Lab 9 – Nanoprocessor Design

Computer Organisation and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

Lab No: 09
Date: 17/08/2022
Group Members:

Navinda - 200452N
Umayanga -200671J
Chamil -200418R
Danush - 200712M
Madhawa -200748D

Assigned Lab Task

In this lab, our task was to build a microprocessor that is capable of executing four simple instructions. First, we built certain components, and Some of them we have already designed in previous labs.

In this lab we designed

1. And develop a 4-bit arithmetic unit that can add and subtract integers. (With help of lab 3)
2. 3-bit adder which is used to increment program counter.
3. 3-bit program counter using D flip flops.
4. A 2-way 4-bit multiplexer (Lab 4)
5. Four 8-way 4-bit mux which can take 8 inputs with 4 bits and gives a 4-bit output (Lab 4)
6. Register bank with seven 4-bit registers.
7. To store our assembly codes, we build a program ROM using 12ROM16x1s.
8. Instruction decoder: activate necessary components based on the instructions we wish to execute.
9. Used 3, 4, and 12-bit buses to connect components.
10. After creating all these components, we created a Nano processor by connecting these components properly.

And then verified functionality using simulation and on the Basys3 board.

VHDL Codes

4-bit Add/Subtract 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 RCA_ADD_SUB is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           sel : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_out : out STD_LOGIC;
           zero : out STD_LOGIC);
end RCA_ADD_SUB;

architecture Behavioral of RCA_ADD_SUB is
component FA_ADD_SUB
 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;
  SIGNAL B_0 : STD_LOGIC_VECTOR (3 DOWNTO 0);
begin
FA_0 : FA_ADD_SUB
 port map (
 A => A0,
 B => B_0(0),
 C_in => sel,
 S => FA0_S,
 C_out => FA0_C);

FA_1 : FA_ADD_SUB
 port map (
 A => A1,
 B => B_0(1),
 C_in => FA0_C,
 S => FA1_S,
 C_out => FA1_C);


 FA_2 :FA_ADD_SUB
  port map (
  A => A2,
  B => B_0(2),
  C_in => FA1_C,
  S => FA2_S,
  C_out => FA2_C);


FA_3 : FA_ADD_SUB
  port map (
  A => A3,
  B => B_0(3),
  C_in => FA2_C,
  S => FA3_S,
  C_out => FA3_C);
S0 <= FA0_S;
S1 <= FA1_S;
S2 <= FA2_S;
```

```
S3 <= FA3_S;
C_out <= FA2_C xor FA3_C;
B_0(0) <=  B0 xor sel;
B_0(1) <=  B1 xor sel;
B_0(2) <=  B2 xor sel;
B_0(3) <=  B3 xor sel;
zero <= not(FA0_S) and not(FA1_S) and not(FA2_S) and not(FA3_S);
end Behavioral;
```

3-bit Adder

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


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 RCA_3_bit is
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           C_out : out STD_LOGIC
       );
end RCA_3_bit;

architecture Behavioral of RCA_3_bit is
component FA
 port (
```

```
 A: in std_logic;
 B: in std_logic;
 C_in: in std_logic;
 S: out std_logic;
 C_out: out std_logic);
 end component;

  SIGNAL FA0_C,  FA1_C, FA2_C : std_logic;
  SIGNAL B_0 : STD_LOGIC_VECTOR (3 DOWNTO 0);
begin
FA_0 : FA
 port map (
 A => A0,
 B => '1',
 C_in => C_in,
 S => S0,
 C_out => FA0_C);

FA_1 : FA
 port map (
 A => A1,
 B => '0' ,
 C_in => FA0_C,
 S => S1,
 C_out => FA1_C);


 FA_2 : FA
  port map (
  A => A2,
  B => '0',
  C_in => FA1_C,
  S => S2,
  C_out => C_out);




end Behavioral;
```

Program Counter

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

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

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

entity Program_Counter is
    Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Res : 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
DF0 : D_FF
PORT MAP(
           D => D(0),
           Res => Res,
           Clk => Clk,
           Q => Q(0));

 DF1 : D_FF
       PORT MAP(
              D => D(1),
              Res => Res,
              Clk => Clk,
```

```
              Q => Q(1));
DF2 : D_FF
      PORT MAP(
             D => D(2),
             Res => Res,
             Clk => Clk,
               Q => Q(2));
end Behavioral;
```

2 way Multiplexer (used for 2 way 3 bit and 4 bit Muxes)

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

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

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

entity MUX_2_to_1 is
    Port ( D0 : in STD_LOGIC;
           D1 : in STD_LOGIC;
           EN : in STD_LOGIC;
           Y : out STD_LOGIC);
end MUX_2_to_1;

architecture Behavioral of MUX_2_to_1 is

begin
    Y <= (not(EN) and D0) or (EN and D1);
```

```
end Behavioral;
```

8-way 4-bit Multiplexer

```
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 way_8_mux is
    Port ( Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 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));
end way_8_mux;

architecture Behavioral of way_8_mux is
component MUX_8_to_1
    port(  D : in STD_LOGIC_VECTOR (7 downto 0);
           S : in STD_LOGIC_VECTOR (2 downto 0);
           EN : in STD_LOGIC;
```

```
           Y : out STD_LOGIC);
 end component;
signal en : STD_LOGIC;
begin

MUX1: MUX_8_to_1
port map(
        D(0) => R0(0),
        D(1) => R1(0),
        D(2) => R2(0),
        D(3) => R3(0),
        D(4) => R4(0),
        D(5) => R5(0),
        D(6) => R6(0),
        D(7) => R7(0),
        S => Sel,
        EN => en,
        Y => Y(0));


MUX2: MUX_8_to_1
        port map(
            D(0) => R0(1),
            D(1) => R1(1),
            D(2) => R2(1),
            D(3) => R3(1),
            D(4) => R4(1),
            D(5) => R5(1),
            D(6) => R6(1),
            D(7) => R7(1),
            S => Sel,
            EN => en,
            Y => Y(1));
MUX3: MUX_8_to_1
        port map(
            D(0) => R0(2),
            D(1) => R1(2),
            D(2) => R2(2),
            D(3) => R3(2),
            D(4) => R4(2),
            D(5) => R5(2),
            D(6) => R6(2),
```

```
            D(7) => R7(2),
            S => Sel,
            EN => en,
            Y => Y(2));
MUX4: MUX_8_to_1
        port map(
            D(0) => R0(3),
            D(1) => R1(3),
            D(2) => R2(3),
            D(3) => R3(3),
            D(4) => R4(3),
            D(5) => R5(3),
            D(6) => R6(3),
            D(7) => R7(3),
            S => Sel,
            EN => en,
            Y => Y(3));
en <= '1';
end Behavioral;
```

Register Bank

```
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 Regsiter_Bank is
    Port ( R_en : in STD_LOGIC_VECTOR (2 downto 0);
            res : in STD_LOGIC;
           Clk_r : in STD_LOGIC;
           R_in1 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in2 : in STD_LOGIC_VECTOR(3 downto 0);
```

```
           R_in3 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in4 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in5 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in6 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in7 : in STD_LOGIC_VECTOR(3 downto 0);
           Bus0 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus1 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus2 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus3 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus4 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus5 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus6 : out STD_LOGIC_VECTOR (3 downto 0);
           Bus7 : out STD_LOGIC_VECTOR (3 downto 0));
end Regsiter_Bank;

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

component Reg
 Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
          En : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Res : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal y : STD_LOGIC_VECTOR (7 downto 0);
begin
three_to_eight_decoder : Decoder_3_to_8
port map(
          I => R_en,
          EN => '1',
          Y => y

);

reg0 : Reg
port map (
          D  => "0000",
```

```
          En => '1',
          Res => res,
          Clk => Clk_r,
          Q => Bus0
);
reg1 : Reg
port map (
          D  => R_in1,
          En => y(1),
          Res => res,
          Clk => Clk_r,
          Q => Bus1
);
reg2 : Reg
port map (
          D  => R_in2,
          En => y(2),
          Res => res,
          Clk => Clk_r,
          Q => Bus2
);
reg3 : Reg
port map (
          D  => R_in3,
          En => y(3),
          Res => res,
          Clk => Clk_r,
          Q => Bus3
);
reg4 : Reg
port map (
          D  => R_in4,
          En => y(4),
          Res => res,
          Clk => Clk_r,
          Q => Bus4
);
reg5 : Reg
port map (
          D  => R_in5,
          En => y(5),
          Res => res,
```

```
          Clk => Clk_r,
          Q => Bus5
);
reg6 : Reg
port map (
          D  => R_in6,
          En => y(6),
          Res => res,
          Clk => Clk_r,
          Q => Bus6
);
reg7 : Reg
port map (
          D  => R_in7,
          En => y(7),
          Res => res,
          Clk => Clk_r,
          Q => Bus7
```

