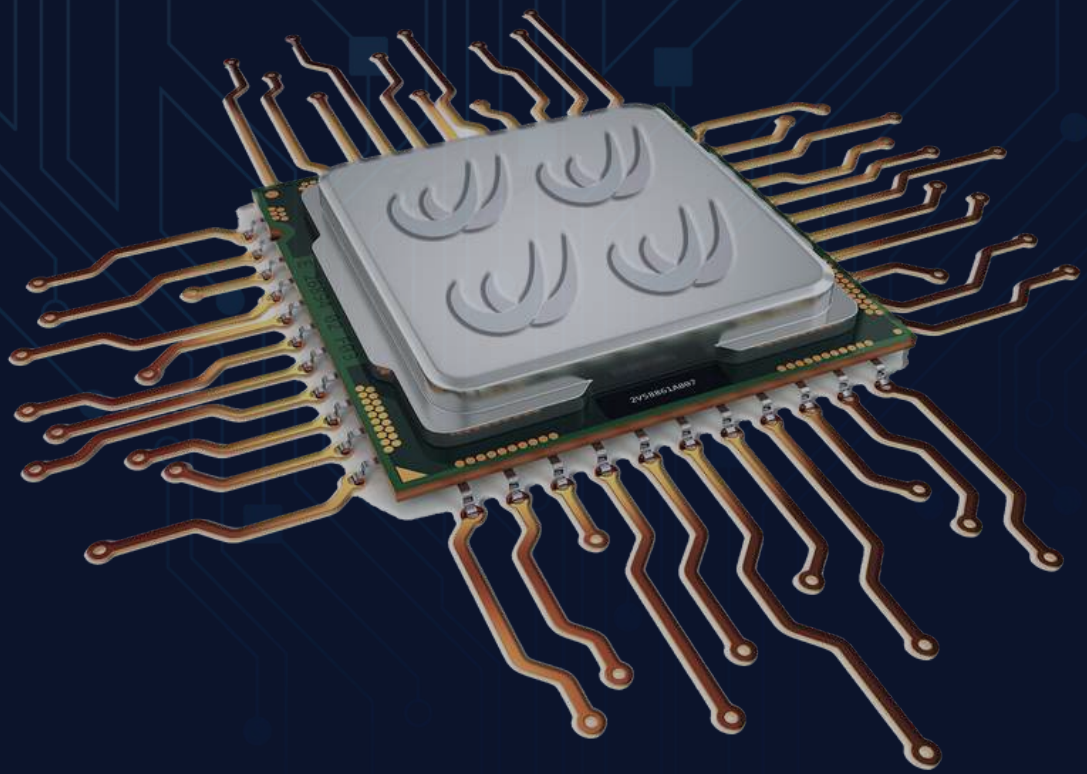
NANO PROCESSOR

CSE 21' | LAB REPORT | LAB 9 & LAB 10

Developed By :

- Gathsara J.A.S. - 210180L
- Gallage D. - 210173T

Group Number - 30



LAB TASK

- design and develop a 4-bit arithmetic unit that can add and subtract signed integers.
- decode instructions to activate necessary components on the processor.
- design and develop k-way b-bit multiplexers or tri-state busses.
- verify their functionality via simulation and on the development board.

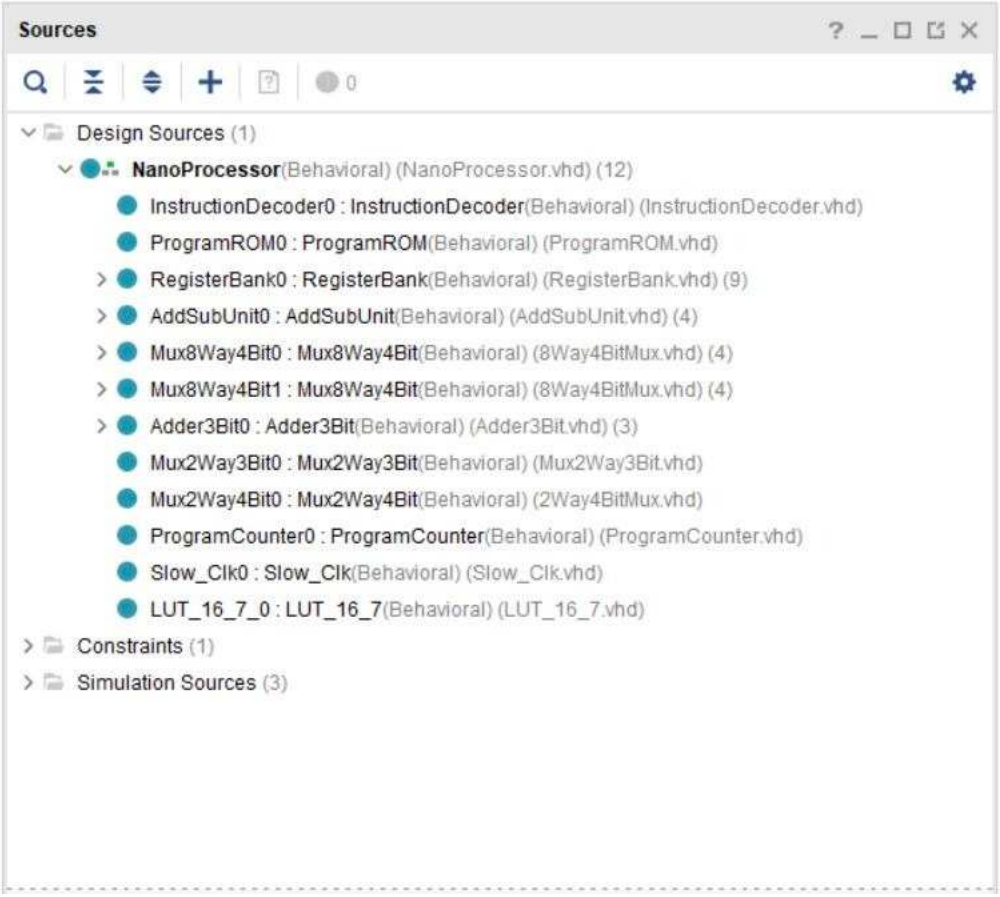
ASSEMBLY PROGRAM

```
MOVI  R7, 0
MOVI  R1, 1
MOVI  R2, 2
MOVI  R3, 3
ADD   R7, R1
ADD   R7, R2
ADD   R7, R3
JZR   R0, 7
```

MACHINE CODE REPRESENTATION

```
101110000000
100010000001
100100000010
100110000011
001110010000
001110100000
001110110000
110000000111
```

VHDL FILES AND TIMING DIAGRAMS



Design Sources

Nano Processor

NANOPROCESSOR.VHD

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

entity NanoProcessor is
    Port ( Clk : in STD_LOGIC;
           Reset : in STD_LOGIC; -- Reset button
           Reg7 : out STD_LOGIC_VECTOR (3 downto 0); -- To Reg7 LED
           Zero : out STD_LOGIC; -- To zero LED
           Overflow : out STD_LOGIC; -- To overflow led
           SevenSegment : out STD_LOGIC_VECTOR(6 downto 0); -- To seven Segment
           AnodeSelect : out STD_LOGIC_VECTOR (3 downto 0); -- To Anode Select Bus
           SwitchIn : in STD_LOGIC_VECTOR(3 downto 0);
           RegDisplay : in STD_LOGIC_VECTOR(2 downto 0));
end NanoProcessor;
```

```
architecture Behavioral of NanoProcessor is

Component InstructionDecoder port(
           InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
           RegisterCheck : in STD_LOGIC_VECTOR (3 downto 0);
           JumpAddress : out STD_LOGIC_VECTOR (2 downto 0);
           AddSubSelect : out STD_LOGIC;
           RegisterEnable : out STD_LOGIC_VECTOR (2 downto 0);
           LoadSelect : out STD_LOGIC;
           ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
           RegASelect : out STD_LOGIC_VECTOR (2 downto 0);
           RegBSelect : out STD_LOGIC_VECTOR (2 downto 0);
           JumpFlag : out STD_LOGIC;
           SwitchIn : in STD_LOGIC_VECTOR(3 downto 0));
end component;

component ProgramROM port(
           MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
           InstructionBus : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component RegisterBank port(
           RegisterIn : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           output0 : out STD_LOGIC_VECTOR (3 downto 0);
           output1 : out STD_LOGIC_VECTOR (3 downto 0);
           output2 : out STD_LOGIC_VECTOR (3 downto 0);
           output3 : out STD_LOGIC_VECTOR (3 downto 0);
           output4 : out STD_LOGIC_VECTOR (3 downto 0);
           output5 : out STD_LOGIC_VECTOR (3 downto 0);
           output6 : out STD_LOGIC_VECTOR (3 downto 0);
           output7 : out STD_LOGIC_VECTOR (3 downto 0);
           RegisterEn : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC := '0');
 end component;

 component AddSubUnit 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);
            AddSubSelect : in STD_LOGIC;
            Overflow : out STD_LOGIC;
            Zero : out STD_LOGIC);
end component;

component Mux8Way4Bit port(
           In0 : in STD_LOGIC_VECTOR (3 downto 0);
           In1 : in STD_LOGIC_VECTOR (3 downto 0);
           In2 : in STD_LOGIC_VECTOR (3 downto 0);
           In3 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            In4 : in STD_LOGIC_VECTOR (3 downto 0);
            In5 : in STD_LOGIC_VECTOR (3 downto 0);
            In6 : in STD_LOGIC_VECTOR (3 downto 0);
            In7 : in STD_LOGIC_VECTOR (3 downto 0);
            OutMux : out STD_LOGIC_VECTOR (3 downto 0);
            Control : in STD_LOGIC_VECTOR (2 downto 0));
end component;

component Adder3Bit port(
            A : in STD_LOGIC_VECTOR (2 downto 0);
            B : in STD_LOGIC_VECTOR (2 downto 0);
            output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux2Way3Bit port(
            In0 : in STD_LOGIC_VECTOR (2 downto 0);
            In1 : in STD_LOGIC_VECTOR (2 downto 0);
            LoadSelect : in STD_LOGIC;
            output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component Mux2Way4Bit port(
            In0 : in STD_LOGIC_VECTOR (3 downto 0);
            In1 : in STD_LOGIC_VECTOR (3 downto 0);
            LoadSelect : in STD_LOGIC;
            output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

component ProgramCounter port(
            input : in STD_LOGIC_VECTOR (2 downto 0);
            output : out STD_LOGIC_VECTOR (2 downto 0);
            Clk : in STD_LOGIC;
            Reset : in STD_LOGIC);
end component;

component Slow_Clk port(
            Clk_in : in STD_LOGIC;
            Clk_out : out STD_LOGIC);
end component;

