <= "000";
      wait for 200ns;
      A <= "010";
      B <= "011";
      wait for 200ns;
      C <= '1';
      wait for 200ns;
      A <= "100";
      B <= "101";
      wait for 200ns;
      C <= '0';
      wait;
   end process;
end Behavioral;
```

Register_4bit

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

entity Register_4bit is
   Port ( Clk : in STD_LOGIC;
          Val : in STD_LOGIC_VECTOR (3 downto 0);
          Sel : in STD_LOGIC;
          Clr : in STD_LOGIC;
          Reg_out : out STD_LOGIC_VECTOR (3 downto 0));
end Register_4bit;

architecture Behavioral of Register_4bit is
   component D_FFwithEN
      port(
         D : in std_logic;
         Res : in std_logic;
         Clk : in std_logic;
         Q : out std_logic;
         EN : in std_logic;
         Qbar : out std_logic);
   end component;
   signal Activate : std_logic;

begin
   D_FF_0 : D_FFwithEN
      port map(
         D => Val(0),
         Res => Clr,
         Clk => Clk,
         EN => Sel,
         Q => Reg_out(0));
   D_FF_1 : D_FFwithEN
      port map(
         D => Val(1),
         Res => Clr,
         Clk =>  Clk,
         EN => Sel,
         Q => Reg_out(1));
   D_FF_2 : D_FFwithEN
      port map(
         D => Val(2),
         Res => Clr,
         Clk =>  Clk,
         EN => Sel,
         Q => Reg_out(2));
   D_FF_3 : D_FFwithEN
      port map(
         D => Val(3),
         Res => Clr,
         Clk =>  Clk,
         EN => Sel,
         Q => Reg_out(3));
end Behavioral;
```


D_FF_0
Clk
Val[3:0]
Sel
Clr
Clk
D
EN
Res
Q
0
Reg_out[3:0]
D_FFwithEN
D_FF_1
1
D_FFwithEN
D_FF_2
2
D_FFwithEN
D_FF_3
3
D_FFwithEN

Register_4bit simulation

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

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

architecture Behavioral of TB_Register_4bit is
   component Register_4bit is
      Port (
         Clk : in std_logic;
         Val : in std_logic_vector;
         Sel : in std_logic;
         Clr : in std_logic;
         Reg_out : out std_logic_vector);
   end component;
   signal Clk, Sel, Clr : std_logic;
   signal Val, Reg_out : std_logic_vector (3 downto 0);

begin
   uut : Register_4bit
      Port map (
         Clk => Clk,
         Val => Val,
         Sel => Sel,
         Clr => Clr,
         Reg_out => Reg_out);
   process
   begin
      Clk <= '0';
      wait for 50ns;
      Clk <= '1';
      wait for 50ns;
   end process;
   process
   begin
      Clr <= '0';
      Sel <= '1';
      wait for 10ns;
      Val <= "1111";
      wait for 100ns;
      Val <= "1001";
      wait for 100ns;
      Val <= "1100";
      wait for 100ns;
      Sel <= '0';
      Val <= "1010";
      wait for 100ns;
      Sel <= '1';
      Val <= "0110";
      wait;
   end process;

end Behavioral;
```

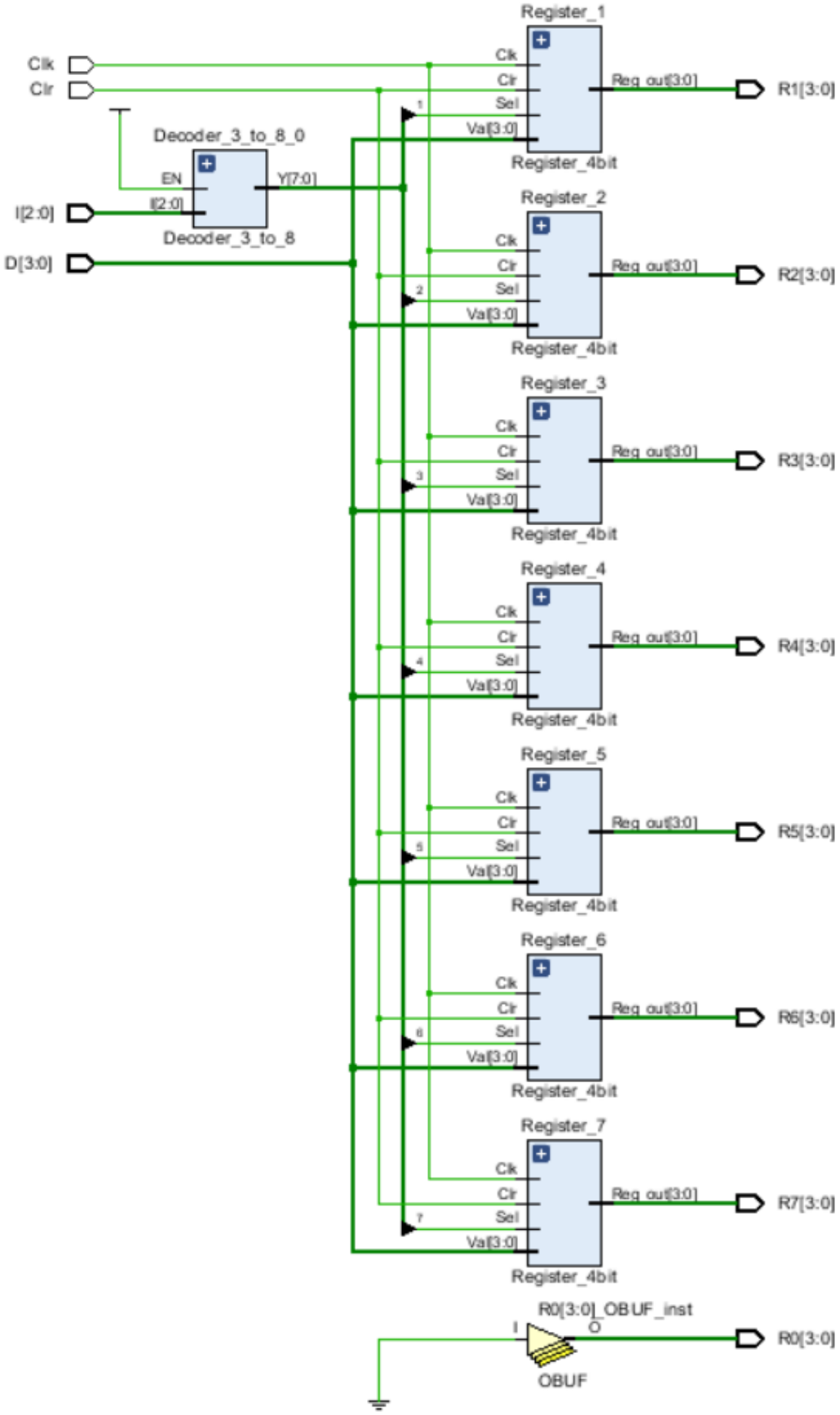
Register Bank

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

entity Register_Bank is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           I : in STD_LOGIC_VECTOR (2 downto 0);
           Clr : in STD_LOGIC;
           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);
           R0 : out STD_LOGIC_VECTOR (3 downto 0));
end Register_Bank;

architecture Behavioral of Register_Bank is
    component Register_4bit
        port(
            Clk : in std_logic;
            Val : in std_logic_vector;
            Sel : in std_logic;
            Clr : in std_logic;
            Reg_out : out std_logic_vector);
    end component;
component Decoder_3_to_8
        port(
            I : in std_logic_vector;
            EN : in std_logic;
            Y : out std_logic_vector);
    end component;
    signal Sel : std_logic_vector (7 downto 0);
--    signal EN0 : std_logic;

begin
    Decoder_3_to_8_0 : Decoder_3_to_8
        port map(
            I => I,
            EN => '1',
            Y => Sel);
    R0 <= "0000";
    Register_1 : Register_4bit
        port map(
            Clk => Clk,
            Val => D,
            Sel => Sel(1),
            Clr => Clr,
```

```
        Reg_out=>R1);

    Register_2 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(2),
        Clr => Clr,
        Reg_out => R2);
    Register_3 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(3),
        Clr => Clr,
        Reg_out => R3);
    Register_4 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(4),
        Clr => Clr,
        Reg_out => R4);
    Register_5 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(5),
        Clr => Clr,
        Reg_out => R5);
    Register_6 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(6),
        Clr => Clr,
        Reg_out => R6);
    Register_7 : Register_4bit
      port map(
        Clk => Clk,
        Val => D,
        Sel => Sel(7),
        Clr => Clr,
        Reg_out => R7);
end Behavioral;
```

