Nano Processor Design Competition

CS1050 – Computer Organization and Digital Design

Group – 60

Vithurshan P.   210676M

Navaneethan B.   210408V

Lab Tasks

- **Designing a 4-bit processor capable of executing 4 instructions.**
- **Designing and developing a 4-bit arithmetic unit that can add and subtract signed integers.**
- **Decoding instructions to activate necessary components on the processor.**
- **Designing and developing k-way b-bit multiplexers or tri-state busses.**
- **Verifying their functionality via simulation and on the development board.**

Assembly program and Machine Code representation

Assembly program:

```
MOVI   R5, 3     ; Store "0011" in R5
MOVI   R6, 1     ; Store "0001" in R6
NEG    R6        ; Store 2's complement of R6 in R6
ADD    R7, R5    ; Add R7 with R5 and Store in R7
ADD    R5, R6    ; Add R5 with R6 and Store in R5
JZR    R5, 7     ; Jump to instruction 111, if R5 holds "0000"
JZR    R0, 3     ; Jump to instruction 011, if R0 holds "0000"
JZR    R5, 7     ; Jump to instruction 111, if R5 holds "0000"
```

Machine Code representation:

```
"101010000011"   -- MOVI   R5, 3
"101100000001"   -- MOVI   R6, 1
"011100000000"   -- NEG    R6
"001111010000"   -- ADD    R7, R5
"001011100000"   -- ADD    R5, R6
"111010000111"   -- JZR    R5, 7
"110000000011"   -- JZR    R0, 3
"111010000111"   -- JZR    R5, 7
```

VHDL Source Codes

1. 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 Add_Sub_Unit_4Bit is
Port ( A:in std_logic_vector(3 downto 0);
       B :in std_logic_vector(3 downto 0);
       Selector : in std_logic;
       S: out std_logic_vector(3 downto 0);
       Zero:out std_logic;
       Overflow : out std_logic);
end Add_Sub_Unit_4Bit;
architecture Behavioral of Add_Sub_Unit_4Bit is
 component FA
 port (
 A: in std_logic;
 B: in std_logic;
 C_in: in std_logic;
 S: out std_logic;
 C_out: out std_logic);
 end component;

 signal FA_S, FA_C: std_logic_vector(3 downto 0);
 signal B_final:std_logic_vector(3 downto 0);
begin

   B_final(0) <= B(0) XOR Selector;
   B_final(1) <= B(1) XOR Selector;
   B_final(2) <= B(2) XOR Selector;
   B_final(3) <= B(3) XOR Selector;

    FA_0 : FA
    port map (
    A => A(0),
    B => B_final(0),
    C_in => Selector,
    S => FA_S(0),
    C_Out => FA_C(0));

    FA_1 : FA
    port map (
    A => A(1),
    B => B_final(1),
    C_in => FA_C(0),
    S => FA_S(1),
    C_Out => FA_C(1));

    FA_2 : FA
    port map (
    A => A(2),
    B => B_final(2),
    C_in => FA_C(1),
    S => FA_S(2),
    C_Out => FA_C(2));

    FA_3 : FA
    port map (
    A => A(3),
    B => B_final(3),
    C_in => FA_C(2),
    S => FA_S(3),
    C_Out => FA_C(3));

Zero <= NOT(FA_S(0) OR FA_S(1) OR FA_S(2) OR FA_S(3));
S <= FA_S;
Overflow <= FA_C(2) XOR FA_C(3);
end Behavioral;
```

2. 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 Adder_3bit is
   Port ( memAddrInp: in std_logic_vector(2 downto 0 );
        memAddrOut: out std_logic_vector(2 downto 0)
       );
end Adder_3bit;

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

signal FA_C : std_logic_vector(2 downto 0);
signal FA_S : std_logic_vector(2 downto 0);

begin
   FA_0 : FA
    port map (
    A => memAddrInp(0),
    B => '0',
    C_in => '1',
    S => FA_S(0),
    C_Out => FA_C(0));

    FA_1 : FA
    port map (
    A => memAddrInp(1),
    B => '0',
    C_in => FA_C(0),
    S => FA_S(1),
    C_Out => FA_C(1));

    FA_2 : FA
    port map (
    A => memAddrInp(2),
    B => '0',
    C_in => FA_C(1),
    S => FA_S(2),
    C_out => FA_C(2)
    );

memAddrOut <= FA_S;

end Behavioral;
```

3. 3-bit 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 ( Clk : in std_logic;
         Address: in std_logic_vector(2 downto 0);
         Reset: in std_logic;
```

```
          MemSelect: 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;
          Qbar : out STD_LOGIC;
          Enable: in std_logic:='1'
          );
end component;

begin

D_FF0:D_FF
   port map(
    D => Address(0),
    Res => Reset,
    Clk => Clk,
    Q => MemSelect(0)
    );

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

D_FF2:D_FF
   port map(
    D => Address(2),
    Res => Reset,
    Clk => Clk,
    Q => MemSelect(2)
    );

end Behavioral;
```

4. 2-way 3-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 TwoWay_3_Bit_Mux is
  Port (InputA :in std_logic_vector(2 downto 0);
        InputB :in std_logic_vector(2 downto 0);
        Selector_JumpFlag : in std_logic;
        Selected_out: out std_logic_vector(2 downto 0)
        );

end TwoWay_3_Bit_Mux;

architecture Behavioral of TwoWay_3_Bit_Mux is
   component Mux_2_to_1
        Port ( Inp1 : in STD_LOGIC;
               Inp2 : in STD_LOGIC;
               Selector : in STD_LOGIC;
               Output : out STD_LOGIC
              );
   end component;

begin
Mux_2_to_1_00:Mux_2_to_1
 port map(
        Inp1 => InputA(0),
        Inp2 => InputB(0),
        Selector => Selector_JumpFlag,
        Output => Selected_out(0)
        );

Mux_2_to_1_01:Mux_2_to_1
 port map(
        Inp1 => InputA(1),
        Inp2 => InputB(1),
        Selector => Selector_JumpFlag,
        Output => Selected_out(1)
        );

 Mux_2_to_1_02:Mux_2_to_1
         port map(
                Inp1 => InputA(2),
                Inp2 => InputB(2),
                Selector => Selector_JumpFlag,
                Output => Selected_out(2)
                );
end Behavioral;
```

5. 2-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 TwoWay_4_Bit_Mux is
Port (  InputA :in std_logic_vector(3 downto 0);
        InputB :in std_logic_vector(3 downto 0);
        Load_Selector : in std_logic;
        Selected_out: out std_logic_vector(3 downto 0)
        );
end TwoWay_4_Bit_Mux;

architecture Behavioral of TwoWay_4_Bit_Mux is
component Mux_2_to_1
        Port ( Inp1 : in STD_LOGIC;
               Inp2 : in STD_LOGIC;
               Selector : in STD_LOGIC;
               Output : out STD_LOGIC
              );
    end component;

begin
Mux_2_to_1_00:Mux_2_to_1
 port map(
        Inp1 => InputA(0),
        Inp2 => InputB(0),
        Selector => Load_Selector,
        Output => Selected_out(0)
        );

Mux_2_to_1_01:Mux_2_to_1
 port map(
        Inp1 => InputA(1),
        Inp2 => InputB(1),
        Selector => Load_Selector,
        Output => Selected_out(1)
        );

 Mux_2_to_1_02:Mux_2_to_1
         port map(
                Inp1 => InputA(2),
                Inp2 => InputB(2),
                Selector => Load_Selector,
                Output => Selected_out(2)
                );

Mux_2_to_1_03:Mux_2_to_1
         port map(
                Inp1 => InputA(3),
                Inp2 => InputB(3),
                Selector => Load_Selector,
                Output => Selected_out(3)
                );
end Behavioral;
```

6. 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 EightWay_4bit_Mux is
   Port (from_Reg0 : in std_logic_vector(3 downto 0 );
        from_Reg1 : in std_logic_vector(3 downto 0 );
        from_Reg2 : in std_logic_vector(3 downto 0 );
        from_Reg3 : in std_logic_vector(3 downto 0 );
        from_Reg4 : in std_logic_vector(3 downto 0 );
        from_Reg5 : in std_logic_vector(3 downto 0 );
        from_Reg6 : in std_logic_vector(3 downto 0 );
        from_Reg7 : in std_logic_vector(3 downto 0 );
        Register_sel: in std_logic_vector(2 downto 0);
        Selected_out: out std_logic_vector(3 downto 0)
        );

end EightWay_4bit_Mux;

architecture Behavioral of EightWay_4bit_Mux is
component Mux_8_to_1
   Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
         D : in STD_LOGIC_VECTOR (7 downto 0);
         Y : out STD_LOGIC
        );

end component;

begin

Mux_8_to_1_0: Mux_8_to_1
 port map(
      D(0) => from_Reg0(0),
      D(1) => from_Reg1(0),
      D(2) => from_Reg2(0),
      D(3) => from_Reg3(0),
      D(4) => from_Reg4(0),
      D(5) => from_Reg5(0),
      D(6) => from_Reg6(0),
      D(7) => from_Reg7(0),
      S => Register_sel,
      Y => Selected_out(0)
      );

Mux_8_to_1_1: Mux_8_to_1
 port map(
      D(0) => from_Reg0(1),
      D(1) => from_Reg1(1),
      D(2) => from_Reg2(1),
      D(3) => from_Reg3(1),
      D(4) => from_Reg4(1),
      D(5) => from_Reg5(1),
      D(6) => from_Reg6(1),
      D(7) => from_Reg7(1),
      S => Register_sel,
      Y => Selected_out(1)
      );

Mux_8_to_1_2: Mux_8_to_1
      port map(
              D(0) => from_Reg0(2),
              D(1) => from_Reg1(2),
              D(2) => from_Reg2(2),
              D(3) => from_Reg3(2),
              D(4) => from_Reg4(2),
              D(5) => from_Reg5(2),
              D(6) => from_Reg6(2),
              D(7) => from_Reg7(2),
              S => Register_sel,
              Y => Selected_out(2)
              );

Mux_8_to_1_3: Mux_8_to_1
      port map(
              D(0) => from_Reg0(3),
              D(1) => from_Reg1(3),
              D(2) => from_Reg2(3),
              D(3) => from_Reg3(3),
              D(4) => from_Reg4(3),
              D(5) => from_Reg5(3),
              D(6) => from_Reg6(3),
              D(7) => from_Reg7(3),
              S => Register_sel,
              Y => Selected_out(3)
              );
end Behavioral;
```