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

SIGNAL InstructionBus : std_logic_vector(11 downto 0);
SIGNAL Mux8Way4BitOut0, Mux8Way4BitOut1, ImmediateValue, RegisterIn, AddSubOut,
lookupTableIn : std_logic_vector(3 downto 0);
SIGNAL JumpAddress, RegisterEnable, RegASelect, RegBSelect, ROMMemorySelect, Adder3BitOut,
ProgramCounterIn : std_logic_vector(2 downto 0);
SIGNAL AddSubSelect, LoadSelect, JumpFlag, SlowClock: std_logic;
```

```
SIGNAL DataBus0, DataBus1, DataBus2, DataBus3, DataBus4, DataBus5, DataBus6, DataBus7 :
std_logic_vector(3 downto 0);

Signal S_temp : std_logic_vector(3 downto 0);

signal count2 : integer := 1;
--signal count3 : std_logic := '0';


begin

InstructionDecoder0 : InstructionDecoder port map(
           InstructionBus => InstructionBus,
           RegisterCheck => Mux8Way4BitOut0,
           JumpAddress => JumpAddress,
           AddSubSelect => AddSubSelect,
           RegisterEnable => RegisterEnable,
           LoadSelect => LoadSelect,
           ImmediateValue => ImmediateValue,
           RegASelect => RegASelect,
           RegBSelect => RegBSelect,
           JumpFlag => JumpFlag,
           SwitchIn => SwitchIn
);

ProgramROM0 : ProgramROM port map(
           MemorySelect => ROMMemorySelect,
           InstructionBus => InstructionBus
);

RegisterBank0 : RegisterBank port map(
           RegisterIn => RegisterIn,
           Clk => SlowClock,
           output0 => DataBus0,
           output1 => DataBus1,
           output2 => DataBus2,
           output3 => DataBus3,
           output4 => DataBus4,
           output5 => DataBus5,
           output6 => DataBus6,
           output7 => DataBus7,
           RegisterEn => RegisterEnable,
           Reset => Reset
);

AddSubUnit0 : AddSubUnit port map(
           A => Mux8Way4BitOut0,
           B => Mux8Way4BitOut1,
           S => AddSubOut,
           AddSubSelect => AddSubSelect,
```

```
            Overflow => Overflow,
            Zero => Zero
);

Mux8Way4Bit0 : Mux8Way4Bit port map(
            In0 => DataBus0,
            In1 => DataBus1,
            In2 => DataBus2,
            In3 => DataBus3,
            In4 => DataBus4,
            In5 => DataBus5,
            In6 => DataBus6,
            In7 => DataBus7,
            OutMux => Mux8Way4BitOut0,
            Control => RegASelect
);

Mux8Way4Bit1 : Mux8Way4Bit port map(
            In0 => DataBus0,
            In1 => DataBus1,
            In2 => DataBus2,
            In3 => DataBus3,
            In4 => DataBus4,
            In5 => DataBus5,
            In6 => DataBus6,
            In7 => DataBus7,
            OutMux => Mux8Way4BitOut1,
            Control => RegBSelect
);

Adder3Bit0 : Adder3Bit port map(
            A => ROMMemorySelect,
            B => "001",
            output => Adder3BitOut
);

Mux2Way3Bit0 : Mux2Way3Bit port map(
            In0 => Adder3BitOut,
            In1 => JumpAddress,
            LoadSelect => JumpFlag,
            output => ProgramCounterIn
);

Mux2Way4Bit0 : Mux2Way4Bit port map(
            In0 => ImmediateValue,
            In1 => AddSubOut,
            LoadSelect => LoadSelect,
            output => RegisterIn
);

ProgramCounter0 : ProgramCounter port map(
```

```
           input => ProgramCounterIn,
           output => ROMMemorySelect,
           Clk => SlowClock,
           Reset => Reset
);

Slow_Clk0 : Slow_Clk port map(
           Clk_in => Clk,
           Clk_out => SlowClock
);

LUT_16_7_0 : LUT_16_7 port map(
        address => S_temp,
        data => SevenSegment
);

Reg7 <= DataBus7;

process(Clk)
begin

    if(rising_edge(Clk)) then
        count2 <= count2 + 1;
        if (count2 = 10000) then -- Register Value Display
            AnodeSelect <= "1110";
            case RegDisplay is
            when "000" =>
                S_temp <= DataBus0;
            when "001" =>
                S_temp <= DataBus1;
            when "010" =>
                S_temp <= DataBus2;
            when "011" =>
                S_temp <= DataBus3;
            when "100" =>
                S_temp <= DataBus4;
            when "101" =>
                S_temp <= DataBus5;
            when "110" =>
                S_temp <= DataBus6;
            when "111" =>
                S_temp <= DataBus7;
            end case;

        elsif (count2 = 20000) then -- RegisterDisplay number display
            AnodeSelect <= "0111";
            lookupTableIn(3) <= '0';
            lookupTableIn(2 downto 0) <= RegDisplay;
            S_temp <= lookupTableIn;
        elsif (count2 = 30000) then -- Counter
               AnodeSelect <= "1101";
```

```
                lookupTableIn(3) <= '0';
                lookupTableIn(2 downto 0) <= ROMMemorySelect;
                S_temp <= lookupTableIn;
                count2 <= 0;
        end if;
    end if;
end process;



end Behavioral;
```

Instruction Decoder

INSTRUCTIONDECODER.VHD

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

entity InstructionDecoder is
    Port ( InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
           RegisterCheck : in STD_LOGIC_VECTOR (3 downto 0);
           JumpAddress : out STD_LOGIC_VECTOR (2 downto 0);
           AddSubSelect : out STD_LOGIC;
           RegisterEnable : out STD_LOGIC_VECTOR (2 downto 0);
           LoadSelect : out STD_LOGIC;
           ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
           RegASelect : out STD_LOGIC_VECTOR (2 downto 0);
           RegBSelect : out STD_LOGIC_VECTOR (2 downto 0);
           JumpFlag : out STD_LOGIC;
           SwitchIn : in STD_LOGIC_VECTOR (3 downto 0));
end InstructionDecoder;

architecture Behavioral of InstructionDecoder is

begin
    process (InstructionBus(11 downto 0),RegisterCheck,SwitchIn)
    begin
            if (InstructionBus(11 downto 10) = "00") then -- Add
                RegASelect <= InstructionBus(9 downto 7);
                RegBSelect <= InstructionBus(6 downto 4);
                AddSubSelect <= '0';
                LoadSelect <= '1';
                RegisterEnable <= InstructionBus(9 downto 7);
                JumpFlag <= '0';

            elsif (InstructionBus(11 downto 10) = "01") then -- Negative
                RegBSelect <= InstructionBus(9 downto 7);
                RegisterEnable <= InstructionBus(9 downto 7);
                LoadSelect <= '1';
```

```
                RegASelect <= "000";
                AddSubSelect <= '1';
                JumpFlag <= '0';

            elsif (InstructionBus(11 downto 10) = "10") then -- Move
                RegisterEnable <= InstructionBus(9 downto 7);
                LoadSelect <= '0';
                JumpFlag <= '0';
                if (InstructionBus(5) = '0') then
                    ImmediateValue <= InstructionBus(3 downto 0);
                else
                    ImmediateValue <= SwitchIn;
                end if;

            else
                JumpAddress <= InstructionBus(2 downto 0); -- Jump
                RegASelect <= InstructionBus(9 downto 7);
                RegisterEnable <= "000";

                if RegisterCheck = "0000" then
                    JumpFlag <= '1';
                else
                    JumpFlag <= '0';
                end if;

            end if;
    end process;



end Behavioral;
```

