LAB 9-10 – NANO PROCESSOR DESIGN COMPETITION

DEPT. OF COMPUTER SCIENCE AND ENGINEERING, UNIVERSITY OF MORATUWA

Participants - **SENARATNE R.R.S.N. 210592C,**

SENEVIRATHNE R.J.M. 210598B

Lab Task –

We had to design a 4-bit Nano processor capable of executing 4 instructions. To build this circuit, we needed to develop the components that we have built in the previous labs.

1) **Designed and developed a 4-bit arithmetic unit that can add/subtract signed integers.**
2) **Designed and developed MUXs (2-way 3-bit, 2-way 4-bit,8-way 4-bit)**
3) **Designed a register bank with 8 registers.**
4) **Designed a program counter and a 3-bit adder which can increment program counter to get the desired register address.**
5) **Designed and build the processor capable of executing 4 instructions (MOVI, ADD, NEG, JZR) that are stored in ROM.**

6) **Decoded instructions to activate necessary components on the processor.**
7) **Write the assembly program to calculate the total of all integers between 1 and 3. Converted the assembly code to machine code and hard code it and stored to ROM. Demonstrate the result using BASYS 3 board.**

Contribution –

210598B – 4- bit Adder/Subtracter
Program Rom
3 – bit Adder
Rom
3-bit PC

(15 hours spent)

210592C – 2 way 3-bit MUX
2 way 4-bit MUX
8way 4-bit MUX
3 – bit Program Counter
Register Bank
Instruction Decoder

(15 hours spent)

There are some modifications of Adders, Rom.

Simulation test bench files are submitted separately as a zip file.

Instructions in machine code

```
101110000001
100010000010
100100000011
001110010000
001110100000
101110000001
110000000000
000000000000
000000000000
```

Assembly program

```
MOVI R7,1    ;   R7 <- 1
MOVI R1,2    ;   R2 <- 2
MOVI R2,3    ;   R2 <- 3
ADD R7       ;   R7 <- R7 + R1
ADD R7       ;   R7 <- R7 + R2
JZR R0,0     ;   IF R0 = 0 JUMP TO LINE 0
```

4-bit Adder/Subtracter

```
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_Subtractor_4bit is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0); --input A
           B : in STD_LOGIC_VECTOR (3 downto 0); --input B
           M : in STD_LOGIC;                  -- Adder Subtractor switch
           S : out STD_LOGIC_VECTOR (3 downto 0);--Output
           C_OUT : out STD_LOGIC;              --Carry or Borrow out
           OVERFLOW: out STD_LOGIC;            --Overflow Flag
           ZERO: out STD_LOGIC                --Zero Flag
           );
end Adder_Subtractor_4bit;

architecture Behavioral of Adder_Subtractor_4bit is
    component FA
       port (
       A: in std_logic;
       B: in std_logic;
       C_in: in std_logic;
       S: out std_logic;
       C_out: out std_logic);
    end component;

    SIGNAL FA0_C, FA1_C, FA2_C, FA3_C,CARRY: std_logic;
    SIGNAL B_temp:STD_LOGIC_VECTOR (3 downto 0);
    SIGNAL S_temp:STD_LOGIC_VECTOR (3 downto 0);
begin

    FA_0 : FA
       port map (
           A => A(0),
           B => B_temp(0),
           C_in => M ,
           S => S_temp(0),
           C_out => FA0_C);

    FA_1 : FA
       port map (
           A => A(1),
```

```
            B => B_temp(1),
            C_in => FA0_C,
            S => S_temp(1),
            C_out => FA1_C);

    FA_2 : FA
       port map (
            A => A(2),
            B => B_temp(2),
            C_in => FA1_C,
            S => S_temp(2),
            C_out => FA2_C);

    FA_3 : FA
       port map (
            A => A(3),
            B => B_temp(3),
            C_in => FA2_C,
            S => S_temp(3),
            C_out => CARRY);


   B_temp(0)<=B(0) XOR M;
   B_temp(1)<=B(1) XOR M;
   B_temp(2)<=B(2) XOR M;
   B_temp(3)<=B(3) XOR M;

    C_OUT<=CARRY;
    S<=S_temp;

    ZERO <=NOT S_temp(0) AND NOT S_temp(1) AND NOT S_temp(2) AND NOT
S_temp(3) AND CARRY;

    OVERFLOW<= M AND NOT(FA2_C AND (A(3) XOR B_temp(3)));

end Behavioral;
```

3 – bit Adder

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/04/2023 12:38:53 AM
-- Design Name:
-- Module Name: Adder_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
```

```
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ( A : in STD_LOGIC_VECTOR (2 downto 0);
          S : 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 FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C, C : std_logic;

begin
   FA_0 : FA
      port map (
         A => A(0),
         B => '1',
         C_in => '0',
         S => S(0),
         C_Out => FA0_C);
   FA_1 : FA
      port map (
         A => A(1),
         B => '0',
         C_in => FA0_C,
         S => S(1),
         C_Out => FA1_C);
```

```
   FA_2 : FA
      port map (
         A => A(2),
         B => '0',
         C_in => FA1_C,
         S => S(2),
         C_Out => C);
end Behavioral;
```

Program Rom

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

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

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

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

architecture Behavioral of ROM is

type rom_type is array (0 to 7) of std_logic_vector(11 downto 0);
signal rom : rom_type :=(
   "101110000001", --0    MOVI R7,1
   "100010000010", --1    MOVI R1,2
   "100100000001", --2    MOVI R2,3
   "001110010000", --3    ADD R7 <- R7 + R1
   "001110100000", --4    ADD R7 <- R7 + R2
   "110000000000", --5    JZR R0,0
   "000000000000", --6
   "000000000000"  --7
  );
begin
   data <= rom(to_integer(unsigned(address)));
end Behavioral;
```

