Lab 9-10 – Nano processor Design Competition

CS1050 Computer Organization and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

210175C – G.G.T.A. Gamage

210429K - NINDUWARA K.G.M.

1. Lab Task
   Design a 4-bit processor capable of executing 4 instructions. The objective of this lab is to develop a 4-bit arithmetic unit that can perform addition and subtraction operations on signed integers. Additionally, we will design and develop the necessary components, such as instruction decoders, k-way b-bit multiplexers, and tri-state busses, to enable instruction execution.

2.

MOVI R2, 3	10 010 000 0011
MOVI R7, 0	10 111 000 0000
MOVI R4, -1	10 100 000 1110
ADD R7, R2	00 111 010 0000
ADD R2, R4	00 010 100 0000
JZR R2, 8	11 010 0000 111
JZR R0,4	11 000 0000 100
ADD R7, R0	00 111 000 0000

3. VHDL Codes

Nano Processor

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 06/06/2023 02:47:24 PM
```

```vhdl
-- Design Name:
-- Module Name: NanoProcessor - 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 NanoProcessor IS
    PORT (
        Clk : IN STD_LOGIC;
        Res : IN STD_LOGIC;
        Zero : OUT STD_LOGIC;
        Overflow : OUT STD_LOGIC;
        S_7Seg : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
        OUT_REG : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        Anode : out STD_LOGIC_VECTOR (3 downto 0)
    );
END NanoProcessor;

ARCHITECTURE Behavioral OF NanoProcessor IS
    COMPONENT InstructionDecoder PORT (
        Instruction : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
        Jump_sel : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        Imd_val : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        Reg_en : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
        Reg_selA : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        Load_sel : OUT STD_LOGIC;
        Reg_selB : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        --MUX_EN_A : OUT STD_LOGIC;
        --MUX_EN_B : OUT STD_LOGIC;
        REG_BANK_EN : OUT STD_LOGIC;
        --Mux_2_to_1_EN : OUT STD_LOGIC;
        Add_sub_sel : OUT STD_LOGIC;
        Jump_address : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        Jump : OUT STD_LOGIC);
    END COMPONENT;

    COMPONENT PC PORT (
        D_in : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        Res : IN STD_LOGIC;
        Clk : IN STD_LOGIC;
        Q_out : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
    END COMPONENT;

    COMPONENT Add_Sub PORT (
        M : IN STD_LOGIC;
        A0 : IN STD_LOGIC;
        A1 : IN STD_LOGIC;
        A2 : IN STD_LOGIC;
        A3 : IN STD_LOGIC;
        B0 : IN STD_LOGIC;
        B1 : IN STD_LOGIC;
        B2 : IN STD_LOGIC;
        B3 : IN STD_LOGIC;
        S0 : OUT STD_LOGIC;
        S1 : OUT STD_LOGIC;
        S2 : OUT STD_LOGIC;
        S3 : OUT STD_LOGIC;
        Zero_flag : OUT STD_LOGIC;
        Sign_flag : OUT STD_LOGIC;
        Overflow_flag : OUT STD_LOGIC;
        Carry_flag : OUT STD_LOGIC
        );
    END COMPONENT;

    COMPONENT adder_3bit PORT (
        A0 : IN STD_LOGIC;
        A1 : IN STD_LOGIC;
        A2 : IN STD_LOGIC;
        B0 : IN STD_LOGIC;
        B1 : IN STD_LOGIC;
        B2 : IN STD_LOGIC;
        C_in : IN STD_LOGIC;
        S0 : OUT STD_LOGIC;
        S1 : OUT STD_LOGIC;
        S2 : OUT STD_LOGIC;
        C_out : OUT STD_LOGIC);
    END COMPONENT;

    COMPONENT Register_Bank PORT (
        Reg_En : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        Clk : IN STD_LOGIC;
        A : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        Reg_Bank_En : IN STD_LOGIC;
        Reset : IN STD_LOGIC;
        B0 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B1 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B2 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B3 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B4 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B5 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B6 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        B7 : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
    END COMPONENT;

    COMPONENT ProgramRom PORT (
        Mem_address : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        Instruction : OUT STD_LOGIC_VECTOR (11 DOWNTO 0));
    END COMPONENT;

    COMPONENT Mux_2_to_1_3bit PORT (
        S : IN STD_LOGIC;
        A : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
    END COMPONENT;

    COMPONENT Mux_2_to_1_4bit PORT (
        S : IN STD_LOGIC;
        A : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
    END COMPONENT;

    COMPONENT Mux_8_to_1_4bit PORT (
        S : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
```

```
        A : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        C : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        E : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        F : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        G : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        H : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
    END COMPONENT;

    COMPONENT Slow_Clk PORT (
        Clk_in : IN STD_LOGIC;
        Clk_out : OUT STD_LOGIC);
    END COMPONENT;

    COMPONENT LUT_16_7 PORT (
        address : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        data : OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
    END COMPONENT;

    SIGNAL Imd_val_out : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Output of InstructionDecoder: Immediate value
    SIGNAL Reg_en_out : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of InstructionDecoder: Register enable
    SIGNAL Reg_selA_out : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of InstructionDecoder: Register select A
    SIGNAL Load_sel_out : STD_LOGIC; -- Output of InstructionDecoder: Load select
    SIGNAL Reg_selB_out : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of InstructionDecoder: Register select B
    --SIGNAL MUX_EN_A_out : STD_LOGIC; -- Output of InstructionDecoder: Multiplexer A enable
    --SIGNAL MUX_EN_B_out : STD_LOGIC; -- Output of InstructionDecoder: Multiplexer B enable
    SIGNAL REG_BANK_EN_out : STD_LOGIC; -- Output of InstructionDecoder: Register bank enable
    --SIGNAL Mux_2_to_1_EN_out : STD_LOGIC; -- Output of InstructionDecoder: Multiplexer 2-to-1 enable
    SIGNAL Add_sub_sel_out : STD_LOGIC; -- Output of InstructionDecoder: Add/subtract select
    SIGNAL Jump_address_out : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of InstructionDecoder: Jump address
    SIGNAL Jump_out : STD_LOGIC; -- Output of InstructionDecoder: Jump

    SIGNAL PC_out : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of PC: Current address
    SIGNAL S4bit : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Outputs of Add_Sub: Sum bits
    SIGNAL Zero_flag, Sign_flag, Overflow_flag, Carry_flag : STD_LOGIC; -- Outputs of Add_Sub: Flags
    SIGNAL Counter : STD_LOGIC_VECTOR(2 DOWNTO 0); -- output of 3bit adder
    SIGNAL C_out : STD_LOGIC; -- Output of adder_3bit: Carry-out