7. 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 Register_Bank is
   Port ( Data_Inp : in STD_LOGIC_VECTOR (3 downto 0);
         Clk : in STD_LOGIC;
         RegEnable : in STD_LOGIC_VECTOR (2 downto 0);
         Reset : in STD_LOGIC;
         out_Reg0 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg1 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg2 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg3 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg4 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg5 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg6 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg7 : out STD_LOGIC_VECTOR (3 downto 0)
         );

end Register_Bank;

architecture Behavioral of Register_Bank is

 component Regist
   Port ( Data_In : in STD_LOGIC_VECTOR(3 downto 0);
        Enable : in STD_LOGIC;
        Clk : in STD_LOGIC;
        Reset : in STD_LOGIC;
        Data_Out : out STD_LOGIC_VECTOR (3 downto 0)
       );
 end component;

 component Decode_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 Y_connect:std_logic_vector(7 downto 0);

begin
Decoder_3_to_8_00: Decode_3_to_8
   port map(
      I => RegEnable,
      EN => '1',
      Y => Y_connect
      );


Reg0: Regist
```

```
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(0),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg0
        );

Reg1: Regist
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(1),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg1
        );

Reg2: Regist
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(2),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg2
        );

Reg3: Regist
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(3),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg3
        );

Reg4: Regist
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(4),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg4
        );

Reg5: Regist
port map(
        Data_In =>Data_Inp,
        Enable => Y_connect(5),
        Clk => Clk,
        Reset => Reset,
        Data_Out => out_Reg5
        );

Reg6: Regist
```

```
port map(
	Data_In =>Data_Inp,
	Enable => Y_connect(6),
	Clk => Clk,
	Reset => Reset,
	Data_Out => out_Reg6
	);

Reg7: Regist
port map(
	Data_In =>Data_Inp,
	Enable => Y_connect(7),
	Clk => Clk,
	Reset => Reset,
	Data_Out => out_Reg7
	);
end Behavioral;
```

8. Program ROM

```
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_Rom is
   Port ( MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
          Instruction : out STD_LOGIC_VECTOR (11 downto 0));
end Program_Rom;

architecture Behavioral of Program_Rom is
type rom_type is array(0 to 7) of std_logic_vector(11 downto 0);

signal Program_Rom : rom_type := (
		"101010000011", -- MOVI R5,3
		"101100000001", -- MOVI R6,1
		"011100000000", -- NEG R6
		"001111010000", -- ADD R7,R5
		"001011100000", -- ADD R5,R6
		"111010000111", -- JZR R5,7
		"110000000011", -- JZR R0,3
		"111010000111" -- JZR R5,7

		);


begin
```

```
Instruction <= Program_Rom(to_integer(unsigned(MemorySelect)));

end Behavioral;
```

9. 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 InstructionDecoder is
   Port ( InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
          RegistCheckForJmp : in STD_LOGIC_VECTOR (3 downto 0);
          RegEnable : out STD_LOGIC_VECTOR (2 downto 0);
          LoadSel : out STD_LOGIC;
          ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
          RegSel_A : out STD_LOGIC_VECTOR (2 downto 0);
          RegSel_B : out STD_LOGIC_VECTOR (2 downto 0);
          Add_Sub_Sel : out STD_LOGIC;
          JmpFlag : out STD_LOGIC;
          AddrToJmp : out STD_LOGIC_VECTOR (2 downto 0));
end InstructionDecoder;

architecture Behavioral of InstructionDecoder 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 jumpCondition1,Y0:std_logic;
begin

OpSelector : Decoder_2_to_4
port map(
   I  => InstructionBus(11 downto 10),
   EN => '1',
   Y(0) => Y0, --add
   Y(1) => Add_Sub_Sel, -- neg
   Y(2) => LoadSel,     --movi
   Y(3) => jumpCondition1);   --jzr

JmpFlag <= jumpCondition1 AND (NOT( RegistCheckForJmp(3) OR  RegistCheckForJmp(2) OR  RegistCheckForJmp(1) OR 
RegistCheckForJmp(0)));
```

```
RegEnable <= InstructionBus(9 downto 7 );
ImmediateValue <= InstructionBus(3 downto 0);
RegSel_B <= InstructionBus(6 downto 4);
RegSel_A <= InstructionBus(9 downto 7);
AddrToJmp <= InstructionBus(2 downto 0);

end Behavioral;
```

10. Look-up Table

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

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

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

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

architecture Behavioral of LUT_16_7 is

type rom_type is array (0 to 15) of std_logic_vector(6 downto 0);

   signal sevenSegment_ROM : rom_type := (
                  "1000000", -- 0
                  "1111001", -- 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;
```

11. 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 = 80000000) then  --use this to test on board
         --if(count= 4) then  -- use this in simulations
            clk_status <= not clk_status;
            Clk_out <= clk_status;
            count <= 1;
         end if;
      end if;
   end process;
end Behavioral;
```

12. 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_STD.ALL;

