**Digital Design and Computer Organization**

Semester 2, 2023

*Lab 9 & 10*

Nano Processor

Team Number: 56

Team Members

* J.P.S.A. Chandrawansha – 210077D
* P.H.T. Dinujaya – 210134C

Tasks:

* Developed the basic logic units such as D flip-flop, tri-state buffer, 4-bit register, slow clock which are needed to build other logical systems of the processor.
* Develop 4-bit arithmetic unit to adding and subtracting signed integers.
* Design decoders to integrate into some other logical parts of the processor like register bank for register selecting and multiplexer for data selection.
* Designed various types of multiplexers by using pre-built decoders and tri-state buffers.
* Created the program-ROM to store our assembly program in binary format.
* Designed and developed the instruction decoder of the processor.
* Assembled all the logical systems we built, to finally construct the whole Nano-processor.
* Test functionality of the various logic parts and the processor throughout series of simulations and confirm that are work fine.

Assembly Code

Assembly instructions

MOVI R7, 10 *; R7 <- 10*

MOVI R1, 1  *; R1 <- 1*

NEG R1      *; R1 <- -R1*

ADD R7, R1  *; R7 <- R7 + R1*

JZR R7, 7   *; If R7 = 0 jump to line 7*

JZR R0, 3   *; If R1 = 0 jump to line 3*

Machine instructions

101110001010

100010000001

010010000000

001110010000

111110000111

110000000011

VHDL Codes

\*Visit [nano-processor](https://github.com/shavinanjitha2002/NanoProcessor.git) git-hub repository for Project Repository.

**HA**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity HA is

    Port ( A : in STD\_LOGIC;

           B : in STD\_LOGIC;

           S : out STD\_LOGIC;

           C : out STD\_LOGIC);

end HA;

architecture Behavioral of HA is

begin

    S <= A XOR B;

    C <= A AND B;

end Behavioral;

**FA**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FA is

    Port ( A : in STD\_LOGIC;

           B : in STD\_LOGIC;

           C\_in : in STD\_LOGIC;

           S : out STD\_LOGIC;

           C\_out : out STD\_LOGIC);

end FA;

architecture Behavioral of FA is

    Component HA

        Port (

            A : in STD\_LOGIC;

            B : in STD\_LOGIC;

            S : out STD\_LOGIC;

            C : out STD\_LOGIC

        );

    End Component;

    SIGNAL HA0\_S, HA0\_C, HA1\_C : STD\_LOGIC;

begin

    HA\_0 : HA

        PORT MAP (

            A => A,

            B => B,

            S => HA0\_S,

            C => HA0\_C

        );

    HA\_1 : HA

        PORT MAP (

            A => HA0\_S,

            B => C\_in,

            S => S,

            C => HA1\_C

        );

    C\_out <= HA0\_C OR HA1\_C;

end Behavioral;

**3 Bit Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Adder\_3 is

    Port ( A\_in : in STD\_LOGIC\_VECTOR (2 downto 0);

           S\_out : out STD\_LOGIC\_VECTOR (2 downto 0));

end Adder\_3;

architecture Behavioral of Adder\_3 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\_out : STD\_LOGIC\_VECTOR(2 downto 0);

begin

    FA\_0 : FA

        PORT MAP(

            A => A\_in(0),

            B => '1',

            S =>  S\_out(0),

            C\_in => '0',

            C\_out => FA\_C\_out(0));

    FA\_1 : FA

        port map(

            A => A\_in(1),

            B => '0',

            S =>  S\_out(1),

            C\_in => FA\_C\_out(0),

            C\_out => FA\_C\_out(1));

    FA\_2 : FA

        PORT MAP(

            A => A\_in(2),

            B => '0',

            S =>  S\_out(2),

            C\_in => FA\_C\_out(1),

            C\_out => FA\_C\_out(2));

end Behavioral;

**Buffer**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Buff\_3 is

    Port ( D : in STD\_LOGIC\_VECTOR (2 downto 0);

           Ctrl : in STD\_LOGIC;

           Y : out STD\_LOGIC\_VECTOR (2 downto 0));

end Buff\_3;

architecture Behavioral of Buff\_3 is

begin

    Y <= D WHEN Ctrl = '1' ELSE "ZZZ";

end Behavioral;

**Mux\_2\_Way\_3**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_2\_way\_3 is

    Port ( I : in STD\_LOGIC;

           A0, A1 : STD\_LOGIC\_VECTOR(2 downto 0);

           Y : out STD\_LOGIC\_VECTOR (2 downto 0));

end Mux\_2\_way\_3;

architecture Behavioral of Mux\_2\_way\_3 is

    Component Buff\_3

        PORT (

            D : in STD\_LOGIC\_VECTOR(2 downto 0);

            Ctrl : in STD\_LOGIC;

            Y : out STD\_LOGIC\_VECTOR(2 downto 0)

        );

    End Component;

    SIGNAL NOT\_I : STD\_LOGIC;

begin

    NOT\_I <= NOT(I);

    Buff\_3\_0 : Buff\_3 PORT MAP(D => A0, Ctrl => NOT\_I, Y => Y);

    Buff\_3\_1 : Buff\_3 PORT MAP(D => A1, Ctrl => I, Y => Y);

end Behavioral;

**4 Bit Buffer**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Buff\_4 is

    Port ( D : in STD\_LOGIC\_VECTOR (3 downto 0);

           Ctrl : in STD\_LOGIC;

           Y : out STD\_LOGIC\_VECTOR (3 downto 0));

end Buff\_4;

architecture Behavioral of Buff\_4 is

begin

    Y <= D WHEN Ctrl = '1' ELSE "ZZZZ";

end Behavioral;

**Mux\_2\_Way\_4**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_2\_way\_4 is

    Port ( I : in STD\_LOGIC;

           A0, A1 : in STD\_LOGIC\_VECTOR (3 downto 0);

           Y : out STD\_LOGIC\_VECTOR (3 downto 0));

end Mux\_2\_way\_4;

architecture Behavioral of Mux\_2\_way\_4 is

    Component Buff\_4

        PORT (

            D : in STD\_LOGIC\_VECTOR(3 downto 0);

            Ctrl : in STD\_LOGIC;

            Y : out STD\_LOGIC\_VECTOR(3 downto 0)

        );

    End Component;

    SIGNAL NOT\_I : STD\_LOGIC;

begin

    NOT\_I <= NOT(I);

    Buff\_4\_0 : Buff\_4 PORT MAP(D => A0, Ctrl => NOT\_I, Y => Y);

    Buff\_4\_1 : Buff\_4 PORT MAP(D => A1, Ctrl => I, Y => Y);

end Behavioral;

**Decoder\_2\_to\_4**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decoder\_2\_to\_4 is

    Port ( I : in STD\_LOGIC\_VECTOR (1 downto 0);

           En : in STD\_LOGIC;

           Y : out STD\_LOGIC\_VECTOR (3 downto 0));

end Decoder\_2\_to\_4;

architecture Behavioral of Decoder\_2\_to\_4 is

begin

    Y(0) <= En AND NOT(I(0)) AND NOT(I(1));

    Y(1) <= En AND I(0) AND NOT(I(1));

    Y(2) <= En AND NOT(I(0)) AND I(1);

    Y(3) <= En AND I(0) AND I(1);

end Behavioral;

**Decoder\_3\_to\_8**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decoder\_3\_to\_8 is

    Port ( I : in STD\_LOGIC\_VECTOR (2 downto 0);

           En : in STD\_LOGIC;

           Y : out STD\_LOGIC\_VECTOR (7 downto 0));

end Decoder\_3\_to\_8;

architecture Behavioral of Decoder\_3\_to\_8 is

    Component Decoder\_2\_to\_4

        PORT (

            I : in STD\_LOGIC\_VECTOR(1 downto 0);