    SIGNAL RB0, RB1, RB2, RB3, RB4, RB5, RB6, RB7 : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Outputs of Register_Bank: Register
values
    SIGNAL Instruction : STD_LOGIC_VECTOR(11 DOWNTO 0); -- Output of ProgramRom: Instruction
    SIGNAL X_2_Way_3bit : STD_LOGIC_VECTOR(2 DOWNTO 0); -- Output of Mux_2_to_1_3bit: Selected input
    SIGNAL X_2_way_4bit : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Output of Mux_2_to_1_4bit: Selected input
```

```vhdl
    SIGNAL X_8_way_4bitA : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Output of Mux_8_to_1_4bit A : Selected input
    SIGNAL X_8_way_4bitB : STD_LOGIC_VECTOR(3 DOWNTO 0); -- Output of Mux_8_to_1_4bit B : Selected input
    SIGNAL Clk_out : STD_LOGIC; -- Output of Slow_Clk: Slow clock
    SIGNAL S_7Seg_out : STD_LOGIC_VECTOR(6 DOWNTO 0);

BEGIN

    LUT : LUT_16_7 PORT MAP(
        address => RB7,
        data => S_7Seg_out
    );

    InstructionDecoder_0 : InstructionDecoder PORT MAP(
        -- Port map connections
        Instruction => Instruction,
        Jump_sel => X_8_way_4bitA,
        Imd_val => Imd_val_out,
        Reg_en => Reg_en_out,
        Reg_selA => Reg_selA_out,
        Load_sel => Load_sel_out,
        Reg_selB => Reg_selB_out,
        --MUX_EN_A => MUX_EN_A_out,
        --MUX_EN_B => MUX_EN_B_out,
        REG_BANK_EN => REG_BANK_EN_out,
        --Mux_2_to_1_EN => Mux_2_to_1_EN_out,
        Add_sub_sel => Add_sub_sel_out,
        Jump_address => Jump_address_out,
        Jump => Jump_out
    );

    PC_0 : PC PORT MAP(
        -- Port map connections
        D_in => X_2_Way_3bit,
        Res => Res,
        Clk => Clk_out,
        Q_out => PC_out
    );

    Add_Sub_0 : Add_Sub PORT MAP(
        -- Port map connections
        M => Add_sub_sel_out,
        A0 => X_8_way_4bitA(0),
        A1 => X_8_way_4bitA(1),
        A2 => X_8_way_4bitA(2),
        A3 => X_8_way_4bitA(3),
        B0 => X_8_way_4bitB(0),
        B1 => X_8_way_4bitB(1),
        B2 => X_8_way_4bitB(2),
        B3 => X_8_way_4bitB(3),
        S0 => S4bit(0),
        S1 => S4bit(1),
        S2 => S4bit(2),
        S3 => S4bit(3),
        Zero_flag => Zero_flag,
        Sign_flag => Sign_flag,
        Overflow_flag => Overflow_flag,
        Carry_flag => Carry_flag
    );

    adder_3bit_0 : adder_3bit PORT MAP(
        A0 => PC_out(0),
        A1 => PC_out(1),
        A2 => PC_out(2),
        B0 => '0',
        B1 => '0',
        B2 => '0',
        C_in => '1',
        S0 => Counter(0),
        S1 => Counter(1),
        S2 => Counter(2),
        C_out => C_out
    );

    Register_Bank_0 : Register_Bank PORT MAP(
        -- Port map connections
        Reg_En => Reg_en_out,
        Clk => Clk_out,
        A => X_2_way_4bit,
        Reg_Bank_En => REG_BANK_EN_out,
        Reset => Res,
        B0 => RB0,
        B1 => RB1,
        B2 => RB2,
        B3 => RB3,
        B4 => RB4,
        B5 => RB5,
        B6 => RB6,
        B7 => RB7
    );
```

```
ProgramRom_0 : ProgramRom PORT MAP(
    -- Port map connections
    Mem_Address => PC_out,
    Instruction => Instruction
);

Mux_2_to_1_3bit_0 : Mux_2_to_1_3bit PORT MAP(
    -- Port map connections
    S => Jump_out,
    A => Jump_address_out,
    B => Counter,
    X => X_2_Way_3bit
);

Mux_2_to_1_4bit_0 : Mux_2_to_1_4bit PORT MAP(
    -- Port map connections
    S => Load_sel_out,
    A => Imd_val_out,
    B => S4bit,
    X => X_2_way_4bit
);

Mux_8_to_1_4bit_A : Mux_8_to_1_4bit PORT MAP(
    -- Port map connections
    S => Reg_selA_out,
    A => RB0,
    B => RB1,
    C => RB2,
    D => RB3,
    E => RB4,
    F => RB5,
    G => RB6,
    H => RB7,
    X => X_8_way_4bitA
);

Mux_8_to_1_4bit_B : Mux_8_to_1_4bit PORT MAP(
    -- Port map connections
    S => Reg_selB_out,
    A => RB0,
    B => RB1,
    C => RB2,
    D => RB3,
    E => RB4,
    F => RB5,
        G => RB6,
        H => RB7,
        X => X_8_way_4bitB
    );

    Slow_Clk_0 : Slow_Clk PORT MAP(
        Clk_in => Clk,
        Clk_out => Clk_out
    );

    OUT_REG <= RB7;
    Overflow <= Overflow_flag;
    Zero <= Zero_flag;
    S_7Seg <= S_7Seg_out;
    Anode <= "1110";

END Behavioral;
```

LUT_16_7

```
----------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 06/08/2023 01:36:19 AM
-- Design Name:
-- Module Name: LUT_16 - 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;
```

```vhdl
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
    "0110001", -- 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;
```

Instruction Decoder

```
--------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 06/05/2023 11:22:34 PM
-- Design Name:
-- Module Name: Instruction_Decoder - 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;
ENTITY InstructionDecoder IS
    PORT (
        Instruction : IN STD_LOGIC_VECTOR (11 DOWNTO 0);
        Jump_sel : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        Imd_val : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        Reg_en : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
        Reg_selA : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        Load_sel : OUT STD_LOGIC;
        Reg_selB : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        --MUX_EN_A : OUT STD_LOGIC;
        --MUX_EN_B : OUT STD_LOGIC;
        REG_BANK_EN : OUT STD_LOGIC;
        --Mux_2_to_1_EN : OUT STD_LOGIC;
        Add_sub_sel : OUT STD_LOGIC;
        Jump_address : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
        Jump : OUT STD_LOGIC);
```

```vhdl
END InstructionDecoder;

ARCHITECTURE Behavioral OF InstructionDecoder IS

BEGIN

    PROCESS (Instruction, Jump_Sel)
    BEGIN

       -- MUX_EN_A <= '0';
        --MUX_EN_B <= '0';
        REG_BANK_EN <= '0';
        Jump <= '0';