Program ROM

PROGRAMROM.VHD

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

entity ProgramROM is
    Port ( MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
           InstructionBus : out STD_LOGIC_VECTOR (11 downto 0));
end ProgramROM;

architecture Behavioral of ProgramROM is
```

```
type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);

signal program_ROM : rom_type := (      "101110000000", -- 0 MOVI R7,0
                                        "100010000001", -- 1 MOVI R1,1
                                        "100100000010", -- 2 MOVI R2,2
                                        "100110000011", -- 3 MOVI R3,3
                                        "001110010000", -- 4 ADD R7,R1
                                        "001110100000", -- 5 ADD R7,R2
                                        "001110110000", -- 6 ADD R7,R3
                                        "110000000111"); -- 7 JZR R0,7


begin

    InstructionBus <= program_ROM(to_integer(unsigned(MemorySelect)));

end Behavioral;
```

Register Bank

REGISTERBANK.VHD

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

entity RegisterBank is
    Port ( RegisterIn : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           output0 : out STD_LOGIC_VECTOR (3 downto 0);
           output1 : out STD_LOGIC_VECTOR (3 downto 0);
           output2 : out STD_LOGIC_VECTOR (3 downto 0);
           output3 : out STD_LOGIC_VECTOR (3 downto 0);
           output4 : out STD_LOGIC_VECTOR (3 downto 0);
           output5 : out STD_LOGIC_VECTOR (3 downto 0);
           output6 : out STD_LOGIC_VECTOR (3 downto 0);
           output7 : out STD_LOGIC_VECTOR (3 downto 0);
           RegisterEn : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC := '0');
end RegisterBank;

architecture Behavioral of RegisterBank is

component Reg 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);
    Reset : in STD_LOGIC);

end component;
```

```
component Decoder_3_to_8 port(

    I : in STD_LOGIC_VECTOR (2 downto 0);
    EN : in STD_LOGIC;
    Y : out STD_LOGIC_VECTOR (7 downto 0));

end component;

SIGNAL Y0 : std_logic_vector(7 downto 0);
SIGNAL HC : std_logic_vector(3 downto 0);
Signal reset0 : std_logic;

begin

Decoder : Decoder_3_to_8 port map(
    I => RegisterEn,
    EN => '1',
    Y => Y0 );

Reg0 : Reg port map(
    D => HC,        -- Hardcoded
    En => '1',
    Clk => Clk,
    Q => output0,
    Reset => reset0 );

Reg1 : Reg port map(
    D => RegisterIn,
    En => Y0(1),
    Clk => Clk,
    Q => output1,
    Reset => reset0 );

Reg2 : Reg port map(
    D => RegisterIn,
    En => Y0(2),
    Clk => Clk,
    Q => output2,
    Reset => reset0 );

Reg3 : Reg port map(
    D => RegisterIn,
    En => Y0(3),
    Clk => Clk,
    Q => output3,
    Reset => reset0 );

Reg4 : Reg port map(
    D => RegisterIn,
    En => Y0(4),
```

```
    Clk => Clk,
    Q => output4,
    Reset => reset0 );

Reg5 : Reg port map(
    D => RegisterIn,
    En => Y0(5),
    Clk => Clk,
    Q => output5,
    Reset => reset0 );

Reg6 : Reg port map(
    D => RegisterIn,
    En => Y0(6),
    Clk => Clk,
    Q => output6,
    Reset => reset0 );

Reg7 : Reg port map(
    D => RegisterIn,
    En => Y0(7),
    Clk => Clk,
    Q => output7,
    Reset => reset0 );

HC <= "0000";
reset0 <= Reset;

end Behavioral;
```

DECODER_3_TO_8.VHD

```
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 : 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);
    Y(3 downto 0) <= Y0;
    Y(7 downto 4) <= Y1;

end Behavioral
```

DECODER_2_TO_4.VHD

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

end Behavioral;
```

REG.VHD

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

entity Reg is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC := '1';
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0) := "0000";
           Reset : in STD_LOGIC := '0');
end Reg;

architecture Behavioral of Reg is

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

end Behavioral;
```

Add Sub Unit

ADDSUBUNIT.VHD

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


entity AddSubUnit 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);
           AddSubSelect : in STD_LOGIC;
           Overflow : out STD_LOGIC;
           Zero : out STD_LOGIC);
end AddSubUnit;

architecture Behavioral of AddSubUnit 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 B2,S2 : std_logic_vector(3 downto 0);
SIGNAL C2 : std_logic_vector(2 downto 0);

begin



FA_0 : FA port map(
    A => A(0),
    B => B2(0),
    C_in => AddSubSelect,
    S => S2(0),
    C_out => C2(0)
);

FA_1 : FA port map(
    A => A(1),
    B => B2(1),
    C_in => C2(0),
    S => S2(1),
    C_out => C2(1)
);

FA_2 : FA port map(
    A => A(2),
```

```
    B => B2(2),
    C_in => C2(1),
    S => S2(2),
    C_out => C2(2)
);

FA_3 : FA port map(
    A => A(3),
    B => B2(3),
    C_in => C2(2),
    S => S2(3),
    C_out => Overflow
);

B2(0) <= AddSubSelect XOR B(0);
B2(1) <= AddSubSelect XOR B(1);
B2(2) <= AddSubSelect XOR B(2);
B2(3) <= AddSubSelect XOR B(3);

S <= S2;

Zero <= NOT(S2(0) OR S2(1) OR S2(2) OR S2(3));

end Behavioral;
```

FA.VHD

```
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;
```

HA.VHD

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

end Behavioral;
```

Mux 8 Way 4 Bit

MUX8WAY4BIT.VHD

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


entity Mux8Way4Bit is
    Port ( In0 : in STD_LOGIC_VECTOR (3 downto 0);
           In1 : in STD_LOGIC_VECTOR (3 downto 0);
           In2 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
           In3 : in STD_LOGIC_VECTOR (3 downto 0);
           In4 : in STD_LOGIC_VECTOR (3 downto 0);
           In5 : in STD_LOGIC_VECTOR (3 downto 0);
           In6 : in STD_LOGIC_VECTOR (3 downto 0);
           In7 : in STD_LOGIC_VECTOR (3 downto 0);
           OutMux : out STD_LOGIC_VECTOR (3 downto 0);
           Control : in STD_LOGIC_VECTOR (2 downto 0));
end Mux8Way4Bit;

architecture Behavioral of Mux8Way4Bit is

component Mux_8_to_1
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           D : in STD_LOGIC_VECTOR (7 downto 0);
           EN : in STD_LOGIC;
           Y : out STD_LOGIC);
end component;

SIGNAL Control0 : std_logic_vector(2 downto 0);

begin

Mux_8_to_1_A : Mux_8_to_1
    port map(
              S => Control0,

              D(0) => In0(0),
              D(1) => In1(0),
              D(2) => In2(0),
              D(3) => In3(0),
              D(4) => In4(0),
              D(5) => In5(0),
              D(6) => In6(0),
              D(7) => In7(0),

              EN => '1',
              Y => OutMux(0)
              );

Mux_8_to_1_B : Mux_8_to_1
   port map(
             S => Control0,

             D(0) => In0(1),
             D(1) => In1(1),
             D(2) => In2(1),
             D(3) => In3(1),
             D(4) => In4(1),
             D(5) => In5(1),
             D(6) => In6(1),
             D(7) => In7(1),
```

```
            EN => '1',
            Y => OutMux(1)
            );

Mux_8_to_1_C : Mux_8_to_1
    port map(
             S => Control0,

             D(0) => In0(2),
             D(1) => In1(2),
             D(2) => In2(2),
             D(3) => In3(2),
             D(4) => In4(2),
             D(5) => In5(2),
             D(6) => In6(2),
             D(7) => In7(2),

             EN => '1',
             Y => OutMux(2)
             );

Mux_8_to_1_D : Mux_8_to_1
   port map(
            S => Control0,

            D(0) => In0(3),
            D(1) => In1(3),
            D(2) => In2(3),
            D(3) => In3(3),
            D(4) => In4(3),
            D(5) => In5(3),
            D(6) => In6(3),
            D(7) => In7(3),

            EN => '1',
            Y => OutMux(3)
            );


Control0 <= Control;

end Behavioral;
```

MUX_8_TO_1.VHD

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

entity Mux_8_to_1 is
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
```

```
          D : in STD_LOGIC_VECTOR (7 downto 0);
          EN : in STD_LOGIC;
          Y : out STD_LOGIC);
end Mux_8_to_1;

architecture Behavioral of Mux_8_to_1 is

component Decoder_3_to_8 port(
    I   : in STD_LOGIC_VECTOR;
    EN  : in STD_LOGIC;
    Y   : out STD_LOGIC_VECTOR);
end component;

signal Dec_out, And_out : STD_LOGIC_VECTOR(7 downto 0);
signal Or_out : STD_LOGIC;

begin

Decoder_3_to_8_0 : Decoder_3_to_8 port map(
    I => S,
    EN => EN,
    Y => Dec_out );

    And_out <= Dec_out AND D;
    Or_out <= And_out(0) OR And_out(1) OR And_out(2) OR And_out(3) OR
         And_out(4) OR And_out(5) OR And_out(6) OR And_out(7);
    Y <= EN and Or_out;

end Behavioral;
```

(Decoder3to8 added previously)

Adder 3 bit

ADDER3BIT.VHD

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

entity Adder3Bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           output : out STD_LOGIC_VECTOR (2 downto 0));
end Adder3Bit;

architecture Behavioral of Adder3Bit is

component FA port (
    A : in std_logic;
```

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

SIGNAL FA_C : std_logic_vector(2 downto 0);

begin

FA_0 : FA port map (
    A => A(0),
    B => B(0),
    S => output(0),
    C_in => '0',
    C_out => FA_C(0));

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

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

end Behavioral;
```

(FA added previously)

Mux 2 Way 3 Bit

MUX2WAY3BIT.VHD

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


entity Mux2Way3Bit is
    Port ( In0 : in STD_LOGIC_VECTOR (2 downto 0);
           In1 : in STD_LOGIC_VECTOR (2 downto 0);
           LoadSelect : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (2 downto 0));
end Mux2Way3Bit;
```

```
architecture Behavioral of Mux2Way3Bit is

signal s0,s1:std_logic_vector(2 downto 0);

begin

    s0(0) <= (NOT(LoadSelect)) AND In0(0);
    s0(1) <= (NOT(LoadSelect)) AND In0(1);
    s0(2) <= (NOT(LoadSelect)) AND In0(2);

    s1(0) <= (LoadSelect) AND In1(0);
    s1(1) <= (LoadSelect) AND In1(1);
    s1(2) <= (LoadSelect) AND In1(2);

    output <= s0 OR s1;

end Behavioral;
```

Mux 2 Way 4 Bit

MUX2WAY4BIT.VHD

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

entity Mux2Way4Bit is
    Port ( In0 : in STD_LOGIC_VECTOR (3 downto 0);
           In1 : in STD_LOGIC_VECTOR (3 downto 0);
           LoadSelect : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (3 downto 0));
end Mux2Way4Bit;

architecture Behavioral of Mux2Way4Bit is

signal s0,s1:std_logic_vector(3 downto 0);

begin

    s0(0) <= (NOT(LoadSelect)) AND In0(0);
    s0(1) <= (NOT(LoadSelect)) AND In0(1);
    s0(2) <= (NOT(LoadSelect)) AND In0(2);
    s0(3) <= (NOT(LoadSelect)) AND In0(3);

    s1(0) <= (LoadSelect) AND In1(0);
    s1(1) <= (LoadSelect) AND In1(1);
    s1(2) <= (LoadSelect) AND In1(2);
    s1(3) <= (LoadSelect) AND In1(3);

    output <= s0 OR s1;
```

```
end Behavioral;
```