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

```
entity Nano_Processor is
   Port ( Clock : in STD_LOGIC;
          RESET : in STD_LOGIC;
          Overflow : out STD_LOGIC;
          Zero : out STD_LOGIC;
          LED : out STD_LOGIC_VECTOR (3 downto 0);
          SevenSegDisplay: out STD_LOGIC_VECTOR (6 downto 0);
          AnodeSignal: out STD_LOGIC_VECTOR ( 3 downto 0) := "1110"
          );

end Nano_Processor;

architecture Behavioral of Nano_Processor is

component Add_Sub_Unit_4Bit
 Port ( A:in std_logic_vector(3 downto 0);
      B :in std_logic_vector(3 downto 0);
      Selector : in std_logic;
      S: out std_logic_vector(3 downto 0);
      Zero:out std_logic;
      Overflow : out std_logic);

end component;

component Adder_3bit
Port ( memAddrInp: in std_logic_vector(2 downto 0 );
          memAddrOut: out std_logic_vector(2 downto 0)
         );

end component;

component Program_Counter
Port ( Clk : in std_logic;
          Address: in std_logic_vector(2 downto 0);
          Reset: in std_logic;
          MemSelect: out std_logic_vector(2 downto 0)
          );

end component;

component EightWay_4bit_Mux
Port (from_Reg0 : in std_logic_vector(3 downto 0 );
         from_Reg1 : in std_logic_vector(3 downto 0 );
         from_Reg2 : in std_logic_vector(3 downto 0 );
         from_Reg3 : in std_logic_vector(3 downto 0 );
         from_Reg4 : in std_logic_vector(3 downto 0 );
         from_Reg5 : in std_logic_vector(3 downto 0 );
         from_Reg6 : in std_logic_vector(3 downto 0 );
         from_Reg7 : in std_logic_vector(3 downto 0 );
         Register_sel: in std_logic_vector(2 downto 0);
         Selected_out: out std_logic_vector(3 downto 0)
         );
end component;
```

```
component TwoWay_4_Bit_Mux
Port (  InputA :in std_logic_vector(3 downto 0);
       InputB :in std_logic_vector(3 downto 0);
       Load_Selector : in std_logic;
       Selected_out: out std_logic_vector(3 downto 0)
       );
end component;

component TwoWay_3_Bit_Mux
 Port (InputA :in std_logic_vector(2 downto 0);
      InputB :in std_logic_vector(2 downto 0);
      Selector_JumpFlag : in std_logic;
      Selected_out: out std_logic_vector(2 downto 0)
      );
end component;

component Register_Bank
Port ( Data_Inp : in STD_LOGIC_VECTOR (3 downto 0);
         Clk : in STD_LOGIC;
         RegEnable : in STD_LOGIC_VECTOR (2 downto 0);
         Reset : in STD_LOGIC;
         out_Reg0 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg1 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg2 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg3 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg4 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg5 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg6 : out STD_LOGIC_VECTOR (3 downto 0);
         out_Reg7 : out STD_LOGIC_VECTOR (3 downto 0)
         );
end component;

component Program_Rom
Port ( MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
         Instruction : out STD_LOGIC_VECTOR (11 downto 0));
end component;

component InstructionDecoder
Port ( InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
         RegistCheckForJmp : in STD_LOGIC_VECTOR (3 downto 0);
         RegEnable : out STD_LOGIC_VECTOR (2 downto 0);
         LoadSel : out STD_LOGIC;
         ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
         RegSel_A : out STD_LOGIC_VECTOR (2 downto 0);
         RegSel_B : out STD_LOGIC_VECTOR (2 downto 0);
         Add_Sub_Sel : out STD_LOGIC;
         JmpFlag : out STD_LOGIC;
         AddrToJmp : out STD_LOGIC_VECTOR (2 downto 0));
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;

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

signal
InpB_2Way_3Bit,addressInp_PC,InpA_2Way_3Bit,RegSelect_A,RegSelect_B,Reg_Enable,Memory_Select:std_logic_vec
tor(2 downto 0);
signal
DataToReg,from_8_Way_Mux_A,from_8_Way_Mux_B,Sum_from_Add_Sub_Unit,Imm_Value,Reg0_Data,Reg1_Data,
Reg2_Data,Reg3_Data,Reg4_Data,Reg5_Data,Reg6_Data,Reg7_Data : std_logic_vector(3 downto 0);
signal Add_Sub_Selector ,LoadSelector,JumpFlag,Slowed_Clock: std_logic;
signal Instruction_Bus:std_logic_vector(11 downto 0);

begin

Instruction_Decoder : InstructionDecoder
port map(
      InstructionBus => Instruction_Bus,
      RegistCheckForJmp => from_8_Way_Mux_A,
      RegEnable => Reg_Enable,
      LoadSel => LoadSelector,
      ImmediateValue => Imm_Value,
      RegSel_A => RegSelect_A,
      RegSel_B => RegSelect_B,
      Add_Sub_Sel => Add_Sub_Selector,
      JmpFlag  =>JumpFlag,
      AddrToJmp => InpB_2Way_3Bit
      );


ProgramROM : Program_Rom
port map(
       MemorySelect => Memory_Select,
       Instruction  => Instruction_Bus
);



ProgramCounter : Program_Counter
port map(
       Clk => Slowed_Clock,
       Address => addressInp_PC,
       Reset  => RESET,
       MemSelect => Memory_Select
       );

Two_Way_3_Bit_Mux : TwoWay_3_Bit_Mux
port map(
      InputA => InpA_2Way_3Bit,
      InputB => InpB_2Way_3Bit,
      Selector_JumpFlag  => JumpFlag,
      Selected_out => addressInp_PC
      );

Adder_3_Bit : Adder_3bit
port map(
      memAddrInp => Memory_Select,
      memAddrOut => InpA_2Way_3Bit
      );

 Two_Way_4_Bit_Mux : TwoWay_4_Bit_Mux
 port map(
       InputA => Sum_from_Add_Sub_Unit,
       InputB => Imm_Value,
       Load_Selector => LoadSelector,
       Selected_out => DataToReg
       );

RegisterBank : Register_Bank
port map(
         Data_Inp => DataToReg,
         Clk => Slowed_Clock,
         RegEnable => Reg_Enable,
         Reset  => RESET,
         out_Reg0 => Reg0_Data,
         out_Reg1 => Reg1_Data,
         out_Reg2 => Reg2_Data,
         out_Reg3 => Reg3_Data,
         out_Reg4 => Reg4_Data,
         out_Reg5 => Reg5_Data,
         out_Reg6 => Reg6_Data,
         out_Reg7 => Reg7_Data
         );

Eight_Way_4Bit_Mux_A: EightWay_4bit_Mux
port map(
        from_Reg0 => Reg0_Data,
        from_Reg1 => Reg1_Data,
        from_Reg2 => Reg2_Data,
        from_Reg3 => Reg3_Data,
        from_Reg4 => Reg4_Data,
        from_Reg5 => Reg5_Data,
        from_Reg6 => Reg6_Data,
        from_Reg7 => Reg7_Data,
        Register_sel => RegSelect_A,
        Selected_out => from_8_Way_Mux_A
        );

Eight_Way_4Bit_Mux_B: EightWay_4bit_Mux
port map(
        from_Reg0 => Reg0_Data,
        from_Reg1 => Reg1_Data,
        from_Reg2 => Reg2_Data,
        from_Reg3 => Reg3_Data,
        from_Reg4 => Reg4_Data,
         from_Reg5 => Reg5_Data,
         from_Reg6 => Reg6_Data,
         from_Reg7 => Reg7_Data,
         Register_sel => RegSelect_B,
         Selected_out => from_8_Way_Mux_B
        );

Add_Sub_Unit : Add_Sub_Unit_4Bit
port map(
      A => from_8_Way_Mux_B,
      B => from_8_Way_Mux_A,
      Selector => Add_Sub_Selector,
      S => Sum_from_Add_Sub_Unit,
      Zero => Zero,
      Overflow => Overflow
    );

LUT_for_7Seg : LUT_16_7
port map(
         address => Reg7_Data,
         data => SevenSegDisplay
         );

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

LED <= Reg7_Data;

end behavioral;
```

VHDL Testbench Codes

1. 4-bit Add/Subtract unit

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

entity Add_Sub_Unit_4Bit_Sim is
end Add_Sub_Unit_4Bit_Sim;

architecture Behavioral of Add_Sub_Unit_4Bit_Sim is
component Add_Sub_Unit_4Bit
   Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
          B : in STD_LOGIC_VECTOR (3 downto 0);
          Selector : in STD_LOGIC;
          S : out STD_LOGIC_VECTOR (3 downto 0);
          Zero : out STD_LOGIC;
          Overflow : out STD_LOGIC);
end component;

signal sel,oflw,zero: std_logic;
signal a, b, s : STD_LOGIC_VECTOR (3 downto 0);
begin

UUT: Add_Sub_Unit_4Bit
port map(
      A => a,
      B => b,
      Selector => sel,
      S => s,
      Zero => zero,
      Overflow => oflw
      );

process begin
   -- 210676: 11 0011 0110 1111 0100
   -- 210408: 1100 1101 0111 1010 00
   a <= "0100";
   b <= "1010";
   sel <= '1';
   wait for 100 ns;

   a <= "1111";
   b <= "0111";
   sel <= '0';
   wait for 100 ns;

   a <= "0110";
   b <= "1101";
   sel <= '0';
   wait for 100 ns;

   a <= "0011";
```

```
    b <= "1100";
    sel <= '1';
    wait for 100 ns;

    a <= "0011";
    b <= "1111";
    sel <= '0';
    wait for 100 ns;

    a <= "1111";
    b <= "1111";
    sel <= '1';
    wait ;
end process;

end Behavioral;
```

2. 3-bit Adder

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

entity Adder_3bit_Sim is
end Adder_3bit_Sim;

architecture Behavioral of Adder_3bit_Sim is
component Adder_3bit
   Port ( memAddrInp : in STD_LOGIC_VECTOR (2 downto 0);
          memAddrOut : out STD_LOGIC_VECTOR (2 downto 0));
end component;

signal input,output : STD_LOGIC_VECTOR (2 downto 0);
begin

UUT: Adder_3bit
port map(
      memAddrInp => input,
      memAddrOut => output
      );

process begin
   -- 210676: 110 011 011 011 110 100
   -- 210408: 110 011 010 111 101 000
   input <= "100";
   wait for 100ns;
   input <= "000";
   wait for 100ns;
   input <= "110";
   wait for 100ns;
   input <= "101";
   wait for 100ns;
   input <= "011";
   wait for 100ns;
   input <= "111";
   wait;
end process;

end Behavioral;
```

3. 3-bit Program Counter

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

entity Program_Counter_Sim is
end entity Program_Counter_Sim;

architecture Behavioral of Program_Counter_Sim is
component Program_Counter
   Port ( Clk : in std_logic;
         Address: in std_logic_vector(2 downto 0);
         Reset: in std_logic;
         MemSelect: out std_logic_vector(2 downto 0));
end component;

signal res : std_logic;
signal clk : std_logic:= '0';
signal addr,memSel : std_logic_vector (2 downto 0);

begin

UUT: Program_Counter
   port map(
      Clk =>clk,
      Address =>addr,
      Reset =>res,
      MemSelect => memSel
   );

process begin
   wait for 50 ns;
   clk <= not (clk);
end process;
process begin
   -- 210676: 110 011 011 011 110 100
   -- 210408: 110 011 010 111 101 000
   res <= '0';
   wait for 50 ns;

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

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

   addr <= "010";
   wait for 100 ns;
   res <= '1';
   wait for 100 ns;

   res <= '0';
   wait for 100 ns;

   addr <= "111";
   wait for 100 ns;

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

   addr <= "000";
   wait;
end process;

end Behavioral;
```

4. 2-way 3-bit Multiplexer

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

entity TwoWay_3_Bit_Mux_Sim is
--  Port ( );
end TwoWay_3_Bit_Mux_Sim;

architecture Behavioral of TwoWay_3_Bit_Mux_Sim is
component TwoWay_3_Bit_Mux
Port (  InputA :in std_logic_vector(2 downto 0);
        InputB :in std_logic_vector(2 downto 0);
        Selector_JumpFlag : in std_logic;
        Selected_out: out std_logic_vector(2 downto 0));
end component;

signal inA, inB, selOut: std_logic_vector(2 downto 0);
signal selJmpFlg: std_logic;
begin

UUT: TwoWay_3_Bit_Mux
port map(
         InputA => inA,
         InputB => inB,
         Selector_JumpFlag => selJmpFlg,
         Selected_out => selOut
        );

process begin
   -- 210676: 110 011 011 011 110 100
   -- 210408: 110 011 010 111 101 000
   inA <= "100";
   inB <= "000";
   selJmpFlg <= '1';
   wait for 100ns;
   inA <= "110";
   inB <= "101";
   selJmpFlg <= '0';
   wait for 100ns;

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

   inA <= "011";
   inB <= "111";
   selJmpFlg <= '0';
   wait;
end process;

end behavioral;
```

5. 2-way 4-bit Multiplexer

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

entity TwoWay_4_Bit_Mux_Sim is
--  Port ( );
end TwoWay_4_Bit_Mux_Sim;

architecture Behavioral of TwoWay_4_Bit_Mux_Sim is
component TwoWay_4_Bit_Mux
Port (  InputA :in std_logic_vector(3 downto 0);
      InputB :in std_logic_vector(3 downto 0);
      Load_Selector : in std_logic;
      Selected_out: out std_logic_vector(3 downto 0));
end component;

signal inA, inB, selOut: std_logic_vector(3 downto 0);
signal loadSel: std_logic;
begin