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 ProgramCounter is
    Port ( Reset : in STD_LOGIC;                --Reset Button
           Clk : in STD_LOGIC;                  --Clock
           A : in STD_LOGIC_VECTOR (2 downto 0);    --ProgramCounter_in
           memory_select : out STD_LOGIC_VECTOR (2 downto 0)); ----
ProgramCounter select Address
end ProgramCounter;

architecture Behavioral of ProgramCounter is
begin
    process(Clk)
    begin
    if(rising_edge(Clk)) then
        if(Reset='1') then
          memory_select<="000";
        else
          memory_select<=A;
        end if;
    end if;
end process;


end Behavioral;
```

MUX_2way_3-bit

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 10:20:28 AM
-- Design Name:
-- Module Name: MUX_2_3bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
```

```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity MUX_2_3bit is
    Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
           B: in STD_LOGIC_VECTOR (2 downto 0);
           S : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (2 downto 0));
end MUX_2_3bit;

architecture Behavioral of MUX_2_3bit is

begin
process (S,A,B)
begin
 if S ='0' then
    Q <= B;
 else
    Q<= A ;
 end if;
 end process;
end Behavioral;
```

MUX_2way_4-bit

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 10:18:45 AM
-- Design Name:
-- Module Name: MUX_2_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
```

```
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

   entity MUX_2_4bit is
      Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
             B : in STD_LOGIC_VECTOR (3 downto 0);
             Q : out STD_LOGIC_VECTOR (3 downto 0);
             S : in STD_LOGIC);
   end MUX_2_4bit;

   architecture Behavioral of MUX_2_4bit is

   signal ls_s : std_logic;
   signal asu_s : std_logic_vector(3 downto 0);
   signal immVal_s : std_logic_vector(3 downto 0);
   signal output : std_logic_vector(3 downto 0);
   begin

      ls_s <= S;
      asu_s <= B;
      immVal_s <= A;
      process (ls_s, asu_s, immVal_s) begin
         if ls_s='1' then
            output<=asu_s;
         else
            output<=immVal_s;
         end if;
      end process;

      Q <= output;
   end Behavioral;
```

MUX_8way_4-bit

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 10:16:43 AM
-- Design Name:
-- Module Name: MUX_8_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity MUX_8_4bit is
    Port ( Reg_Sel : in STD_LOGIC_VECTOR (2 downto 0);
           R0 : in STD_LOGIC_VECTOR (3 downto 0);
           R1 : in STD_LOGIC_VECTOR (3 downto 0);
           R2 : in STD_LOGIC_VECTOR (3 downto 0);
           R3 : in STD_LOGIC_VECTOR (3 downto 0);
           R4 : in STD_LOGIC_VECTOR (3 downto 0);
           R5 : in STD_LOGIC_VECTOR (3 downto 0);
           R6 : in STD_LOGIC_VECTOR (3 downto 0);
           R7 : in STD_LOGIC_VECTOR (3 downto 0);
           Mux_Out  : out STD_LOGIC_VECTOR (3 downto 0));
end MUX_8_4bit;
```

```
architecture Behavioral of MUX_8_4bit is
component Decoder_3_to_8
   port(I : in STD_LOGIC_VECTOR (2 downto 0);
   EN : in STD_LOGIC;
   Y : out STD_LOGIC_VECTOR (7 downto 0));
end component;

signal Decoder_Out : std_logic_vector(7 downto 0);
signal Mux_Output : std_logic_vector(3 downto 0);
signal D0 : std_logic_vector(3 downto 0);
signal A0 : std_logic_vector(3 downto 0);
signal D1 : std_logic_vector(3 downto 0);
signal A1 : std_logic_vector(3 downto 0);
signal D2 : std_logic_vector(3 downto 0);
signal A2 : std_logic_vector(3 downto 0);
signal D3 : std_logic_vector(3 downto 0);
signal A3 : std_logic_vector(3 downto 0);
signal D4 : std_logic_vector(3 downto 0);
signal A4 : std_logic_vector(3 downto 0);
signal D5 : std_logic_vector(3 downto 0);
signal A5 : std_logic_vector(3 downto 0);
signal D6 : std_logic_vector(3 downto 0);
signal A6 : std_logic_vector(3 downto 0);
signal D7 : std_logic_vector(3 downto 0);
signal A7 : std_logic_vector(3 downto 0);


begin
Decoder_3_to_8_0 : Decoder_3_to_8
   port map (
      I => Reg_Sel,
      EN => '1',
      Y => Decoder_Out
   );
   D0 <= R0 AND A0;
   D1 <= R1 AND A1;
   D2 <= R2 AND A2;
   D3 <= R3 AND A3;
   D4 <= R4 AND A4;
   D5 <= R5 AND A5;
   D6 <= R6 AND A6;
   D7 <= R7 AND A7;

   A0 <= (others=> Decoder_Out(0));
   A1 <= (others=> Decoder_Out(1));
   A2 <= (others=> Decoder_Out(2));
   A3 <= (others=> Decoder_Out(3));
   A4 <= (others=> Decoder_Out(4));
   A5 <= (others=> Decoder_Out(5));
   A6 <= (others=> Decoder_Out(6));
   A7 <= (others=> Decoder_Out(7));
```

```
   Mux_Out <= D0 or D1 or D2 or D3 or D4 or D5 or D6 or D7;

end Behavioral;
```

Register Bank

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 09:58:59 AM
-- Design Name:
-- Module Name: Register_Bank - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity Reg_Bank is
   Port ( Clk : in STD_LOGIC;
          Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Reset :in STD_LOGIC;
          Reg_Bank_In : in STD_LOGIC_VECTOR (3 downto 0);
          REG_0_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_1_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_2_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_3_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_4_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_5_OUT : out STD_LOGIC_VECTOR (3 downto 0);
```

```
        REG_6_OUT : out STD_LOGIC_VECTOR (3 downto 0);
        REG_7_OUT : out STD_LOGIC_VECTOR (3 downto 0));
end Reg_Bank;

architecture Behavioral of Reg_Bank is
component REG
   port(
      D : in STD_LOGIC_VECTOR (3 downto 0);
      En : in STD_LOGIC;
      Clk : in STD_LOGIC;
      Reset :in STD_LOGIC;
      Q : out STD_LOGIC_VECTOR (3 downto 0)
    );
 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 decoder_out :  std_LOGIC_VECTOR(7 downto 0);

begin

   Reg_Bank_Decoder : Decoder_3_to_8
      port map(
         I => Reg_EN,
         EN => '1',
         Y => decoder_out
         );

 REG_0 : REG
      port map(
        D => "0000",
        EN=> decoder_out(0),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_0_OUT
       );
   REG_1 : REG
      port map(
        D => Reg_Bank_In,
        EN=> decoder_out(1),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_1_OUT
        );
    REG_2 : REG
      port map(
        D => Reg_Bank_In,
```