        IF (Instruction(11 DOWNTO 10) = "00") THEN -- add
          --  MUX_EN_A <= '1';
          -- MUX_EN_B <= '1';
            Reg_selA <= Instruction(9 DOWNTO 7);
            Reg_selB <= Instruction(6 DOWNTO 4);
            Add_sub_sel <= '0';
           -- Mux_2_to_1_EN <= '1';
            Load_sel <= '0';
            REG_BANK_EN <= '1';
            Reg_en <= Instruction(9 DOWNTO 7);

        ELSIF (Instruction(11 DOWNTO 10) = "01") THEN -- neg
           -- MUX_EN_A <= '1';
            --MUX_EN_B <= '1';
            Reg_selA <= "000";
            Reg_selB <= Instruction(9 DOWNTO 7);
            Add_sub_sel <= '1';
            --Mux_2_to_1_EN <= '1';
            Load_sel <= '0';
            REG_BANK_EN <= '1';
            Reg_en <= Instruction(9 DOWNTO 7);

        ELSIF (Instruction(11 DOWNTO 10) = "10") THEN --movi
            REG_BANK_EN <= '1';
            Reg_en <= Instruction(9 DOWNTO 7);
            --Mux_2_to_1_EN <= '1';
            Load_sel <= '1';
            Imd_val <= Instruction(3 DOWNTO 0);

        ELSIF (Instruction(11 DOWNTO 10) = "11") THEN  --jump
            -- JZR R1, 7   ; If R1 = 0 jump to line 7 format: 11 RRR 0000 ddd
            Reg_selA <= Instruction(9 DOWNTO 7); -- GET THE VALUE OF THE REGISTER
RRR IN THE REGISTER A
            IF (Jump_sel = "0000") THEN
                Jump <= '1'; -- set jump flag
                Jump_address <= Instruction(2 DOWNTO 0);
            END IF;

        END IF;
    END PROCESS;
END Behavioral;
```

Program Counter

```
--------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 05/23/2023 03:58:14 PM
-- Design Name:
-- Module Name: PC - 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;
```

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

ENTITY PC IS
    PORT (
        D_in : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        Res : IN STD_LOGIC;
        Clk : IN STD_LOGIC;
        Q_out : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
END PC;

ARCHITECTURE Behavioral OF PC IS
    COMPONENT D_FF
        PORT (
            D : IN STD_LOGIC;
            Res : IN STD_LOGIC;
            Clk : IN STD_LOGIC;
            Q : OUT STD_LOGIC);
    END COMPONENT;

BEGIN

    D_FF0 : D_FF
    PORT MAP(
        D => D_in(0),
        Res => Res,
        Clk => Clk,
        Q => Q_out(0)
    );

    D_FF1 : D_FF
    PORT MAP(
        D => D_in(1),
        Res => Res,
        Clk => Clk,
        Q => Q_out(1)
    );

    D_FF2 : D_FF
    PORT MAP(
        D => D_in(2),
        Res => Res,
        Clk => Clk,
        Q => Q_out(2)
    );
END Behavioral;
```

Adder\ Substracter

```
---------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 03/16/2023 11:47:57 AM
-- Design Name:
-- Module Name: RCA_4 - 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 Add_Sub IS
    PORT (
        M : IN STD_LOGIC;
        A0 : IN STD_LOGIC;
        A1 : IN STD_LOGIC;
        A2 : IN STD_LOGIC;
        A3 : IN STD_LOGIC;
        B0 : IN STD_LOGIC;
        B1 : IN STD_LOGIC;
        B2 : IN STD_LOGIC;
        B3 : IN STD_LOGIC;
        S0 : OUT STD_LOGIC;
        S1 : OUT STD_LOGIC;
        S2 : OUT STD_LOGIC;
        S3 : OUT STD_LOGIC;
        Zero_flag : OUT STD_LOGIC;
        Sign_flag : OUT STD_LOGIC;
        Overflow_flag : OUT STD_LOGIC;
        Carry_flag : OUT STD_LOGIC
    );
END Add_Sub;

ARCHITECTURE Behavioral OF Add_Sub IS
    COMPONENT FA
        PORT (
            A : IN STD_LOGIC;
            B : IN STD_LOGIC;
            C_in : IN STD_LOGIC;
            S : OUT STD_LOGIC;
            C_out : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL FA0_S, FA0_C, FA1_S, FA1_C, FA2_S, FA2_C, FA3_S, FA3_C : STD_LOGIC;
    SIGNAL NB0, NB1, NB2, NB3 : STD_LOGIC;
    SIGNAL S : STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

    NB0 <= (B0 XOR M);
    NB1 <= (B1 XOR M);
    NB2 <= (B2 XOR M);
    NB3 <= (B3 XOR M);

    FA_0 : FA
    PORT MAP(
        A => A0,
        B => NB0,
        C_in => M,
        S => S(0),
        C_Out => FA0_C);
```

```vhdl
    FA_1 : FA
    PORT MAP(
        A => A1,
        B => NB1,
        C_in => FA0_C,
        S => S(1),
        C_Out => FA1_C);

    FA_2 : FA
    PORT MAP(
        A => A2,
        B => NB2,
        C_in => FA1_C,
        S => S(2),
        C_Out => FA2_C);
    FA_3 : FA
    PORT MAP(
        A => A3,
        B => NB3,
        C_in => FA2_C,
        S => S(3),
        C_Out => Carry_flag);
    S0 <= S(0);
    S1 <= S(1);
    S2 <= S(2);
    S3 <= S(3);

    Zero_flag <= (NOT S(0)) AND (NOT S(1)) AND (NOT S(2)) AND (NOT S(3));

    Sign_flag <= S(3);

    Overflow_flag <= (NOT M) AND (A3 XNOR B3) AND (A3 XOR S(3));

END Behavioral;
```

Adder_3bit

```vhdl
----------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 05/23/2023 03:41:39 PM
-- Design Name:
```

```vhdl
-- Module Name: RCA_3 - 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 (
        A0 : IN STD_LOGIC;
        A1 : IN STD_LOGIC;
        A2 : IN STD_LOGIC;

        B0 : IN STD_LOGIC;
        B1 : IN STD_LOGIC;
        B2 : IN STD_LOGIC;

        C_in : IN STD_LOGIC;
        S0 : OUT STD_LOGIC;
        S1 : OUT STD_LOGIC;
        S2 : OUT STD_LOGIC;

        C_out : OUT STD_LOGIC);
END 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 : STD_LOGIC;

BEGIN
    FA_0 : FA
    PORT MAP(
        A => A0,
        B => B0,
        C_in => C_in,
        S => S0,
        C_Out => FA0_C);

    FA_1 : FA
    PORT MAP(
        A => A1,
        B => B1,
        C_in => FA0_C,
        S => S1,
        C_Out => FA1_C);

    FA_2 : FA
    PORT MAP(
        A => A2,
        B => B2,
        C_in => FA1_C,
        S => S2,
        C_Out => C_out);
END Behavioral;
```