D flip flop with ENABLE

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

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

architecture Behavioral of D_FFwithEN is

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


Register_1
Clk
Clr
Sel
Va[3:0]
Reg_out[3:0]
R1[3:0]
Register_4bit
Clk
Decoder_3_to_8_0
EN
I[2:0]
Y[7:0]
Decoder_3_to_8
I[2:0]
D[3:0]
Register_2
R2[3:0]
Register_3
R3[3:0]
Register_4
R4[3:0]
Register_5
R5[3:0]
Register_6
R6[3:0]
Register_7
R7[3:0]
R0[3:0]_OBUF_inst
I
O
OBUF
R0[3:0]

Register Bank simulation

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


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

architecture Behavioral of TB_Register_Bank is
   component Register_Bank is
      Port ( D : in STD_LOGIC_VECTOR ;
           Clk : in STD_LOGIC;
           I : in STD_LOGIC_VECTOR ;
           Clr : in STD_LOGIC;
           R1 : out STD_LOGIC_VECTOR;
           R2 : out STD_LOGIC_VECTOR ;
           R3 : out STD_LOGIC_VECTOR ;
           R4 : out STD_LOGIC_VECTOR ;
           R5 : out STD_LOGIC_VECTOR ;
           R6 : out STD_LOGIC_VECTOR ;
           R7 : out STD_LOGIC_VECTOR ;
           R0 : out STD_LOGIC_VECTOR );
   end component;
   signal D,R1,R2,R3,R4,R5,R6,R7,R0 : std_logic_vector(3 downto 0);
   signal I : STD_LOGIC_VECTOR (2 downto 0);
   signal Clr, Clk : STD_LOGIC;
begin
   UUT : Register_Bank
   port map (
     D => D ,
     I => I ,
     Clr => Clr,
     Clk => Clk,
     R1 => R1,
     R2 => R2,
     R3 => R3,
     R4 => R4,
     R5 => R5,
     R6 => R6,
     R7 => R7,
     R0 => R0
   );
   process
   begin
     Clk <= '0';
     wait for 20ns;
     Clk <= '1';
     wait for 20ns;
   end process;
   process
   begin
     Clr <= '1';
     wait for 10ns;
     wait for 100ns;
     Clr <= '0';
```

```
        D <= "0001";
        I <= "001";
        wait for 100ns;
        D <= "0101";
        I <= "010";
        wait for 100ns;
        D <= "1100";
        I <= "111";
        wait for 100ns;
        D <= "1111";
        I <= "000";
        wait for 100 ns;
        Clr <= '1';
        wait;

    end process;

end Behavioral;
```

Program ROM

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


entity ROM is
   Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
          Q : 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 := (
---------- Countdown from 7 to 1.--------------------------
--          "100010000111", -- MOVI R1, 111
--          "100100000001", -- MOVI R2, 1
--          "010100000000", -- NEG R2
--          "000010100000", -- ADD R1, R2
--          "110010000111", -- JZR R1, 7
--          "111110000011", -- JZR R0, 3
--          "110000000111", -- JZR R0, 7
--          "110000000111"  -- JZR R0, 7
--          );

---------- Add numbers from 3 to 1-----------------------------
           "100010000011", -- MOVI R1, 3
           "100100000001", -- MOVI R2, 1
           "010100000000", -- NEG R2
           "001110010000", -- ADD R7, R1
           "000010100000", -- ADD R1, R2
           "110010000111", -- JZR R1, 7
           "110000000011", -- JZR R0, 3
           "110000000111"  -- JZR R0, 7
           );
begin
   Q <= ROM(to_integer(unsigned(S)));
end Behavioral;
```

ROM simulation

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

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

architecture Behavioral of TB_ROM is
   component ROM is
      Port (
         S : in std_logic_vector;
         Q : out std_logic_vector);
   end component;
   signal S : std_logic_vector(2 downto 0);
   signal Q : std_logic_vector(11 downto 0);