```
        EN=> decoder_out(2),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_2_OUT
        );
    REG_3 : REG
      port map(
        D => Reg_Bank_In,
        EN=> decoder_out(3),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_3_OUT
        );
    REG_4 : REG
      port map(
        D => Reg_Bank_In,
        EN=> decoder_out(4),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_4_OUT
        );
    REG_5 : REG
      port map(
        D => Reg_Bank_In,
        EN=> decoder_out(5),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_5_OUT
        );
    REG_6 : REG
      port map(
        D => Reg_Bank_In,
        EN=> decoder_out(6),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_6_OUT
        );
    REG_7 : REG
      port map(
        D =>Reg_Bank_In,
        EN=> decoder_out(7),
        Clk => Clk,
        Reset=>Reset,
        Q=> REG_7_OUT
        );

end Behavioral;
```

Instruction Decoder

```
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 10:21:32 AM
-- Design Name:
-- Module Name: InstructionDecorder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ( Instruction : in STD_LOGIC_VECTOR (11 downto 0); --Instruction
          RCJ : in STD_LOGIC_VECTOR (3 downto 0);          --Register check for jump
          RegEn : out STD_LOGIC_VECTOR (2 downto 0);       --Register Enable
          LS : out STD_LOGIC;                              --Load Select
          Immediatevalue : out STD_LOGIC_VECTOR (3 downto 0);--Immediate Value
          Reg_1 : out STD_LOGIC_VECTOR (2 downto 0);        --Register Select 1
          Reg_2 : out STD_LOGIC_VECTOR (2 downto 0);        --Register Select 2
          Add_Sub_Select : out STD_LOGIC;                  --Add_Sub_Select
          JumpFlag : out STD_LOGIC;                        --Jump Flag
          ATJ : out STD_LOGIC_VECTOR (2 downto 0));         --Address to jump
end InstructionDecoder;

architecture Behavioral of InstructionDecoder is


SIGNAL regEn_temp : std_logic_vector(2 downto 0);
SIGNAL regChkForJump : std_logic_vector(3 downto 0);
SIGNAL reg_select_temp_1 : std_logic_vector(2 downto 0);
SIGNAL op : std_logic_vector(1 downto 0);
SIGNAL reg_1_temp : std_logic_vector(2 downto 0);
```

```
SIGNAL reg_2_temp : std_logic_vector(2 downto 0);
SIGNAL immediateValue_temp : std_logic_vector(3 downto 0);
SIGNAL jumpAddress : std_logic_vector(2 downto 0);
SIGNAL reg_select_temp_2 : std_logic_vector(2 downto 0);
SIGNAL LS_temp : std_logic;
SIGNAL addSub_temp : std_logic;
SIGNAL immVal_temp : std_logic_vector(3 downto 0);

SIGNAL jumpSel : std_logic := '0';
begin
   op <= Instruction(11 downto 10);
   reg_1_temp <= Instruction(9 downto 7);
   reg_2_temp <= Instruction(6 downto 4);
   immediateValue_temp <= Instruction(3 downto 0);
   process (op, reg_1_temp, reg_2_temp, immediateValue_temp)

   begin
     jumpSel <= '0';
      if op = "00" then
         reg_select_temp_1<=reg_1_temp;
         reg_select_temp_2<=reg_2_temp;
         addSub_temp<='0';
         LS_temp<='1';
         regEn_temp<=reg_1_temp;
      elsif op = "01"then
         reg_select_temp_1<=reg_1_temp;
         reg_select_temp_2<="000";

         addSub_temp<='1';
         LS_temp<='1';
         regEn_temp<=reg_1_temp;
      elsif op = "10" then

         immVal_temp<=immediateValue_temp;
         LS_temp <='0';
         regEn_temp<=reg_1_temp;
      elsif op = "11" then
         reg_select_temp_1<=reg_1_temp;
         jumpAddress<=immediateValue_temp(2 downto 0);
         if regChkForJump= "0000" then
            jumpSel<='1';
         else
            jumpSel<='0';
         end if;
      end if;
   end process;

   JumpFlag <= jumpSel;
   regChkForJump <= RCJ;
   RegEn <= regEn_temp;
   LS <= LS_temp;
   Immediatevalue <= immVal_temp;
   Reg_1 <= reg_select_temp_1;
```