Register Bank

```
--------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
```

```vhdl
-- Create Date: 06/01/2023 04:31:10 PM
-- 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 Register_Bank is
    port (
        Reg_En: in std_logic_vector(2 downto 0);
        Clk: in std_logic;
        A: in std_logic_vector(3 downto 0);
        Reg_Bank_En: in std_logic;
        Reset: in std_logic;
        B0: out std_logic_vector(3 downto 0);
        B1: out std_logic_vector(3 downto 0);
        B2: out std_logic_vector(3 downto 0);
        B3: out std_logic_vector(3 downto 0);
        B4: out std_logic_vector(3 downto 0);
        B5: out std_logic_vector(3 downto 0);
        B6: out std_logic_vector(3 downto 0);
        B7: out std_logic_vector(3 downto 0)
    );
end Register_Bank;

architecture Behavioral of Register_Bank is
component Reg
    port(
          D : in STD_LOGIC_VECTOR (3 downto 0);
          En : in STD_LOGIC;
          Clk : in STD_LOGIC;
          Res : in STD_LOGIC;
          Q : out STD_LOGIC_VECTOR (3 downto 0)
    );
end component;

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

   signal YB : std_logic_vector(7 downto 0);


begin
    Decoder_3_to_8_1 : Decoder_3_to_8 port map(
        I => Reg_En,
        EN => Reg_Bank_En,
        Y => YB
    );

    R0: Reg port map(
        D => "0000",
        En => '1',
        Clk => Clk,
        Res => Reset,
        Q => B0
    );
    R1: Reg port map(
        D => A,
        En => YB(1),
        Clk => Clk,
        Res => Reset,
        Q => B1
    );
    R2: Reg port map(
        D => A,
        En => YB(2),
        Clk => Clk,
        Res => Reset,
        Q => B2
    );
    R3: Reg port map(
        D => A,
        En => YB(3),
        Clk => Clk,
        Res => Reset,
        Q => B3
    );
    R4: Reg port map(
        D => A,
        En => YB(4),
        Clk => Clk,
        Res => Reset,
        Q => B4
    );
    R5: Reg port map(
        D => A,
        En => YB(5),
        Clk => Clk,
        Res => Reset,
        Q => B5
    );
    R6: Reg port map(
        D => A,
        En => YB(6),
        Clk => Clk,
        Res => Reset,
        Q => B6
    );
    R7: Reg port map(
        D => A,
        En => YB(7),
        Clk => Clk,
        Res => Reset,
        Q => B7
    );

end Behavioral;
```

Program ROM

```
----------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 06/06/2023 01:52:25 PM
-- Design Name:
-- Module Name: Program_ROM - 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;
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 ProgramRom IS
    PORT (
        Mem_address : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        Instruction : OUT STD_LOGIC_VECTOR (11 DOWNTO 0));
END ProgramRom;

ARCHITECTURE Behavioral OF ProgramRom IS
    TYPE rom_type IS ARRAY (0 TO 7) OF STD_LOGIC_VECTOR(11 DOWNTO 0);
    SIGNAL progam_ROM : rom_type := (
        "100100000011", --MOVI R2, 3                10 RRR 000 0011
        "101110000000", --MOVI R7, 0                10 RRR 000 0000
        "101000001111", --MOVI R4, -1               10 RRR 000 1110
        "001110100000", --ADD R7, R2                00 RRR RRR 0000
        "000101000000", --ADD R2, R4                00 RRR RRR 0000
        "110100000111", --JZR R2, 8                 11 RRR 0000 111
        "110000000011", --JZR R0,4                  11 RRR 0000 100
        "001110000000" --ADD R7, R0                 00 RRR RRR 0000
    );

BEGIN
    Instruction <= progam_ROM(to_integer(unsigned(Mem_address)));
END Behavioral;
```

2 way 3 bit MUX

```
--------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 05/31/2023 12:11:03 AM
-- Design Name:
-- Module Name: 2_way_3_bit_mux - 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_to_1_3bit IS
    PORT (
        S : IN STD_LOGIC;
        A : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
END Mux_2_to_1_3bit;

ARCHITECTURE Behavioral OF Mux_2_to_1_3bit IS

BEGIN
    X <= A WHEN (S = '1') ELSE
    B;

END Behavioral;
```

2 way 4 bit MUX

```
---------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 05/31/2023 12:36:47 AM
-- Design Name:
-- Module Name: Mux_2_to_1_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_to_1_4bit IS
    PORT (
        S : IN STD_LOGIC;
        A : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END Mux_2_to_1_4bit;

ARCHITECTURE Behavioral OF Mux_2_to_1_4bit IS

BEGIN
    X <= A WHEN (S = '1') ELSE
        B;

END Behavioral;
```

8 way 4 bit MUX

```
--------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 05/31/2023 12:50:47 AM
-- Design Name:
-- Module Name: Mux_8_to_1_4bit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
```

```vhdl
-- 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_to_1_4bit IS
    PORT (
        S : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        A : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        C : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        E : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        F : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        G : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        H : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        X : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END Mux_8_to_1_4bit;

ARCHITECTURE Behavioral OF Mux_8_to_1_4bit IS

BEGIN
    WITH s SELECT
        X <= A WHEN "000",
        B WHEN "001",
        C WHEN "010",
        D WHEN "011",
        E WHEN "100",
        F WHEN "101",
        G WHEN "110",
        H WHEN OTHERS;
```