Program ROM

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

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

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

entity ROM is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : 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 nanoprocessor_ROM : rom_type := (
    "000000000000",
    "100010000011",
    "100100000010",
    "010100000000",
    "000010100000",
    "001110010000",
    "111110000111",
    "110000000001"

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

end Behavioral;
```

Instruction Decoder

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

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

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

entity Ins_Decoder is
    Port ( ROM : in STD_LOGIC_VECTOR (11 downto 0);
            Reg_value : in STD_LOGIC_VECTOR(3 downto 0);
           Reg_en : out STD_LOGIC_VECTOR (2 downto 0);
           L_Sel : out STD_LOGIC;
           In_Val : out STD_LOGIC_VECTOR (3 downto 0);
           Reg_sel0 : out STD_LOGIC_VECTOR (2 downto 0);
           Reg_sel1 : out STD_LOGIC_VECTOR (2 downto 0);
```

```
            AU_sel : out STD_LOGIC;
            JMP : out STD_LOGIC;
            JMP_ADD : out STD_LOGIC_VECTOR (2 downto 0));
end Ins_Decoder;

architecture Behavioral of Ins_Decoder is
signal regA,regB : STD_LOGIC_VECTOR(2 downto 0);
begin
    regA <= ROM(9 downto 7);
    regB <= ROM(6 downto 4);
    --moving to registers
    L_Sel <= ROM(11) and not(ROM(10));
    In_Val <=  ROM(3 downto 0);
    Reg_en <= regA;

    --adding
    AU_Sel <= not(ROM(11)) and ROM(10);
    Reg_Sel0 <= regA;
    Reg_Sel1 <= regB;

-- jump if zero
    JMP <= (ROM(11) and ROM(10)) and( not(Reg_value(0)) and
not(Reg_value(1)) and not(Reg_value(2)) and not(Reg_value(3)));
    JMP_ADD <= ROM(2 downto 0);

end Behavioral;
```

Slow Clock

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

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

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

```
entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           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 = 2) then
                clk_status <= NOT( clk_status);
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

Seven Segment

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

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

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity Seven_Segment is
Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0)  );
end Seven_Segment;
architecture Behavioral of Seven_Segment is
```

```
component LUT_16_7
port ( address : in STD_LOGIC_VECTOR (3 downto 0);
           data : out STD_LOGIC_VECTOR (6 downto 0)

           );
end component;
begin
LUT0 : LUT_16_7
    port map(
        address => A,
        data => S_7Seg
    );
end Behavioral;
```

Nano Processor

```
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_S

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

entity Nanoprocessor is
    Port ( CLK : in STD_LOGIC;
           RES : in STD_LOGIC;
           OVRFLW : out STD_LOGIC;
           ZERO : out STD_LOGIC;
           SEG7 : out STD_LOGIC_VECTOR (3 downto 0));
end Nanoprocessor;

architecture Behavioral of Nanoprocessor is
component Ins_Decoder
 Port ( ROM : in STD_LOGIC_VECTOR (11 downto 0);
```

```
           Reg_value : in STD_LOGIC_VECTOR(3 downto 0);
          Reg_en : out STD_LOGIC_VECTOR (2 downto 0);
          L_Sel : out STD_LOGIC;
          In_Val : out STD_LOGIC_VECTOR (3 downto 0);
          Reg_sel0 : out STD_LOGIC_VECTOR (2 downto 0);
          Reg_sel1 : out STD_LOGIC_VECTOR (2 downto 0);
          AU_sel : out STD_LOGIC;
          JMP : out STD_LOGIC;
          JMP_ADD : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component ROM
Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component MUX_2_to_1
    Port ( D0 : in STD_LOGIC;
           D1 : in STD_LOGIC;
           EN : in STD_LOGIC;
           Y : out STD_LOGIC);
end component;

component RCA
Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           C_out : out STD_LOGIC
       );
end component;
component Program_Counter
Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
       Clk : in STD_LOGIC;
       Res : in STD_LOGIC;
       Q : out STD_LOGIC_VECTOR (2 downto 0));
end component;

---green over---yellow begin--
```

```
component RCA_ADD_SUB
Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           sel : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_out : out STD_LOGIC;
           zero : out STD_LOGIC);
end component;
component way_8_mux
Port ( Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 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));
end component;

component Regsiter_Bank
Port (     R_en : in STD_LOGIC_VECTOR (2 downto 0);
           Clk_r : in STD_LOGIC;
           R_in0 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in1 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in2 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in3 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in4 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in5 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in6 : in STD_LOGIC_VECTOR(3 downto 0);
           R_in7 : in STD_LOGIC_VECTOR(3 downto 0);
```

```
            Bus0 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus1 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus2 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus3 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus4 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus5 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus6 : out STD_LOGIC_VECTOR (3 downto 0);
            Bus7 : out STD_LOGIC_VECTOR (3 downto 0));

end component;
signal data0 : STD_LOGIC_VECTOR(11 downto 0);
signal jmp,cout,lsel,sel : STD_LOGIC;
signal a0,a1,a2,a3,s0,s1,s2,y0,y1,y2,o0,o1,o2,o3 : STD_LOGIC;
signal jmp_add,q,regsel0,regsel1,ren:  STD_LOGIC_VECTOR (2 downto 0);
signal i0,i1,inval,r0,r1,r2,r3,r4,r5,r6,r7:  STD_LOGIC_VECTOR (3
downto 0);

begin
Instruction_decoder : Ins_Decoder
PORT MAP (
          ROM  => data0,
          Reg_value => i0,
          Reg_en  => ren,
          L_Sel => lsel,
          In_Val => inval,
          Reg_sel0  => regsel0,
          Reg_sel1 => regsel1,
          AU_sel => sel,
          JMP => jmp,
          JMP_ADD  => jmp_add
);

ROMprocessor : ROM
PORT MAP(
    data => data0,
    address=> q
);

MUX2_0: MUX_2_to_1
port map(
    D0 => s0,
    D1 => jmp_add(0),
```

```
    EN => jmp,
    Y => y0
);
MUX2_1: MUX_2_to_1
port map(
        D0 =>s1,
        D1 => jmp_add(1),
        EN => jmp,
        Y => y1
);
MUX2_2: MUX_2_to_1
port map(
   D0 =>s2,
   D1 => jmp_add(2),
   EN => jmp,
   Y => y2
);

RCA_3bit : RCA
port map(
   A0 => q(0),
   A1 => q(1),
   A2 => q(2),
   C_in => '0',
   S0 => s0,
   S1 => s1,
   S2 =>s2 ,
   C_out => cout

);

PC : Program_Counter
port map(
    D(0) => y0,
    D(1) => y1,
    D(2) => y2,
   Clk => CLK,
   Res => RES,
   Q => q
);
--yellow
```

```
RCA_for_adding_subtracting : RCA_ADD_SUB
port map(
           A0 => i0(0),
           A1 =>i0(1),
           A2 =>i0(2),
           A3 =>i0(3),
           B0 => i1(0),
           B1 =>i1(1),
           C_in => '0',
           sel =>sel,
           B2 =>i1(2),
           B3 =>i1(3),
           S0 => a0 ,
           S1 => a1,
           S2 => a2,
           S3 => a3,
           C_out => OVRFLW,
           zero => ZERO

);

MUX_0 : way_8_mux
port map(
           Sel => regsel0 ,
           Y =>i0 ,
           R0 => r0,
           R1 => r1,
           R2 => r2,
           R3 => r3,
           R4 => r4,
           R5 => r5,
           R6 => r6,
           R7 => r7

);

MUX_1 : way_8_mux
port map(
           Sel => regsel1 ,
           Y => i1,
           R0 => r0,
          R1 => r1,
```

```
          R2 => r2,
          R3 => r3,
          R4 => r4,
          R5 => r5,
          R6 => r6,
          R7 => r7

);
-- orange part
Ins_Dec_Mux0 : Mux_2_to_1
port map (
    D0 => a0,
   D1 => inval(0),
   EN => jmp,
   Y => o0
);
Ins_Dec_Mux1 : Mux_2_to_1
port map (
    D0 => a1,
   D1 => inval(1),
   EN => lsel,
   Y => o1
);
Ins_Dec_Mux2 : Mux_2_to_1
port map (
    D0 => a2,
   D1 => inval(2),
   EN => lsel,
   Y => o2
);
Ins_Dec_Mux3 : Mux_2_to_1
port map (
    D0 => a3,
   D1 => inval(3),
   EN =>  lsel,
   Y => o3
);

reg_bank : Regsiter_Bank
PORT MAP(
           R_en => ren,
           Clk_r =>  CLK,
```

```
            R_in0(0) => o0,
            R_in0(1) =>o1,
            R_in0(2) =>o2,
            R_in0(3) =>o3,
            R_in1(0) => o0,
            R_in1(1) =>o1,
            R_in1(2) =>o2,
            R_in1(3) =>o3,
            R_in2(0) => o0,
            R_in2(1) =>o1,
            R_in2(2) =>o2,
            R_in2(3) =>o3,
            R_in3(0) => o0,
            R_in3(1) =>o1,
            R_in3(2) =>o2,
            R_in3(3) =>o3,
            R_in4(0) => o0,
           R_in4(1) =>o1,
           R_in4(2) =>o2,
           R_in4(3) =>o3,
           R_in5(0) => o0,
           R_in5(1) =>o1,
           R_in5(2) =>o2,
           R_in5(3) =>o3,
           R_in6(0) => o0,
           R_in6(1) =>o1,
           R_in6(2) =>o2,
           R_in6(3) =>o3,
           R_in7(0) => o0,
           R_in7(1) =>o1,
            R_in7(2) =>o2,
           R_in7(3) => o3,
            Bus0 => r0,
            Bus1 =>r1,
            Bus2 =>r2,
            Bus3 =>r3,
            Bus4 => r4,
            Bus5 =>r5,
            Bus6 =>r6,
            Bus7 =>r7
);
```

```
end Behavioral;
```