            En : in STD\_LOGIC;

            Y : out STD\_LOGIC\_VECTOR(3 downto 0)

        );

    End Component;

    SIGNAL En\_0, En\_1 : STD\_LOGIC;

begin

    Decoder\_2\_to\_4\_0 : Decoder\_2\_to\_4

        PORT MAP (

            I => I(1 downto 0),

            En => En\_0,

            Y => Y(3 downto 0)

        );

    Decoder\_2\_to\_4\_1 : Decoder\_2\_to\_4

        PORT MAP (

            I => I(1 downto 0),

            En => En\_1,

            Y => Y(7 downto 4)

        );

    En\_0 <= En AND NOT(I(2));

    En\_1 <= En AND I(2);

end Behavioral;

**Register**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Reg is

    Port ( Clk : in STD\_LOGIC;

           En : in STD\_LOGIC;

           Rst : in STD\_LOGIC;

           D : in STD\_LOGIC\_VECTOR (3 downto 0);

           Q : out STD\_LOGIC\_VECTOR (3 downto 0));

end Reg;

architecture Behavioral of Reg is

begin

    process (Clk, Rst) begin

        if Rst'event AND Rst = '1' then Q <= "0000"; -- reset the register asynchronously

        else if (rising\_edge(Clk)) then -- respont at the rising edge of the clock pulse

            if En = '1' then -- store the 4-bit fi En is asserted

                Q <= D;

            end if;

        end if;

        end if;

    end process;

end Behavioral;

**Register Bank**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Reg\_Bank is

    Port ( I : in STD\_LOGIC\_VECTOR (3 downto 0);

           Clk : in STD\_LOGIC;

           Rst : in STD\_LOGIC;

           Reg\_En : in STD\_LOGIC\_VECTOR (2 downto 0);

           R0 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R1 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R2 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R3 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R4 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R5 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R6 : out STD\_LOGIC\_VECTOR (3 downto 0);

           R7 : 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);

            Clk : in STD\_LOGIC;

            Rst : in STD\_LOGIC;

            En : 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\_Y : STD\_LOGIC\_VECTOR(7 downto 0);

begin

    Decoder\_3\_to\_8\_0 : Decoder\_3\_to\_8

        PORT MAP(

            I => Reg\_En,

            En => '1',

            Y => Decoder\_Y

        );

    Reg\_0 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(0),

            Q => R0

        );

    Reg\_1 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(1),

            Q => R1

        );

    Reg\_2 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(2),

            Q => R2

        );

    Reg\_3 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(3),

            Q => R3

        );

    Reg\_4 : Reg

        PORT MAP (

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(4),

            Q => R4

        );

    Reg\_5 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(5),

            Q => R5

        );

    Reg\_6 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(6),

            Q => R6

        );

    Reg\_7 : Reg

        PORT MAP(

            D => I,

            Clk => Clk,

            Rst => Rst,

            En => Decoder\_Y(7),

            Q => R7

        );

end Behavioral;

**4 Bit Adder and Subtractor**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Add\_Subtract is

    Port ( A\_in : in STD\_LOGIC\_VECTOR (3 downto 0);

           B\_in : in STD\_LOGIC\_VECTOR (3 downto 0);

           S\_out : out STD\_LOGIC\_VECTOR (3 downto 0);

           M : in STD\_LOGIC;

           C\_out : out STD\_LOGIC;

           V : out STD\_LOGIC);

end Add\_Subtract;

architecture Behavioral of Add\_Subtract 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\_out, B\_in\_temp : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    B\_in\_temp(0) <= B\_in(0) XOR M;

    B\_in\_temp(1) <= B\_in(1) XOR M;

    B\_in\_temp(2) <= B\_in(2) XOR M;

    B\_in\_temp(3) <= B\_in(3) XOR M;

    FA\_0: FA

        PORT MAP(

            A => A\_in(0),

            B => B\_in\_temp(0),

            C\_in => M,

            S => S\_out(0),

            C\_out => FA\_C\_out(0));

    FA\_1: FA

        PORT MAP(

            A => A\_in(1),

            B => B\_in\_temp(1),

            C\_in => FA\_C\_out(0),

            S => S\_out(1),

            C\_out => FA\_C\_out(1));

    FA\_2: FA

        PORT MAP(

            A => A\_in(2),

            B => B\_in\_temp(2),

            C\_in => FA\_C\_out(1),

            S => S\_out(2),

            C\_out => FA\_C\_out(2));

    FA\_3: FA

        PORT MAP(

            A => A\_in(3),

            B => B\_in\_temp(3),

            C\_in => FA\_C\_out(2),

            S => S\_out(3),

            C\_out => FA\_C\_out(3));

    C\_out <= FA\_C\_out(3);

    V <= FA\_C\_out(3) XOR FA\_C\_out(2);

end Behavioral;

**Mux\_8\_Way\_4**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_8\_way\_4 is

    Port ( Reg\_Sel : in STD\_LOGIC\_VECTOR (2 downto 0);

           A0, A1, A2, A3, A4, A5, A6, A7 : in STD\_LOGIC\_VECTOR(3 downto 0);

           Y : out STD\_LOGIC\_VECTOR (3 downto 0));

end Mux\_8\_way\_4;

architecture Behavioral of Mux\_8\_way\_4 is

    Component Decoder\_3\_to\_8

        PORT (

            I : in STD\_LOGIC\_VECTOR(2 downto 0);

            En : in STD\_LOGIC;

            Y : out STD\_LOGIC\_VECTOR(7 downto 0)

        );

    End Component;

    Component Buff\_4

        PORT (

            D : in STD\_LOGIC\_VECTOR(3 downto 0);

            Ctrl : in STD\_LOGIC;

            Y : out STD\_LOGIC\_VECTOR(3 downto 0)

        );

    End Component;

    SIGNAL Decoder\_Y : STD\_LOGIC\_VECTOR(7 downto 0);

begin

    Decoder\_3\_to\_8\_0 : Decoder\_3\_to\_8

        PORT MAP(

            I => Reg\_Sel,

            En => '1',

            Y => Decoder\_Y

        );

    -- use the yti state buffers to pass selected 4-bits

    Buff\_4\_0 : Buff\_4 PORT MAP(D => A0, Ctrl => Decoder\_Y(0), Y => Y);

    Buff\_4\_1 : Buff\_4 PORT MAP(D => A1, Ctrl => Decoder\_Y(1), Y => Y);

    Buff\_4\_2 : Buff\_4 PORT MAP(D => A2, Ctrl => Decoder\_Y(2), Y => Y);

    Buff\_4\_3 : Buff\_4 PORT MAP(D => A3, Ctrl => Decoder\_Y(3), Y => Y);

    Buff\_4\_4 : Buff\_4 PORT MAP(D => A4, Ctrl => Decoder\_Y(4), Y => Y);

    Buff\_4\_5 : Buff\_4 PORT MAP(D => A5, Ctrl => Decoder\_Y(5), Y => Y);

    Buff\_4\_6 : Buff\_4 PORT MAP(D => A6, Ctrl => Decoder\_Y(6), Y => Y);

    Buff\_4\_7 : Buff\_4 PORT MAP(D => A7, Ctrl => Decoder\_Y(7), Y => Y);

end Behavioral;

**D Flip FLop**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity D\_FF is

    Port ( Clk : in STD\_LOGIC;

           D : in STD\_LOGIC;

           Rst : in STD\_LOGIC;

           En : in STD\_LOGIC;