begin
   uut: ROM port map(
      S => S,
      Q => Q);
   process
   begin
      for i in 0 to 5 loop
         S <= std_logic_vector(to_unsigned(i,3));
         wait for 100ns;
      end loop;
      wait;
   end process;
end Behavioral;
```

Instruction Decoder

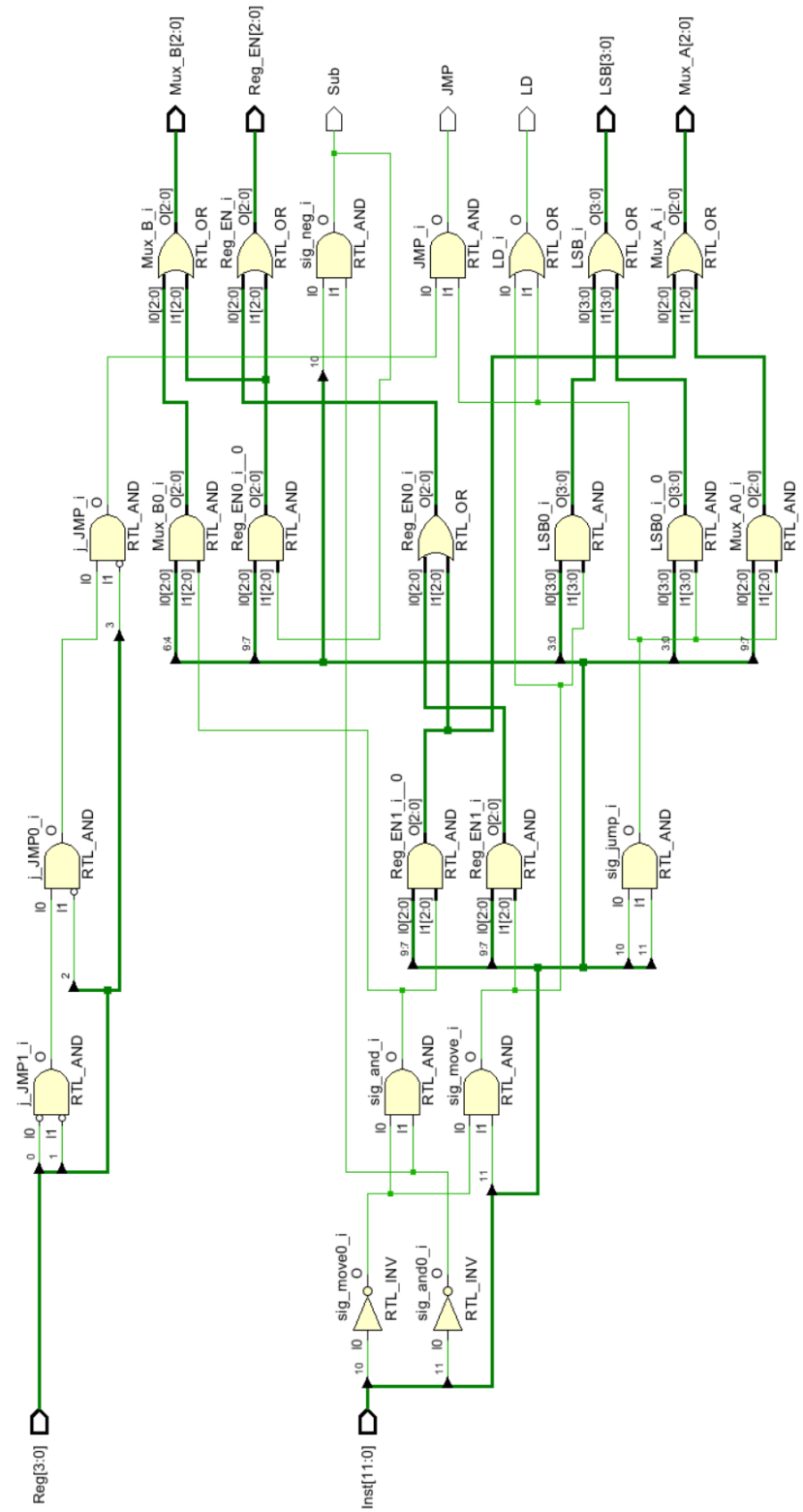
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Instruction_Decoder is
    Port ( Inst : in STD_LOGIC_VECTOR (11 downto 0);
           Reg : in STD_LOGIC_VECTOR (3 downto 0);
           LSB : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Mux_A : out STD_LOGIC_VECTOR (2 downto 0);
           LD : out STD_LOGIC;
           Mux_B : out STD_LOGIC_VECTOR (2 downto 0);
           Sub : out STD_LOGIC;
           JMP : out STD_LOGIC);
end Instruction_Decoder;

architecture Behavioral of Instruction_Decoder is
    component D_FF
        port(
            D : in std_logic;
            Res : in std_logic;
            Clk : in std_logic;
            Q : out std_logic;
            Qbar : out std_logic);
    end component;
    signal sig_move, sig_and, sig_neg, sig_jump : std_logic;
    signal sig_move2, sig_and2, sig_neg2, sig_jump2 : std_logic_vector (2 downto 0);
    signal sig_move3, sig_and3, sig_neg3, sig_jump3 : std_logic_vector( 3 downto 0);
    signal m_LSB, a_LSB, n_LSB, j_LSB, LSB0 : std_logic_vector(3 downto 0);
    signal m_Reg_EN, a_Reg_EN, n_Reg_EN, j_Reg_EN, Reg_EN0 : std_logic_vector (2 downto 0);
    signal m_Mux_A, a_Mux_A, n_Mux_A, j_Mux_A, Mux_A0 : std_logic_vector (2 downto 0);
    signal m_LD, a_LD, n_LD, j_LD, LD0 : std_logic;
    signal m_Mux_B, a_Mux_B, n_Mux_B, j_Mux_B, Mux_B0 : std_logic_vector ( 2 downto 0);
    signal m_Sub, a_Sub, n_Sub, j_Sub, Sub0 : std_logic;
    signal m_JMP, a_JMP, n_JMP, j_JMP, JMP0 : std_logic;
    signal Res : std_logic;

begin


    sig_move <= (NOT Inst(10)) AND (Inst(11));
    sig_and <= (NOT Inst(10)) AND (NOT Inst(11));
    sig_neg <= (Inst(10)) AND (NOT Inst(11));
    sig_jump <= (Inst(10)) AND (Inst(11));

    sig_move2(0) <= sig_move; sig_move2(1) <= sig_move; sig_move2(2) <= sig_move;
    sig_and2(0) <= sig_and; sig_and2(1) <= sig_and; sig_and2(2) <= sig_and;
    sig_neg2(0) <= sig_neg; sig_neg2(1) <= sig_neg; sig_neg2(2) <= sig_neg;
    sig_jump2(0) <= sig_jump; sig_jump2(1) <= sig_jump; sig_jump2(2) <= sig_jump;
    sig_move3(0) <= sig_move; sig_move3(1) <= sig_move; sig_move3(2) <= sig_move; sig_move3(3) <= sig_move;
    sig_and3(0) <= sig_and; sig_and3(1) <= sig_and; sig_and3(2) <= sig_and; sig_and3(3) <= sig_and;
    sig_neg3(0) <= sig_neg; sig_neg3(1) <= sig_neg; sig_neg3(2) <= sig_neg; sig_neg3(3) <= sig_neg;
    sig_jump3(0) <= sig_jump; sig_jump3(1) <= sig_jump; sig_jump3(2) <= sig_jump; sig_jump3(3) <= sig_jump;

    --move instruction
    m_LSB <= Inst(3 downto 0);
```

```
    m_Reg_EN <= Inst(9 downto 7);
    m_Mux_A <= "000";
    m_LD <= '1';
    m_Mux_B <= "000";
    m_Sub <= '0';
    m_JMP <= '0';

    --add instruction
    a_LSB <= "0000";
    a_Reg_EN <= Inst(9 downto 7);
    a_Mux_A <= Inst(9 downto 7);
    a_LD <= '0';
    a_Mux_B <= Inst(6 downto 4);
    a_Sub <= '0';
    a_JMP <= '0';

    --neg instruction
    n_LSB <= "0000";
    n_Reg_EN <= Inst(9 downto 7);
    n_Mux_A <= "000";
    n_LD <= '0';
    n_Mux_B <= Inst(9 downto 7);
    n_Sub <= '1';
    n_JMP <= '0';

    --jump instruction
    j_LSB <= Inst(3 downto 0);
    j_Reg_EN <= "000";
    j_Mux_A <= Inst(9 downto 7);
    j_LD <= '1';
    j_Mux_B <= "000";
    j_Sub <= '0';
    j_JMP <= (NOT Reg(0)) AND (NOT Reg(1)) AND (NOT Reg(2)) AND (NOT Reg(3));

    LSB <= (m_LSB AND sig_move3) OR (a_LSB AND sig_and3) OR (n_LSB AND sig_neg3) OR (j_LSB AND sig_jump3);
    Reg_EN <= (m_Reg_EN AND sig_move2) OR (a_Reg_EN AND sig_and2) OR (n_Reg_EN AND sig_neg2) OR (j_Reg_EN AND
sig_jump2);
    Mux_A <= (m_Mux_A AND sig_move2) OR (a_Mux_A AND sig_and2) OR (n_Mux_A AND sig_neg2) OR (j_Mux_A AND
sig_jump2);
    LD <= (m_LD AND sig_move) OR (a_LD AND sig_and) OR (n_LD AND sig_neg) OR (j_LD AND sig_jump);
    Mux_B <= (m_Mux_B AND sig_move2) OR (a_Mux_B AND sig_and2) OR (n_Mux_B AND sig_neg2) OR (j_Mux_B AND
sig_jump2);
    Sub <= (m_Sub AND sig_move) OR (a_Sub AND sig_and) OR (n_Sub AND sig_neg) OR (j_Sub AND sig_jump);
    JMP <= (m_JMP AND sig_move) OR (a_JMP AND sig_and) OR (n_JMP AND sig_neg) OR (j_JMP AND sig_jump);
end Behavioral;
```

Instruction Decoder simulation

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

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

architecture Behavioral of TB_Instruction_Decoder is
   component Instruction_Decoder is
      Port (  Inst : in std_logic_vector;
            Reg : in std_logic_vector;
            LSB : out std_logic_vector;
            Reg_EN : out std_logic_vector;
            Mux_A : out std_logic_vector;
            LD : out std_logic;
            Mux_B : out std_logic_vector;
            Sub : out std_logic;
            JMP : out std_logic);
   end component;
   signal Inst : std_logic_vector (11 downto 0);
   signal LD, Sub, JMP : std_logic;
   signal Reg, LSB : std_logic_vector (3 downto 0);
   signal Mux_A, Mux_B, Reg_EN : std_logic_vector (2 downto 0);

begin
   uut: Instruction_Decoder port map(
      Inst => Inst,
      Reg => Reg,
      LSB => LSB,
      Reg_EN => Reg_EN,
      Mux_A => Mux_A,
      LD => LD,
      Mux_B => Mux_B,
      Sub => Sub,
      JMP => JMP);

   process
   begin
      wait for 5ns;
      Reg <= "0000";
      Inst <= "100010000010";
      wait for 80ns;
      Inst <= "100100000001";
      wait for 80ns;
      Inst <= "010100000000";
      wait for 80ns;
      Inst <= "000010100000";
      wait for 80ns;
      Inst <= "110010000111";
      wait for 80ns;
      Reg <= "0100";
      wait for 80ns;
      Inst <= "111110000011";
      wait;
```