Full Adder (imported)

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

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

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

entity FA_ADD_SUB 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_ADD_SUB;

architecture Behavioral of FA_ADD_SUB is
component HA_ADD_SUB
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_ADD_SUB
    PORT MAP (
    A => A,
    B => B,
    C => HA0_C,
    S => HA0_S
```

```
    );

HA_1 : HA_ADD_SUB
    PORT MAP (
   A => HA0_S,
       B => C_in,
       C => HA1_C,
       S => HA1_S

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

Half Adder (imported)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity HA_ADD_SUB is
    Port ( A : in STD_LOGIC;
    B : in STD_LOGIC;
    S : out STD_LOGIC;
    C : out STD_LOGIC);
end HA_ADD_SUB;
architecture Behavioral of HA_ADD_SUB is
    begin
        S <= A XOR B;
        C <= A AND B;
end Behavioral;
```

D Flip Flop (imported)

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

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

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

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

architecture Behavioral of D_FF is

begin
    process (Clk) begin
        if (rising_edge(Clk)) then
            if Res = '1' then
                Q <= '0';

            else
                Q<= D;

              end if;
           end if;
        end process;

end Behavioral;
```

Registers (imported)

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

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

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

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

architecture Behavioral of Reg is

begin
    process(Clk)
    begin
        if(rising_edge(Clk)) then
            if (res = '1') then
                Q <="0000";
            else
            if (En = '1') then
                Q <= D;
            end if;
            end if;
        end if;
    end process;

end Behavioral;
```

3 to 8 Decoder (Imported)

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

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

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

entity Decoder_3_to_8 is
    Port ( I : in STD_LOGIC_VECTOR (2 downto 0);
           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 (1 downto 0);
           EN : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR (3 downto 0));
end component;
signal EN0,EN1,I0,I1, I2 : STD_LOGIC;
begin
Decoder_2_to_4_0 : Decoder_2_to_4
    PORT MAP (
        I(0) => I0,
        I(1) => I1,
        EN => EN0,
        Y => Y(3 downto 0)
    );
Decoder_2_to_4_1 : Decoder_2_to_4
        PORT MAP (
        I(0) => I0,
        I(1) => I1,
        EN => EN1,
```

```
        Y => Y(7 downto 4)
    );



    I0 <= I(0);
    I1 <= I(1);
    EN0 <= NOT(I(2)) AND EN;
    EN1 <= I(2) AND EN;


end Behavioral;
```

2 to 4 Decoder (Imported)

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

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

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

entity Decoder_2_to_4 is
    Port ( I : in STD_LOGIC_VECTOR (1 downto 0);
           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) <= EN AND NOT(I(0)) AND NOT(I(1));
Y(1) <= EN AND I(0) AND NOT(I(1));
```

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


end Behavioral;
```

Simulation Files

Nanoprocessor (Test Bench)

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


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

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

architecture Behavioral of TB_Nanoprocessor is
component Nanoprocessor is
    Port ( CLK : in STD_LOGIC;
           RES : in STD_LOGIC;
           OVRFLW : out STD_LOGIC;
           ZERO : out STD_LOGIC;
           Anode : out STD_LOGIC_VECTOR  (3 downto 0);
           SEG7 : out STD_LOGIC_VECTOR (6 downto 0));
end component;
```

```
        signal    CLK :  STD_LOGIC := '0';
  signal         RES :  STD_LOGIC;
          signal OVRFLW :  STD_LOGIC;
     signal      ZERO :  STD_LOGIC;
   signal        Anode :  STD_LOGIC_VECTOR  (3 downto 0);
   signal SEG7 :  STD_LOGIC_VECTOR (6 downto 0);

--200452N Navinda Perera 11 0000 1111 0000 0100

begin
UUT : Nanoprocessor
port map(CLK => CLK,
           RES => RES,
           OVRFLW => OVRFLW,
           ZERO => ZERO,
           Anode => Anode,
           SEG7 => SEG7

);
process
begin
wait for 10ns;
CLK <= NOT(CLK);
end process;

process
begin
RES <= '1';
wait for 90ns;
RES <= '0';
wait;
end process;
end Behavioral;
```

PC (Test Bench)

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

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

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

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

architecture Behavioral of TB_PC is
component Program_Counter is
    Port ( D : in STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Res : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end component;

  signal   d:  STD_LOGIC_VECTOR (2 downto 0);
  signal   clk : STD_LOGIC :='0';
  signal   res : STD_LOGIC :='0';
  signal q :  STD_LOGIC_VECTOR (2 downto 0);


begin




UUT : Program_Counter
port map(
       D => d,
        Clk => clk,
        Res => res,
        Q => q
);
process
begin
wait for 10ns;
clk <= NOT(clk);
end process;
```

```
process
begin
D <= "000";
wait for 20ns;
D <= "001";
wait for 20ns;
D <= "010";
wait for 20ns;
D <= "011";
wait for 20ns;
D <= "100";
wait for 20ns;
D <= "101";
wait for 20ns;
D <= "110";
wait for 20ns;
D <= "111";
wait;
end process;
end Behavioral;
```

ROM (Test Bench)

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

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

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

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

```
architecture Behavioral of TB_ROM is
component ROM is
    Port ( address : in STD_LOGIC_VECTOR (2 downto 0);
           data : out STD_LOGIC_VECTOR (11 downto 0));
end component;

  signal   ad :  STD_LOGIC_VECTOR (2 downto 0);
  signal   dt : STD_LOGIC_VECTOR (11 downto 0);

begin
UUT : ROM
port map(address => ad,
        data => dt
);

process
begin
ad <= "000";
wait for 20ns;
ad <= "001";
wait for 20ns;
ad <= "010";
wait for 20ns;
ad <= "011";
wait for 20ns;
ad <="100";
wait for 20ns;
ad<= "101";
wait for 20ns;
ad <= "110";
wait for 20ns;
ad <="111";
wait;
end process;
end Behavioral;
```

4-bit Add/Subtract Unit (Test Bench)

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

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

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

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

architecture Behavioral of TB_RCA is
component RCA
 Port ( A0 : in STD_LOGIC;
          A1 : in STD_LOGIC;
          A2 : in STD_LOGIC;
          A3 : in STD_LOGIC;
          B0 : in STD_LOGIC;
          B1 : in STD_LOGIC;
          B2 : in STD_LOGIC;
          B3 : in STD_LOGIC;
          C_in : in STD_LOGIC;
          sel : in STD_LOGIC;
          S0 : out STD_LOGIC;
          S1 : out STD_LOGIC;
          S2 : out STD_LOGIC;
          S3 : out STD_LOGIC;
          zero : out STD_LOGIC;
          C_out : out STD_LOGIC);
```