           Q : out STD\_LOGIC);

end D\_FF;

architecture Behavioral of D\_FF is

begin

    process (Clk, Rst) begin -- procss begin when clock changes

        if Rst'event AND Rst = '1' then Q <= '0'; -- reset the D flip-flop asynchronously

        else if (rising\_edge(Clk)) then  -- respond at clock rising edge

            if En = '1' then -- store D if En is asserted

                Q <= D;

             end if;

         end if;

         end if;

    end process;

end Behavioral;

**Slow Clock**

IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Slow\_Clk is

    Port ( Clk\_in : in STD\_LOGIC;

           Clk\_out : out STD\_LOGIC);

end Slow\_Clk;

architecture Behavioral of Slow\_Clk is

    SIGNAL Count : integer := 1;

    SIGNAL Clk\_status : STD\_LOGIC := '0';

begin

    -- in this we slow down the clock speed for get visible output

    process (Clk\_in) begin

        IF (rising\_edge(Clk\_in)) THEN

            Count <= count + 1;

            IF (Count = 2) THEN -- for this we are considering only 2 otherwise count = 5000000

                Clk\_status <= NOT Clk\_status;

                Clk\_out <= Clk\_status;

                Count <= 1;

            END IF;

        END IF;

    end process;

end Behavioral;

**3 Bit Counter**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Count\_3 is

    Port ( Clk : in STD\_LOGIC;

           rst : in STD\_LOGIC;

           Add\_in : in STD\_LOGIC\_VECTOR (2 downto 0);

           Add\_out : out STD\_LOGIC\_VECTOR (2 downto 0));

end Count\_3;

architecture Behavioral of Count\_3 is

Component Slow\_Clk

   PORT(

        Clk\_in : in STD\_LOGIC;

        Clk\_out : out STD\_LOGIC);

End Component;

Component D\_FF

    PORT(

        Clk : in STD\_LOGIC;

        D : in STD\_LOGIC;

        Rst : in STD\_LOGIC;

        En : in STD\_LOGIC;

        Q : out STD\_LOGIC);

End Component;

    SIGNAL Clk\_Slow: STD\_LOGIC;

begin

    -- this is for slow down the clock speed

    Slow\_Clk\_0 : Slow\_Clk

        PORT MAP(

            Clk\_in => Clk,

            Clk\_out => Clk\_Slow);

    -- left bit store

    D\_FF\_0 : D\_FF

        PORT MAP(

            D => Add\_in(0),

            Rst => rst,

            En => '1',

            Clk => Clk, -- Clk\_Slow

            Q => Add\_out(0));

    -- middle bit store

    D\_FF\_1 : D\_FF

        PORT MAP(

            D => Add\_in(1),

            Rst => rst,

            En => '1',

            Clk => Clk, -- Clk\_Slow

            Q => Add\_out(1));

    -- right bit store

    D\_FF\_2 : D\_FF

        PORT MAP(

            D => Add\_in(2),

            Rst => rst,

            En => '1',

            Clk => Clk, -- Clk\_Slow

            Q => Add\_out(2));

end Behavioral;

**ROM**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity ROM\_8 is

    Port ( address : in STD\_LOGIC\_VECTOR (2 downto 0);

           ins : out STD\_LOGIC\_VECTOR (11 downto 0));

end ROM\_8;

architecture Behavioral of ROM\_8 is

    type rom\_type is array (0 to 7) of std\_logic\_vector(11 downto 0);

    signal mem\_8 : rom\_type := ( "101110001010",

                                 "100010000001",

                                 "001110010000",

                                 "000000000000",

                                 "000000000000",

                                 "000000000000",

                                 "000000000000",

                                 "000000000000"

    );

begin

    ins <= mem\_8(to\_integer(unsigned(address)));

end Behavioral;

**Look up table for seven segment display**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use ieee.numeric\_std.all;

entity LUT\_7\_Display is

    Port ( address : in STD\_LOGIC\_VECTOR (3 downto 0);

           data : out STD\_LOGIC\_VECTOR (6 downto 0));

end LUT\_7\_Display;

architecture Behavioral of LUT\_7\_Display 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;

Simulations VHDL Codes

**FA Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FA\_Sim is

--  Port ( );

end FA\_Sim;

architecture Behavioral of FA\_Sim 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 A, B, C\_in : STD\_LOGIC; -- input signals

    SIGNAL S, C\_out : STD\_LOGIC; -- output signals

begin

    UUT : FA PORT MAP (

        A => A,

        B => B,

        C\_in => C\_in,

        S => S,

        C\_out => C\_out

    );

    process

    begin

        -- set the A and B values

        A <= '0';

        B <= '0';

        C\_in <= '0';

        wait for 10ns;

        A <= '1';

        wait for 10ns;

        B <= '1';

        wait for 10ns;

        A <= '0';

        WAIT FOR 10ns;

        C\_in <= '1';

        wait for 10ns;

        B <= '0';

        wait for 10ns;

        A <= '1';

        wait for 10ns;

        B <= '1';

        wait for 10ns;

        wait; -- wait forever

     end process;

end Behavioral;

**Adder and Subtractor Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Add\_Subtract\_Sim is

--  Port ( );

end Add\_Subtract\_Sim;

architecture Behavioral of Add\_Subtract\_Sim is

Component Add\_Subtract

    PORT(

        A\_in : in STD\_LOGIC\_VECTOR (3 downto 0);

        B\_in : in STD\_LOGIC\_VECTOR (3 downto 0);

        S\_out : out STD\_LOGIC\_VECTOR (3 downto 0);

        M : in STD\_LOGIC;

        C\_out : out STD\_LOGIC;

        V : out STD\_LOGIC

    );

End Component;

    SIGNAL A\_in, B\_in, S\_out: STD\_LOGIC\_VECTOR(3 downto 0);

    SIGNAL M, v, C\_out: STD\_LOGIC;

begin

    UUT : Add\_Subtract

        PORT MAP(

            A\_in => A\_in,

            B\_in => B\_in,

            M => M,

            S\_out => S\_out,

            V => V,

            C\_out => C\_out

        );

    process

    begin

        M <= '0';

        A\_in <= "0000";

        B\_in <= "0000";

        wait for 10ns;

        A\_in <= "0001";

        B\_in <= "0001";

        wait for 10ns;

        A\_in <= "0010";

        B\_in <= "0010";

        wait for 10ns;

        A\_in <= "0100";

        B\_in <= "0100";

        wait for 10ns;

        A\_in <= "0100";

        B\_in <= "0100";

        wait for 10ns;

        A\_in <= "0110";

        B\_in <= "1100";

        wait for 10ns;

        A\_in <= "0111";

        B\_in <= "1000";

        wait for 10ns;

    end process;

end Behavioral;

**3 Bit Adder**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Adder\_3\_Sim is

--  Port ( );

end Adder\_3\_Sim;

architecture Behavioral of Adder\_3\_Sim is

Component Adder\_3

    Port(

        A\_in : in STD\_LOGIC\_VECTOR (2 downto 0);

        S\_out : out STD\_LOGIC\_VECTOR (2 downto 0));

End Component;

    SIGNAL A\_in, S\_out : STD\_LOGIC\_VECTOR(2 downto 0);

begin

    UUT : Adder\_3

        PORT MAP(

            A\_in => A\_in,

            S\_out => S\_out);

    process

    begin

        A\_in <= "000";

        wait for 10ns;

        A\_in <= "001";

        wait for 10ns;

        A\_in <= "010";

        wait for 10ns;

        A\_in <= "011";

        wait for 10ns;

        A\_in <= "100";

        wait for 10ns;

        A\_in <= "101";

        wait for 10ns;

        A\_in <= "110";

        wait for 10ns;

        A\_in <= "111";

        wait;

    end process;

end Behavioral;