```
    end process;
end Behavioral;
```



LUT 16_7-segmant

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

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

architecture Behavioral of LUT_16_7 is
   type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);
      signal sevenSegment_ROM  : rom_type := (
         "1000000", --0
         "1111001", --1
         "0100100", --2
         "0110000", --3
         "0011001", --4
         "0010010", --5
         "0000010", --6
         "1111000", --7
         "0000000", --8
         "0010000", --9
         "0001000", --a
         "0000011", --b
         "1000110", --c
         "0100001", --d
         "0000110", --e
         "0001110"     );  –f
begin
   data <= sevenSegment_ROM(to_integer(unsigned(address)));
end Behavioral;
```

Slow Clock

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

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


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

begin
   process (Clk_in) begin
      if (rising_edge (Clk_in)) then
         count <= count + 1;
--          if (count = 50000000) then -- For the board.
         if (count = 1) then -- For simulations.
            clk_status <= not clk_status;
            Clk_out <= clk_status;
            count <= 1;
         end if;
      end if;
   end process;
end Behavioral;
```

NANOPROCESSOR – VHDL code

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

entity Nanoprocessor is
   Port ( Clr : in STD_LOGIC;
          Clk : in STD_LOGIC;
          R : out STD_LOGIC_VECTOR (3 downto 0);
          Overflow : out STD_LOGIC;
          Zero : out STD_LOGIC;
          Seven_Seg : out std_logic_vector (6 downto 0));
end Nanoprocessor;

architecture Behavioral of Nanoprocessor is
   component Slow_Clk is
      Port (  Clk_in : in std_logic;
              Clk_out : out std_logic);
   end component;
   component Mux_8_to_1_4bit is
      Port ( S : in STD_LOGIC_VECTOR;
             R0 : in STD_LOGIC_VECTOR;
             R1 : in STD_LOGIC_VECTOR;
             R2 : in STD_LOGIC_VECTOR;
             R3 : in STD_LOGIC_VECTOR;
             R4 : in STD_LOGIC_VECTOR;
             R5 : in STD_LOGIC_VECTOR;
             R6 : in STD_LOGIC_VECTOR;
             R7 : in STD_LOGIC_VECTOR;
             Q : out STD_LOGIC_VECTOR);
   end component;
   component Register_Bank is
      Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
             Clk : in STD_LOGIC;
             I : in STD_LOGIC_VECTOR (2 downto 0);
             Clr : in STD_LOGIC;
             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);
             R0 : out STD_LOGIC_VECTOR (3 downto 0));
   end component;
   component Instruction_Decoder is
```

```
    Port ( Inst : in STD_LOGIC_VECTOR (11 downto 0);
           Reg : in STD_LOGIC_VECTOR (3 downto 0);
           LSB : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_EN : out STD_LOGIC_VECTOR (2 downto 0);
           Mux_A : out STD_LOGIC_VECTOR (2 downto 0);
           LD : out STD_LOGIC;
           Mux_B : out STD_LOGIC_VECTOR (2 downto 0);
           Sub : out STD_LOGIC;
           JMP : out STD_LOGIC);
  end component;
  component Mux_2_to_1_4bit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
  end component;
  component Mux_2_to_1_3bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
  end component;
  component ROM is
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           Q : out STD_LOGIC_VECTOR (11 downto 0));
  end component;
  component Program_Counter is
     Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
            Clr : in STD_LOGIC;
            Clk : in STD_LOGIC;
            Q : out STD_LOGIC_VECTOR (2 downto 0));
  end component;
  component Adder_3bit is
    Port (  A : in STD_LOGIC_VECTOR (2 downto 0);
            S : out STD_LOGIC_VECTOR(2 downto 0);
            carry : out STD_LOGIC);
  end component;
  component Add_Sub_4bit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           B : in STD_LOGIC_VECTOR (3 downto 0);
           S : out STD_LOGIC_VECTOR (3 downto 0);
           M : in STD_LOGIC;
           overflow : out STD_LOGIC);
  end component;

  component LUT_16_7 is
        Port ( address : in STD_LOGIC_VECTOR (3 downto 0);
               data : out STD_LOGIC_VECTOR (6 downto 0));
  end component;
  signal PC_ROM, Add3_MuxC, PC_MuxC : std_logic_vector(2 downto 0);
  signal ROM_Decoder : std_logic_vector(11 downto 0);
  signal Decoder_MuxD : std_logic_vector(3 downto 0);
```

```
    signal Decoder_MuxC, Decoder_MuxDSelc, Decoder_Adder : std_logic;
    signal Decoder_RegBank, Decoder_MuxA, Decoder_MuxB : std_logic_vector(2 downto 0);
    signal MuxD_Adder, MuxD_RegBank : std_logic_vector(3 downto 0);
    signal R0,R1,R2,R3,R4,R5,R6,R7 : std_logic_vector(3 downto 0);
    signal MuxA_Adder, MuxB_Adder : std_logic_vector(3 downto 0);
    signal Slw_Clk : std_logic;
    signal Adder_Cout : std_logic;

begin
    LUT : LUT_16_7
        Port map(
            address => R7,
            data => Seven_Seg);

    Slow_Clk_0 : Slow_Clk
        Port map(
            Clk_in => Clk,
            Clk_out => Slw_Clk);

    Adder : Add_Sub_4bit
        Port map(
            A => MuxA_Adder,
            B => MuxB_Adder,
            M => Decoder_Adder,
            overflow => overflow,
            S => MuxD_Adder);
    MuxA : Mux_8_to_1_4bit
        Port map (
            R0 => R0,
            R1 => R1,
            R2 => R2,
            R3 => R3,
            R4 => R4,
            R5 => R5,
            R6 => R6,
            R7 => R7,
            S => Decoder_MuxA,
            Q => MuxA_Adder);
    MuxB : Mux_8_to_1_4bit
        Port map (
            R0 => R0,
            R1 => R1,
            R2 => R2,
            R3 => R3,
            R4 => R4,
            R5 => R5,
            R6 => R6,
            R7 => R7,
            S => Decoder_MuxB,
            Q => MuxB_Adder);
    Register_Bank_0 : Register_Bank
        Port map (
```