```
end component;
signal a0,a1,a2,a3,b0,b1,b2,b3,c_in,s0,s1,s2,s3,c_out,sel,zero :
STD_LOGIC;
begin
UUT: RCA
PORT MAP(
     A0 => a0,
     A1 => a1,
     A2 => a2,
     A3 => a3,
     B0 => b0,
     B1 => b1,
     B2 => b2,
     B3 => b3,
     C_in => c_in,
     S0 => s0,
     S1 => s1,
     S2 => s2,
     S3 => s3,
     C_out => c_out,
     sel => sel,
     zero => zero



);
process
begin
--index number 200452N
-- index number in binary 11 0000 1111 0000 0100


    sel <= '0';
-- 0000+0100 = 0100
      A0 <= '0';
     A1 <=  '0';
     A2 <= '0';
     A3 <= '0';
     B0 <=  '0';
     B1 <=  '0';
     B2 <= '1';
     B3 <= '0';
```

```
    c_in <= '0';

      wait for 100ns;

 -- 0000 + 1111 = 1111
      sel <= '1';
      A0 <= '0';
      A1 <=  '0';
      A2 <= '0';
      A3 <= '0';
      B0 <=  '1';
      B1 <=  '1';
      B2 <= '1';
      B3 <= '1';
      c_in <= '0';


       wait for 100ns;

-- 0101 +1011 = 10000

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

       wait for 100ns;

 --0111 + 1111 = 10110
     A0 <= '1';
      A1 <=  '1';
      A2 <= '1';
      A3 <= '0';
      B0 <=  '1';
      B1 <=  '1';
      B2 <= '1';
      B3 <= '1';
```

```
       c_in <= '0';

 wait for 100ns;

 -- 0001+ 0001 = 0010
      A0 <= '1';
      A1 <=  '0';
      A2 <= '0';
      A3 <= '0';
      B0 <=  '1';
      B1 <=  '0';
      B2 <= '0';
      B3 <= '0';
      c_in <= '0';

          wait;

end process;

end Behavioral;
```

2 way 3 bit(Test Bench)

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

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

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

entity TB_MUX2to1 is
--  Port ( );
end TB_MUX2to1;
```

```
architecture Behavioral of TB_MUX2to1 is
component MUX_2_to_1
    Port ( D0 : in STD_LOGIC;
           D1 : in STD_LOGIC;
           EN : in STD_LOGIC;
           Y : out STD_LOGIC);
end component;
signal d0,d1,en,y : STD_LOGIC;
begin
UUT: MUX_2_to_1
PORT MAP(
D0 => d0,
D1 => d1,
EN => en,
Y => y
);
process
begin
wait for 20ns;
en <= '0';
d0 <= '1';
d1 <= '0';
wait for 100ns;
en <= '1';
d1 <= '0';
wait for 100ns;
en <= '0';
d1 <= '0';
wait;
end process;
end Behavioral;
```

Instruction Decoder(Test Bench)

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
```

```
--use IEEE.NUMERIC_STD.ALL;

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

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

architecture Behavioral of TB_INS_Dec is
component Ins_Decoder
Port (      ROM : in STD_LOGIC_VECTOR (11 downto 0);
            Reg_value : in STD_LOGIC_VECTOR(3 downto 0);
            Reg_en : out STD_LOGIC_VECTOR (2 downto 0);
            L_Sel : out STD_LOGIC;
            In_Val : out STD_LOGIC_VECTOR (3 downto 0);
            Reg_sel0 : out STD_LOGIC_VECTOR (2 downto 0);
            Reg_sel1 : out STD_LOGIC_VECTOR (2 downto 0);
            AU_sel : out STD_LOGIC;
            JMP : out STD_LOGIC;
            JMP_ADD : out STD_LOGIC_VECTOR (2 downto 0));
end component;
 signal    ROM :  STD_LOGIC_VECTOR (11 downto 0);
  signal  Reg_value :  STD_LOGIC_VECTOR(3 downto 0);
  signal  Reg_en :  STD_LOGIC_VECTOR (2 downto 0);
    signal   L_Sel :  STD_LOGIC;
  signal          In_Val :  STD_LOGIC_VECTOR (3 downto 0);
  signal          Reg_sel0 :  STD_LOGIC_VECTOR (2 downto 0);
  signal          Reg_sel1 :  STD_LOGIC_VECTOR (2 downto 0);
  signal          AU_sel : STD_LOGIC;
  signal          JMP : STD_LOGIC;
  signal          JMP_ADD :  STD_LOGIC_VECTOR (2 downto 0);
begin
Dec1 : Ins_Decoder
PORT MAP (
            ROM => ROM,
            Reg_value =>  Reg_Value ,
            Reg_en =>  Reg_en,
            L_Sel => L_Sel,
            In_Val => In_Val,
```

```
            Reg_sel0 => Reg_Sel0,
            Reg_sel1 => Reg_Sel1,
            AU_sel => AU_sel ,
            JMP => JMP ,
            JMP_ADD => JMP_ADD);

        process
            begin
                ROM <= "100010001011";
                wait;
            end process;
end Behavioral;
```

Slow Clock (Test Bench)

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

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

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

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

architecture Behavioral of TB_SLOW is
component Slow_Clk
Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end component;
signal clkin : STD_LOGIC := '0';
signal clkout : STD_LOGIC;
begin
```

```
UUT : Slow_Clk
 PORT MAP(
     Clk_in => clkin,
     Clk_out => clkout
   );

process
    begin
        wait for 20ns;
        clkin <= NOT(clkin);
end process;
end Behavioral;
```

8 Way 4 Bit Mux (Test Bench)

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

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

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

entity TB_8MUX is
--  Port ( );
end TB_8MUX;

architecture Behavioral of TB_8MUX is

component way_8_mux
    Port ( Sel : in STD_LOGIC_VECTOR (2 downto 0);
           Y : out STD_LOGIC_VECTOR (3 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));
end component;

 signal  sel :  STD_LOGIC_VECTOR (2 downto 0);
 signal  y,r0,r1,r2,r3,r4,r5,r6,r7 : STD_LOGIC_VECTOR (3 downto 0);

begin
    UUT: way_8_mux
        port map(
            Sel => sel,
            Y => y,
            R0 => r0,
            R1 => r1,
            R2 => r2,
            R3 =>r3,
            R4 => r4,
            R5 => r5,
            R6 =>r6,
            R7 =>r7 );
    process
    begin
        sel <= "001";
         r0 <= "0000";
        r1 <= "1111";
        r2 <= "0000";
        r3 <= "0000";
        r4 <= "0000";
        r5 <= "0000";
        r6 <= "0000";
        r7 <= "0000";
        wait;

    end process;

end Behavioral;
```

Seven Segment Display(Test Bench)

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

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

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

entity TB_AU_7_Seg is
--  Port ( );
end TB_AU_7_Seg;

architecture Behavioral of TB_AU_7_Seg is

component AU_7_seg
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           RegSel : in STD_LOGIC;
           S_LED : out STD_LOGIC_VECTOR (3 downto 0);
           S_7Seg : out STD_LOGIC_VECTOR (6 downto 0);
           Carry : out STD_LOGIC;
           Zero : out STD_LOGIC;
           Anode : out STD_LOGIC_VECTOR (3 downto 0));
end component;

    signal A, S_LED : STD_LOGIC_VECTOR (3 downto 0) := "0000";
    signal Anode : STD_LOGIC_VECTOR (3 downto 0);
    signal S_7Seg : STD_LOGIC_VECTOR (6 downto 0);
    signal RegSel, Carry, Zero, Clk : STD_LOGIC := '0';

begin
    UUT: AU_7_seg
        port map(
```

```
            A => A,
            Clk => Clk,
            RegSel => RegSel,
            S_LED => S_LED,
            S_7Seg => S_7Seg,
            Carry => Carry,
            Zero => Zero,
            Anode => Anode);

    process
    begin
        Clk <= NOT(Clk);
        wait for 20 ns;
    end process;

    process
    begin

        RegSel <= '0';
        A <= "0100";
        wait for 83ns;

        RegSel <= '1';
        A <= "1101";
        wait for 92ns;

        RegSel <= '0';
        A <= "1110";
        wait for 85ns;

        RegSel <= '1';
        A <= "0000";
        wait for 83ns;

        RegSel <= '0';
        A <= "0011";
        wait for 81ns;

        RegSel <= '1';
        A <= "1011";
        wait for 85ns;
```

```
        RegSel <= '0';
        A <= "0000";
        wait for 82ns;

        RegSel <= '1';
        A <= "0000";
        wait for 82ns;

    end process;

end Behavioral;
```