**D Flip Flop Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity D\_FF\_Sim is

--  Port ( );

end D\_FF\_Sim;

architecture Behavioral of D\_FF\_Sim is

    Component D\_FF

        PORT (

            Clk : in STD\_LOGIC;

            En : in STD\_LOGIC;

            Rst : in STD\_LOGIC;

            D : in STD\_LOGIC;

            Q : out STD\_LOGIC

        );

    End Component;

    SIGNAL Clk, En, Rst, D : STD\_LOGIC;

    SIGNAL Q : STD\_LOGIC;

begin

    UUT : D\_FF PORT MAP (

        Clk => Clk,

        Rst => Rst,

        En => En,

        D => D,

        Q => Q

    );

    clock\_process : process -- background clock process

    begin

        Clk <= '0';

        wait for 5ns;

        Clk <= '1';

        wait for 5ns;

    end process;

    D\_FF\_process : process -- flip flop process

    begin

        En <= '0';

        D <= '0';

        wait for 10ns;

        D <= '1';

        wait for 10ns;

        En <= '1';

        wait for 10ns;

        D <= '0';

        wait for 10ns;

        D <= '1';

        wait for 10ns;

        -- reset the flip flop

        Rst <= '1';

        wait for 10ns;

        D <= '1';

        wait for 10ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Decoder 2 to 4 Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decoder\_2\_to\_4\_Sim is

--  Port ( );

end Decoder\_2\_to\_4\_Sim;

architecture Behavioral of Decoder\_2\_to\_4\_Sim 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 I : STD\_LOGIC\_VECTOR(1 downto 0);

    SIGNAL En : STD\_LOGIC;

    SIGNAL Y : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Decoder\_2\_to\_4

        PORT MAP(

            I => I,

            En => En,

            Y => Y

        );

    process

    begin

        En <= '0';

        I <= "00";

        wait for 5ns;

        En <= '1';

        wait for 5ns;

        I <= "01";

        wait for 5ns;

        I <= "10";

        wait for 5ns;

        I <= "11";

        wait for 5ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Decoder 3 to 8 Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Decoder\_3\_to\_8\_Sim is

--  Port ( );

end Decoder\_3\_to\_8\_Sim;

architecture Behavioral of Decoder\_3\_to\_8\_Sim 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 I : STD\_LOGIC\_VECTOR(2 downto 0);

    SIGNAL En : STD\_LOGIC;

    SIGNAL Y : STD\_LOGIC\_VECTOR(7 downto 0);

begin

    UUT : Decoder\_3\_to\_8

        PORT MAP(

            I => I,

            En => En,

            Y => Y

        );

    process

    begin

        En <= '0';

        I <= "000";

        wait for 5ns;

        En <= '1';

        wait for 5ns;

        I <= "001";

        wait for 5ns;

        I <= "010";

        wait for 5ns;

        I <= "011";

        wait for 5ns;

        I <= "100";

        wait for 5ns;

        I <= "111";

        wait for 5ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Instruction Decoder Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Ins\_Decoder\_Sim is

--  Port ( );

end Ins\_Decoder\_Sim;

architecture Behavioral of Ins\_Decoder\_Sim is

    Component Ins\_Decoder

        PORT (

             ins\_bus : in STD\_LOGIC\_VECTOR (11 downto 0);

             jmp\_check : in STD\_LOGIC\_VECTOR(3 downto 0);

             reg\_enb : out STD\_LOGIC\_VECTOR (2 downto 0);

             load\_sel : out STD\_LOGIC;

             im\_val : out STD\_LOGIC\_VECTOR(3 downto 0);

             reg\_sel\_1 : out STD\_LOGIC\_VECTOR (2 downto 0);

             reg\_sel\_2 : out STD\_LOGIC\_VECTOR (2 downto 0);

             add\_sub\_sel : out STD\_LOGIC;

             jmp : out STD\_LOGIC;

             jmp\_addr : out STD\_LOGIC\_VECTOR(2 downto 0)

        );

    End Component;

    SIGNAL ins\_bus : STD\_LOGIC\_VECTOR(11 downto 0);

    SIGNAL jmp\_check, im\_val : STD\_LOGIC\_VECTOR(3 downto 0);

    SIGNAL reg\_enb, reg\_sel\_1, reg\_sel\_2, jmp\_addr : STD\_LOGIC\_VECTOR(2 downto 0);

    SIGNAL load\_sel, add\_sub\_sel, jmp : STD\_LOGIC;

begin

    UUT : Ins\_Decoder PORT MAP (

        ins\_bus => ins\_bus,

        jmp\_check => jmp\_check,

        reg\_enb => reg\_enb,

        load\_sel => load\_sel,

        im\_val => im\_val,

        reg\_sel\_1 => reg\_sel\_1,

        reg\_sel\_2 => reg\_sel\_2,

        add\_sub\_sel => add\_sub\_sel,

        jmp => jmp,

        jmp\_addr => jmp\_addr

    );

    process

    begin

        -- test couple of 12 bit instructions on the instruction decoder

        -- MOVI R1, 10

        ins\_bus <= "100010001010";

        wait for 10ns;

        -- MOVI R2, 1

        ins\_bus <= "100100000001";

        wait for 10ns;

        -- NEG R2

        ins\_bus <= "010100000000";

        wait for 10ns;

        wait; --wait forever

    end process;

end Behavioral;

**Seven Segment Look up Table Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity LUT\_7\_Display\_Sim is

--  Port ( );

end LUT\_7\_Display\_Sim;

architecture Behavioral of LUT\_7\_Display\_Sim is

Component LUT\_7\_Display

    PORT(

        address : in STD\_LOGIC\_VECTOR (3 downto 0);

        data : out STD\_LOGIC\_VECTOR (6 downto 0));

End Component;

    SIGNAL d : STD\_LOGIC\_VECTOR(6 downto 0);

    SIGNAL a : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : LUT\_7\_Display

    PORT MAP(

        data => d,

        address => a);

    process

    begin

        a <= "0000";

        wait for 10ns;

        a <= "0001";

        wait for 10ns;

        a <= "0010";

        wait for 10ns;

        a <= "0011";

        wait for 10ns;

        a <= "0100";

        wait for 10ns;

        a <= "0101";

        wait for 10ns;

        a <= "0110";

        wait for 10ns;

        a <= "0111";

        wait for 10ns;

        a <= "1000";

        wait for 10ns;

        a <= "1001";

        wait for 10ns;

        a <= "1010";

        wait for 10ns;

        a <= "1011";

        wait for 10ns;

        a <= "1100";

        wait for 10ns;

        a <= "1101";

        wait for 10ns;

        a <= "1110";

        wait for 10ns;

        a <= "1111";

        wait;

    end process;

end Behavioral;

**Mux 2 Way 3 Bit Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_2\_way\_3\_Sim is

--  Port ( );

end Mux\_2\_way\_3\_Sim;

architecture Behavioral of Mux\_2\_way\_3\_Sim is

    Component Mux\_2\_way\_3

        PORT (

            I : in STD\_LOGIC;

            A0, A1 : in STD\_LOGIC\_VECTOR(2 downto 0);

            Y : out STD\_LOGIC\_VECTOR(2 downto 0)

        );

    End Component;

    SIGNAL I : STD\_LOGIC;

    SIGNAL A0, A1, Y : STD\_LOGIC\_VECTOR(2 downto 0);

begin

   UUT : Mux\_2\_way\_3

        PORT MAP(

            I => I,

            A0 => A0,

            A1 => A1,

            Y => Y

        );

    process

    begin

        -- set the initial data

        A0 <= "010";

        A1 <= "110";

        -- select the data path

        I <= '0';

        wait for 5ns;

        I <= '1';

        wait for 5ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Mux 2 Way 4 Bit Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_2\_way\_4\_Sim is