Program Counter

PROGRAMCOUNTER.VHD

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

entity ProgramCounter is
    Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
           output : out STD_LOGIC_VECTOR (2 downto 0);
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC);
end ProgramCounter;

architecture Behavioral of ProgramCounter is

begin

process (Clk) begin
    if (rising_edge(Clk)) then
        if Reset = '1' then
            output <= "000";
        else
            output <= input;
        end if;
    end if;
end process;


end Behavioral;
```

Slow Clock

SLOWCLOCK.VHD

```
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 = 100000000) then -- Roughly 2 to 3 seconds
                clk_status <= not(clk_status);
                Clk_out <= clk_status;
                count <= 1;
            end if;
        end if;
    end process;

end Behavioral;
```

LUT_16_7

LUT_16_7.VHD

```
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;
```

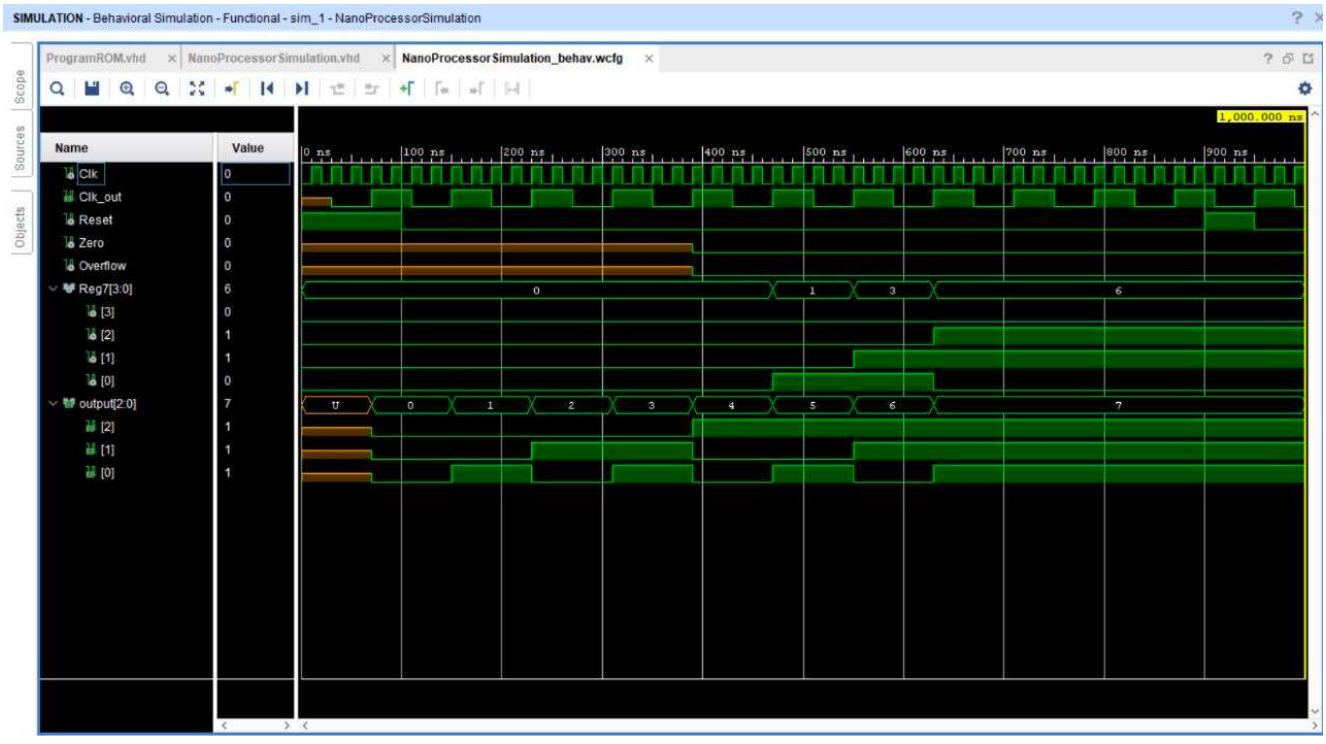
Simulation Sources

Nano Processor

NANOPROCESSORSIMULATION.VHD

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

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

architecture Behavioral of NanoProcessorSimulation is

component NanoProcessor port(
           Clk : in STD_LOGIC;
           Reset : in STD_LOGIC;
           Reg7 : out STD_LOGIC_VECTOR (3 downto 0);
           Zero : out STD_LOGIC;
           Overflow : out STD_LOGIC;
           SevenSegment : out STD_LOGIC_VECTOR(6 downto 0);
           AnodeSelect : out STD_LOGIC_VECTOR(3 downto 0);
           SwitchIn : in STD_LOGIC_VECTOR(3 downto 0));
end component;

SIGNAL Clk,Reset,Zero,Overflow : std_logic;
SIGNAL Reg7,AnodeSelect,SwitchIn : std_logic_vector(3 downto 0);
SIGNAL SevenSegment : std_logic_vector(6 downto 0);

begin

UUT : NanoProcessor port map(
           Clk => Clk,
           Reset => Reset,
           Reg7 => Reg7,
           Zero => Zero,
           Overflow => Overflow,
           SevenSegment => SevenSegment,
           AnodeSelect => AnodeSelect,
           SwitchIn => SwitchIn
);

process
```

```
begin

    Clk <= '0';
    wait for 5ns;

    Clk <= '1';
    wait for 5ns;

end process;

process
begin
    SwitchIn <= "0110";

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

    Reset <= '0';
    wait for 800ns;

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

end process;

end Behavioral;
```

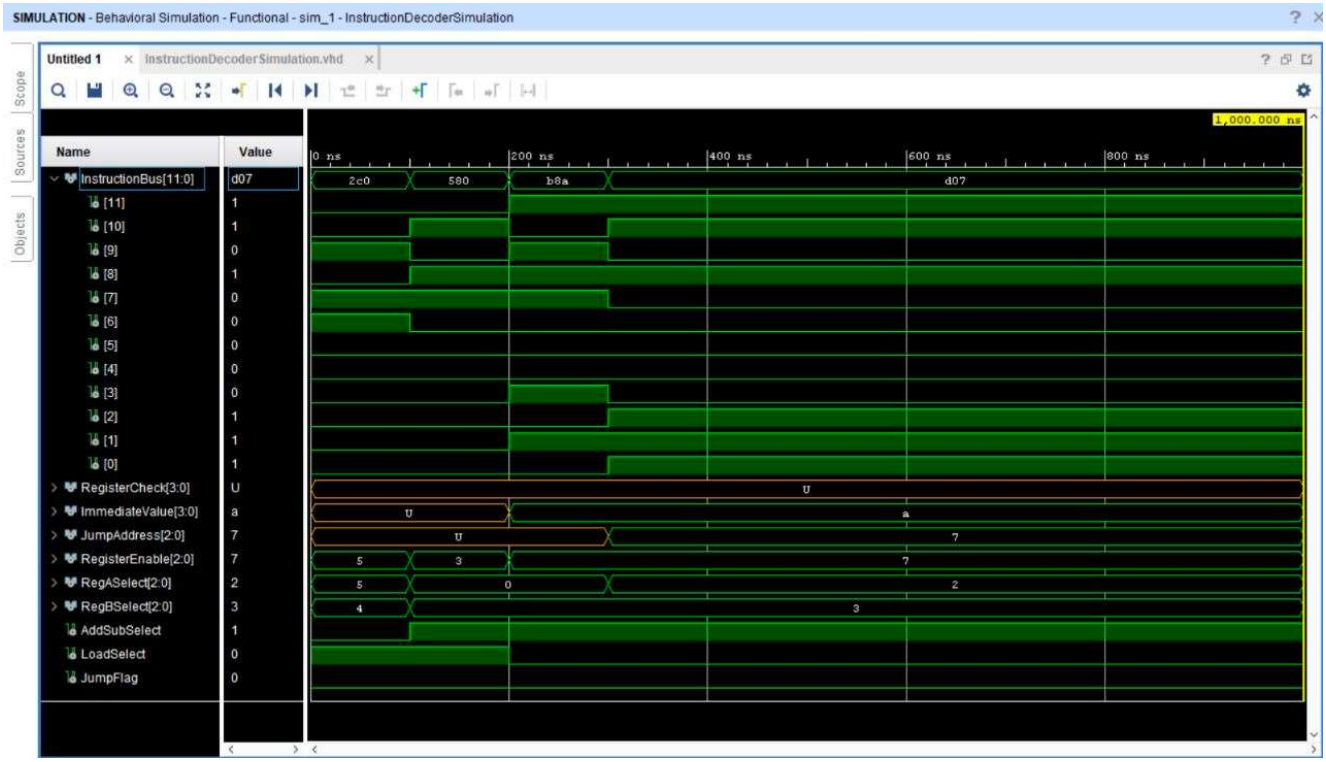


Instruction Decoder

INSTRUCTIONDECODERSIMULATION.VHD

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

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

architecture Behavioral of InstructionDecoderSimulation is

Component InstructionDecoder
    Port (  InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
            RegisterCheck : in STD_LOGIC_VECTOR (3 downto 0);
            JumpAddress : out STD_LOGIC_VECTOR (2 downto 0);
            AddSubSelect : out STD_LOGIC;
            RegisterEnable : out STD_LOGIC_VECTOR (2 downto 0);
            LoadSelect : out STD_LOGIC;
            ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
            RegASelect : out STD_LOGIC_VECTOR (2 downto 0);
            RegBSelect : out STD_LOGIC_VECTOR (2 downto 0);
            JumpFlag : out STD_LOGIC);
end component;