UUT: TwoWay_4_Bit_Mux
port map(
       InputA => inA,
       InputB => inB,
       Load_Selector => loadSel,
       Selected_out => selOut
      );

process begin
   -- 210676: 11 0011 0110 1111 0100
   -- 210408: 11 0011 0101 1110 1000
   inA <= "0100";
   inB <= "1000";
   loadSel <= '0';
   wait for 100ns;
```

```
  inA <= "1111";
  inB <= "1110";
  loadSel <= '1';
  wait for 100ns;

  loadSel <= '0';
  wait for 100ns;

  inA <= "0110";
  inB <= "0101";
  wait for 100ns;

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

  loadSel <= '0';
  wait;
end process;

end behavioral;
```

6. 8-way 4-bit Multiplexer

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

entity EightWay_4bit_Mux_Sim is
--  Port ( );
end EightWay_4bit_Mux_Sim;

architecture Behavioral of EightWay_4bit_Mux_Sim is
component EightWay_4bit_Mux
Port ( from_Reg0 : in std_logic_vector(3 downto 0 );
    from_Reg1 : in std_logic_vector(3 downto 0 );
    from_Reg2 : in std_logic_vector(3 downto 0 );
    from_Reg3 : in std_logic_vector(3 downto 0 );
    from_Reg4 : in std_logic_vector(3 downto 0 );
    from_Reg5 : in std_logic_vector(3 downto 0 );
    from_Reg6 : in std_logic_vector(3 downto 0 );
    from_Reg7 : in std_logic_vector(3 downto 0 );
    Register_sel: in std_logic_vector(2 downto 0);
    Selected_out: out std_logic_vector(3 downto 0));
end component;

signal reg_0, reg_1, reg_2, reg_3, reg_4, reg_5, reg_6, reg_7, selOut: std_logic_vector(3 downto 0);
signal regSel: std_logic_vector(2 downto 0);
begin

UUT: EightWay_4bit_Mux
port map(
     from_Reg0 => reg_0,
     from_Reg1 => reg_1,
     from_Reg2 => reg_2,
     from_Reg3 => reg_3,
     from_Reg4 => reg_4,
     from_Reg5 => reg_5,
     from_Reg6 => reg_6,
     from_Reg7 => reg_7,
     Register_sel => regSel,
     Selected_out => selOut
    );

process begin
  -- 210676: 11 0011 0110 1111 0100
  -- 210408: 1100 1101 0111 1010 00
  reg_0 <= "0100";
  reg_1 <= "1111";
  reg_2 <= "0110";
  reg_3 <= "0011";
  reg_4 <= "1100";
  reg_5 <= "1101";
  reg_6 <= "0111";
  reg_7 <= "1010";

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

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

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

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

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

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

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

  regSel <= "111";
  wait;
end process;

end behavioral;
```

7. Register Bank

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

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

```
end RegisterBank_Sim;

architecture Behavioral of RegisterBank_Sim is
component Register_Bank
Port ( Data_Inp : in STD_LOGIC_VECTOR (3 downto 0);
      Clk : in STD_LOGIC;
      RegEnable : in STD_LOGIC_VECTOR (2 downto 0);
      Reset : in STD_LOGIC;
      out_Reg0 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg1 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg2 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg3 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg4 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg5 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg6 : out STD_LOGIC_VECTOR (3 downto 0);
      out_Reg7 : out STD_LOGIC_VECTOR (3 downto 0)
        );
end component;

signal out_Reg0,out_Reg1,out_Reg2,out_Reg3,out_Reg4,out_Reg5,out_Reg6,out_Reg7, Data_Inp: std_logic_vector(3
downto 0);
signal RegEnable: std_logic_vector(2 downto 0);
signal Reset,Clk: std_logic:= '0';
begin

UUT: Register_Bank
 port map (
      Data_Inp   => Data_Inp,
      Clk        => Clk,
      RegEnable  => RegEnable,
      Reset      => Reset,
      out_Reg0   => out_Reg0,
      out_Reg1   => out_Reg1,
      out_Reg2   => out_Reg2,
      out_Reg3   => out_Reg3,
      out_Reg4   => out_Reg4,
      out_Reg5   => out_Reg5,
      out_Reg6   => out_Reg6,
      out_Reg7   => out_Reg7
  );

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

process begin
   -- 210676: 11 0011 0110 1111 0100
   -- 210408: 1100 1101 0111 1010 00

   Reset <='1';
   wait for 5ns;
   Reset <='0';
   Data_Inp <= "0100";
```

```
    RegEnable <= "000";
    wait for 50ns;

    Data_Inp <= "1111";
    RegEnable <= "001";
    wait for 50ns;

    Data_Inp <= "0110";
    RegEnable <= "010";
    wait for 50ns;

    Data_Inp <= "0011";
    RegEnable <= "011";
    wait for 50ns;

    Data_Inp <= "1100";
    RegEnable <= "100";
    wait for 50ns;

    Data_Inp <= "1101";
    RegEnable <= "101";
    wait for 50ns;

    Data_Inp <= "0111";
    RegEnable <= "110";
    wait for 50ns;

    Data_Inp <= "1010";
    RegEnable <= "111";
    wait;


end process;

end behavioral;
```

8. Program ROM

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

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

architecture Behavioral of ProgramRom_Sim is
component Program_Rom
Port ( MemorySelect : in STD_LOGIC_VECTOR (2 downto 0);
        Instruction : out STD_LOGIC_VECTOR (11 downto 0));
end component;

signal MemorySelect:STD_LOGIC_VECTOR (2 downto 0);
signal Instruction: STD_LOGIC_VECTOR (11 downto 0);
begin
```

```
UUT: Program_Rom
port map(
      MemorySelect =>MemorySelect,
      Instruction => Instruction
     );

process begin
   -- 210676: 110 011 011 011 110 100
   -- 210408: 110 011 010 111 101 000

   MemorySelect <= "100";
   wait for 50ns;

   MemorySelect <= "110";
   wait for 50ns;

   MemorySelect <= "011";
   wait for 50ns;

   MemorySelect <= "000";
   wait for 50ns;

   MemorySelect <= "101";
   wait for 50ns;

   MemorySelect <= "111";
   wait for 50ns;

   MemorySelect <= "010";
   wait for 50ns;

end process;

end behavioral;
```

9. Instruction Decoder

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


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

architecture Behavioral of InstructionDecoder_Sim is
component  InstructionDecoder
Port ( InstructionBus : in STD_LOGIC_VECTOR (11 downto 0);
     RegistCheckForJmp : in STD_LOGIC_VECTOR (3 downto 0);
     RegEnable : out STD_LOGIC_VECTOR (2 downto 0);
     LoadSel : out STD_LOGIC;
     ImmediateValue : out STD_LOGIC_VECTOR (3 downto 0);
     RegSel_A : out STD_LOGIC_VECTOR (2 downto 0);
     RegSel_B : out STD_LOGIC_VECTOR (2 downto 0);
     Add_Sub_Sel : out STD_LOGIC;
     JmpFlag : out STD_LOGIC;
     AddrToJmp : out STD_LOGIC_VECTOR (2 downto 0) );
end component;

signal ins : std_logic_vector(11 downto 0);
signal regJmpCheck, immVal : std_logic_vector(3 downto 0);
signal regEn, regSelA, regSelB, jmpAddr : std_logic_vector(2 downto 0);
signal loadSel, addSubSel, flag : std_logic;

begin

UUT: InstructionDecoder
port map (
        InstructionBus => ins,
        RegistCheckForJmp => regJmpCheck,
        RegEnable => regEn,
        LoadSel => loadSel,
        ImmediateValue => immVal,
        RegSel_A => regSelA,
        RegSel_B => regSelB,
        Add_Sub_Sel => addSubSel,
        JmpFlag => flag,
        AddrToJmp => jmpAddr
      );

process begin
   ins <= "101010000011"; -- MOVI R5, 3
   wait for 100ns;

   ins <= "101100000001"; -- MOVI R6, 1
   wait for 100ns;

   ins <= "011100000000"; -- NEG R6
   wait for 100 ns;

   ins <= "001111010000"; -- ADD R7, R5
   wait for 100 ns;

   ins <= "001011100000"; -- ADD R5, R6
   wait for 100 ns;

   ins <= "111010000111"; -- JZR R5, 7
   wait for 100 ns;

   regJmpCheck <= "0000";
   wait for 100 ns;

   regJmpCheck <= "0110";
   wait;
end process;

end Behavioral;
```

10. Look-up Table

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

architecture Behavioral of LUT_Sim is

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

end component;
signal address : STD_LOGIC_VECTOR (3 downto 0);
signal data:STD_LOGIC_VECTOR (6 downto 0);
begin

UUT : LUT_16_7
   port map(
      address => address,
      data => data
      );

process
   begin
   -- RegNo :210408
   -- In binary : 11 0011 0101 1110 1000

   address <= "1000";
   wait for 200ns;

   address <= "1110";
   wait for 200ns;

   address <= "0101";
   wait for 200ns;

   address <= "0011";
   wait;

end process;
end Behavioral;
```

11. Slow Clock

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


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

architecture Behavioral of Slow_Clk_Sim is

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

signal Clk_in:std_logic :='0';
signal Clk_out:std_logic :='0';

begin
UUT : Slow_Clk
port map(
   Clk_in => Clk_in,
   Clk_out => Clk_out
   );


process begin
   wait for 2ns;
   Clk_in <= NOT(Clk_in);


end process;

end Behavioral;
```