--  Port ( );

end Mux\_2\_way\_4\_Sim;

architecture Behavioral of Mux\_2\_way\_4\_Sim is

    Component Mux\_2\_way\_4

        PORT (

            I : in STD\_LOGIC;

            A0, A1 : in STD\_LOGIC\_VECTOR(3 downto 0);

            Y : out STD\_LOGIC\_VECTOR(3 downto 0)

        );

    End Component;

    SIGNAL I : STD\_LOGIC;

    SIGNAL A0, A1, Y : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Mux\_2\_way\_4

        PORT MAP(

            I => I,

            A0 => A0,

            A1 => A1,

            Y => Y

        );

    process

    begin

        -- set theh initial data

        A0 <= "1010";

        A1 <= "1110";

        I <= '0';

        wait for 5ns;

        I <= '1';

        wait for 5ns;

        wait; --wait forever

    end process;

end Behavioral;

**Mux 8 way 4 Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_8\_way\_4\_Sim is

--  Port ( );

end Mux\_8\_way\_4\_Sim;

architecture Behavioral of Mux\_8\_way\_4\_Sim is

    Component Mux\_8\_way\_4

        PORT (

            Reg\_Sel : in STD\_LOGIC\_VECTOR (2 downto 0);

            A0, A1, A2, A3, A4, A5, A6, A7 : in STD\_LOGIC\_VECTOR(3 downto 0);

            Y : out STD\_LOGIC\_VECTOR (3 downto 0)

        );

    End Component;

    SIGNAL Reg\_Sel : STD\_LOGIC\_VECTOR(2 downto 0);

    SIGNAL A0, A1, A2, A3, A4, A5, A6, A7 : STD\_LOGIC\_VECTOR(3 downto 0);

    SIGNAL Y : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Mux\_8\_way\_4 PORT MAP(

        Reg\_Sel => Reg\_Sel,

        Y => Y,

        A0 => A0,

        A1 => A1,

        A2 => A2,

        A3 => A3,

        A4 => A4,

        A5 => A5,

        A6 => A6,

        A7 => A7

    );

    process

    begin

        -- set the initial data of A0 - A7

        A0 <= "0110";

        A1 <= "1101";

        A2 <= "1111";

        A3 <= "0001";

        A4 <= "1000";

        A5 <= "1011";

        A6 <= "0000";

        A7 <= "1110";

        -- set the register selecter values

        Reg\_Sel <= "000";

        wait for 5ns;

        Reg\_Sel <= "001";

        wait for 5ns;

        Reg\_Sel <= "010";

        wait for 5ns;

        Reg\_Sel <= "011";

        wait for 5ns;

        Reg\_Sel <= "100";

        wait for 5ns;

        Reg\_Sel <= "101";

        wait for 5ns;

        Reg\_Sel <= "110";

        wait for 5ns;

        Reg\_Sel <= "111";

        wait for 5ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Register Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Reg\_Sim is

--  Port ( );

end Reg\_Sim;

architecture Behavioral of Reg\_Sim is

    Component Reg

        PORT (

            Clk : in STD\_LOGIC;

            En : in STD\_LOGIC;

            Rst : in STD\_LOGIC;

            D : in STD\_LOGIC\_VECTOR(3 downto 0);

            Q : out STD\_LOGIC\_VECTOR(3 downto 0)

        );

    End Component;

    SIGNAL Clk, En, Rst : STD\_LOGIC;

    SIGNAL D : STD\_LOGIC\_VECTOR(3 downto 0);

    SIGNAL Q : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Reg PORT MAP (

        Clk => Clk,

        Rst => Rst,

        En => En,

        D => D,

        Q => Q

    );

    clock\_process : process -- background clock process

    begin

        Clk <= '0';

        wait for 5ns;

        Clk <= '1';

        wait for 5ns;

    end process;

    reg\_process : process

    begin

        En <= '0';

        D <= "0110";

        wait for 10ns;

        D <= "1010";

        wait for 10ns;

        En <= '1';

        wait for 10ns;

        D <= "0110";

        wait for 10ns;

        -- reset the register

        Rst <= '1';

        wait for 10ns;

        D <= "1110";

        wait for 10ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Register Bank Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Reg\_Bank\_Sim is

--  Port ( );

end Reg\_Bank\_Sim;

architecture Behavioral of Reg\_Bank\_Sim is

    Component Reg\_Bank

        PORT (

            I : in STD\_LOGIC\_VECTOR (3 downto 0);

            Clk : in STD\_LOGIC;

            Rst : in STD\_LOGIC;

            Reg\_En : in STD\_LOGIC\_VECTOR (2 downto 0);

            R0 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R1 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R2 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R3 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R4 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R5 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R6 : out STD\_LOGIC\_VECTOR (3 downto 0);

            R7 : out STD\_LOGIC\_VECTOR (3 downto 0)

        );

    End Component;

    SIGNAL I : STD\_LOGIC\_VECTOR(3 downto 0);

    SIGNAL Clk, Rst : STD\_LOGIC;

    SIGNAL Reg\_En : STD\_LOGIC\_VECTOR(2 downto 0);

    SIGNAL R0, R1, R2, R3, R4, R5, R6, R7 : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Reg\_Bank

        PORT MAP(

            I => I,

            Clk => Clk,

            Rst => Rst,

            Reg\_En => Reg\_En,

            R0 => R0,

            R1 => R1,

            R2 => R2,

            R3 => R3,

            R4 => R4,

            R5 => R5,

            R6 => R6,

            R7 => R7

        );

    clock\_process : process -- background clock process

    begin

        Clk <= '0';

        wait for 5ns;

        Clk <= '1';

        wait for 5ns;

    end process;

    reg\_process : process -- register bank process

    begin

        I <= "1000";

        Reg\_En <= "001";

        wait for 10ns;

        I <= "0100";

        Reg\_En <= "010";

        wait for 10ns;

        -- reset the regsiter bank

        Rst <= '1';

        wait for 10ns;

        I <= "0110";

        Reg\_En <= "011";

        wait for 10ns;

        wait; -- wait forever

    end process;

end Behavioral;

**Slow Clock Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Slow\_Clk\_Sim is

--  Port ( );

end Slow\_Clk\_Sim;

architecture Behavioral of Slow\_Clk\_Sim is

Component Slow\_Clk is

    PORT (

        Clk\_in : in STD\_LOGIC;

        Clk\_out : out STD\_LOGIC);

End Component;

    SIGNAL Clk\_in :  STD\_LOGIC := '0';

    SIGNAL Clk\_out :  STD\_LOGIC;

begin

    UUT : Slow\_Clk

        PORT MAP(

        Clk\_in => Clk\_in,

        Clk\_out => Clk\_out);

    clock\_process: process

    begin

        Clk\_in <= '0';

        wait for 5 ns;

        Clk\_in <= '1';

        wait for 5 ns;

    end process;

end Behavioral;

**Processor Simulation**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Processor\_Sim is

--  Port ( );

end Processor\_Sim;

architecture Behavioral of Processor\_Sim is

Component Processor

    PORT(

        Rst : in STD\_LOGIC;

        Clk : in STD\_LOGIC;

        Carry : out STD\_LOGIC;

        Zeroes : out STD\_LOGIC;

        R7\_out : out STD\_LOGIC\_VECTOR (3 downto 0));

End Component;

    SIGNAL Rst, Clk, Carry, Zeroes : STD\_LOGIC;

    SIGNAL R7\_out : STD\_LOGIC\_VECTOR(3 downto 0);

begin

    UUT : Processor

        PORT MAP(

            Rst => Rst,

            Clk => Clk,

            Carry => Carry,

            Zeroes => Zeroes,

            R7\_out => R7\_out);

    clk\_process : process

    begin

            Clk <= '0';

            wait for 10ns;