Signal InstructionBus : STD_LOGIC_VECTOR (11 downto 0);
Signal RegisterCheck, ImmediateValue : STD_LOGIC_VECTOR (3 downto 0);
Signal JumpAddress, RegisterEnable, RegASelect, RegBSelect : STD_LOGIC_VECTOR (2 downto 0);
Signal AddSubSelect, LoadSelect, JumpFlag : STD_LOGIC;

begin

UUT: InstructionDecoder port map(
        InstructionBus => InstructionBus,
        RegisterCheck => RegisterCheck,
        JumpAddress => JumpAddress,
        AddSubSelect => AddSubSelect,
        RegisterEnable => RegisterEnable,
        LoadSelect => LoadSelect,
        ImmediateValue => ImmediateValue,
        RegASelect => RegASelect,
        RegBSelect => RegBSelect,
        JumpFlag => JumpFlag);


process begin
    InstructionBus <= "001011000000";
    wait for 100ns;

    InstructionBus <= "010110000000";
    wait for 100ns;
```

```
    InstructionBus <= "101110001010";
    wait for 100ns;

    InstructionBus <= "110100000111";
    wait;

end process;

end Behavioral;
```

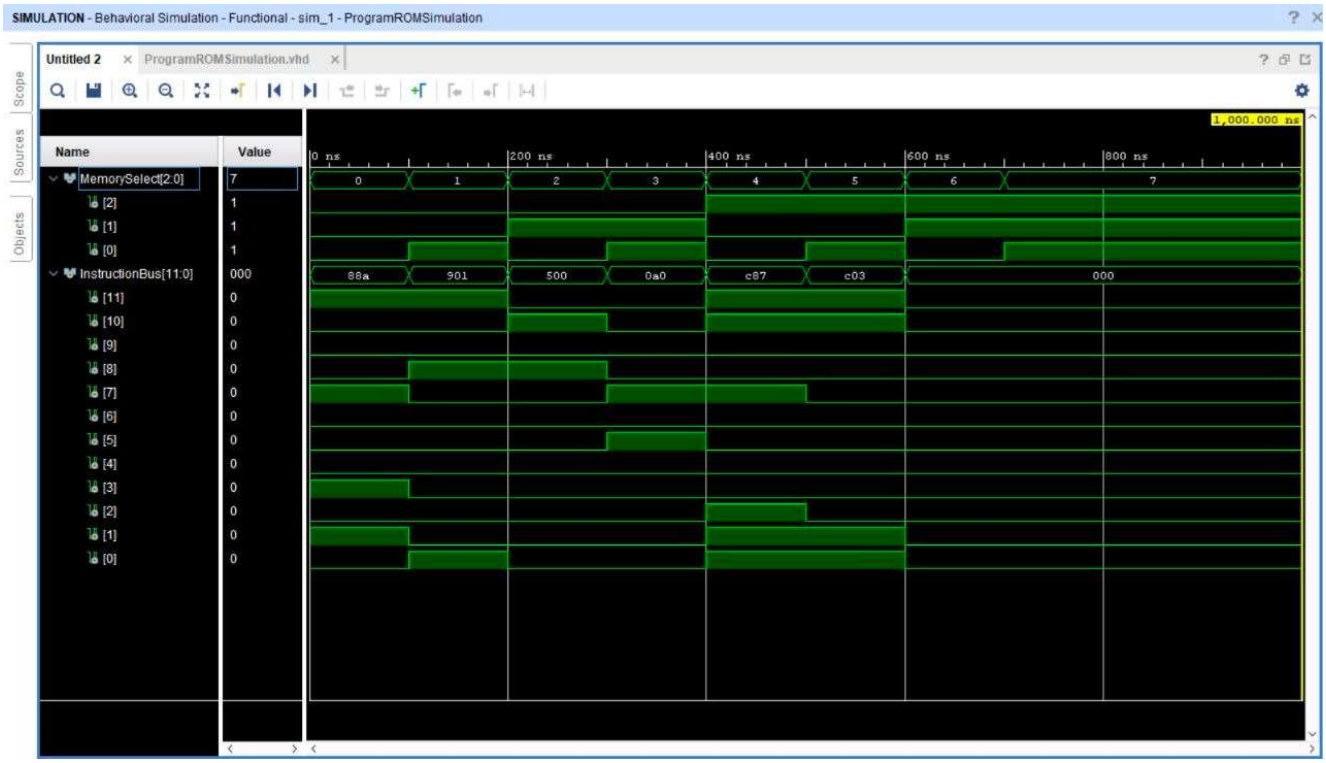


Program ROM

PROGRAMROMSIMULATION.VHD

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

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

architecture Behavioral of ProgramROMSimulation is

component ProgramROM
        Port ( MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
           InstructionBus : out STD_LOGIC_VECTOR (11 downto 0));
```

```
end component;

SIGNAL MemorySelect : STD_LOGIC_VECTOR (2 downto 0);
SIGNAL InstructionBus : STD_LOGIC_VECTOR (11 downto 0);

begin

UUT : ProgramROM port map(
        MemorySelect => MemorySelect,
        InstructionBus => InstructionBus
);

process
begin

MemorySelect <= "000";
wait for 100ns;

MemorySelect <= "001";
wait for 100ns;

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

MemorySelect <= "011";
wait for 100ns;

MemorySelect <= "100";
wait for 100ns;

MemorySelect <= "101";
wait for 100ns;

MemorySelect <= "110";
wait for 100ns;

MemorySelect <= "111";
wait;

end process;


end Behavioral;
```



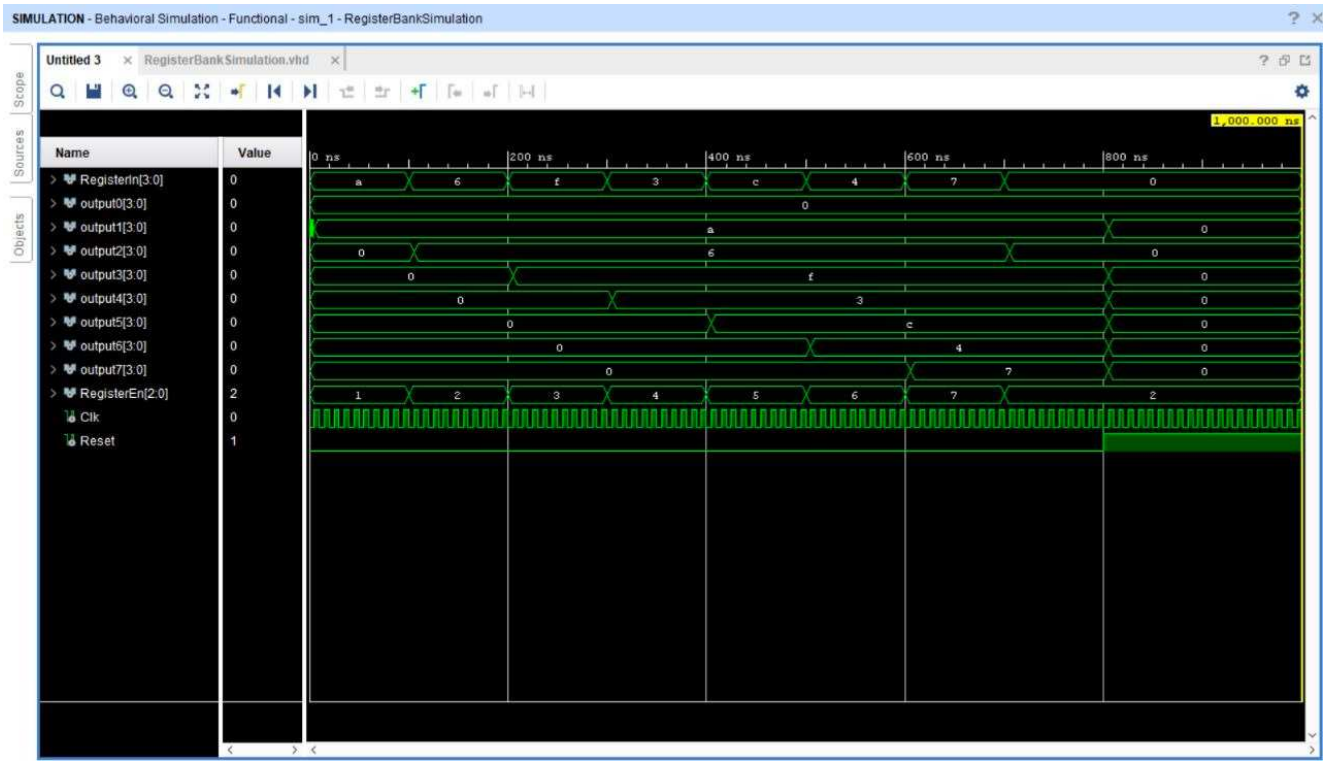
Register Bank

REGISTERBANKSIMULATION.VHD

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


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

architecture Behavioral of RegisterBankSimulation is

component RegisterBank port(
           RegisterIn : in STD_LOGIC_VECTOR (3 downto 0);
           Clk : in STD_LOGIC;
           output0 : out STD_LOGIC_VECTOR (3 downto 0);
           output1 : out STD_LOGIC_VECTOR (3 downto 0);
           output2 : out STD_LOGIC_VECTOR (3 downto 0);
           output3 : out STD_LOGIC_VECTOR (3 downto 0);
           output4 : out STD_LOGIC_VECTOR (3 downto 0);
           output5 : out STD_LOGIC_VECTOR (3 downto 0);
           output6 : out STD_LOGIC_VECTOR (3 downto 0);
           output7 : out STD_LOGIC_VECTOR (3 downto 0);
           RegisterEn : in STD_LOGIC_VECTOR (2 downto 0);
           Reset : in STD_LOGIC);

end component;
```

```
SIGNAL RegisterIn, output0, output1, output2, output3,  output4, output5, output6, output7
: std_logic_vector(3 downto 0);
SIGNAL RegisterEn : std_logic_vector(2 downto 0);
SIGNAL Clk,Reset : std_logic;

begin

UUT: RegisterBank port map(
          RegisterIn => RegisterIn,
          Clk => Clk,
          output0 => output0,
          output1 => output1,
          output2 => output2,
          output3 => output3,
          output4 => output4,
          output5 => output5,
          output6 => output6,
          output7 => output7,
          RegisterEn => RegisterEn,
          Reset => Reset
);

process begin

    Clk <= '0';
    wait for 5ns;