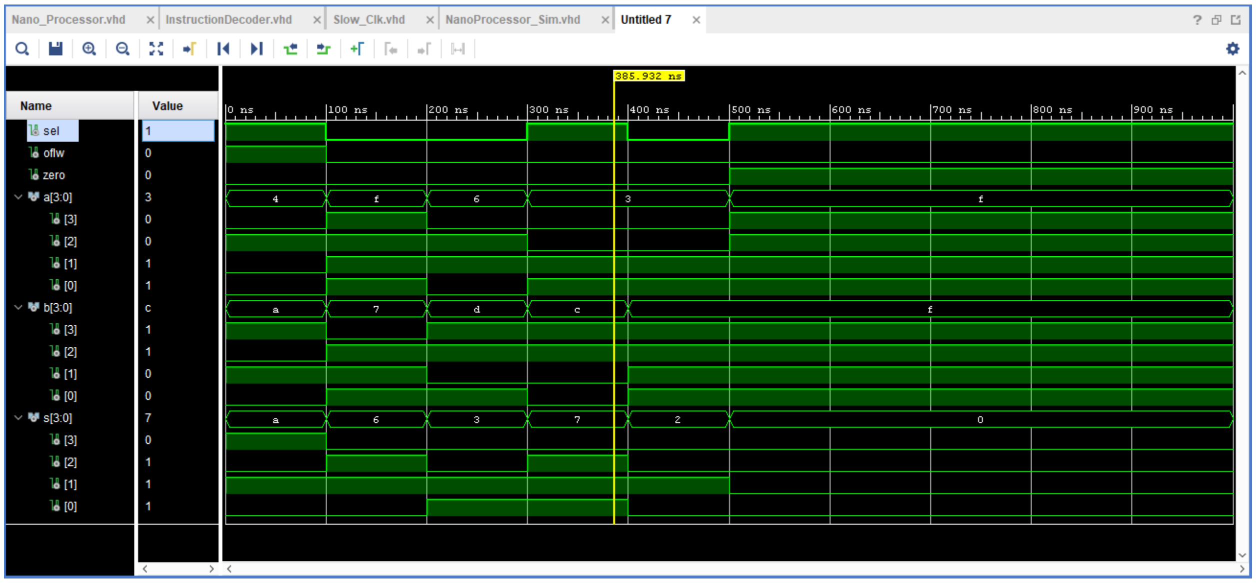
12. Nano Processor

```
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_STD.ALL;

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

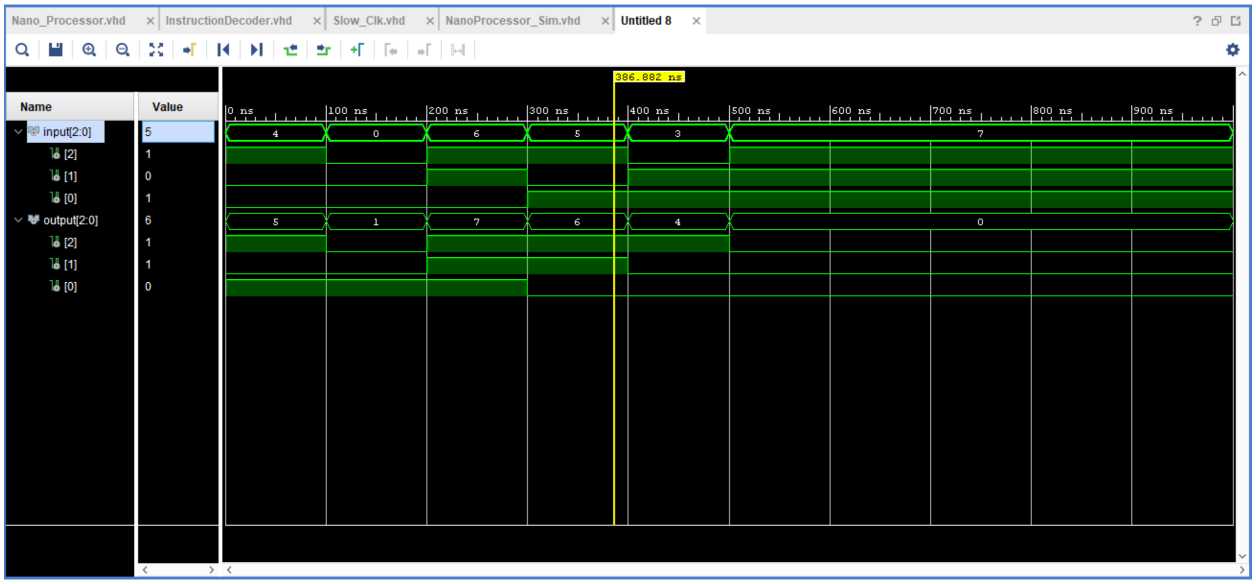
```
--use UNISIM.VComponents.all;

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

architecture Behavioral of NanoProcessor_Sim is

component Nano_Processor
Port ( Clock : in STD_LOGIC;
      RESET : in STD_LOGIC;
      Overflow : out STD_LOGIC;
      Zero : out STD_LOGIC;
      LED : out STD_LOGIC_VECTOR (3 downto 0);
      SevenSegDisplay: out STD_LOGIC_VECTOR (6 downto 0);
      AnodeSignal: out STD_LOGIC_VECTOR ( 3 downto 0) := "0001");

end component;
signal Overflow,Zero : std_logic;
signal RESET:std_logic := '1';
signal Clock:std_logic := '0';
signal LED,AnodeSignal: std_logic_vector(3 downto 0);
signal SevenSegDisplay: std_logic_vector(6 downto 0);

begin

UUT: Nano_Processor
port map(
       Clock => Clock,
       RESET => RESET,
       Overflow => Overflow,
       Zero => Zero,
       LED =>LED,
       SevenSegDisplay => SevenSegDisplay,
       AnodeSignal => AnodeSignal
       );

process
begin
   wait for 3ns;
   Clock <= NOT(Clock);
end process;

process begin
   -- wait for 20ns;
   RESET <= '1';
   wait for 76 ns;
   RESET <= '0';

   wait;
end process;