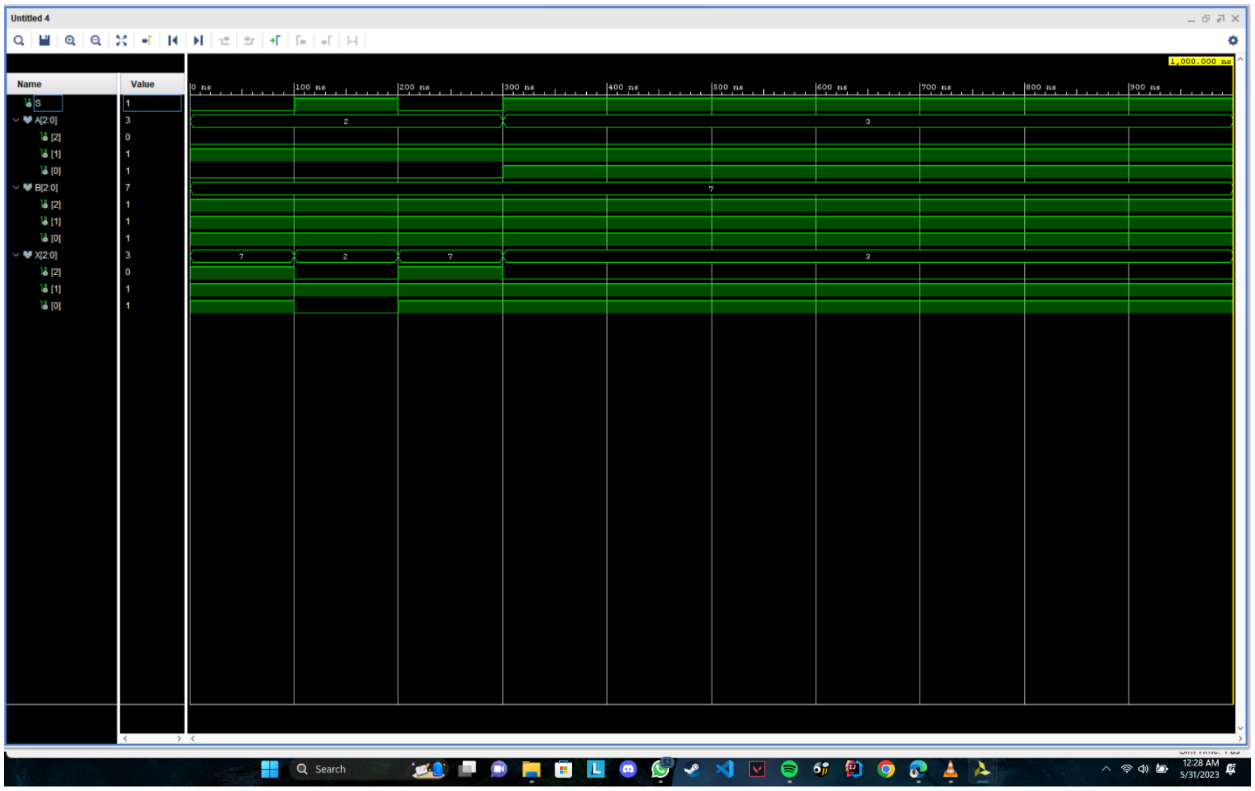
```
END behavioral;
```

Slow Clock

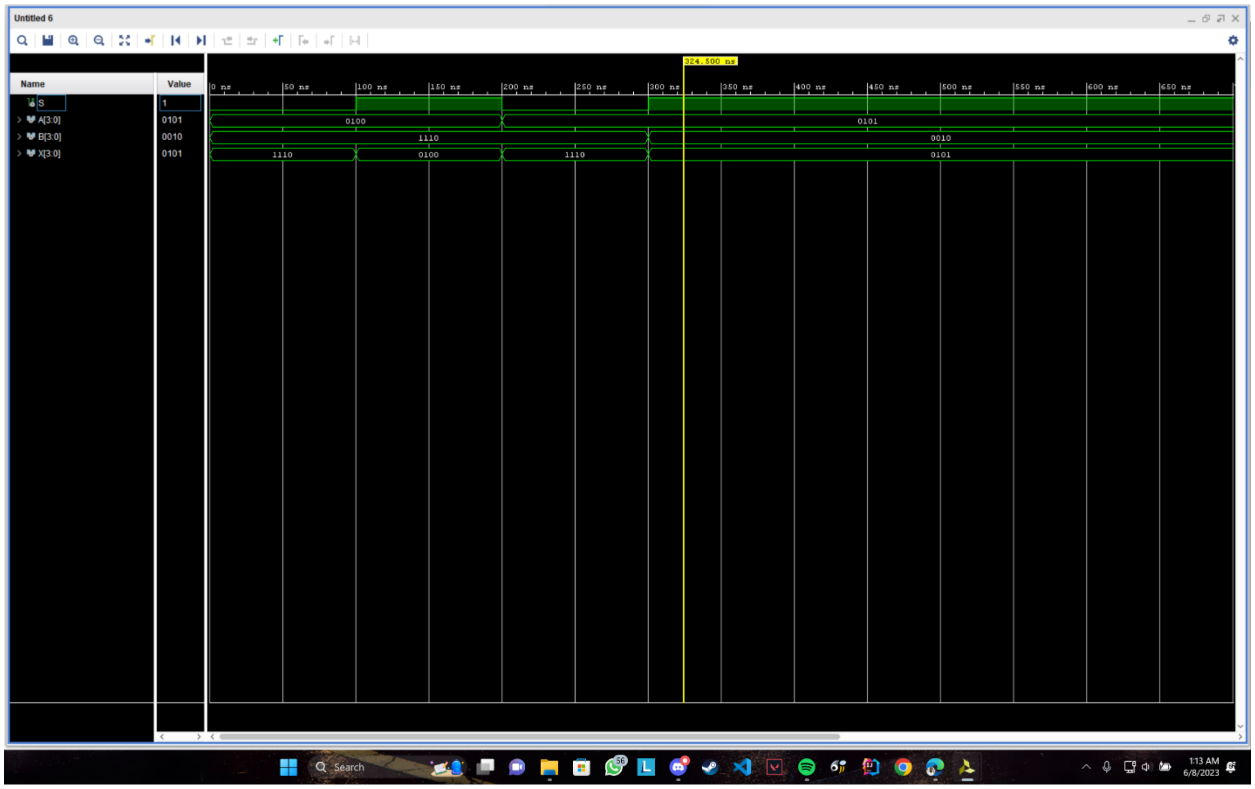
```
----------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date: 04/04/2023 12:13:55 PM
-- Design Name:
-- Module Name: Slow_Clk - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- 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 Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;
```

```vhdl
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 = 1) then
            Clk_status <= not Clk_status;
            Clk_out <= Clk_status;
            count <= 1;
        end if;
    end if;
    end process;

end Behavioral;
```