```
    Reg_2 <= reg_select_temp_2;
    Add_Sub_Select <= addSub_temp;
    ATJ <= jumpAddress;
end Behavioral;
```

Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/09/2023 10:25:44 AM
-- Design Name:
-- Module Name: nano_processor - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


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 ( pushButton : in STD_LOGIC;
           Imm_Value:out std_logic_vector(3 downto 0);
           Clk : in STD_LOGIC;
           nextInsVal : out std_logic_vector(2 downto 0);
           RegisterEnable:out std_logic_vector(2 downto 0);
           RegisterBank_DataIn: out std_logic_vector(3 downto 0);
```

```
        LED : out STD_LOGIC_VECTOR (3 downto 0);
        LED_R6 : out std_logic_vector(3 downto 0);
        LED_R5 : out std_logic_vector(3 downto 0);
        LED_OVERFLOW:out STD_LOGIC;
        LED_ZERO:out STD_LOGIC;
        Instruction_temp : out std_logic_vector(11 downto 0)


        );
end nano_processor;

architecture Behavioral of nano_processor is
COMPONENT Slow_Clk is
   Port ( Clk_in : in STD_LOGIC;
        Clk_out : out STD_LOGIC);
end COMPONENT;


--Instruction Decoder
COMPONENT InstructionDecoder is
   Port ( Instruction : in STD_LOGIC_VECTOR (11 downto 0);
        RCJ : in STD_LOGIC_VECTOR (3 downto 0);
        RegEn : out STD_LOGIC_VECTOR (2 downto 0);
        LS : out STD_LOGIC;
        Immediatevalue : out STD_LOGIC_VECTOR (3 downto 0);
        Reg_1 : out STD_LOGIC_VECTOR (2 downto 0);
        Reg_2 : out STD_LOGIC_VECTOR (2 downto 0);
        Add_Sub_Select : out STD_LOGIC;
        JumpFlag : out STD_LOGIC;
        ATJ : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;


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

--Adder 3bit
COMPONENT Adder_3bit is
   Port ( A : in STD_LOGIC_VECTOR(2 DOWNTO 0);
        B : in STD_LOGIC_VECTOR(2 DOWNTO 0);
        S : out STD_LOGIC_VECTOR(2 DOWNTO 0);
        C_out : out STD_LOGIC);
END COMPONENT ;

--MUX 2 3bit
COMPONENT MUX_2_3bit is
   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
        B: in STD_LOGIC_VECTOR (2 downto 0);
        S : in STD_LOGIC;
        Q : out STD_LOGIC_VECTOR (2 downto 0));
```

```
end COMPONENT;

--Program Counter
COMPONENT ProgramCounter is
   Port ( Reset : in STD_LOGIC;
          Clk : in STD_LOGIC;
          A : in STD_LOGIC_VECTOR (2 downto 0);
          memory_select : out STD_LOGIC_VECTOR (2 downto 0));
end COMPONENT;

--Register Bank
COMPONENT Reg_Bank is
   Port ( Clk : in STD_LOGIC;
          Reg_EN : in STD_LOGIC_VECTOR (2 downto 0);
          Reset :in STD_LOGIC;
          Reg_Bank_In : in STD_LOGIC_VECTOR (3 downto 0);
          REG_0_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_1_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_2_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_3_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_4_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_5_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_6_OUT : out STD_LOGIC_VECTOR (3 downto 0);
          REG_7_OUT : out STD_LOGIC_VECTOR (3 downto 0));
end COMPONENT;

--MUX 8 4bit
COMPONENT MUX_8_4bit is
   Port ( Reg_Sel : in STD_LOGIC_VECTOR (2 downto 0);
          R0 : in STD_LOGIC_VECTOR (3 downto 0);
          R1 : in STD_LOGIC_VECTOR (3 downto 0);
          R2 : in STD_LOGIC_VECTOR (3 downto 0);
          R3 : in STD_LOGIC_VECTOR (3 downto 0);
          R4 : in STD_LOGIC_VECTOR (3 downto 0);
          R5 : in STD_LOGIC_VECTOR (3 downto 0);
          R6 : in STD_LOGIC_VECTOR (3 downto 0);
          R7 : in STD_LOGIC_VECTOR (3 downto 0);
          Mux_Out  : out STD_LOGIC_VECTOR (3 downto 0));
end COMPONENT;

--MUX 2 4bit
COMPONENT MUX_2_4bit is
   Port (A : in STD_LOGIC_VECTOR (3 downto 0);
          B: in STD_LOGIC_VECTOR (3 downto 0);
          S : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (3 downto 0));
end COMPONENT;



--Adder_Subtractor 4bit
COMPONENT Adder_Subtractor_4bit IS
   Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
```

```
        B : in STD_LOGIC_VECTOR (3 downto 0);
        M : in STD_LOGIC;
        S : out STD_LOGIC_VECTOR (3 downto 0);
        C_OUT : out STD_LOGIC;
        OVERFLOW: out STD_LOGIC;
        ZERO: out STD_LOGIC );
end COMPONENT;


SIGNAL clk_temp:STD_LOGIC;
SIGNAL clk_temp2:STD_LOGIC;

---Program counter
SIGNAL Next_Instruction:STD_LOGIC_VECTOR(2 DOWNTO 0);

--Adder 3 bit
SIGNAL THREE_BIT_ADDER_OUT:STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL C_OUTT:STD_LOGIC;

--Adder_Subtractor
SIGNAL M,OVERFLOW,ZERO,C_OUT:STD_LOGIC;

--Register Bank
SIGNAL R0,R1,R2,R3,R4,R5,R6,R7,Reg_Bank_In:STD_LOGIC_VECTOR(3 DOWNTO
0);
signal pushButton_temp : std_logic;

--instruction decoder
SIGNAL Reg_1,Reg_2,Reg_EN:STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL LS_temp:STD_LOGIC;
SIGNAL ASU_temp:STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL Immediatevalue_temp:STD_LOGIC_VECTOR(3 DOWNTO 0);

------Decoder
SIGNAL Instruction: STD_LOGIC_VECTOR(11 DOWNTO 0);
SIGNAL JumpFlag:STD_LOGIC;
SIGNAL ATJ:STD_LOGIC_VECTOR(2 DOWNTO 0);

---ROM
SIGNAL memory_select:STD_LOGIC_VECTOR(2 DOWNTO 0);

-----MUX 8 4 bit
SIGNAL MUX_8_OUT0,MUX_8_OUT1:STD_LOGIC_VECTOR(3 DOWNTO 0);

begin

Program_Counter:ProgramCounter
   Port map(
   Reset=>pushButton_temp,
   Clk=>clk_temp,
--    Clk=>clk_temp2,
   A=>Next_Instruction,
   memory_select=>memory_select
```

```
    );


Reg_Bank_0 :Reg_Bank
    port map(
    Clk=>clk_temp,
    Reg_EN=>Reg_EN,
    Reset=>pushButton_temp,
    Reg_Bank_In=>Reg_Bank_In,
    REG_0_OUT=>R0,
    REG_1_OUT=>R1,
    REG_2_OUT=>R2,
    REG_3_OUT=>R3,
    REG_4_OUT=>R4,
    REG_5_OUT=>R5,
    REG_6_OUT=>R6,
    REG_7_OUT=>R7
    );

MUX_2_3bit_0:MUX_2_3bit
        port map(
        A=>ATJ,
        B=>THREE_BIT_ADDER_OUT,
        S=>JumpFlag,
        Q=>Next_Instruction
        );

Instruction_Decoder:InstructionDecoder
    Port map(
    Instruction=>Instruction,
    RCJ =>MUX_8_OUT0,
    RegEn =>Reg_EN,
    LS =>LS_temp,
    Immediatevalue=>Immediatevalue_temp,
    Reg_1=>Reg_1,
    Reg_2 =>Reg_2,
    Add_Sub_Select =>M,
    JumpFlag=>JumpFlag,
    ATJ=>ATJ
    );

Adder_3bit_0:Adder_3bit
        Port map(
        A=>memory_select,
        B => "001",
        S=>THREE_BIT_ADDER_OUT,
        C_out=>C_OUTT
        );

MUX_8_4bit_0:MUX_8_4bit
    port map(
     Reg_Sel =>Reg_1,
     R0 =>R0,
```

```
    R1 =>R1,
    R2 =>R2,
    R3 =>R3,
    R4 =>R4,
    R5 =>R5,
    R6 =>R6,
    R7 =>R7,
    Mux_Out =>MUX_8_OUT0
    );
MUX_8_4bit_1:MUX_8_4bit
    port map(
    Reg_Sel =>Reg_2,
    R0 =>R0,
    R1 =>R1,
    R2 =>R2,
    R3 =>R3,
    R4 =>R4,
    R5 =>R5,
    R6 =>R6,
    R7 =>R7,
    Mux_Out =>MUX_8_OUT1
    );
MUX_2_4bit_0: MUX_2_4bit
   port map(
   A=>Immediatevalue_temp,
   B=>ASU_temp,
   Q=>Reg_Bank_In,
   S=>LS_temp
   );

Adder_Subtractor: Adder_Subtractor_4bit
   Port map (
   A =>MUX_8_OUT0,
   B =>MUX_8_OUT1,
   M  =>M,
   S =>ASU_temp,
   C_OUT =>C_OUT,
   ZERO =>ZERO,
   OVERFLOW => OVERFLOW
   );


ROM_0 :ROM
   Port map(
   address=>memory_select,
   data=>Instruction
   );

Slow_Clk_0 :Slow_Clk
      Port map(
      Clk_in=>Clk,
      Clk_out=>clk_temp2
      );
```