end Behavioral;
```

Timing Diagrams

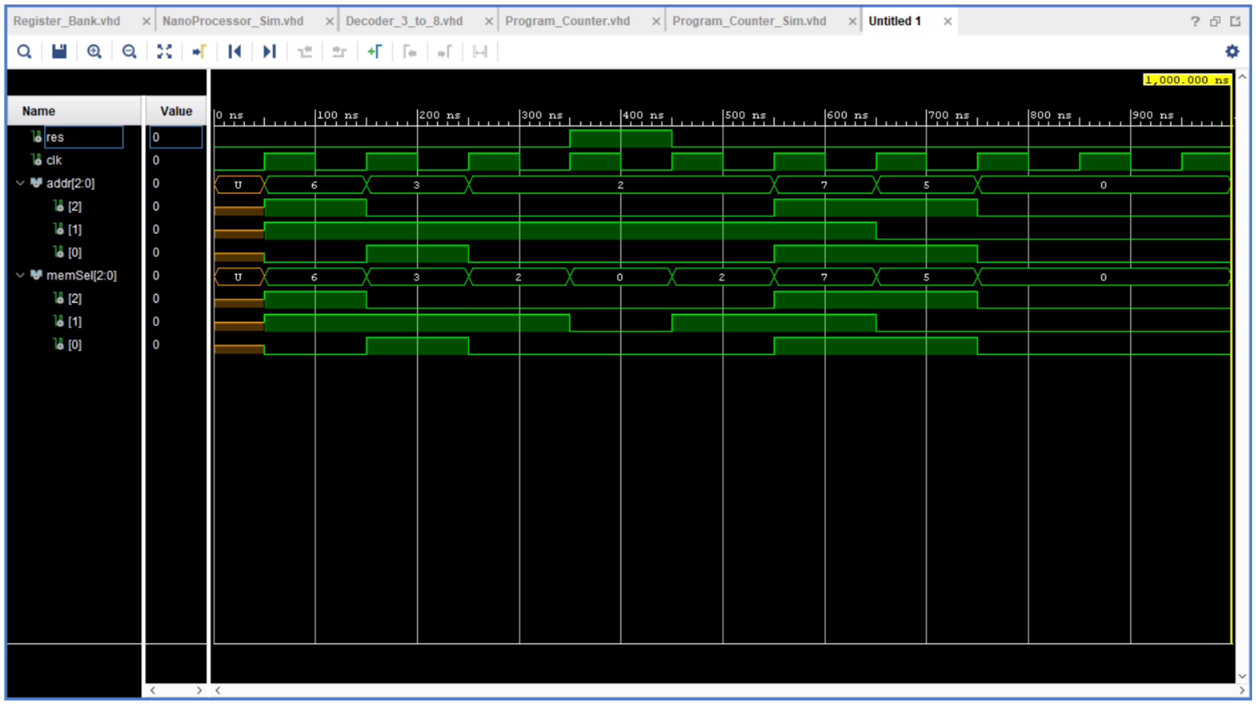
1. **4-bit Add/Subtract unit**



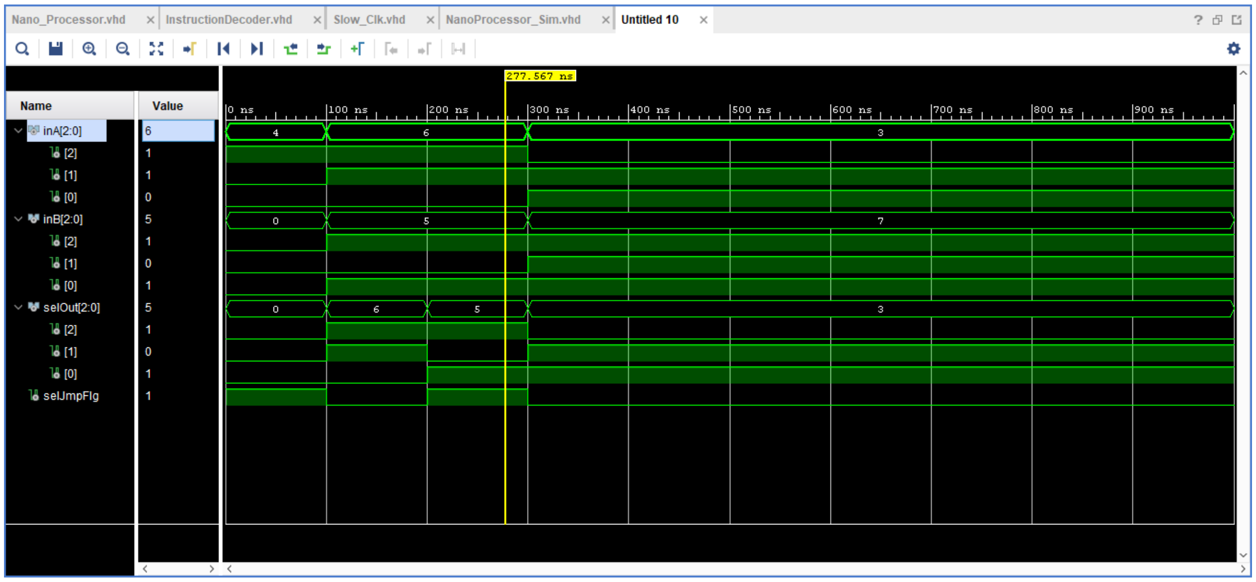
2. **3-bit Adder**



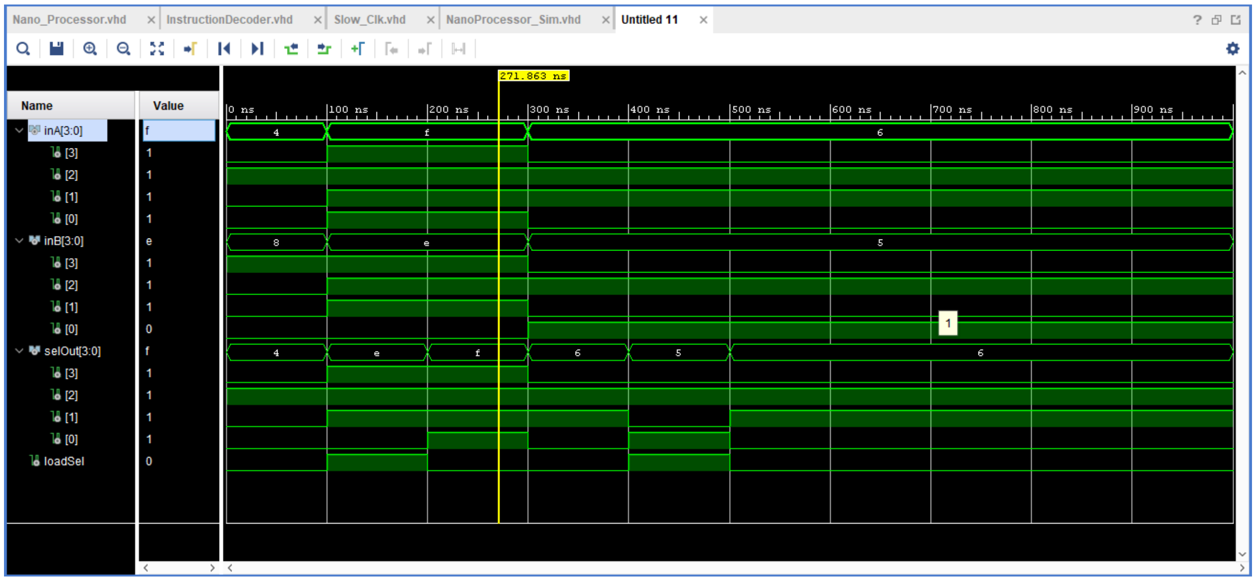
3. 3-bit Program Counter



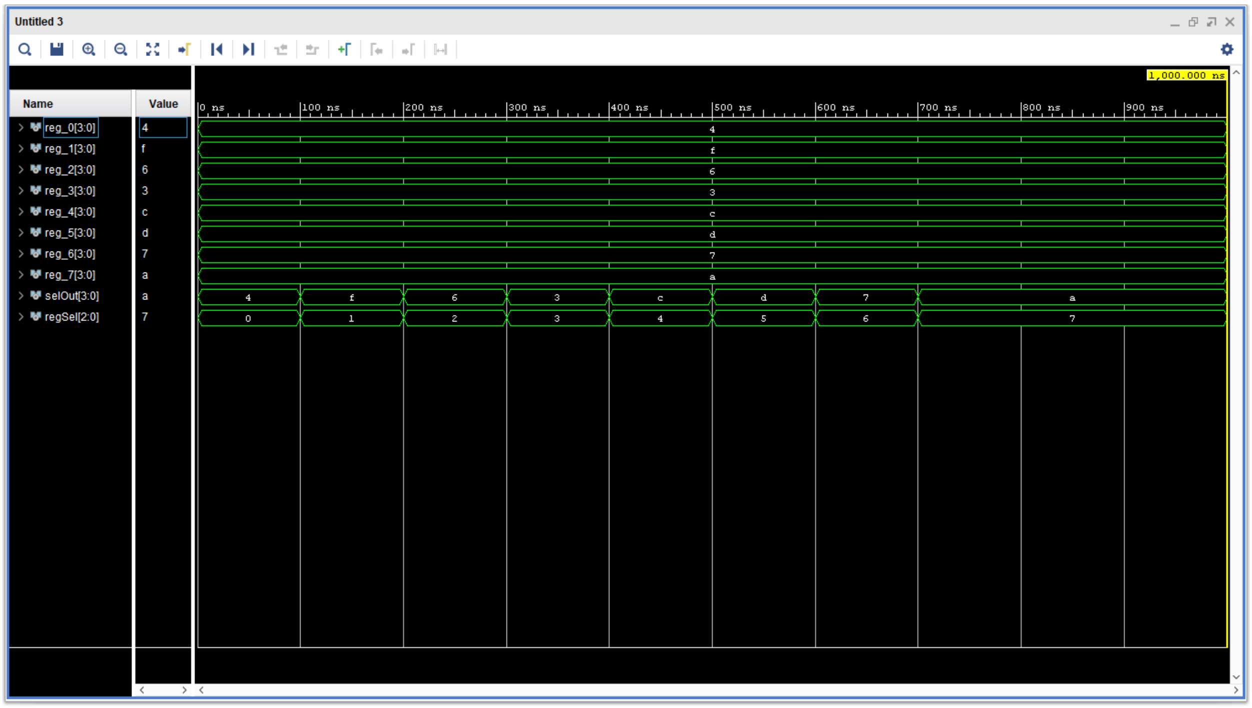
4. 2-way 3-bit Multiplexer



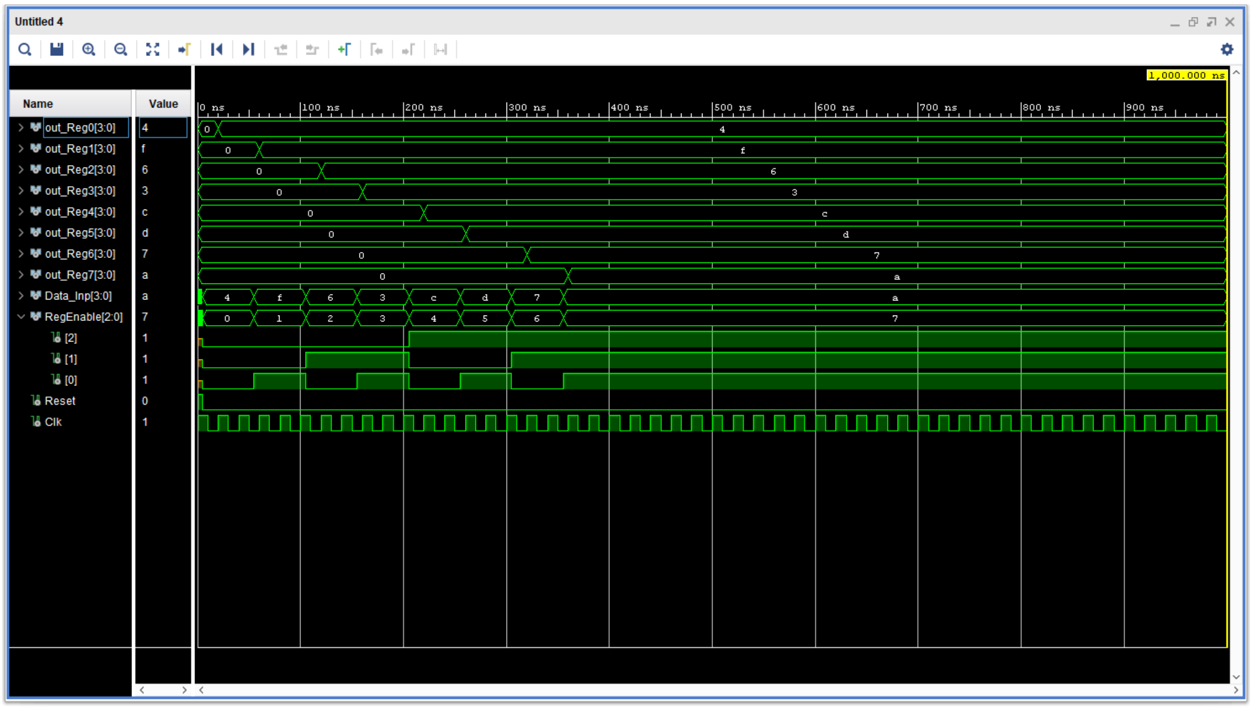
5. 2-way 4-bit Multiplexer



6. 8-way 4-bit Multiplexer

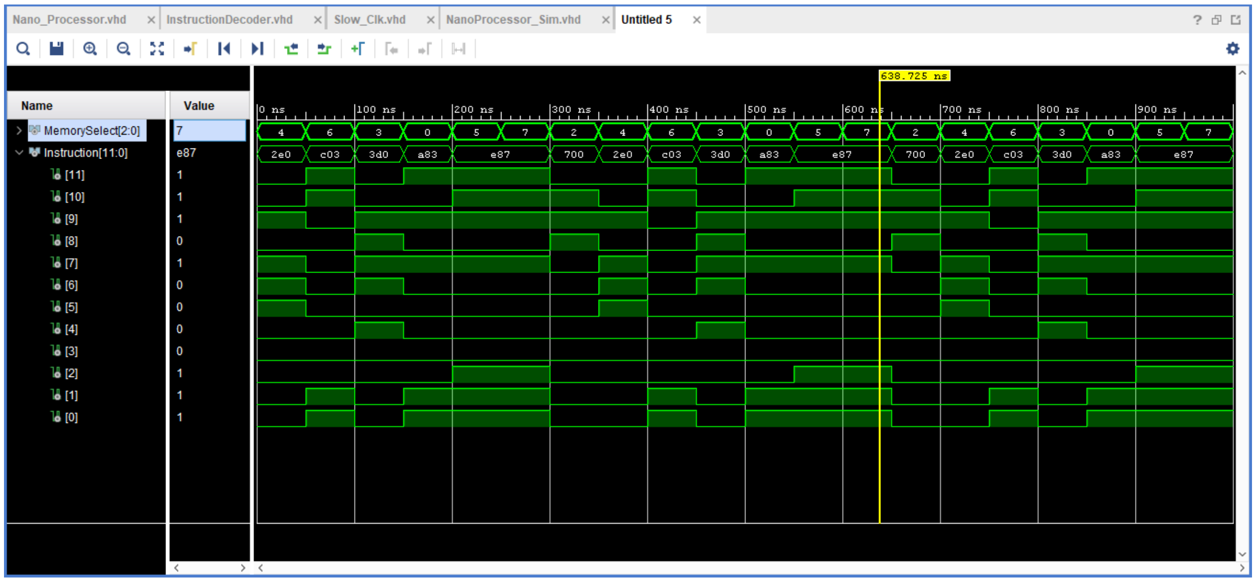


7. Register Bank



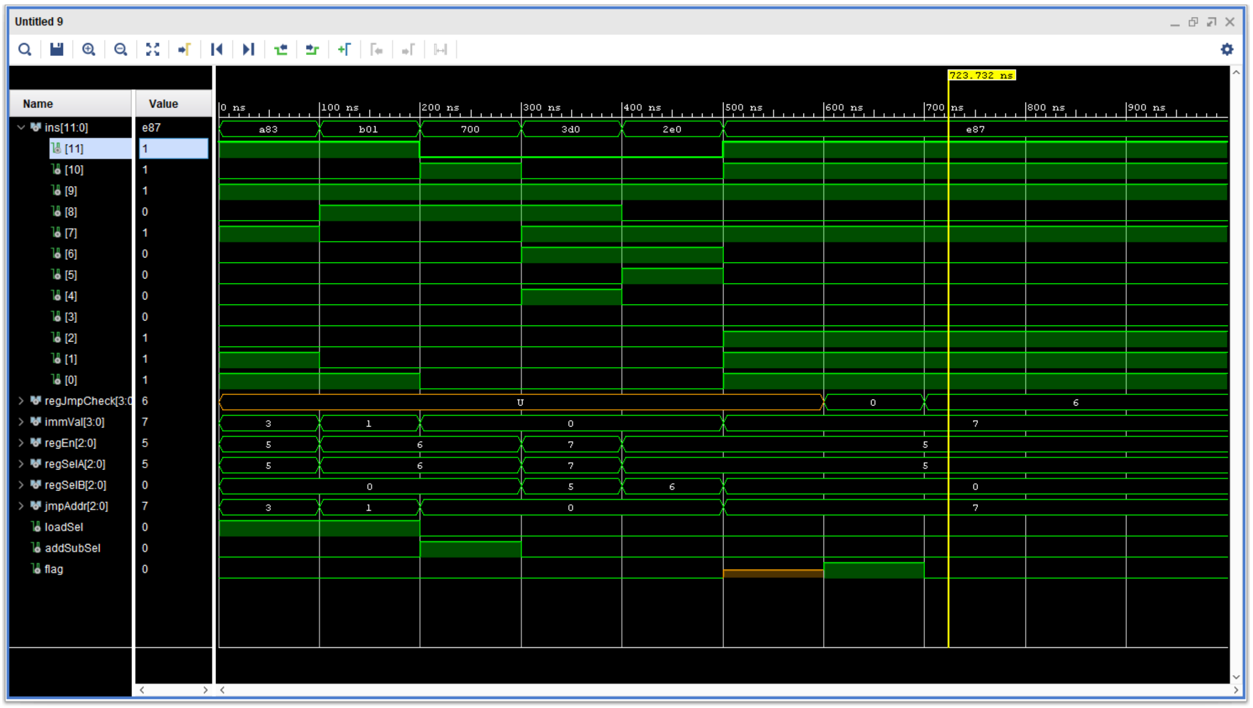
Name	Value
out_Reg0[3:0]	4
out_Reg1[3:0]	f
out_Reg2[3:0]	6
out_Reg3[3:0]	3
out_Reg4[3:0]	c
out_Reg5[3:0]	d
out_Reg6[3:0]	7
out_Reg7[3:0]	a
Data_Inp[3:0]	a
RegEnable[2:0]	7
[2]	1
[1]	1
[0]	1
Reset	0
Clk	1

8. Program ROM



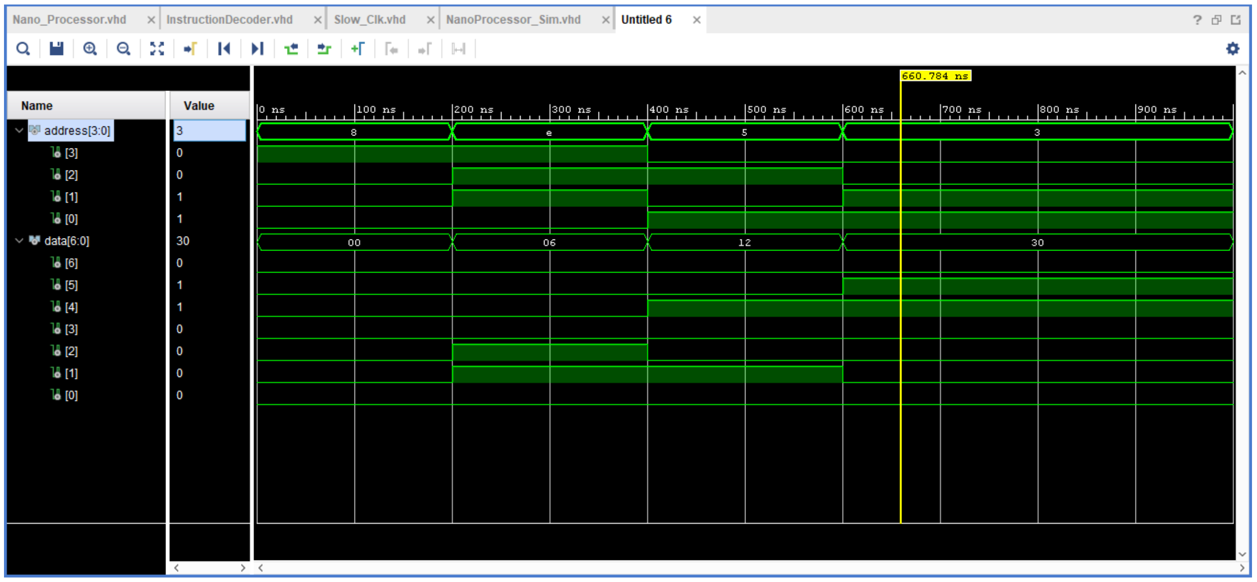
Name	Value
MemorySelect[2:0]	7
Instruction[11:0]	e87
[11]	1
[10]	1
[9]	1
[8]	0
[7]	1
[6]	0
[5]	0
[4]	0
[3]	0
[2]	1
[1]	1
[0]	1

9. Instruction Decoder



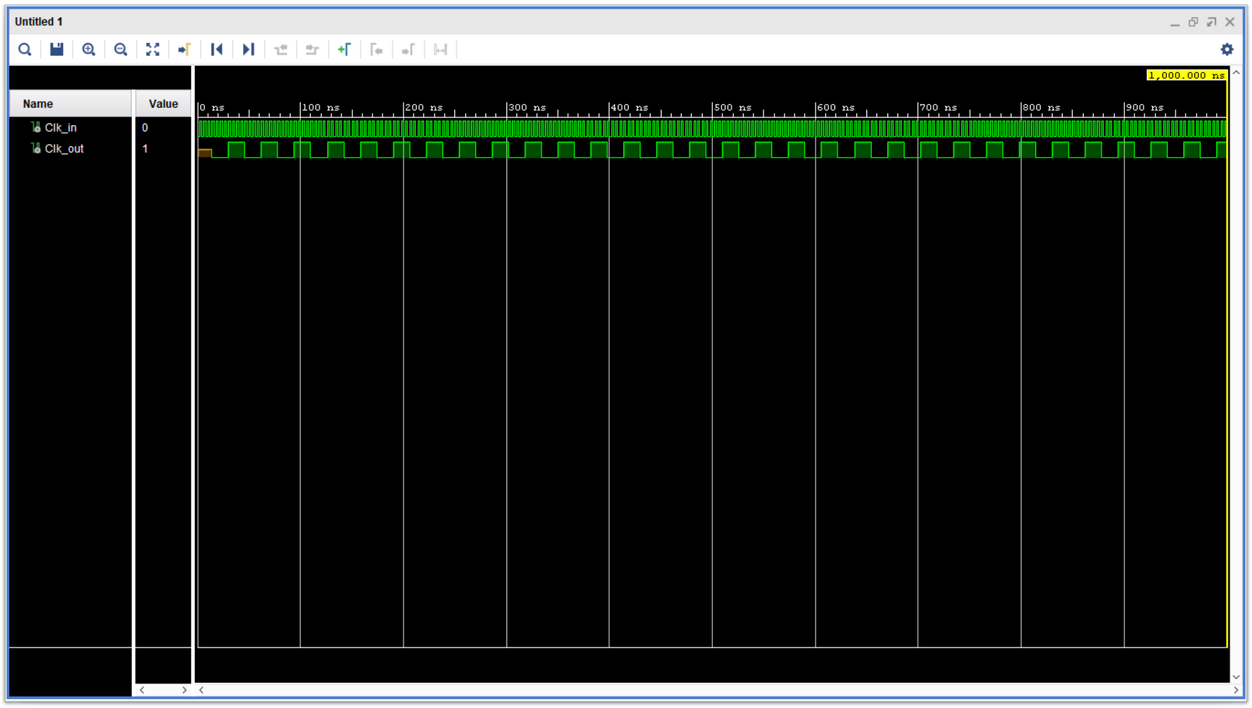
Name	Value
ins[11:0]	e87
[11]	1
[10]	1
[9]	1
[8]	0
[7]	1
[6]	0
[5]	0
[4]	0
[3]	0
[2]	1
[1]	1
[0]	1
regJmpCheck[3:0	6
immVal[3:0]	7
regEn[2:0]	5
regSelA[2:0]	5
regSelB[2:0]	0
jmpAddr[2:0]	7
loadSel	0
addSubSel	0
flag	0

10. Look-up Table

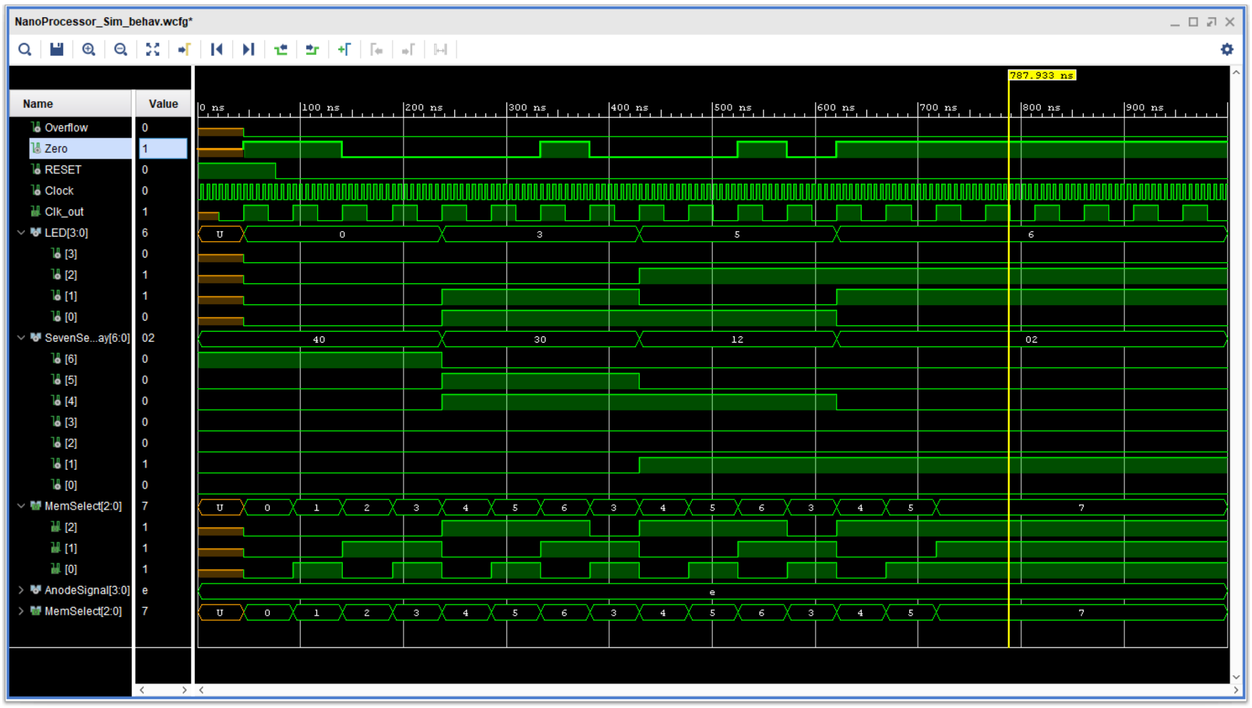


Name	Value
address[3:0]	3
[3]	0
[2]	0
[1]	1
[0]	1
data[6:0]	30
[6]	0
[5]	1
[4]	1
[3]	0
[2]	0
[1]	0
[0]	0

11. Slow Clock



12. Nano Processor



Constraint File (Basys3.xdc)

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports Clock]
	set_property IOSTANDARD LVCMOS33 [get_ports Clock]
	create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports Clock]

## Switches
#set_property PACKAGE_PIN V17 [get_ports {A[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
#set_property PACKAGE_PIN V16 [get_ports {A[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
#set_property PACKAGE_PIN W16 [get_ports {A[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
#set_property PACKAGE_PIN W17 [get_ports {A[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
```

```
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {RegSel}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {RegSel}]


## LEDs
set_property PACKAGE_PIN U16 [get_ports {LED[0]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {LED[1]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {LED[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {LED[3]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
```

```
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