    Clk <= '1';
    wait for 5ns;

end process;

process begin

    Reset <= '0';

    RegisterEn <= "001";
    RegisterIn <= "1010";
    wait for 100 ns;

    RegisterEn <= "010";
    RegisterIn <= "0110";
    wait for 100 ns;

    RegisterEn <= "011";
    RegisterIn <= "1111";
    wait for 100 ns;

    RegisterEn <= "100";
    RegisterIn <= "0011";
    wait for 100 ns;
```

```
    RegisterEn <= "101";
    RegisterIn <= "1100";
    wait for 100 ns;

    RegisterEn <= "110";
    RegisterIn <= "0100";
    wait for 100 ns;

    RegisterEn <= "111";
    RegisterIn <= "0111";
    wait for 100 ns;

     RegisterEn <= "010";
     RegisterIn <= "0000";
     wait for 100 ns;

     Reset <= '1';
     wait;


end process;

end Behavioral;
```

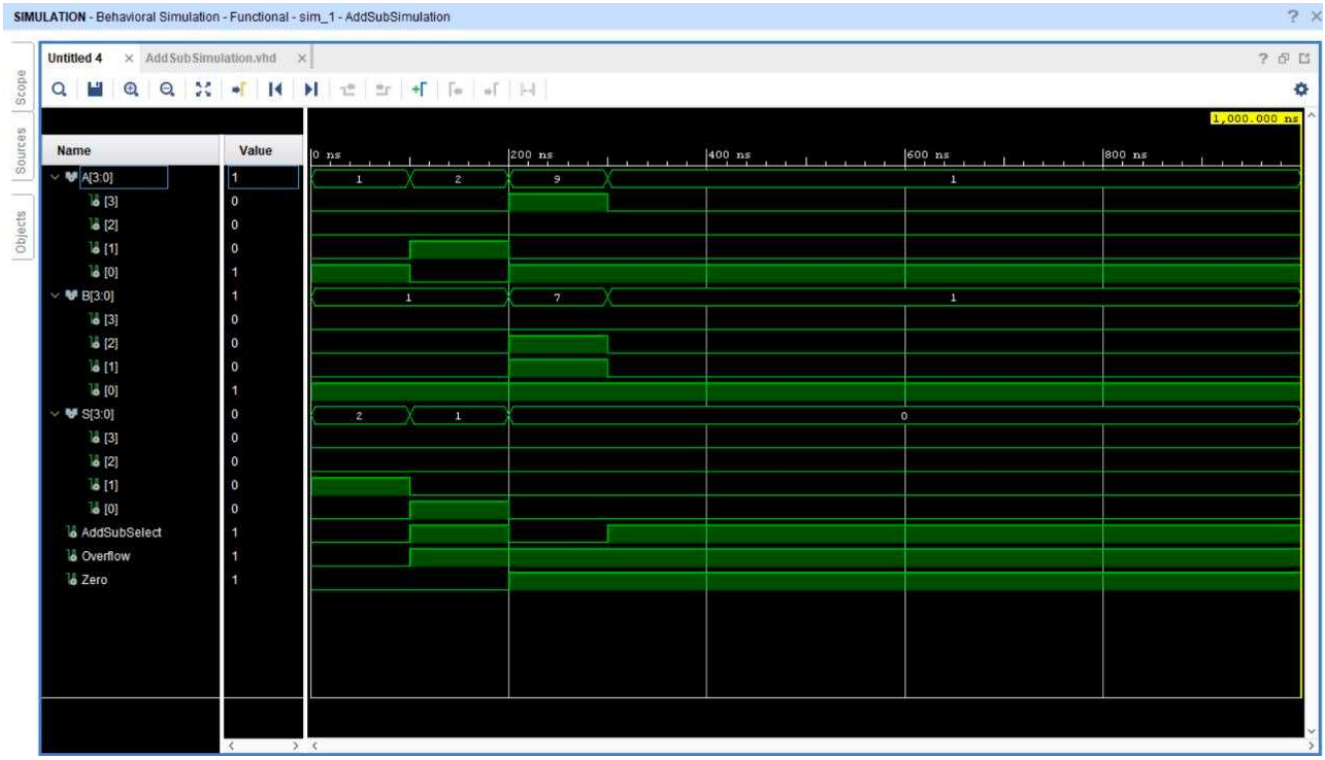


Add Sub Unit

ADDSUBSIMULATION.VHD

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

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

architecture Behavioral of AddSubSimulation is

component AddSubUnit
    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);
           AddSubSelect : in STD_LOGIC;
           Overflow : out STD_LOGIC;
           Zero : out STD_LOGIC);
end component;

SIGNAL A,B,S : std_logic_vector(3 downto 0);
SIGNAL AddSubSelect,Overflow,Zero : std_logic;

begin

UUT : AddSubUnit
    port map(
        A => A,
        B => B,
        S => S,
        AddSubSelect => AddSubSelect,
        Overflow => Overflow,
        Zero => Zero
    );

process
begin

A <= ("0001");
B <= ("0001");
AddSubSelect <= '0';
wait for 100ns;

A <= ("0010");
B <= ("0001");
AddSubSelect <= '1';
wait for 100ns;

A <= ("1001");
B <= ("0111");
```

```
AddSubSelect <= '0';
wait for 100ns;

A <= ("0001");
B <= ("0001");
AddSubSelect <= '1';
wait;

end process;


end Behavioral;
```



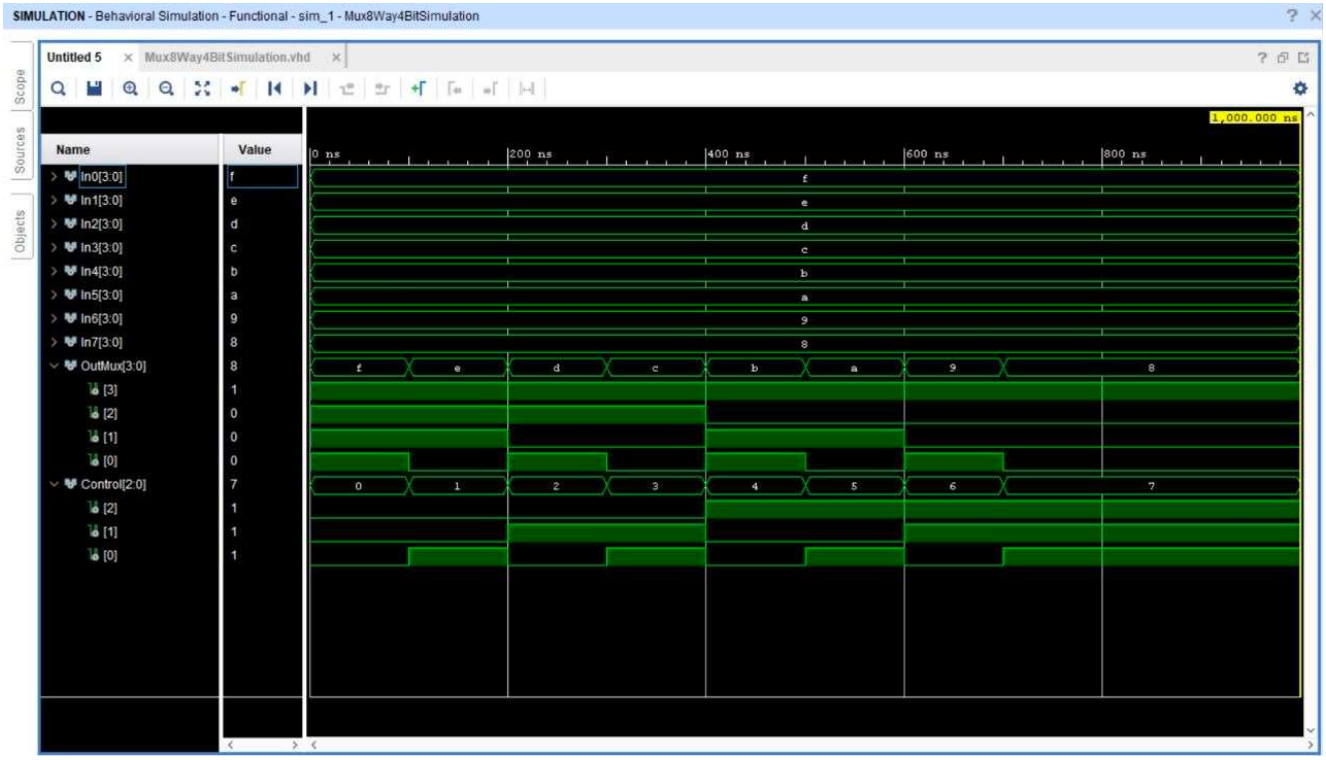
Mux 8 Way 4 Bit

MUX8WAY4BITSIMULATION.VHD

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

entity Mux8Way4BitSimulation is
--  Port ( );
end Mux8Way4BitSimulation;

architecture Behavioral of Mux8Way4BitSimulation is

component Mux8Way4Bit Port(
          In0 : in STD_LOGIC_VECTOR (3 downto 0);
          In1 : in STD_LOGIC_VECTOR (3 downto 0);
```