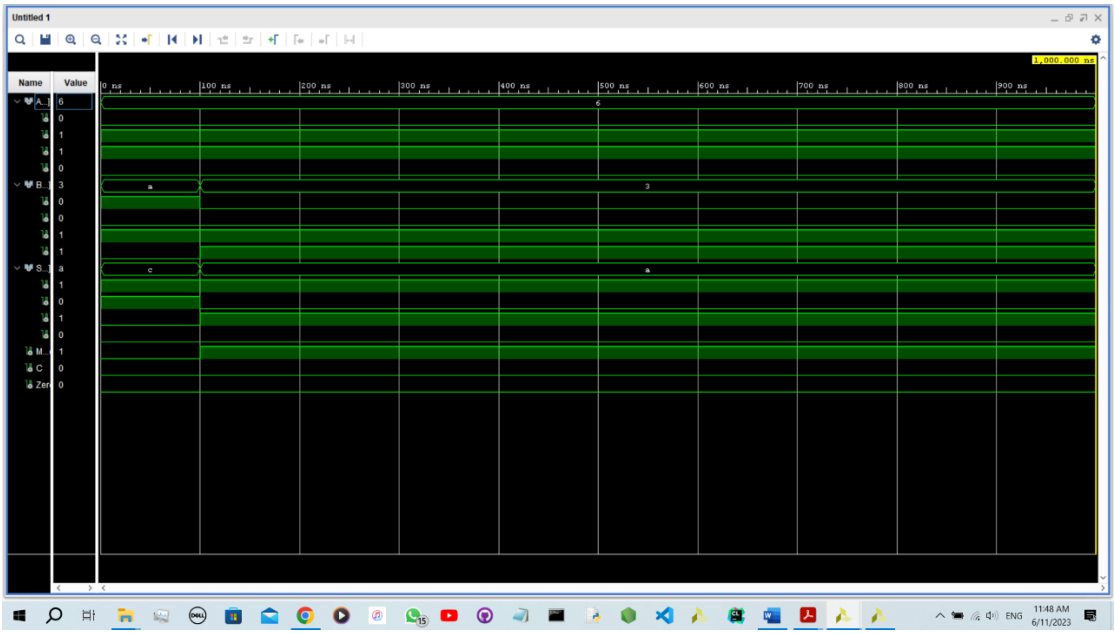
```
    clk_temp <= Clk;
    nextInsVal <= Next_Instruction;
    pushButton_temp <= pushButton;
    LED<=R7;
    LED_R6 <= R6;
    LED_R5 <= R5;
    LED_OVERFLOW<=OVERFLOW;
    LED_ZERO<=ZERO;
    RegisterEnable<=Reg_EN;
    RegisterBank_DataIn<=Reg_Bank_In;
    Imm_Value<=Immediatevalue_temp;
    Instruction_temp <= Instruction;

end Behavioral;
```