set_property PACKAGE_PIN P1 [get_ports {Zero}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Zero}]
set_property PACKAGE_PIN L1 [get_ports {Overflow}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Overflow}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {SevenSegDisplay[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[0]}]
set_property PACKAGE_PIN W6 [get_ports {SevenSegDisplay[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[1]}]
set_property PACKAGE_PIN U8 [get_ports {SevenSegDisplay[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[2]}]
set_property PACKAGE_PIN V8 [get_ports {SevenSegDisplay[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[3]}]
set_property PACKAGE_PIN U5 [get_ports {SevenSegDisplay[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[4]}]
set_property PACKAGE_PIN V5 [get_ports {SevenSegDisplay[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[5]}]
set_property PACKAGE_PIN U7 [get_ports {SevenSegDisplay[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {SevenSegDisplay[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
```

```
        #set_property IOSTANDARD LVCMOS33 [get_ports dp]

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


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btnC]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnC]
#set_property PACKAGE_PIN T18 [get_ports btnU]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnU]
#set_property PACKAGE_PIN W19 [get_ports btnL]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnL]
#set_property PACKAGE_PIN T17 [get_ports btnR]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnR]
set_property PACKAGE_PIN U17 [get_ports RESET]
        set_property IOSTANDARD LVCMOS33 [get_ports RESET]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
```

```
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]



##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
```

```
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]



##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
```

```
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]


##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
```

```
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]



##VGA Connector
#set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
#set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
#set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
#set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
#set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
#set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