            Clk <= '1';

            wait for 10ns;

    end process;

    processor\_process : process

    begin

        Rst <= '1';

        wait;

    end process;

end Behavioral;

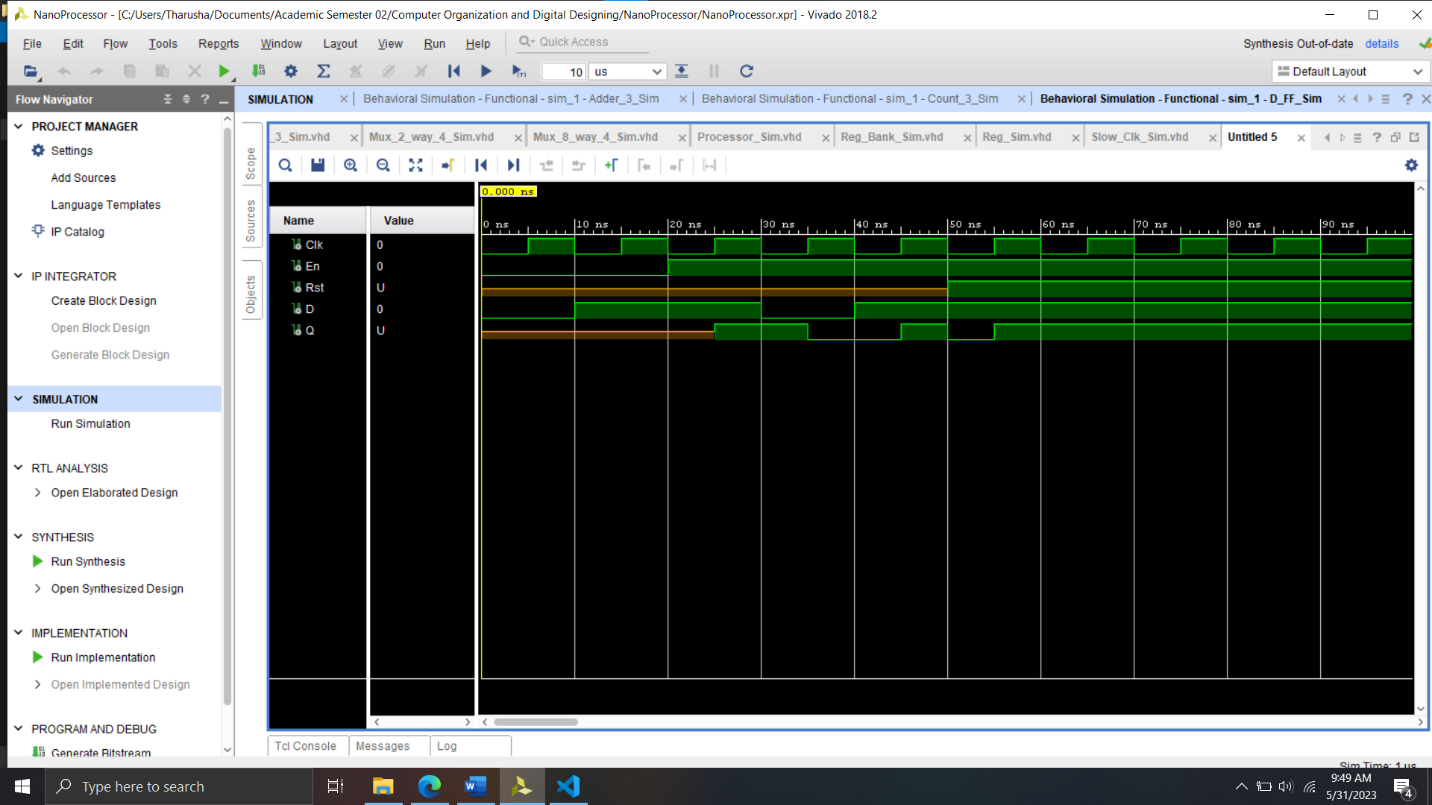
Timing Diagrams

***Sim FA***

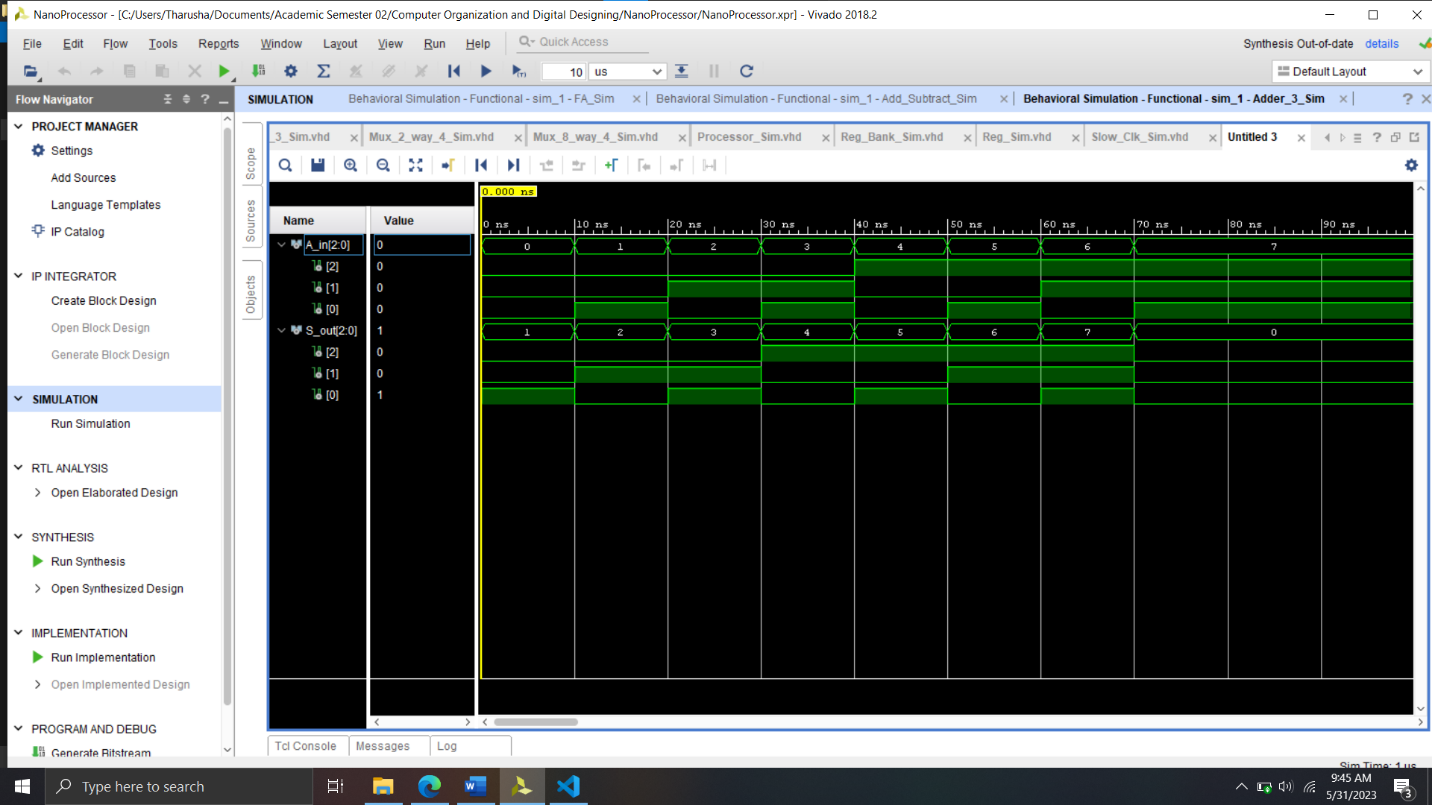