Register Bank(Test Bench)

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

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

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

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

architecture Behavioral of TB_Register is
component Regsiter_Bank
 Port ( R_en : in STD_LOGIC_VECTOR (2 downto 0);
           res : in STD_LOGIC;
          Clk_r : in STD_LOGIC;
          R_in1 : in STD_LOGIC_VECTOR(3 downto 0);
          R_in2 : in STD_LOGIC_VECTOR(3 downto 0);
          R_in3 : in STD_LOGIC_VECTOR(3 downto 0);
          R_in4 : in STD_LOGIC_VECTOR(3 downto 0);
          R_in5 : in STD_LOGIC_VECTOR(3 downto 0);
```

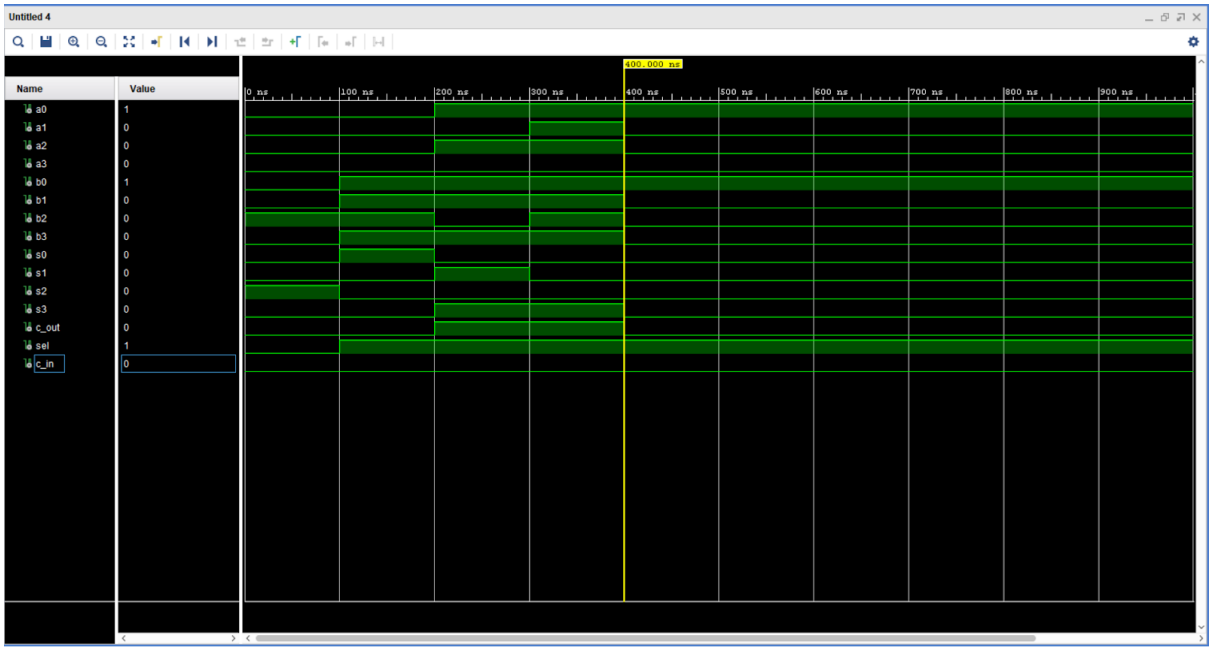
```
          R_in6 : in STD_LOGIC_VECTOR(3 downto 0);
          R_in7 : in STD_LOGIC_VECTOR(3 downto 0);
          Bus0 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus1 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus2 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus3 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus4 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus5 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus6 : out STD_LOGIC_VECTOR (3 downto 0);
          Bus7 : out STD_LOGIC_VECTOR (3 downto 0));
end component;

 signal        R_en :  STD_LOGIC_VECTOR (2 downto 0);
 signal          res :  STD_LOGIC;
signal          Clk_r :  STD_LOGIC := '0';
 signal         Bus0 :  STD_LOGIC_VECTOR (3 downto 0);
  signal        Bus1 : STD_LOGIC_VECTOR (3 downto 0);
 signal         Bus2 :  STD_LOGIC_VECTOR (3 downto 0);
 signal         Bus3 :  STD_LOGIC_VECTOR (3 downto 0);
  signal        Bus4 :  STD_LOGIC_VECTOR (3 downto 0);
  signal        Bus5 :  STD_LOGIC_VECTOR (3 downto 0);
  signal        Bus6 :  STD_LOGIC_VECTOR (3 downto 0);
signal          Bus7 :  STD_LOGIC_VECTOR (3 downto 0);
signal r :  STD_LOGIC_VECTOR (3 downto 0);




begin
UUT : Regsiter_Bank
port map(
          R_en => R_en   ,
          res =>  res ,
          Clk_r => Clk_r   ,
          R_in1 => r ,
          R_in2 =>   r ,
          R_in3 =>   r   ,
          R_in4=>   r ,
          R_in5 =>   r  ,
          R_in6 =>  r  ,
          R_in7  =>   r  ,
```

```
          Bus0 => Bus0 ,
          Bus1 =>   Bus1 ,
          Bus2 => Bus2   ,
          Bus3 =>  Bus3  ,
          Bus4=>  Bus4 ,
          Bus5=> Bus5  ,
          Bus6 => Bus6  ,
          Bus7=> Bus7
);


process
begin
wait for 20ns;
Clk_r <= NOT(Clk_r);
end process;

-- 200418R -- 1 1000 1110 1110 0010

process
begin
 R_en <= "001";
 r<= "0010";
 res <= '0';
wait for 100ns;
 r<= "1110";
 R_en <= "010";
wait for 100ns;
R_en <= "011";
 r<= "1110";
wait for 100ns;
R_en <= "100";
 r<= "1000";

wait for 100ns;
R_en <= "101";
 r<= "0010";

wait for 100ns;
R_en <= "110";
 r<= "1110";
```

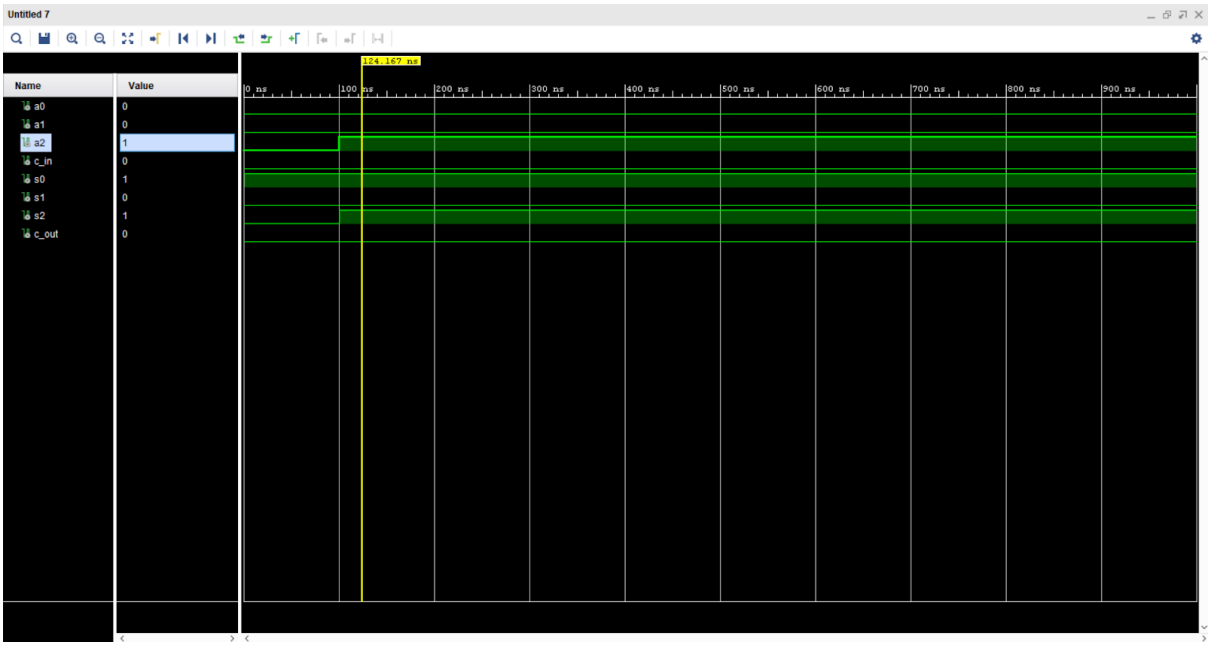
```
wait for 100ns;
R_en <= "111";
 r<= "1110";