#set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
#set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
#set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
#set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]
#set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]
#set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]
#set_property PACKAGE_PIN P19 [get_ports Hsync]
	#set_property IOSTANDARD LVCMOS33 [get_ports Hsync]
#set_property PACKAGE_PIN R19 [get_ports Vsync]
```

```
	#set_property IOSTANDARD LVCMOS33 [get_ports Vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]
	#set_property IOSTANDARD LVCMOS33 [get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]
	#set_property IOSTANDARD LVCMOS33 [get_ports RsTx]


##USB HID (PS/2)
#set_property PACKAGE_PIN C17 [get_ports PS2Clk]
	#set_property IOSTANDARD LVCMOS33 [get_ports PS2Clk]
	#set_property PULLUP true [get_ports PS2Clk]
#set_property PACKAGE_PIN B17 [get_ports PS2Data]
	#set_property IOSTANDARD LVCMOS33 [get_ports PS2Data]
	#set_property PULLUP true [get_ports PS2Data]


##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
	#set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]
```

Conclusion

In this lab, we designed a 4-bit nano processor capable of executing instructions and performing basic arithmetic operations on signed integers.

To achieve this, we needed to extend several components including the add/subtract unit, adder, Program Counter, multiplexers, Register Bank, Program ROM, and Instruction Decoder. Teamwork and communication were crucial as we worked simultaneously on different components.

The building process involved several steps such as designing the Instruction Decoder, building and testing sub-components, creating a top-level design, writing an Assembly program, connecting inputs and outputs, and finally testing the nano processor on the development board.

In conclusion the lab provided hands-on experience in processor design, instruction decoding, multiplexer implementation, simulation, and hardware testing. The collaborative nature of the project also emphasized the importance of teamwork, communication, and coordination when working on complex tasks.

Members' Contributions

1. Vithurshan P. (210676M)

   Designing and developing the relevant components.

2. Navaneethan B. (210408V)

   Designing the simulations for the components and simulating.

We both together combined Nano processor using the developed component.

Total time spent on this project - approximately 30-35 hours.

THANK YOU!