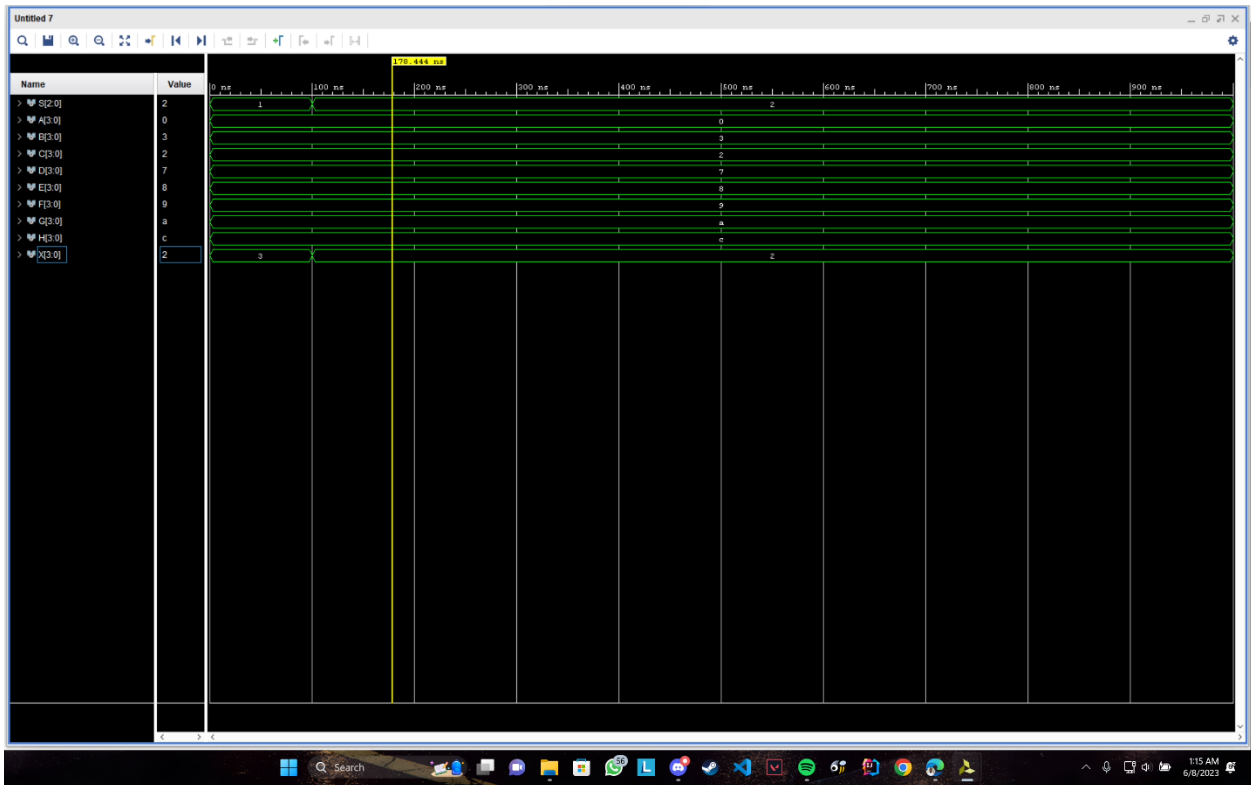
2 WAY 3 BIT MUX



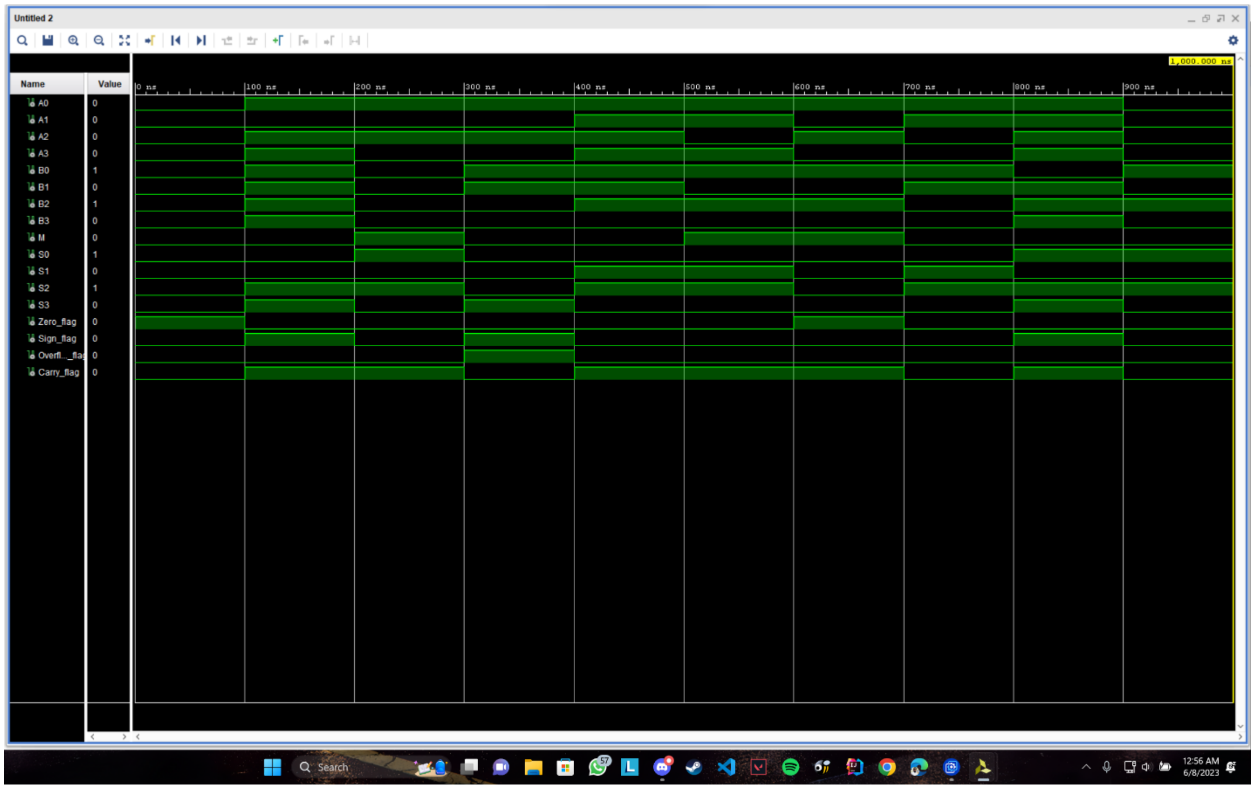
2 WAY 4 BIT MUX



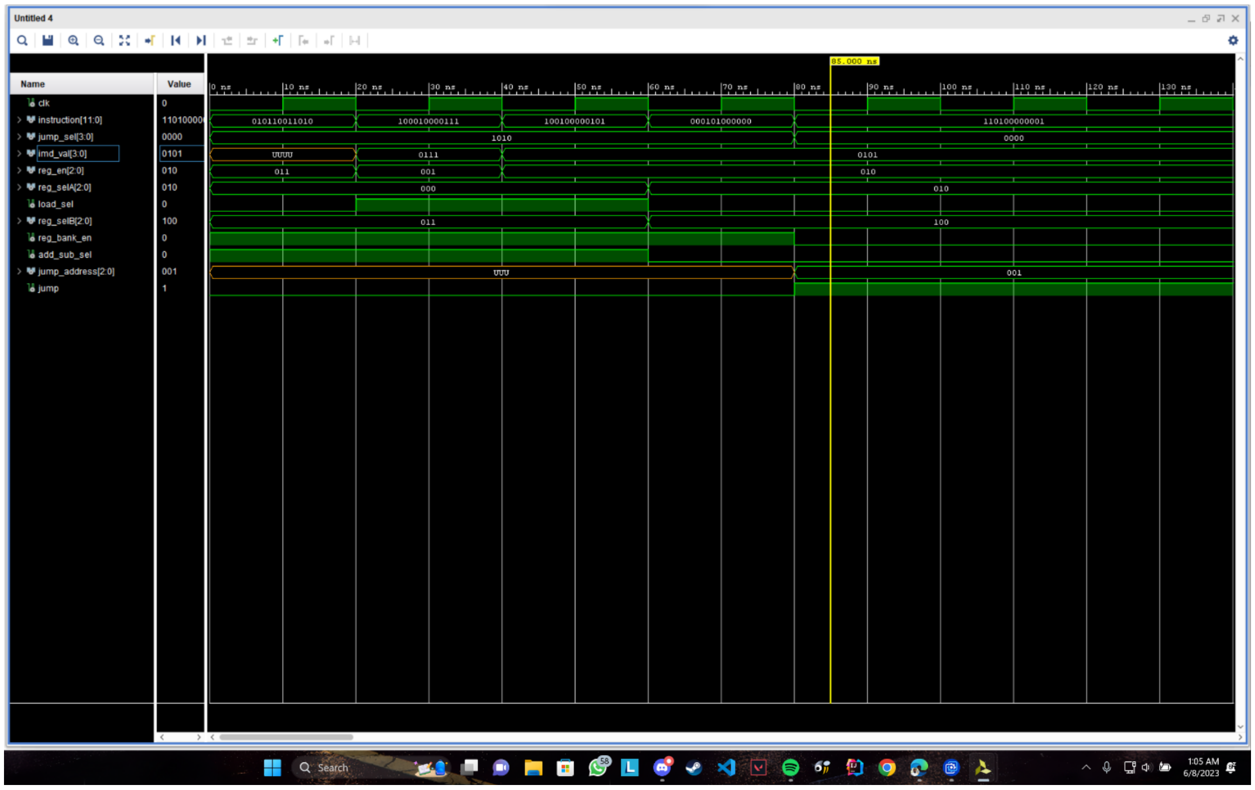
8 WAY 4 BIT MUX



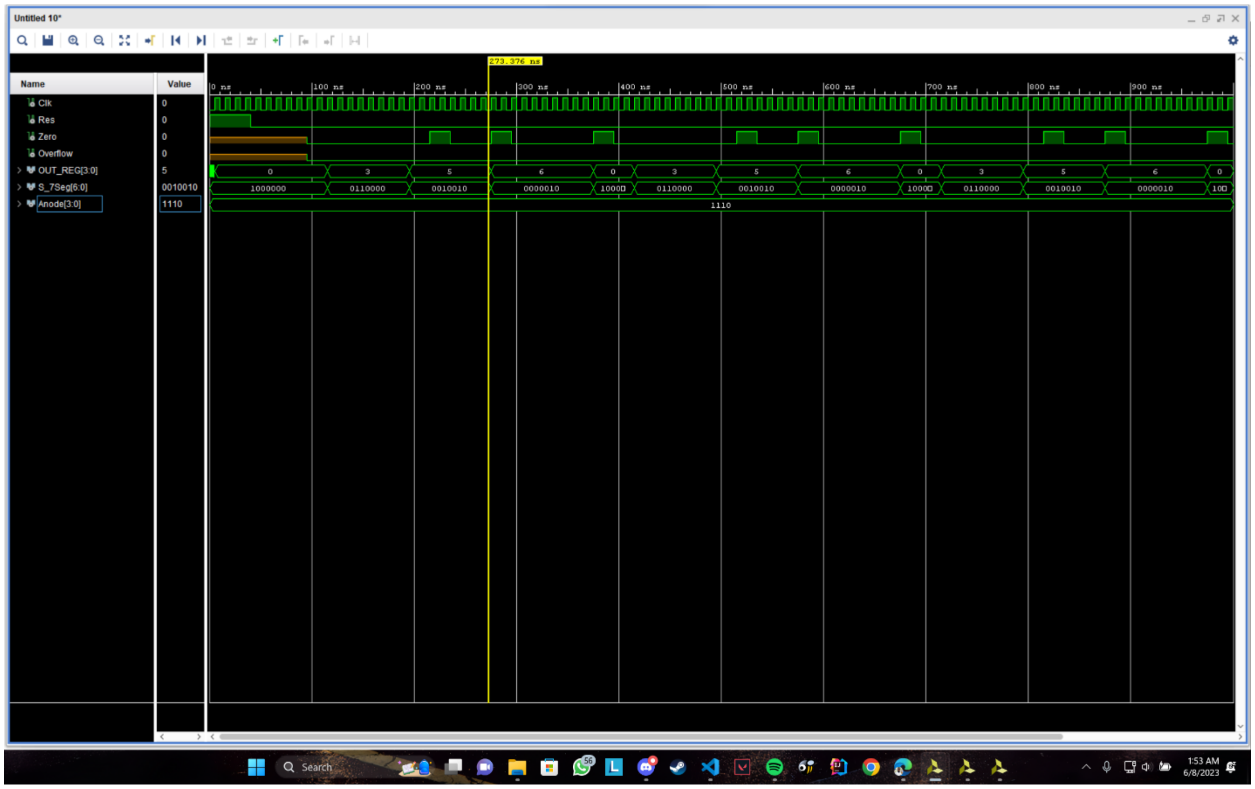
Add_Sub sim



InstrcutionDecoder_SIM

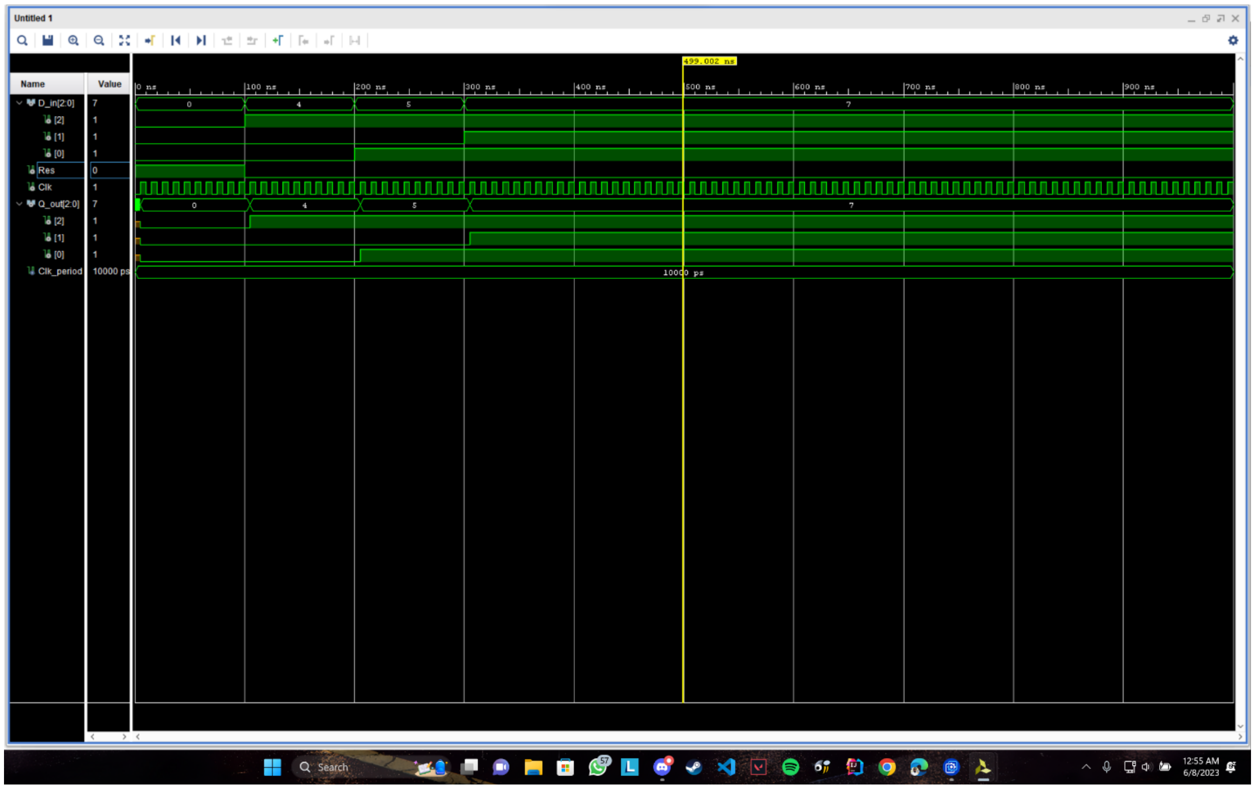


Nano_sim



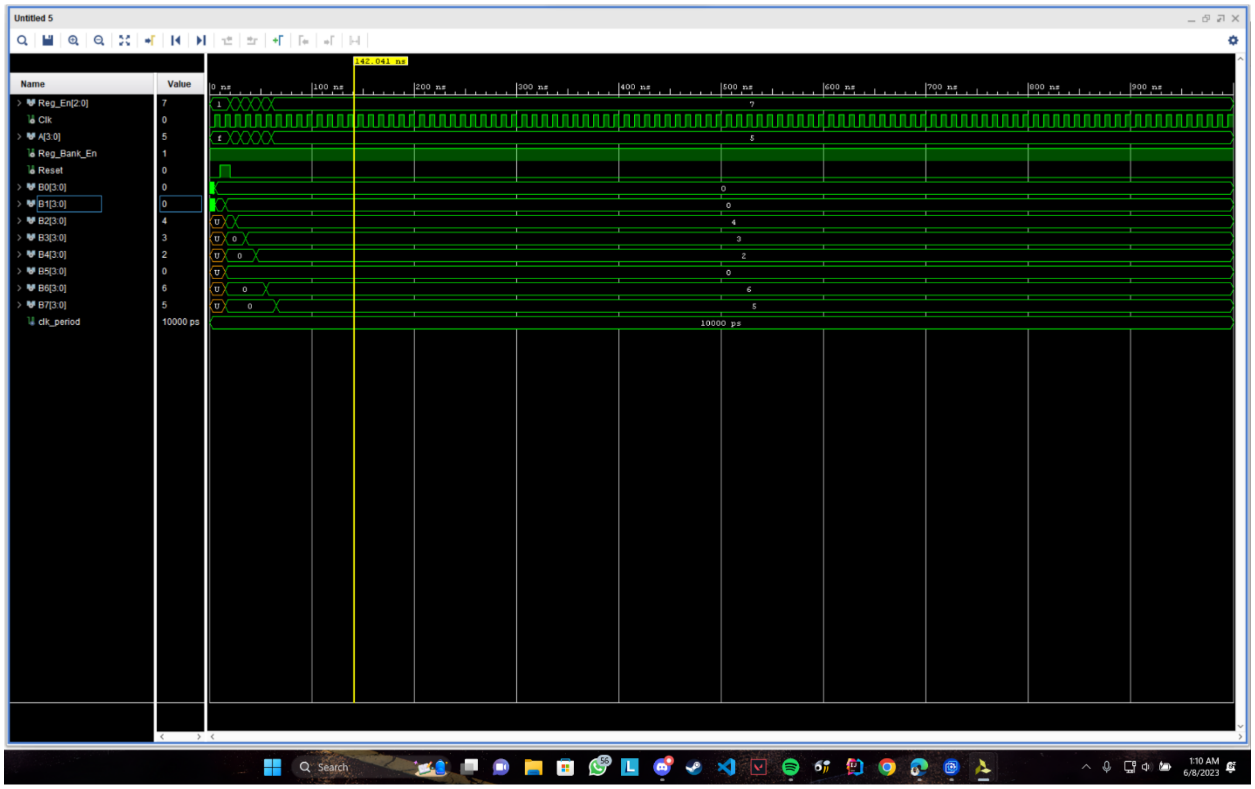
Name	Value
Clk	0
Res	0
Zero	0
Overflow	0
OUT_REG[3:0]	5
S_7Seg[6:0]	0010010
Anode[3:0]	1110

PC_Sim



Name	Value
D_in[2:0]	7
[2]	1
[1]	1
[0]	1
Res	0
Clk	1
Q_out[2:0]	7
[2]	1
[1]	1
[0]	1
Clk_period	10000 ps

Register_Bank_Sim



Conclusion

In this lab, we designed and developed a 4-bit nanoprocessor that can execute four instructions: MOVI, ADD, NEG, and JZR. We used various components such as a 4-bit add/subtract unit, a 3-bit program counter, an 8-way 4-bit multiplexer, a register bank, a program ROM, and an instruction decoder. We tested our design using simulation and on the BASYS 3 board. We also wrote an assembly program to calculate the sum of integers from 1 to 3 and verified the result on R7. We learned how to apply the principles of computer organization and digital design to build a simple microprocessor and how to use VHDL to implement and test our circuits. Our experiment succeeded in meeting the learning outcomes of the lab.

Team members

We shared the components and created them.

Gamage – Instruction Decoder, Program ROM, Register Bank, PC

Ninduwara – All MUXes, 2 Adders

After creating each component we discussed and made the nanoprocessor together.