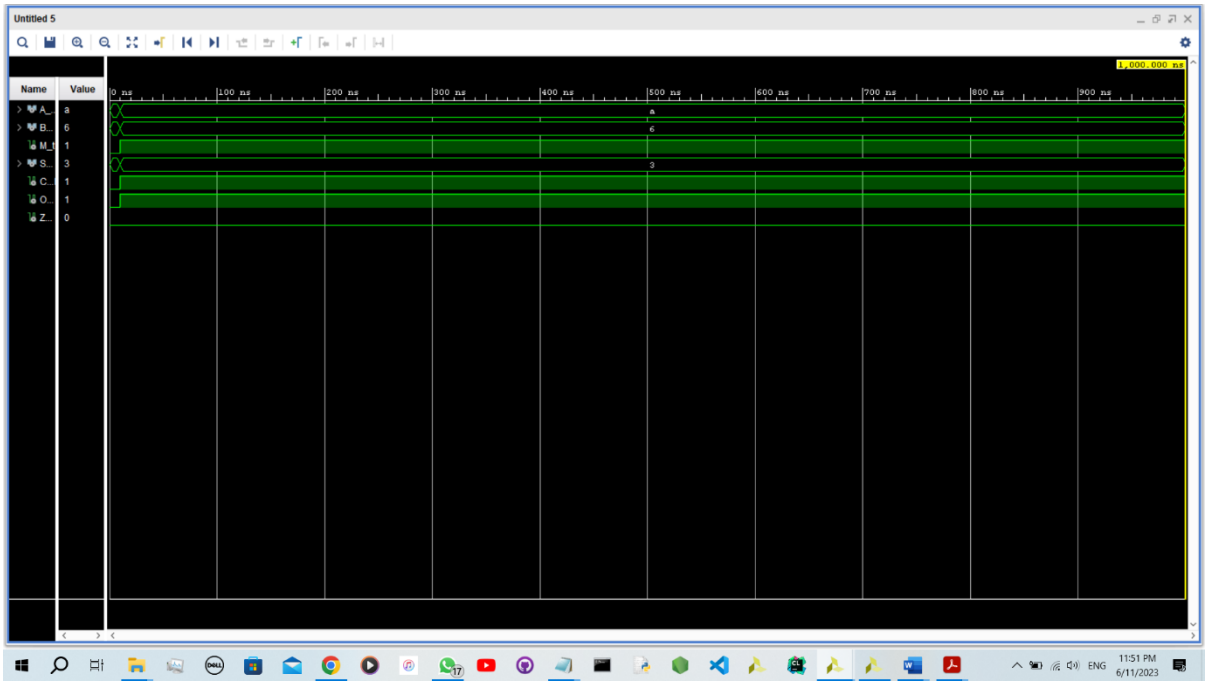
Timing Diagrams

4-bit Adder/Subtracter

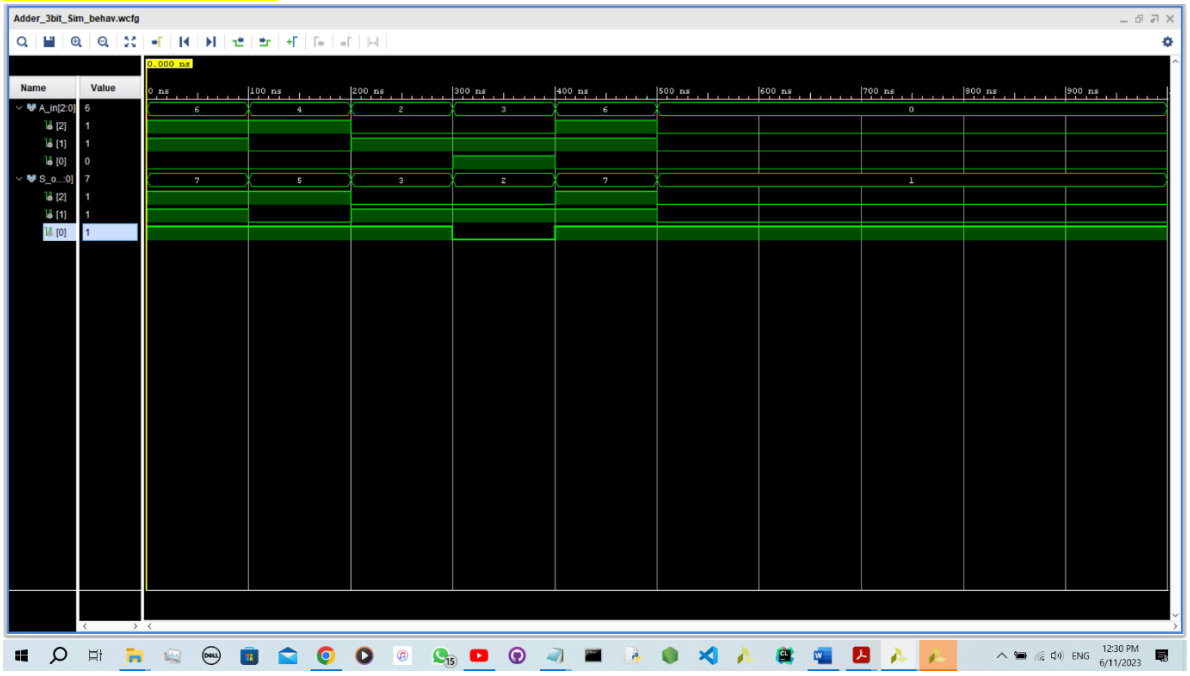
For addition



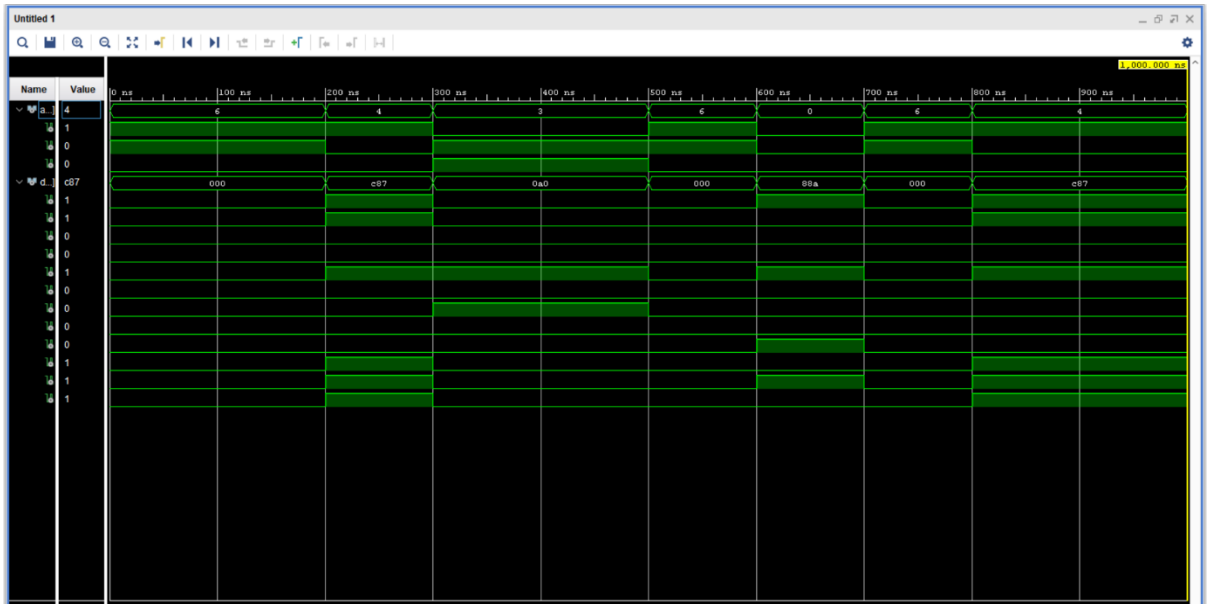
For subtraction



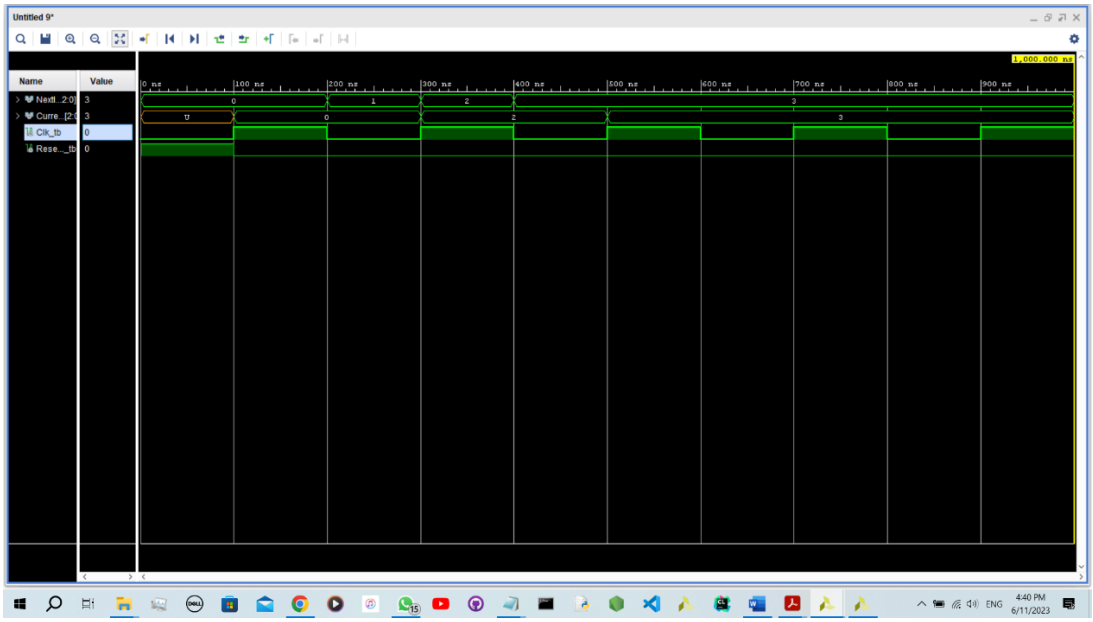
3 – bit Adder



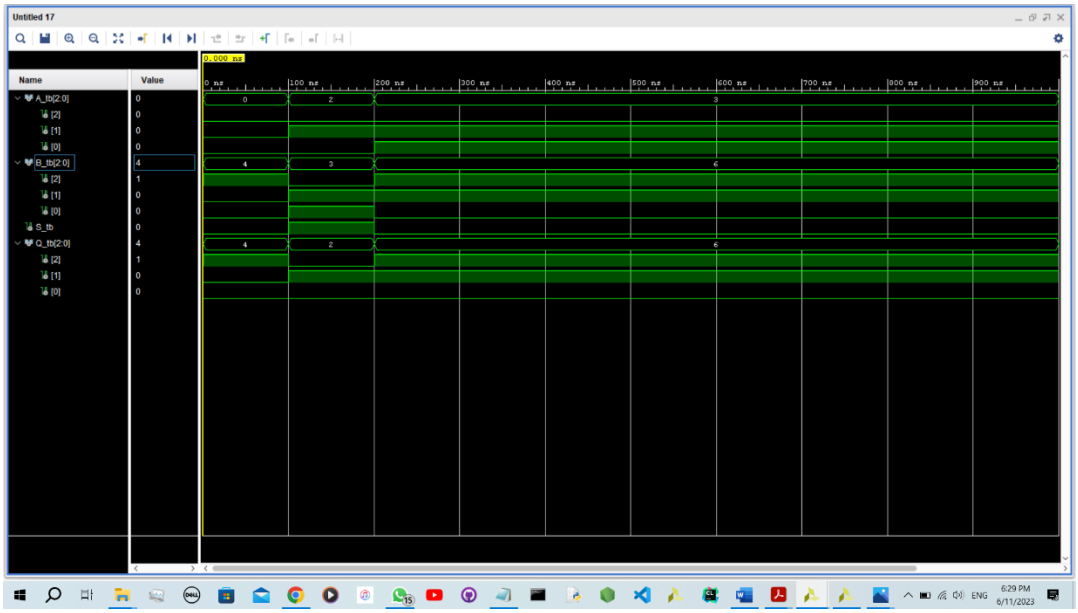
Program Rom



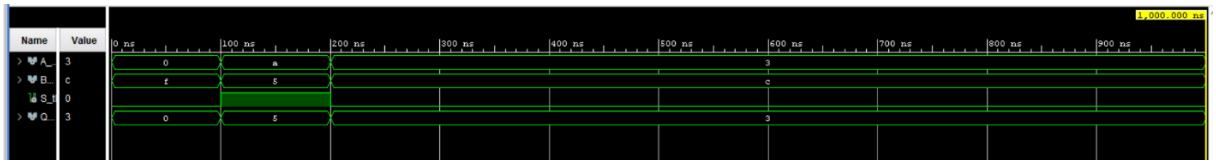
3 – bit Program Counter



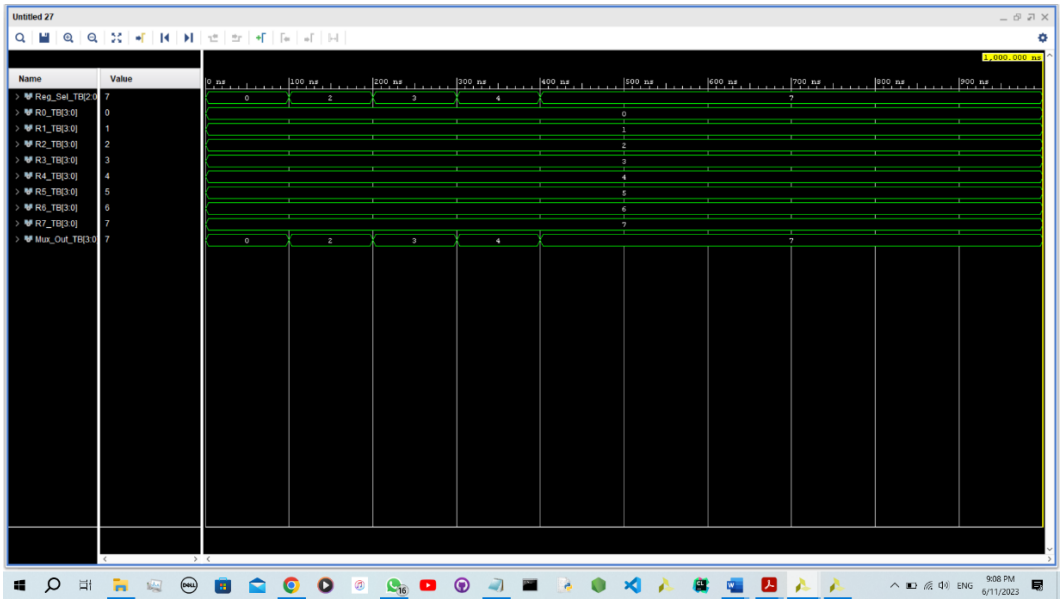
MUX_2way_3-bit



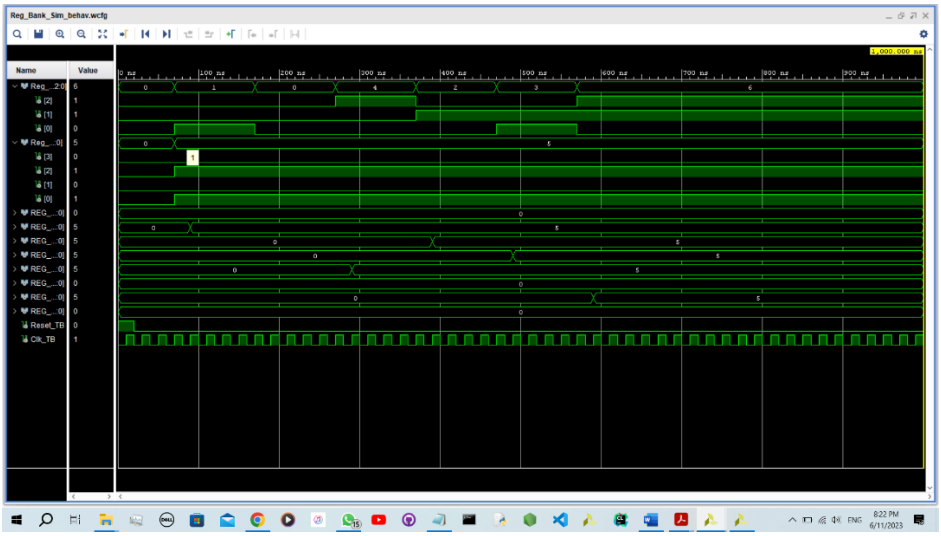