***A screenshot of a computer

Description automatically generated***

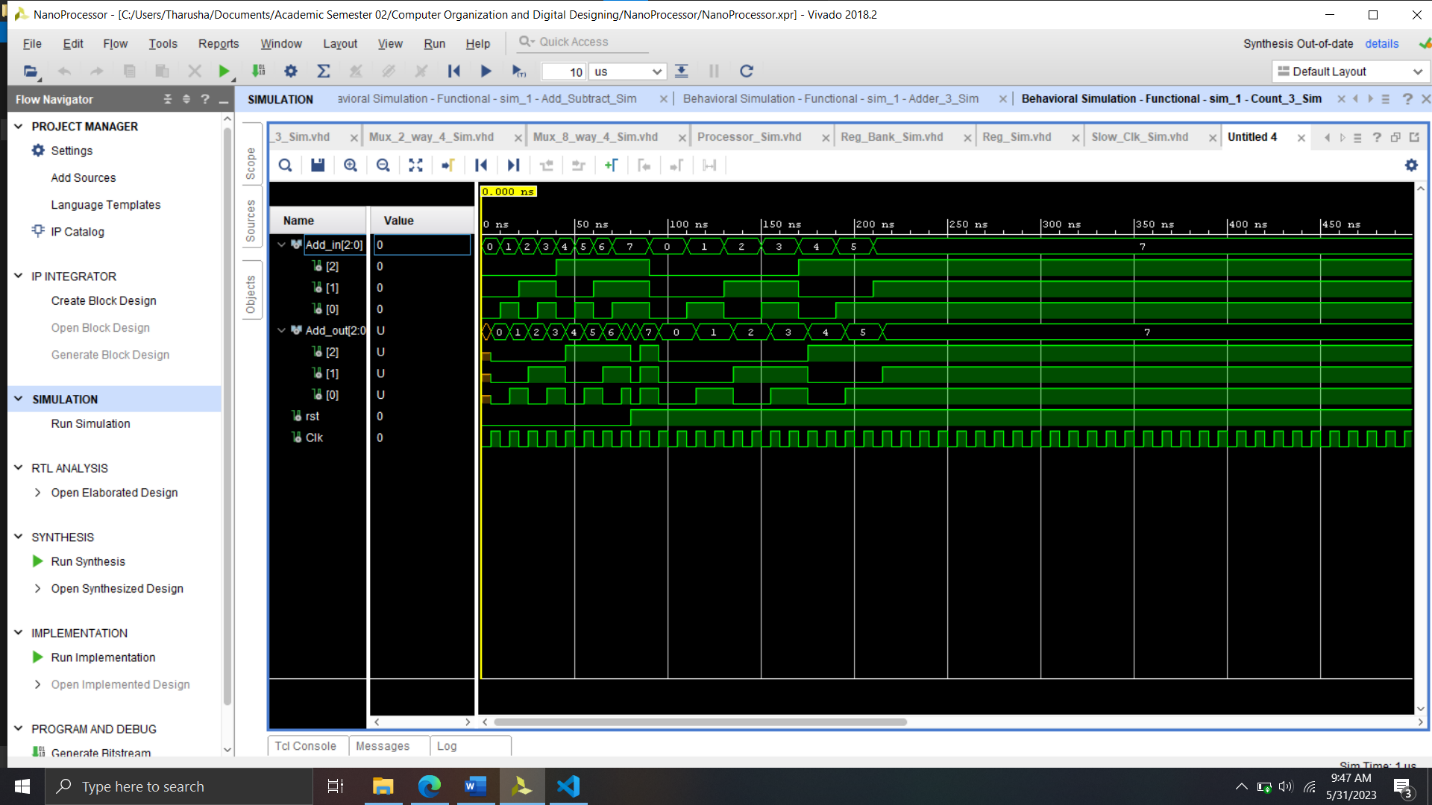
***Sim D\_FF***

******

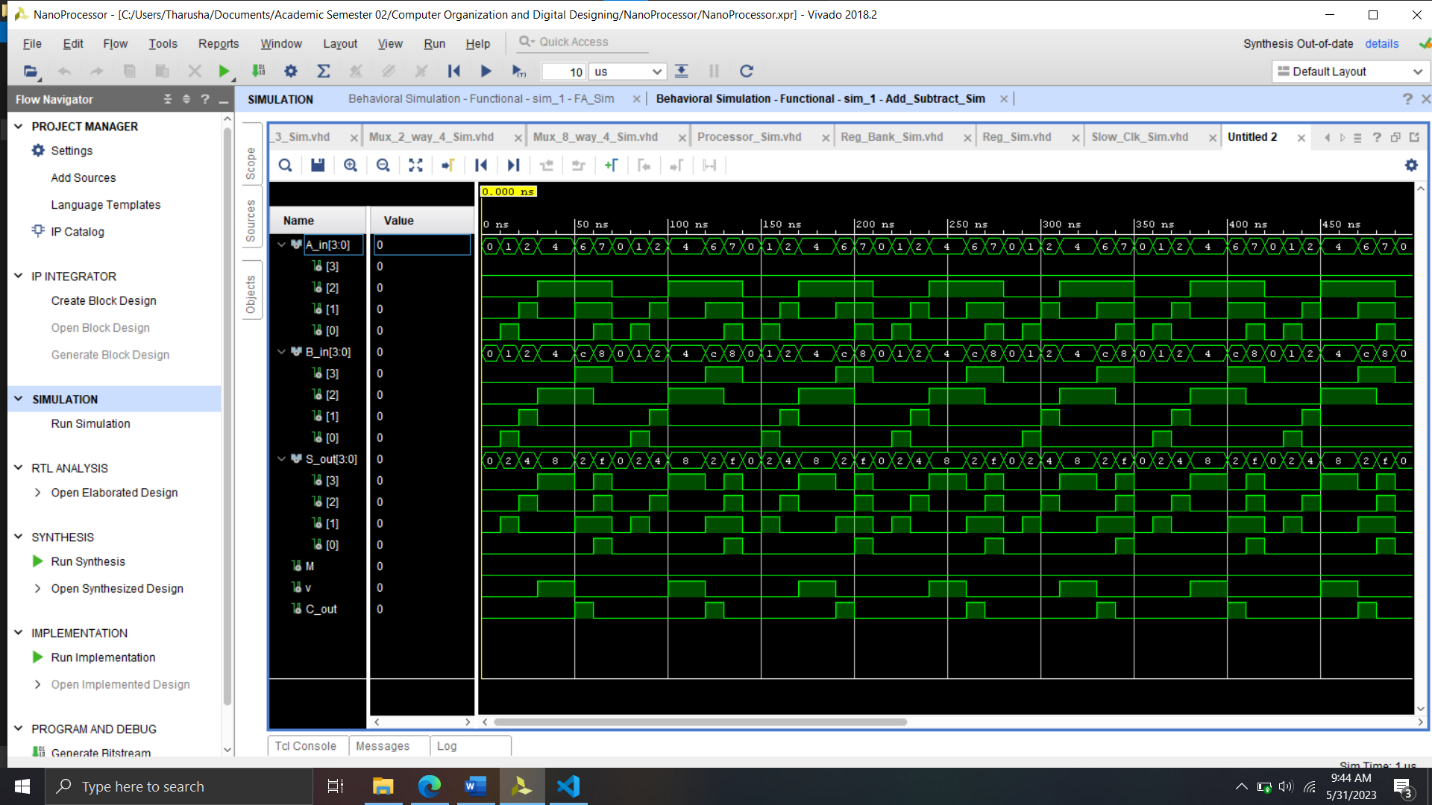
***Sim 3-bit Adder***

******

***Sim 3-bit Counter***

******

***Sim Adder and Subtractor***