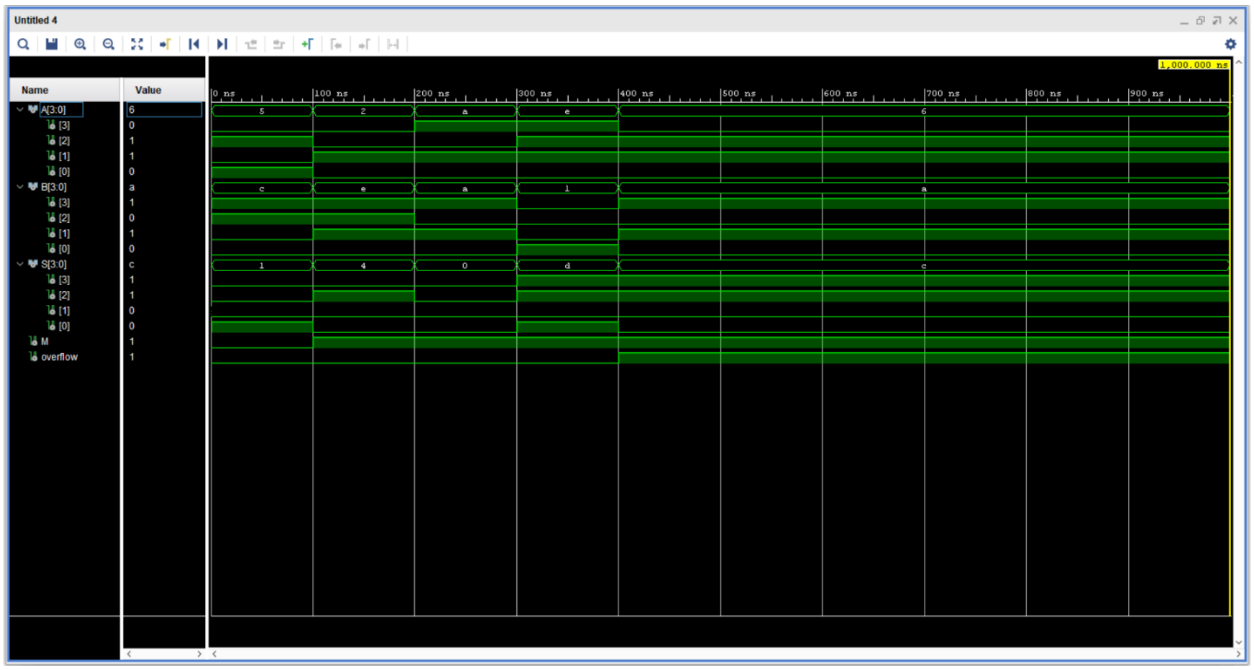
```
      D => MuxD_RegBank,
      Clk => Slw_Clk,
      I => Decoder_RegBank,
      Clr => Clr,
      R0 => R0,
      R1 => R1,
      R2 => R2,
      R3 => R3,
      R4 => R4,
      R5 => R5,
      R6 => R6,
      R7 => R7);
  MuxC : Mux_2_to_1_3bit
    Port map (
      A => Add3_MuxC,
      B => Decoder_MuxD(2 downto 0),
      S => Decoder_MuxC,
      Q => PC_MuxC);
  Adder_3bit_0 : Adder_3bit
    Port map (
      A => PC_ROM,
      Carry => Adder_Cout,
      S => Add3_MuxC);
  Instruction_Decoder_0 : Instruction_Decoder
    Port map (
      Inst => ROM_Decoder,
      Reg => MuxA_Adder,
      LSB => Decoder_MuxD,
      Reg_EN => Decoder_RegBank,
      Mux_A => Decoder_MuxA,
      Mux_B => Decoder_MuxB,
      LD => Decoder_MuxDSelc,
      Sub => Decoder_Adder,
      JMP => Decoder_MuxC);
  MuxD : Mux_2_to_1_4bit
    Port map (
      A => MuxD_Adder,
      B => Decoder_MuxD,
      S => Decoder_MuxDSelc,
      Q => MuxD_RegBank);
  Program_Counter_0 : Program_Counter
    Port map (
      D => PC_MuxC,
      Clr => Clr,
      Clk => Slw_Clk,
      Q => PC_ROM);
  ROM_0 : ROM
    Port map (
      S => PC_ROM,
      Q => ROM_Decoder);
  R <= R7;
```

```
    Zero <= (NOT MuxD_Adder(0)) AND (NOT MuxD_Adder(1)) AND (NOT MuxD_Adder(2)) AND (NOT
            MuxD_Adder(3));
end Behavioral;
```

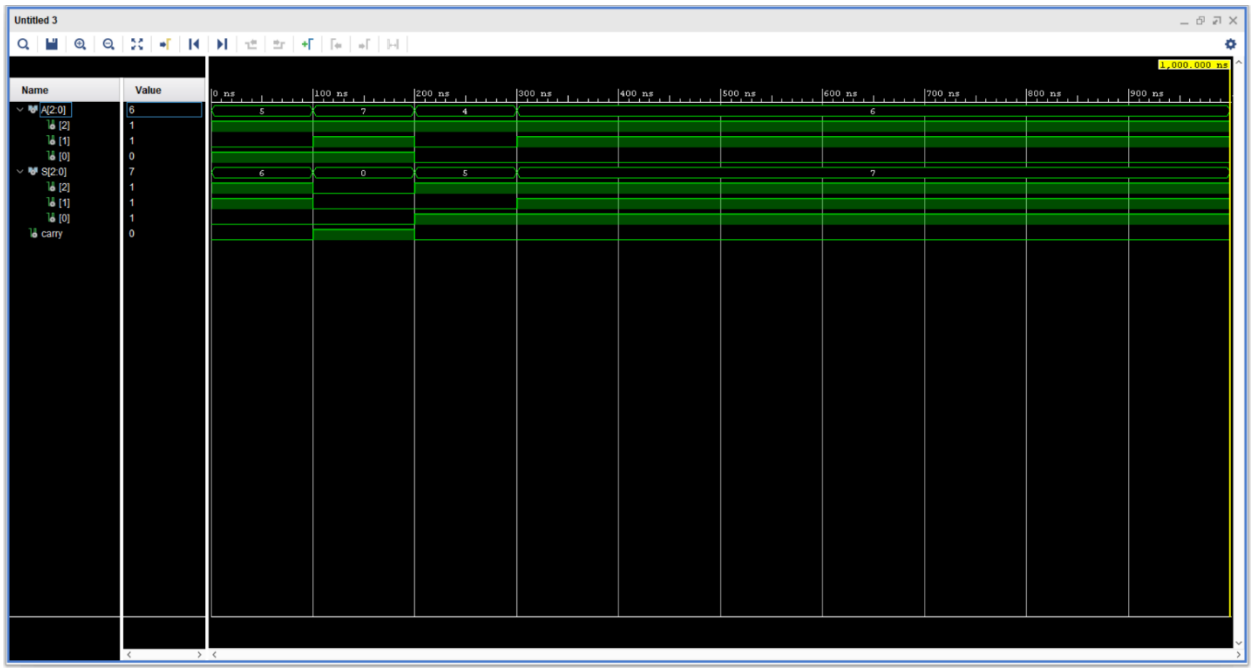
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

architecture Behavioral of TB_Nanoprocessor is
   component Nanoprocessor is
      Port (
         Clr : in std_logic;
         Clk : in std_logic;
         R : out std_logic_vector;
         Overflow : out std_logic;
         Zero : out std_logic;
         Seven_Seg : out std_logic_vector );
   end component;
   signal Clr, Clk, Overflow, Zero : std_logic;
   signal R : std_logic_vector( 3 downto 0);
   signal Seven_Seg : std_logic_vector (6 downto 0) ;

begin
   uut : Nanoprocessor
      port map (
         Clr => Clr,
         Clk => Clk,
         R => R,
         Overflow => Overflow,
         Zero => Zero,
         Seven_Seg => Seven_Seg);
   process
   begin
      Clk <= '0';
      wait for 5ns;
      Clk <= '1';
      wait for 5ns;
   end process;
   process
   begin
      Clr <= '1';
      wait for 20ns;
      Clr <= '0';
      wait;
   end process;