MUX_2way_4-bit



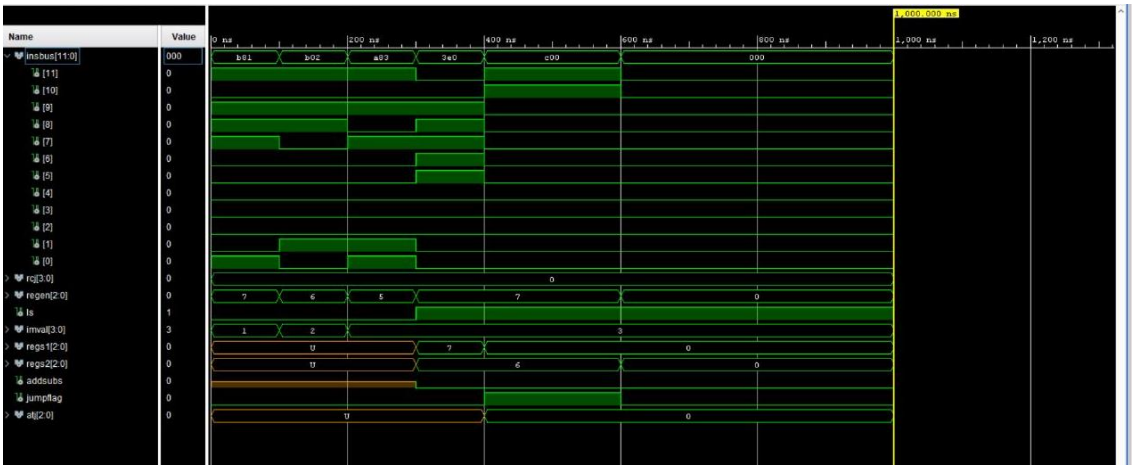
MUX_8way_4-bit



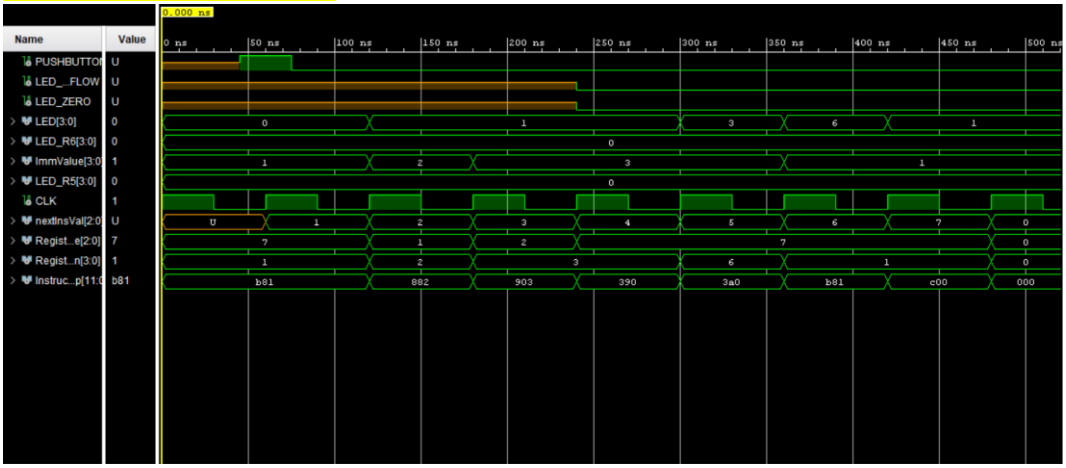
Register Bank



Instruction Decoder



Nano Processor



Conclusions from the Lab

In this lab, we designed and implemented essential components to create a functional nanoprocessor. These components included an add/subtract unit, adder, program counter, multiplexers, register bank, ROM, and instruction decoder. We successfully tested the processor by writing an Assembly program to calculate the total of integers from 1 to 3, converting it to machine code, and executing it on the development board. We faced challenges with component integration, timing, and module activation, but overcame them through efficient logic and thorough testing. The lab provided hands-on experience in computer organization, digital design, and teamwork skills, enhancing our understanding and appreciation for processor design.