```
           In2 : in STD_LOGIC_VECTOR (3 downto 0);
           In3 : in STD_LOGIC_VECTOR (3 downto 0);
           In4 : in STD_LOGIC_VECTOR (3 downto 0);
           In5 : in STD_LOGIC_VECTOR (3 downto 0);
           In6 : in STD_LOGIC_VECTOR (3 downto 0);
           In7 : in STD_LOGIC_VECTOR (3 downto 0);
           OutMux : out STD_LOGIC_VECTOR (3 downto 0);
           Control : in STD_LOGIC_VECTOR (2 downto 0));
end component;

SIGNAL In0,In1,In2,In3,In4,In5,In6,In7,OutMux : std_logic_vector(3 downto 0);
SIGNAL Control : std_logic_vector(2 downto 0);

begin

UUT : Mux8Way4Bit Port map(
           In0 => In0,
           In1 => In1,
           In2 => In2,
           In3 => In3,
           In4 => In4,
           In5 => In5,
           In6 => In6,
           In7 => In7,
           OutMux => OutMux,
           Control => Control
           );
process
begin

In0 <= "1111";
In1 <= "1110";
In2 <= "1101";
In3 <= "1100";
In4 <= "1011";
In5 <= "1010";
In6 <= "1001";
In7 <= "1000";

Control <= "000";
wait for 100ns;

Control <= "001";
wait for 100ns;

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

Control <= "011";
wait for 100ns;
```

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

Control <= "101";
wait for 100ns;

Control <= "110";
wait for 100ns;

Control <= "111";
wait;

end process;

end Behavioral;
```



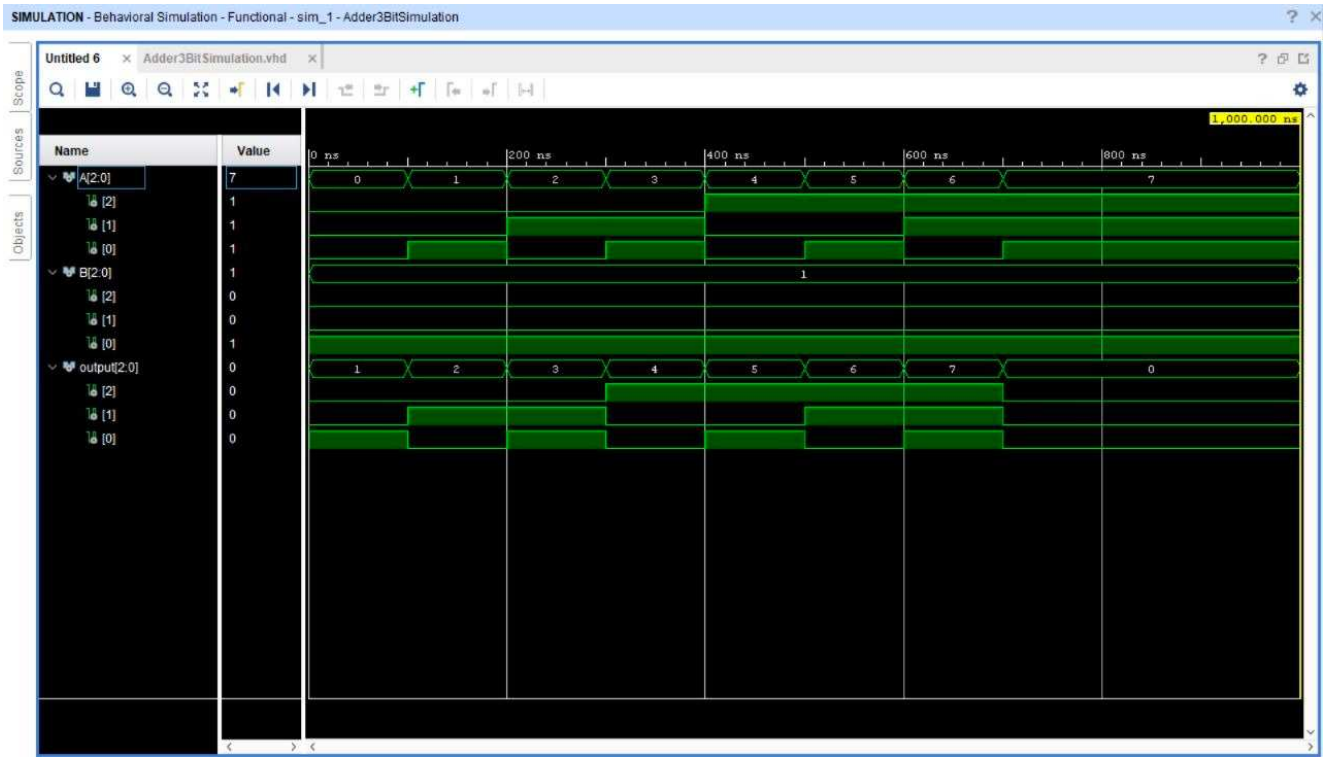
Adder 3 bit

ADDER3BITSIMULATION.VHD

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

entity Adder3BitSimulation is
--  Port ( );
end Adder3BitSimulation;

architecture Behavioral of Adder3BitSimulation is

component Adder3Bit Port (
```

```
           A : in STD_LOGIC_VECTOR (2 downto 0);
           B : in STD_LOGIC_VECTOR (2 downto 0);
           output : out STD_LOGIC_VECTOR (2 downto 0));

end component;

SIGNAL A,B,output : std_logic_vector( 2 downto 0);

begin

UUT : Adder3Bit port map(
        A => A,
        B => B,
        output => output
);

process
begin

A <= "000";
B <= "001";

wait for 100ns;

A <= "001";
B <= "001";

wait for 100ns;

A <= "010";
B <= "001";

wait for 100ns;

A <= "011";
B <= "001";

wait for 100ns;


A <= "100";
B <= "001";

wait for 100ns;


A <= "101";
B <= "001";

wait for 100ns;
```

```
A <= "110";
B <= "001";

wait for 100ns;

A <= "111";
B <= "001";

wait;

end process;


end Behavioral;
```

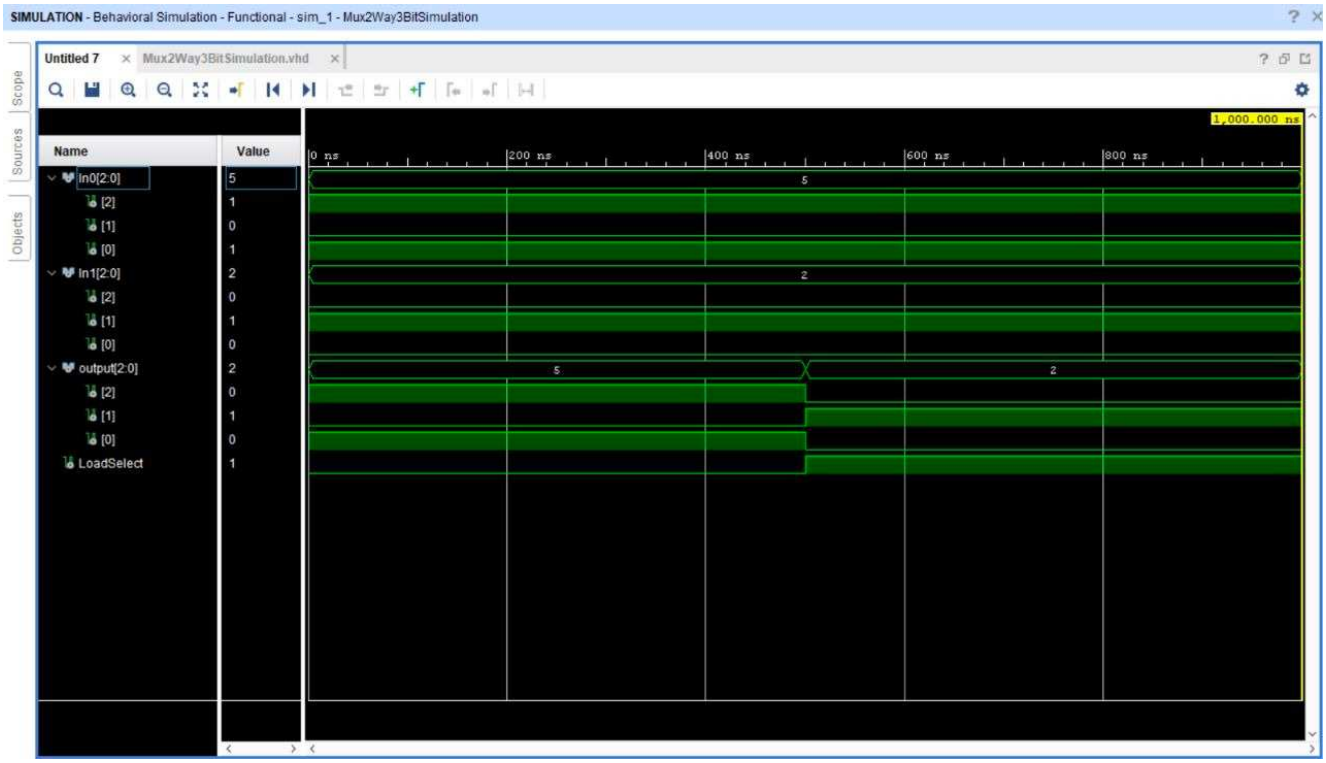


Mux 2 Way 3 Bit

MUX2WAY3BITSIMULATION.VHD

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


entity Mux2Way3BitSimulation is
--  Port ( );
end Mux2Way3BitSimulation;

architecture Behavioral of Mux2Way3BitSimulation is

component Mux2Way3Bit Port(
          In0 : in STD_LOGIC_VECTOR (2 downto 0);
```

```
           In1 : in STD_LOGIC_VECTOR (2 downto 0);
           LoadSelect : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR (2 downto 0));
end component;

SIGNAL In0,In1,output : std_logic_vector(2 downto 0);
SIGNAL LoadSelect : std_logic;

begin

UUT : Mux2Way3Bit Port map(
           In0 => In0,
           In1 => In1,
           LoadSelect => LoadSelect,
           output => output
           );

process
begin

In0 <= "101";
In1 <= "010";

LoadSelect <= '0';
wait for 500ns;
LoadSelect <= '1';
wait;


end process;

end Behavioral;
```