end Behavioral;
```

Timing Diagrams

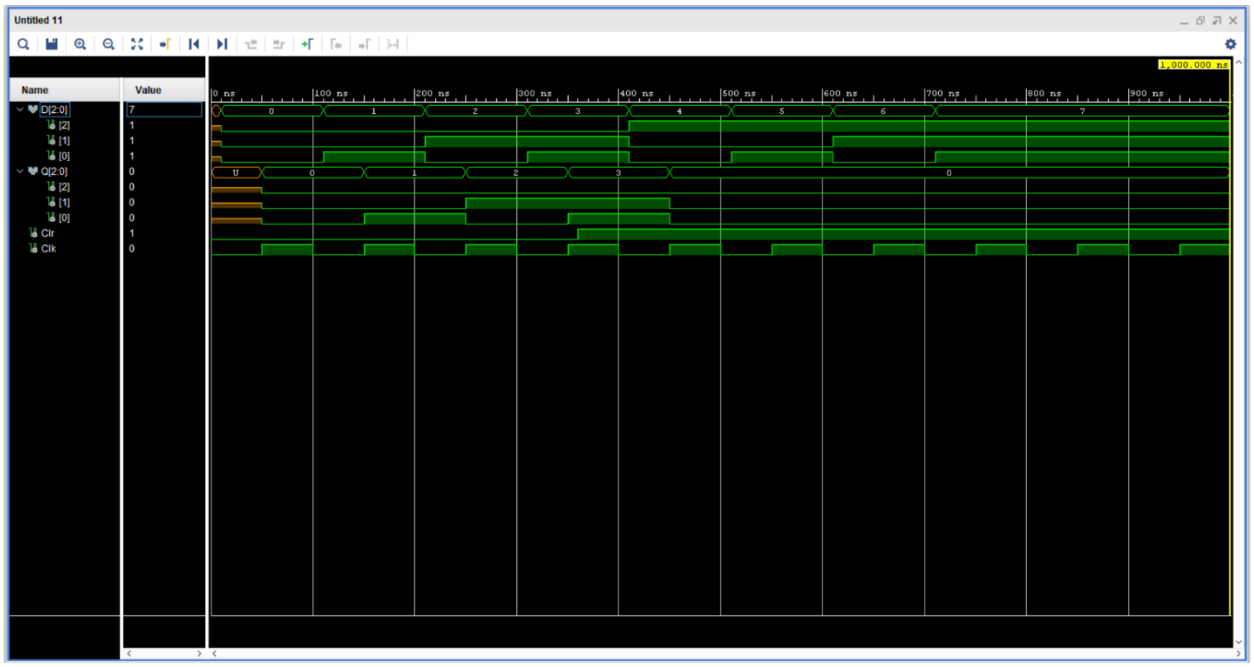
4 bit add/sub



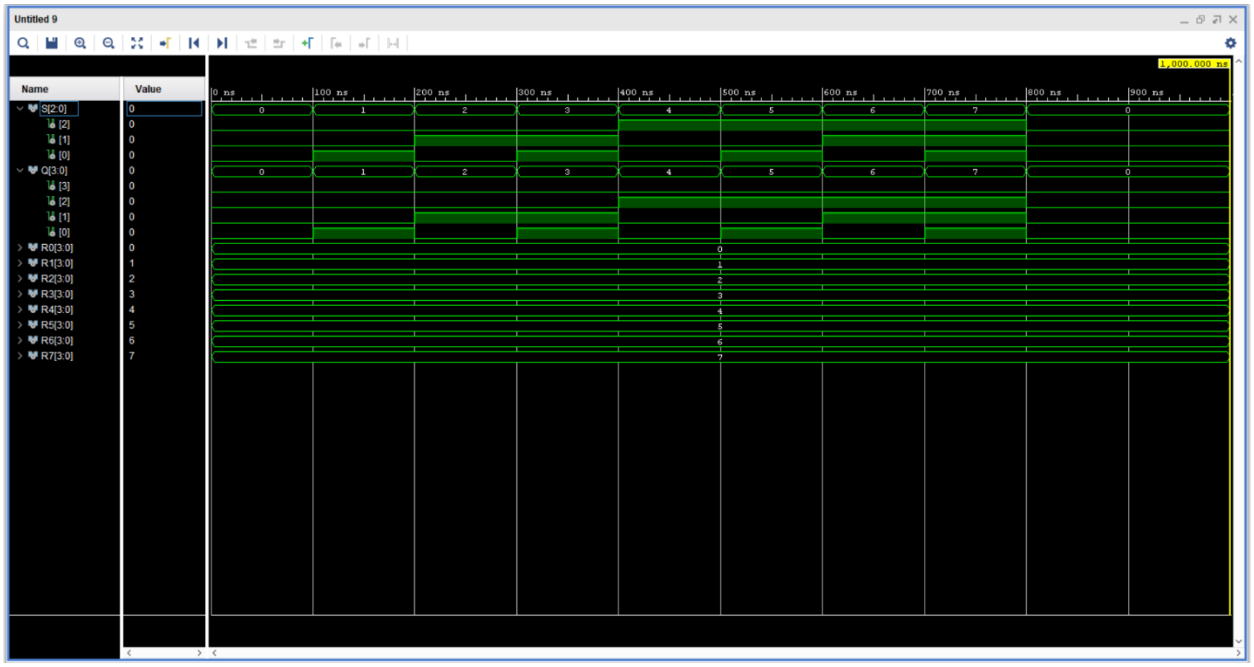
3 bit Adder



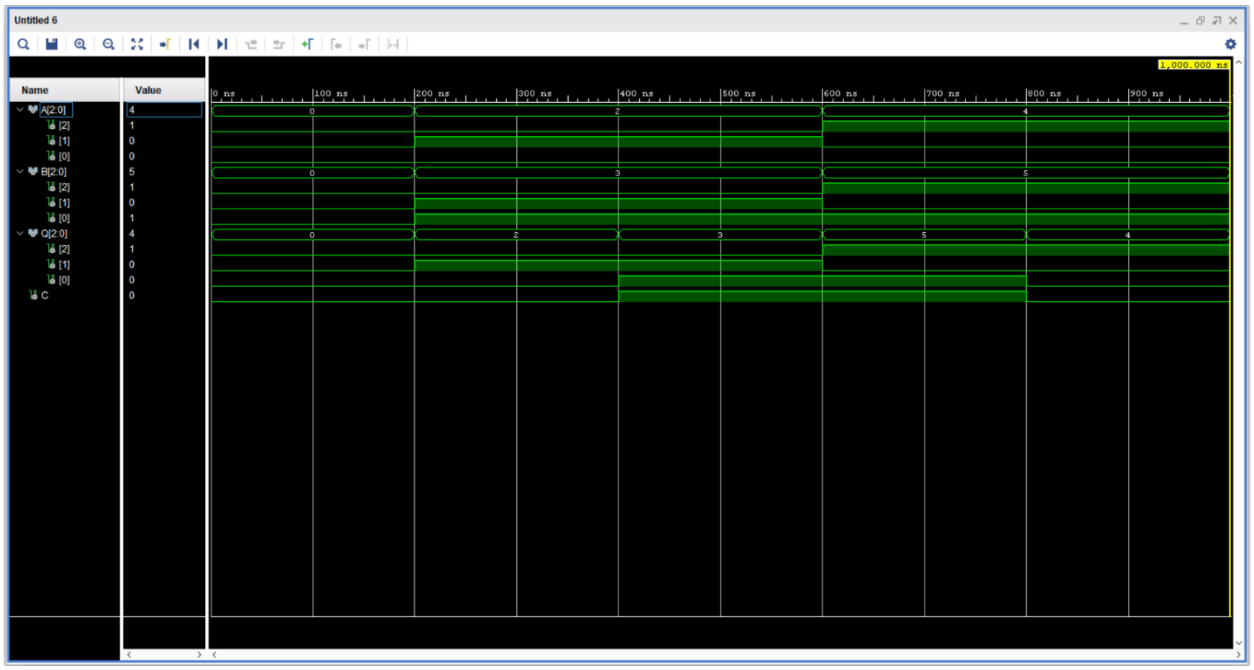
Program Counter



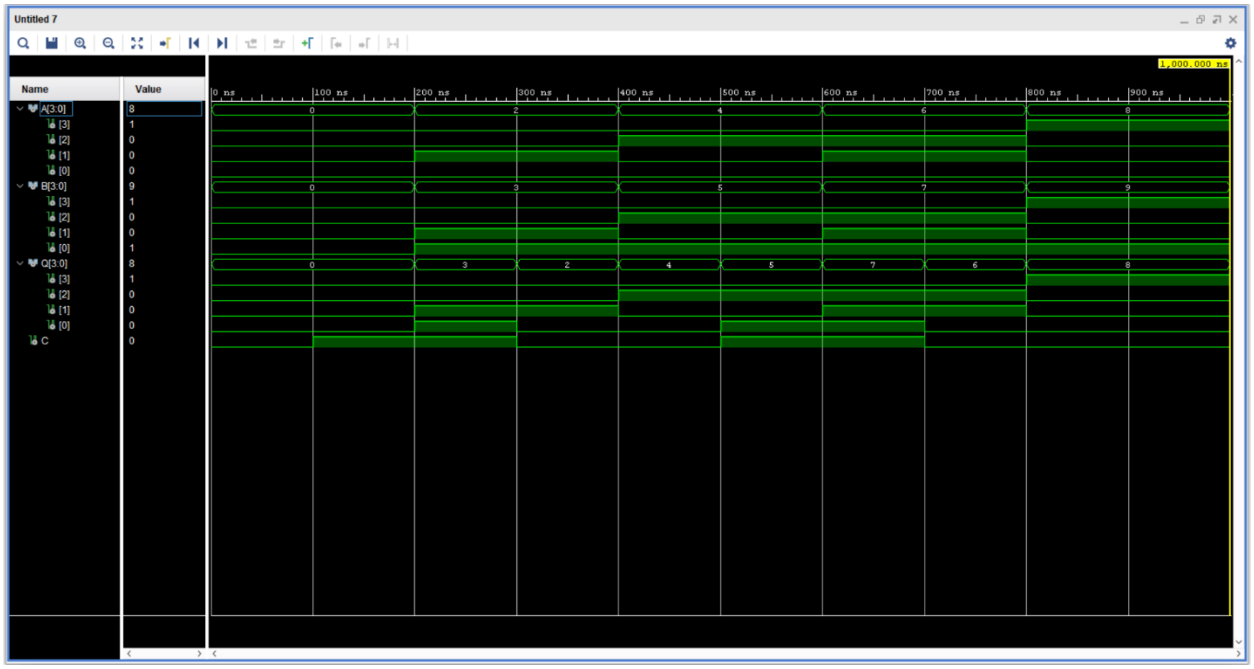
8 to 1 4 bit multiplexer



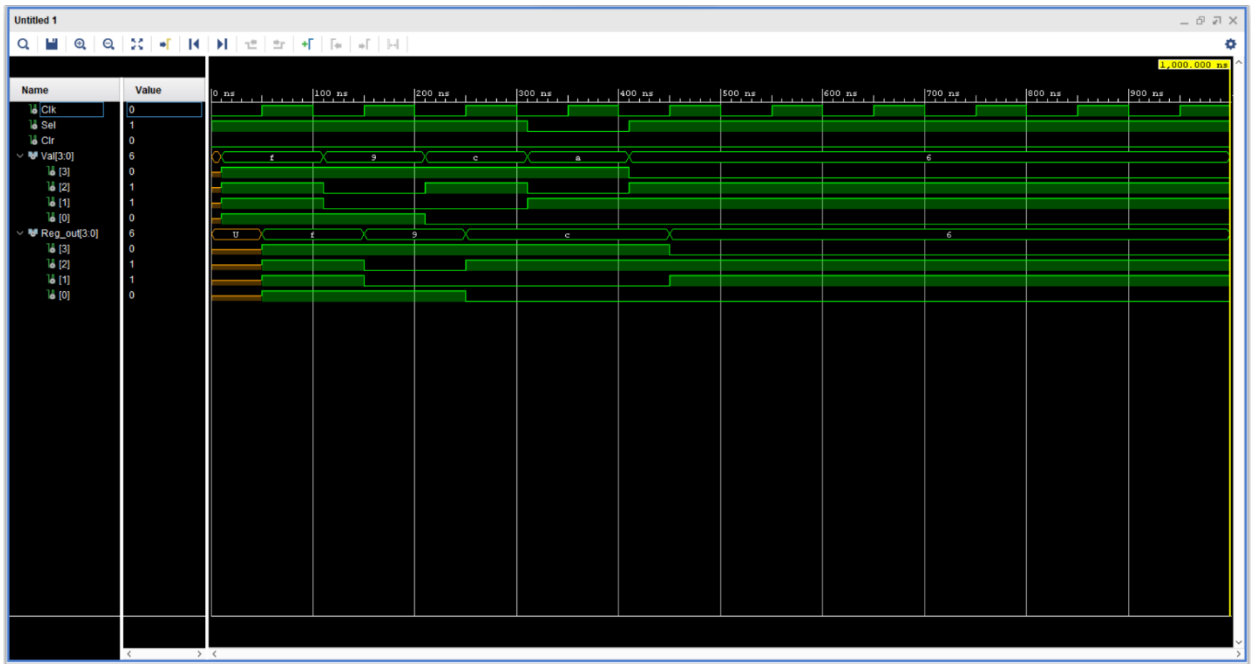
2 to 1 3bit multiplexer



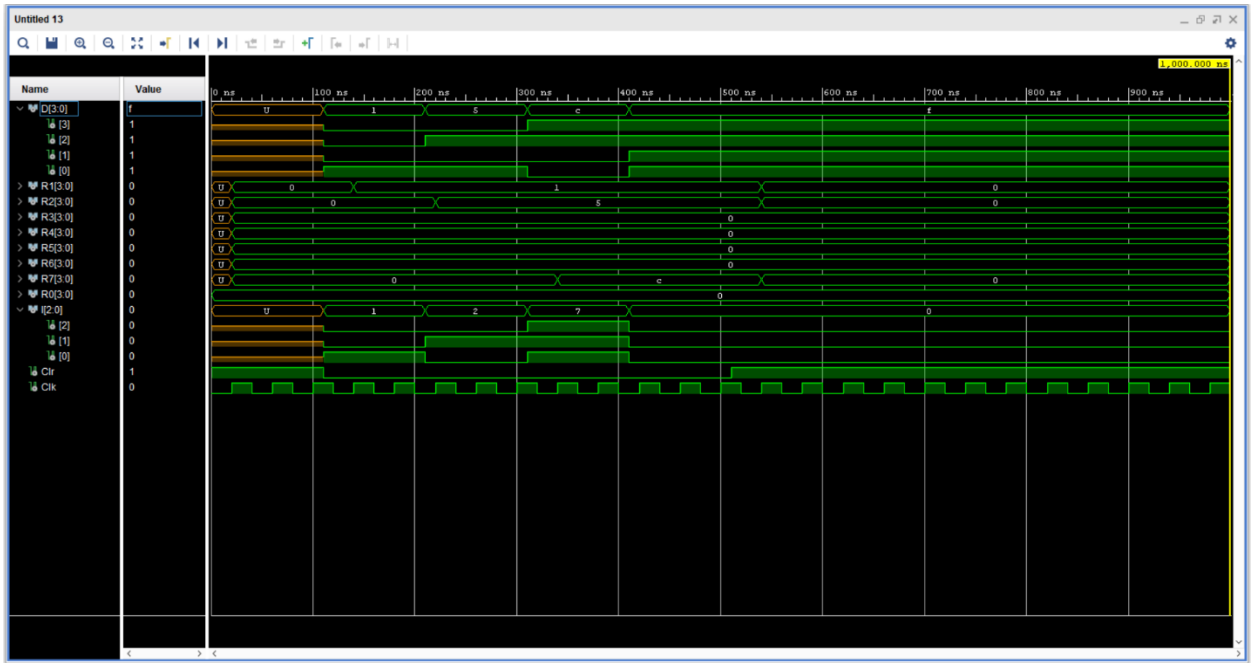
2 to 1 4bit multiplexer



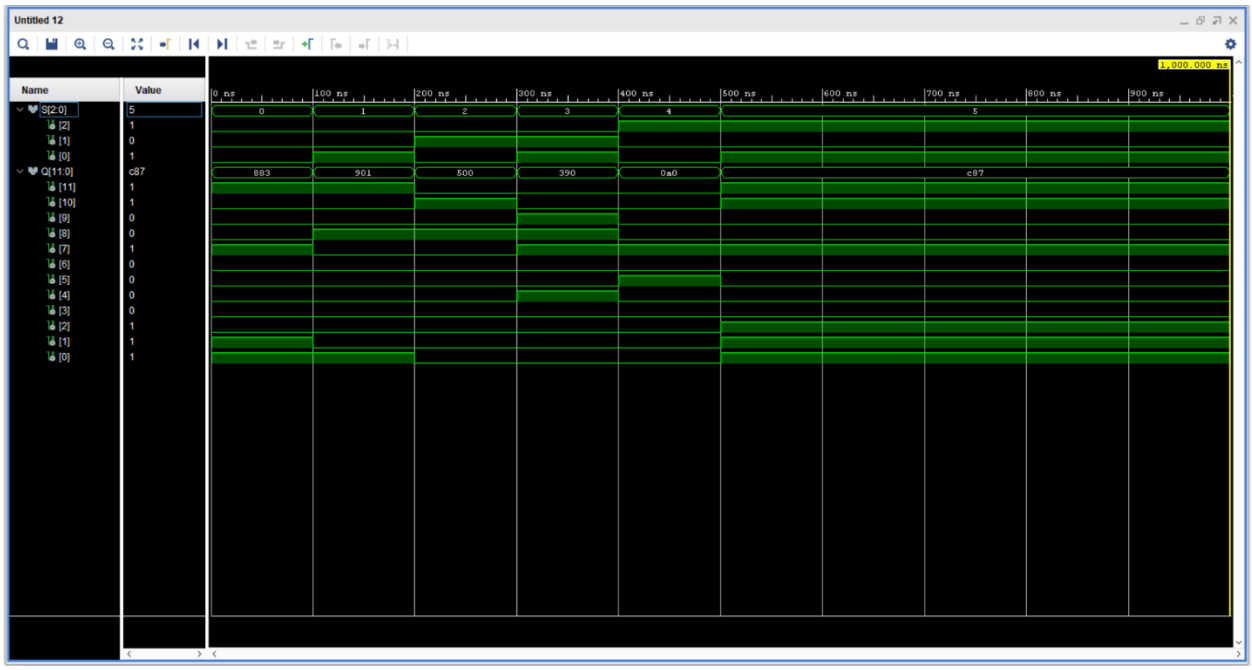
Register_4bit



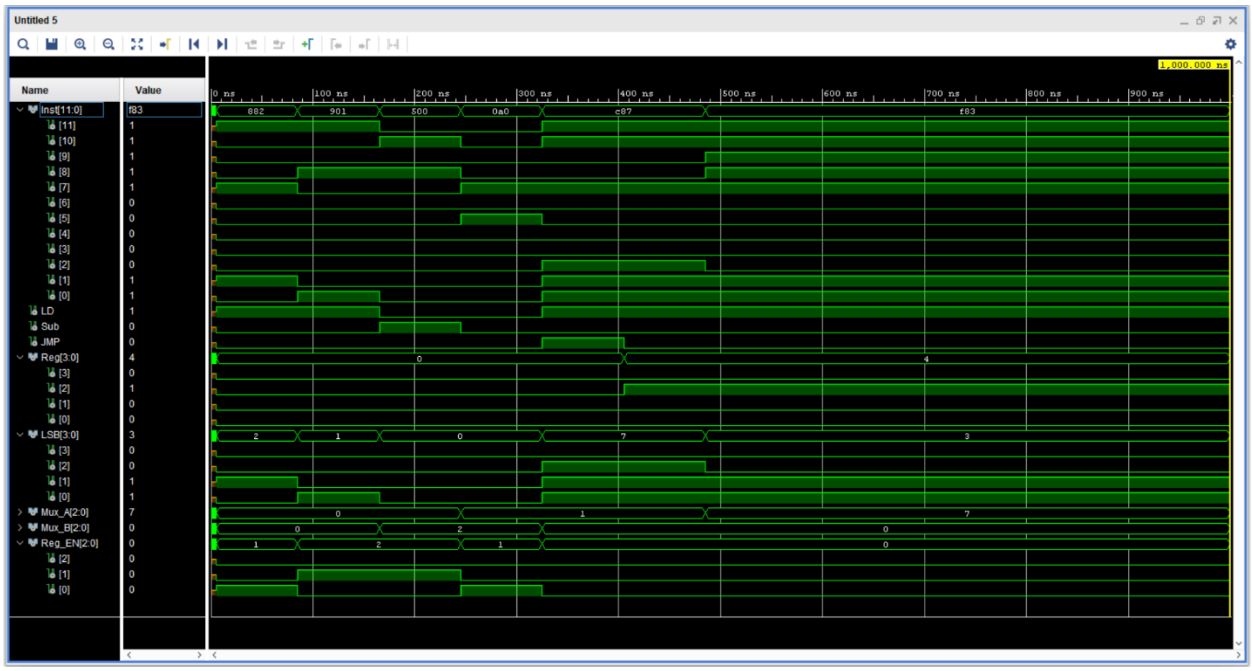
Register Bank



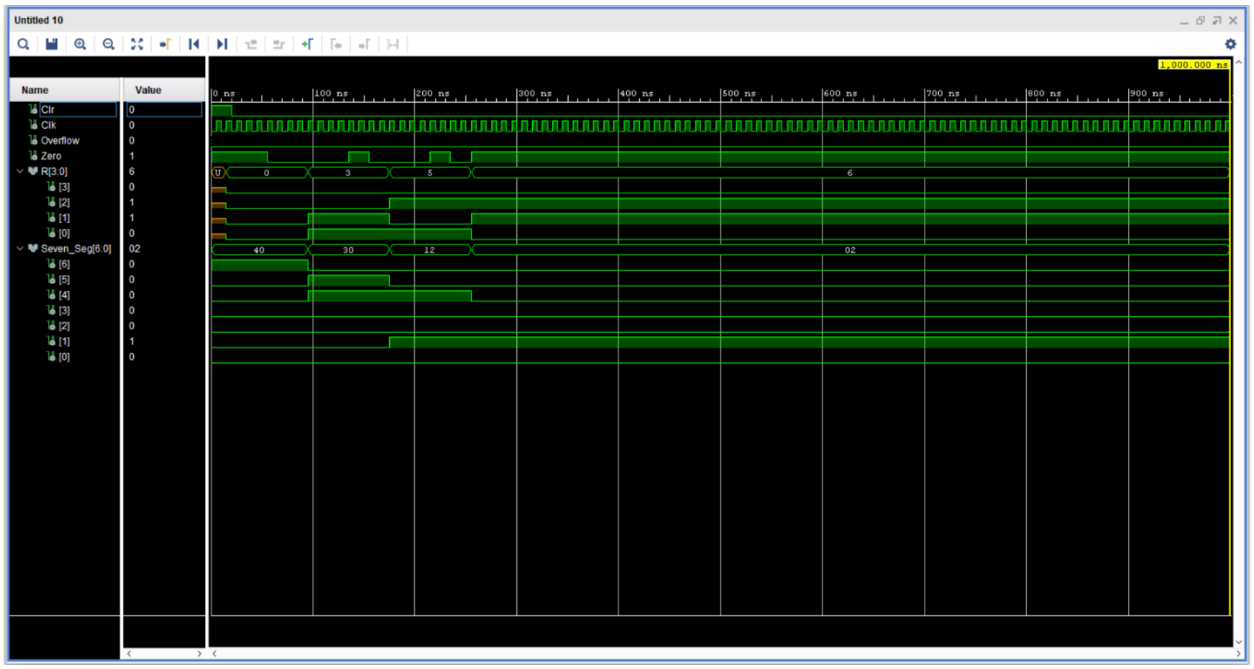
ROM



Instruction Decoder



Nano processor



Conclusion

In this lab, we learnt how to design and develop a 4-bit arithmetic unit that can add and subtract signed integers. We also learned how to decode instructions to activate the necessary components on the processor. Additionally, we focused on designing and developing k-way b-bit multiplexers and verifying their functionality through simulation and on the development board. As a team, we collaborated and shared ideas to identify and address the mistakes and drawbacks involved in microprocessor creation and simulation implementation on the board correctly.

Contribution

K.C.K.Manawathilake – Designing the design and simulation VHDL files for 8_to_1_4bit; 2_to_1_4bit ; 2_to_1_3bit multiplexers , ROM , Instruction Decoder and Slow_clock. Making the 'Nanoprocessor' VHDL file connecting all the components.

S.M.A.N.A. Manchanayake – Designing the design and simulation VHDL files for 4-bit adder/subtracter , 3-bit adder , 3-bit Program Counter , Register Bank , LUT_16-7 seg. Renaming the final files and making the lab report.

--both did the making of XDC file and checking on the board

Time spent : 2hr x 5 days + 3 hrs X 2 days + 5hr X 1 day

= approximately 20-22 hours

(almost equal time was spent by two of us as we did the work together at the same time)