wait;
end process;



end Behavioral;
```

Timing Diagrams

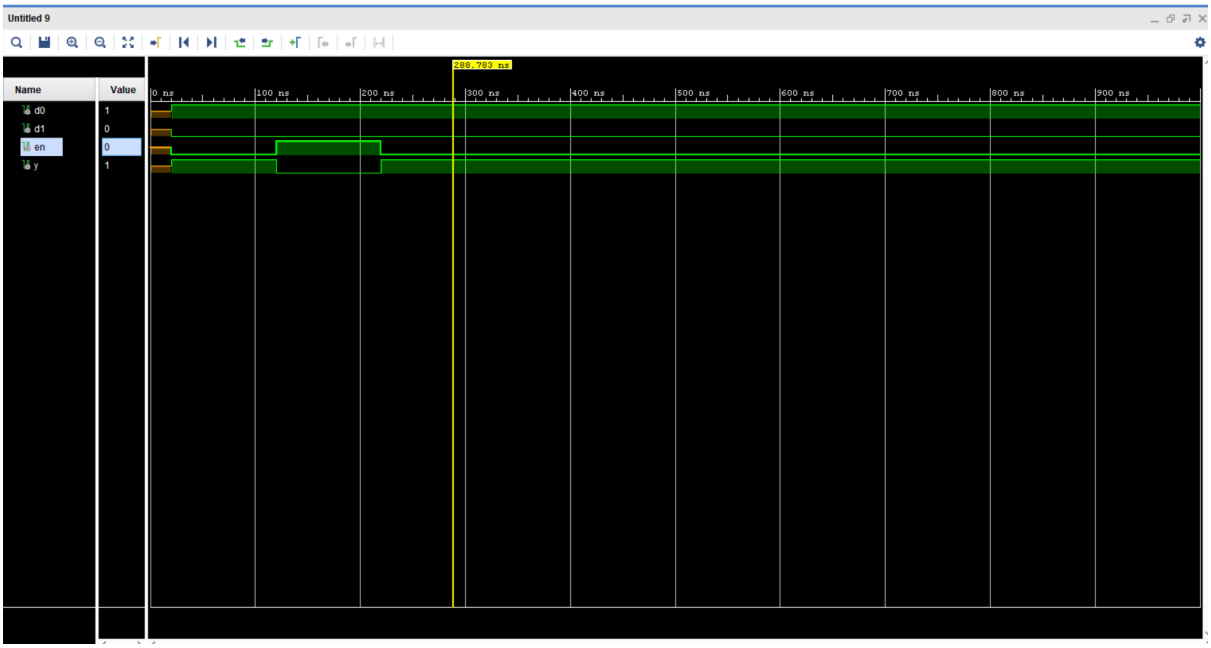
4-bit Add/Subtract Unit



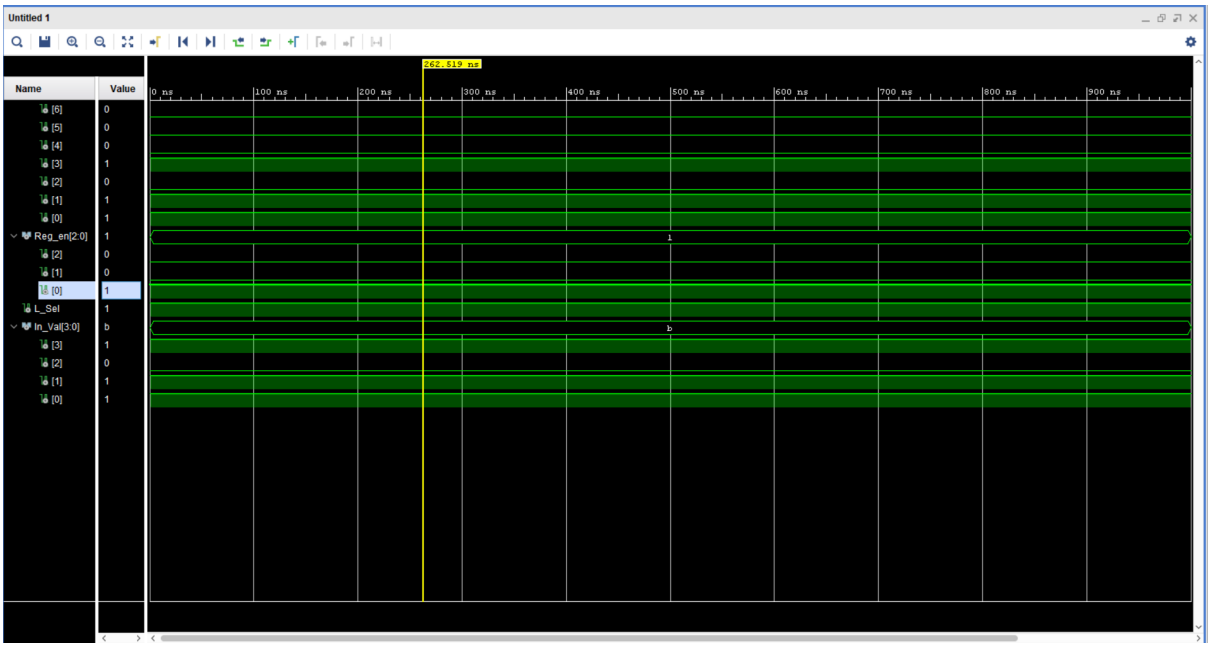
3-bit Adder



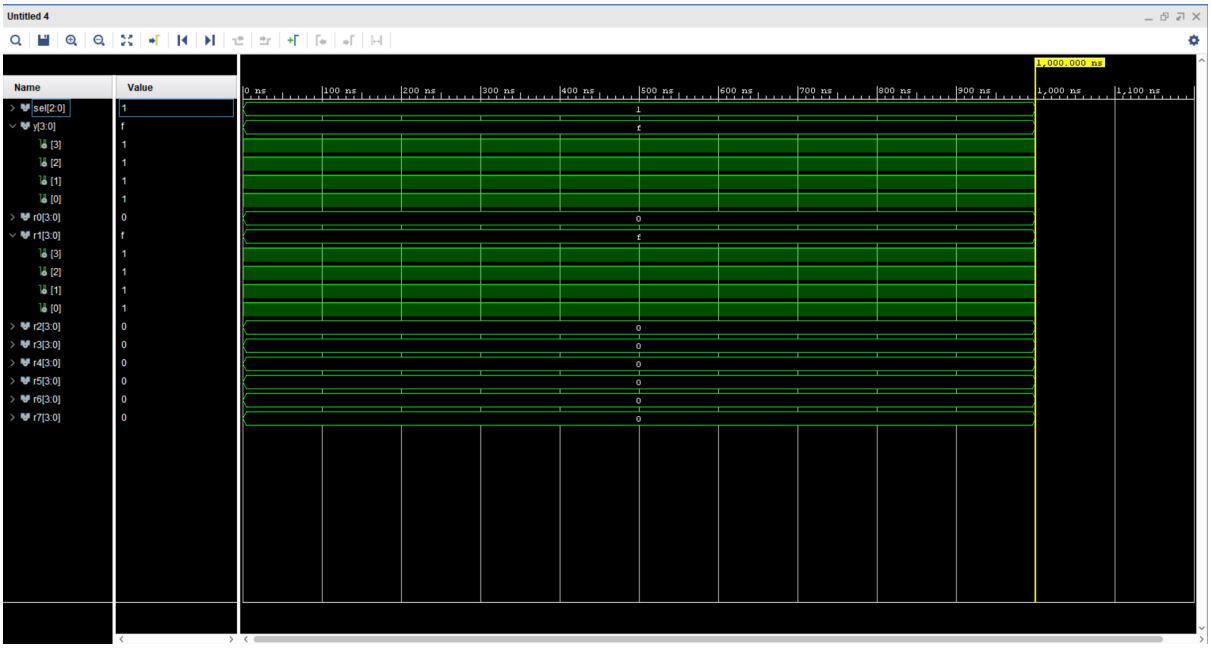
2 way 3 bit Mux



Instruction Decoder

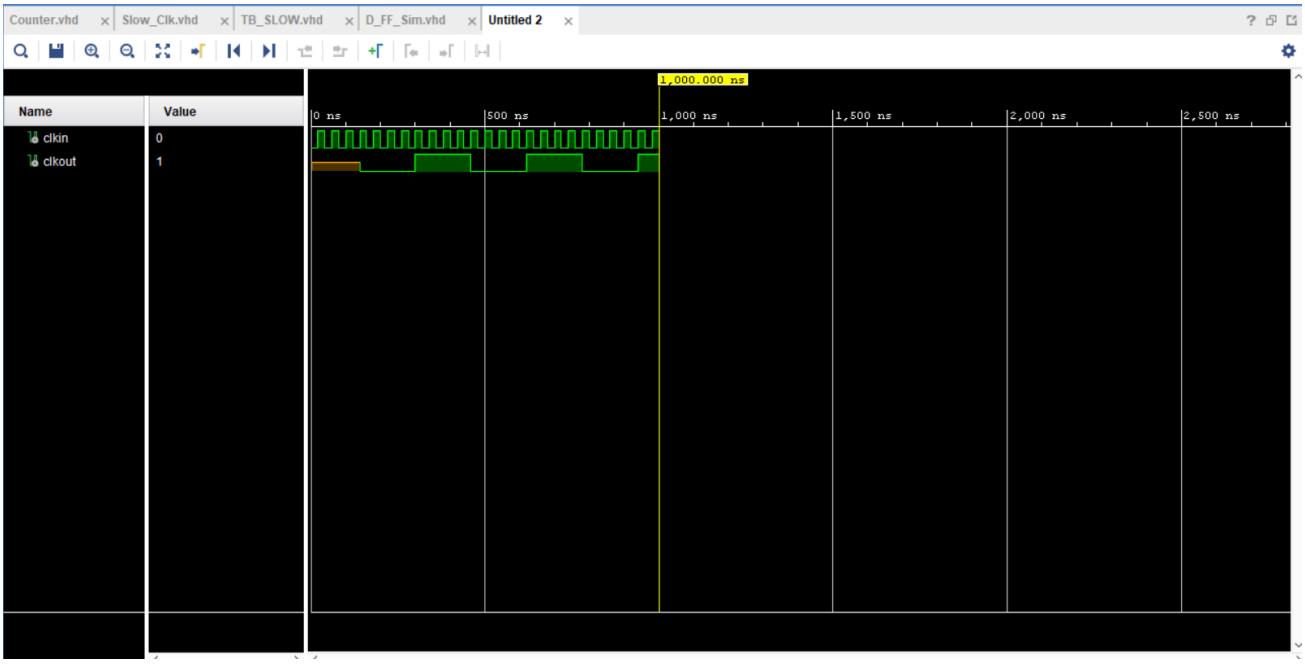


8 way 4 bit Mux

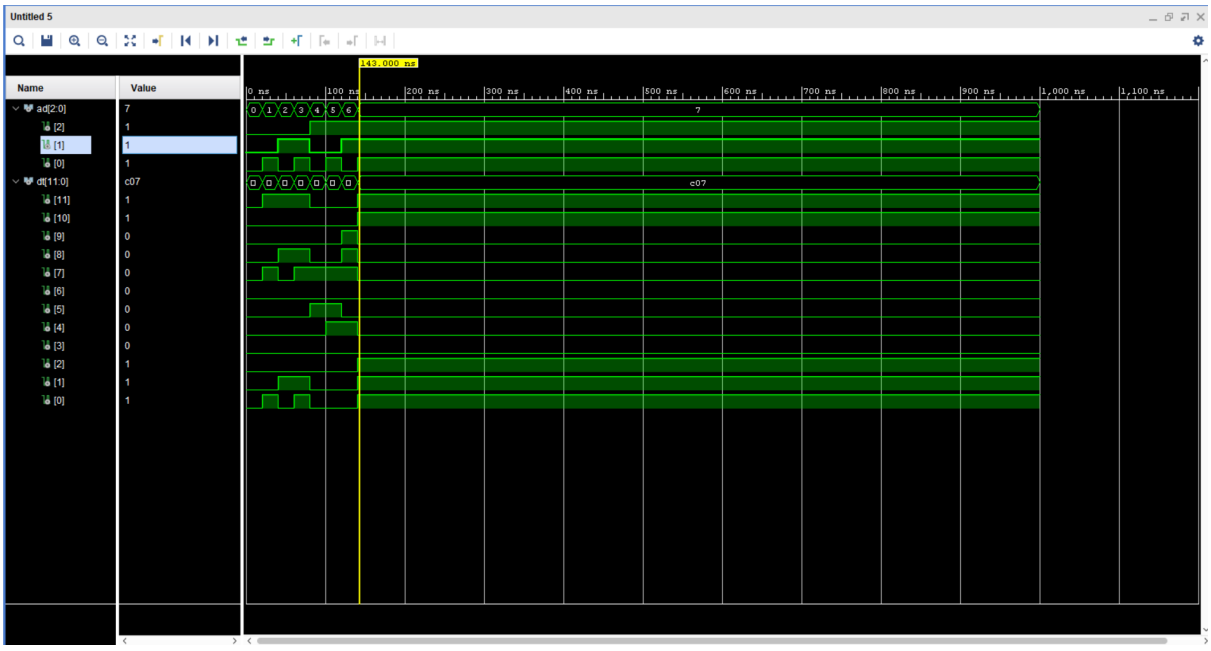


Seven Segment Display

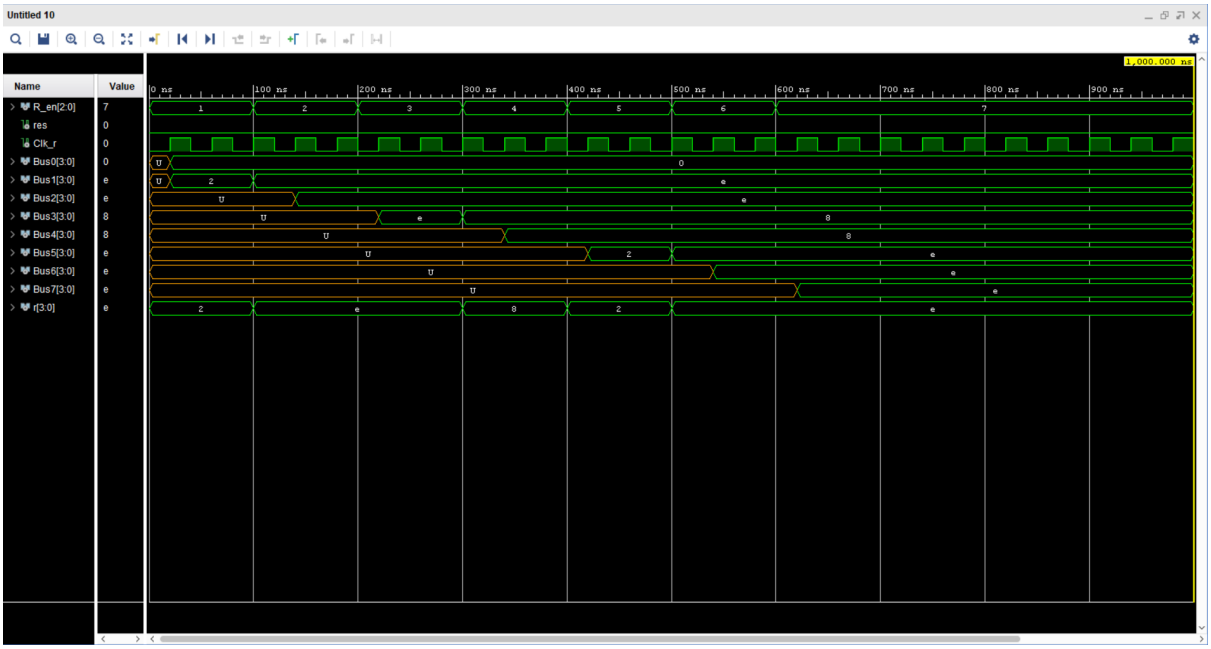
Slow Clock



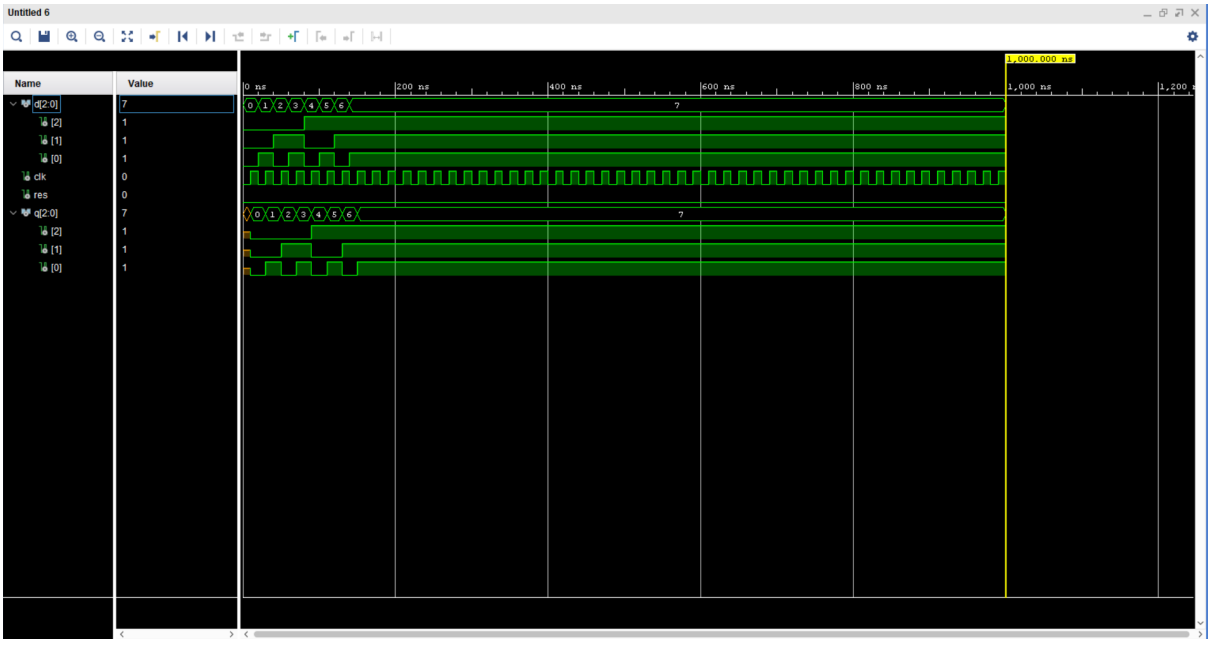
ROM



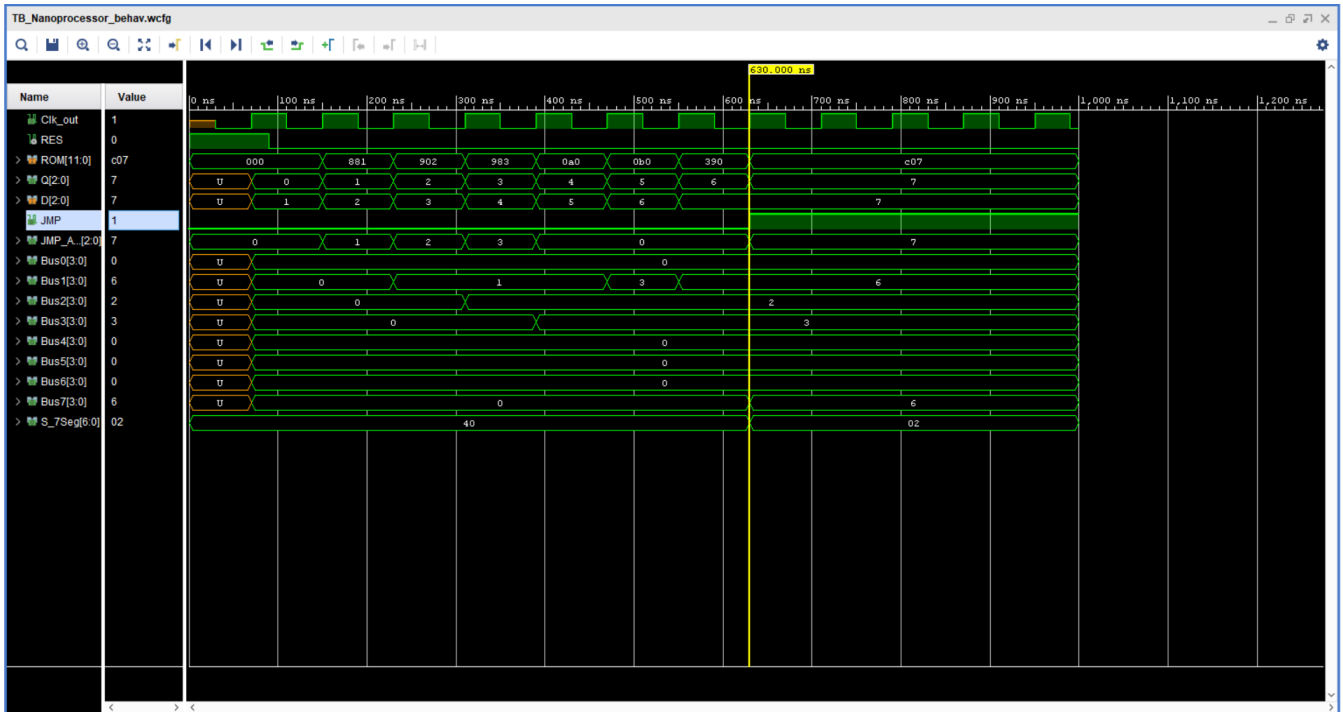
Register Bank



PC



Nanoprocessor



Assembly Program

1. Do Nothing
2. MOVI R1,1
3. MOVI R2,2
4. MOVI R3,3
5. ADD R1,R2
6. ADD R1,R3
7. ADD,R7,R1
8. JZR R0,8 (If R0=0 Jump to Line 8)

Machine Code

1. 0000 0000 0000
2. 1000 1000 0001
3. 1001 0000 0010
4. 1001 1000 0011
5. 0000 1010 0000
6. 0000 1011 0000
7. 0011 1001 0000
8. 1100 0000 0111

Conclusions

- Usage of buses simplified the design rather than using so many wires.