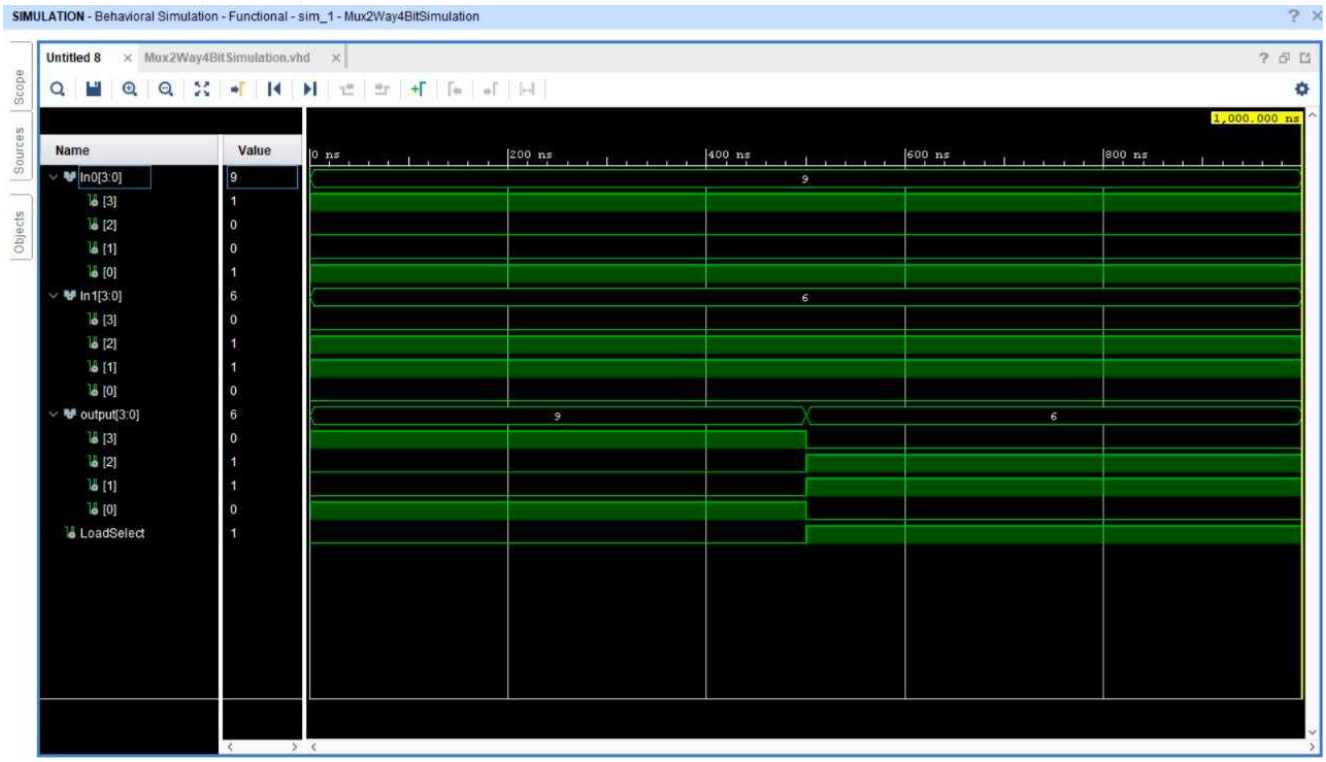


Mux 2 Way 4 Bit

MUX2WAY4BITSIMULATION.VHD

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

entity Mux2Way4BitSimulation is
    --  Port ( );
end Mux2Way4BitSimulation;

architecture Behavioral of Mux2Way4BitSimulation is

component Mux2Way4Bit Port(
          In0 : in STD_LOGIC_VECTOR (3 downto 0);
          In1 : in STD_LOGIC_VECTOR (3 downto 0);
          LoadSelect : in STD_LOGIC;
          output : out STD_LOGIC_VECTOR (3 downto 0));
end component;

SIGNAL In0,In1,output : std_logic_vector(3 downto 0);
SIGNAL LoadSelect : std_logic;

begin

UUT : Mux2Way4Bit Port map(
          In0 => In0,
          In1 => In1,
```

```
            LoadSelect => LoadSelect,
            output => output
            );

process
begin

In0 <= "1001";
In1 <= "0110";

LoadSelect <= '0';
wait for 500ns;
LoadSelect <= '1';
wait;


end process;


end Behavioral;
```



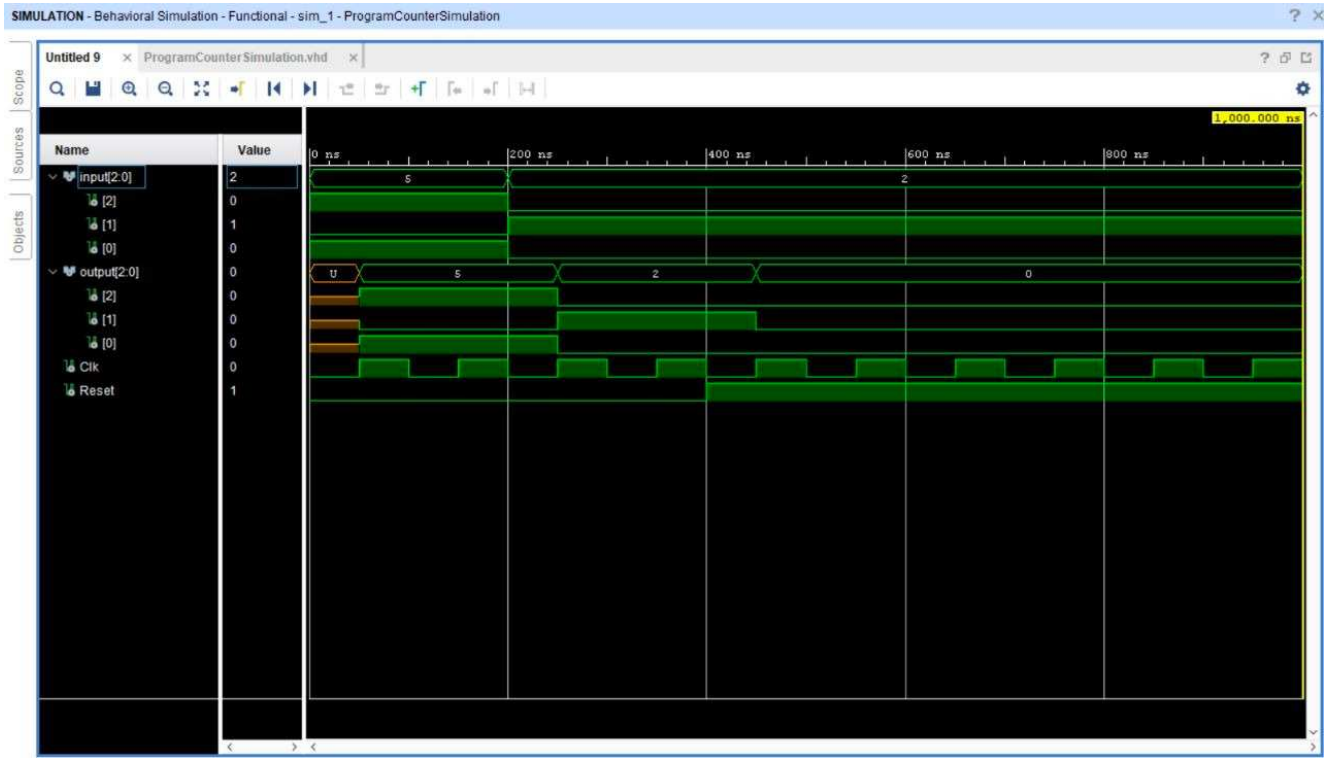
Program Counter

PROGRAMCOUNTERSIMULATION.VHD

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

entity ProgramCounterSimulation is
--  Port ( );
```

```
end ProgramCounterSimulation;

architecture Behavioral of ProgramCounterSimulation is

component ProgramCounter
     Port ( input : in STD_LOGIC_VECTOR (2 downto 0);
          output : out STD_LOGIC_VECTOR (2 downto 0);
          Clk : in STD_LOGIC;
          Reset : in STD_LOGIC);

end component;

SIGNAL input,output : std_logic_vector (2 downto 0);
SIGNAL Clk,Reset : std_logic;

begin

UUT : ProgramCounter port map(
    input => input,
    output => output,
    Clk => Clk,
    Reset => Reset
    );


process
begin
Clk <= '0';
wait for 50ns;

Clk <= '1';
wait for 50ns;


end process;


process
begin

reset <= '0';

input <= "101";
wait for 200ns;

input <= "010";
wait for 200ns;

reset <= '1';
wait;
```

```
end process;


end Behavioral;
```



CONCLUSIONS

From this lab we were able to sharpen our knowledge on the VHDL, Vivado and Computer digital design. Also, we were able to design and develop a 4-bit arithmetic unit that can add and subtract signed integers, decode instructions to activate necessary components on the processor, design and develop k-way b-bit multiplexers. verify their functionality via simulation and on the development board. This gave a good understanding on Nano Processors.

CONTRIBUTION FROM TEAM MEMBERS

Gathsara J.A.S –

- Making Design Sources and coding them.
- Made Upgrades on Reprogrammable design.

Gallage D. –

- Making the simulation files and testing the test bench.
- Made Upgrades on Additional Instructions adding.
- Made Seven Segment Up and Running.

NUMBER OF HOURS SPENT

Working about 28 hours Total.