- We can calculate only the total of integers between 1 and 3 as our microprocessor is only 4 bit wide.
- We need to hardcode assembly instructions as binary values, as microprocessors only understand machine language. We have achieved this via the help of ROM.
- Results are generated by the clock input.
- The Res input can reset the nanoprocessor.
- We could import required code files from previous labs and use them in building the processor.
- Proper performance of sub components was ensured before completing the microprocessor by simulating them using xsim simulator.
- The basic knowledge about how a microprocessor is internally structured and how those components work inside was received through the project.
- Team working skills were highlighted through this project, as we could focus on sub parts separately and finally combine them to form a microprocessor.
- Teamwork is essential in the computer science field.

Final LUT/FF counts

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	42	0	20800	0.20
LUT as Logic	42	0	20800	0.20
LUT as Memory	0	0	9600	0.00
Slice Registers	53	0	41600	0.13
Register as Flip Flop	53	0	41600	0.13
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

7. Primitives

Ref Name	Used	Functional Category
FDRE	53	Flop & Latch
LUT4	19	LUT
OBUF	17	IO
LUT5	14	LUT
LUT6	12	LUT
CARRY4	8	CarryLogic
LUT3	7	LUT
LUT2	2	LUT
IBUF	2	IO
LUT1	1	LUT
BUFG	1	Clock

Individual Contribution

200452N - PERERA D.N.M	4 Bit Adder Subtractor Instruction Decoder Final connections (nanoprocessor.vhd) Final Simulation(TB_nanoprocessor.vhd)	10 Hours Spent
200748D- Aponso G.M.M.K	8 way 4 bit Muxes 1 8 way 4 bit Muxes 2 Lab Report	10 Hours spent
Umayanga S A I 200671J	3-bit RCA 2 way 3 bit Mux 2 way 4 bit Mux	12 hours spent
Chamil 200418R	Register Bank 7 Segment Display Slow clock	12 Hours spent
AWADR Wickramasinghe 200712M	ROM Program Counter	10 Hours spent

END OF REPORT