******

***Sim Decoder 2 to 4***

***A screenshot of a computer

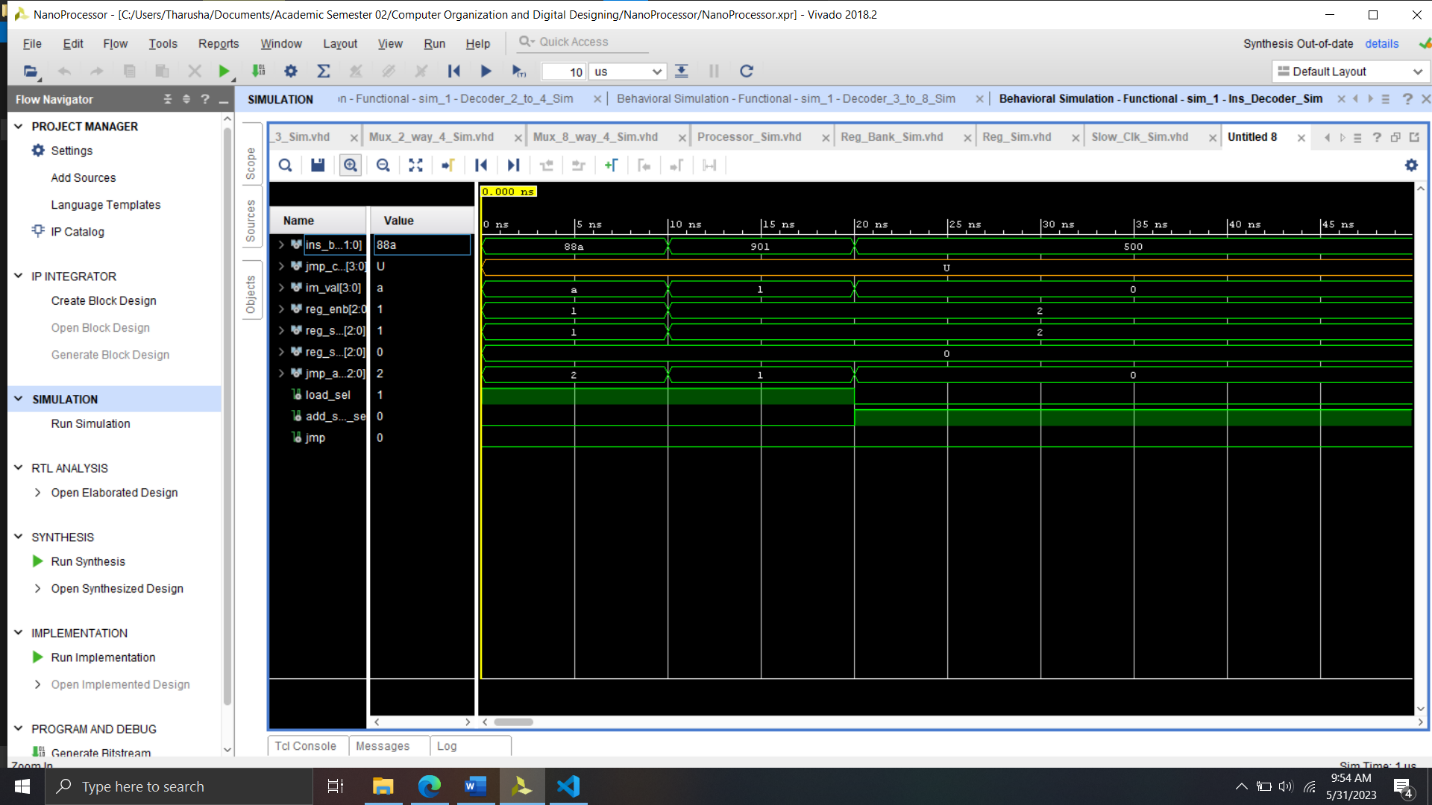
Description automatically generated***

***Sim Decoder 3 to 8***

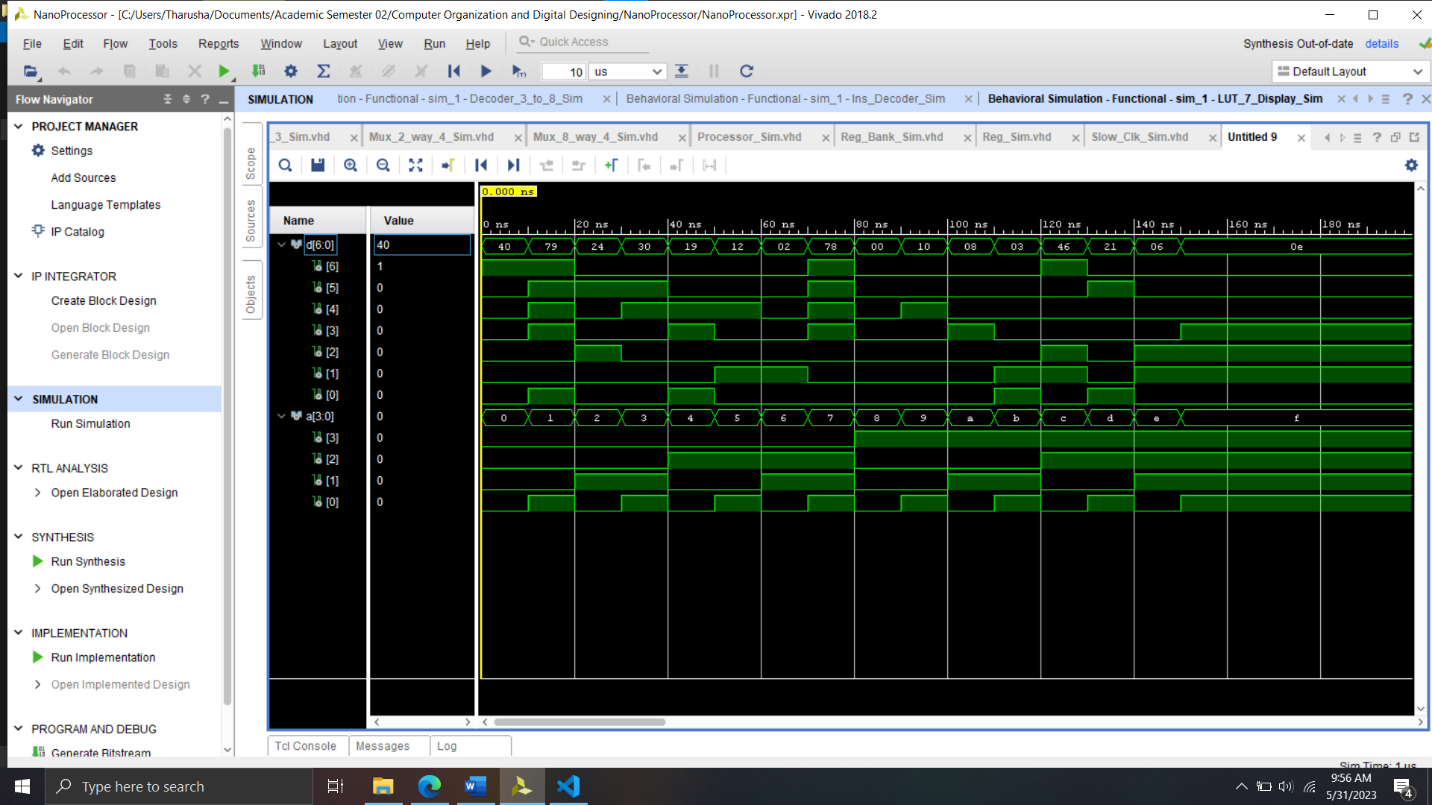
***A screenshot of a computer

Description automatically generated***

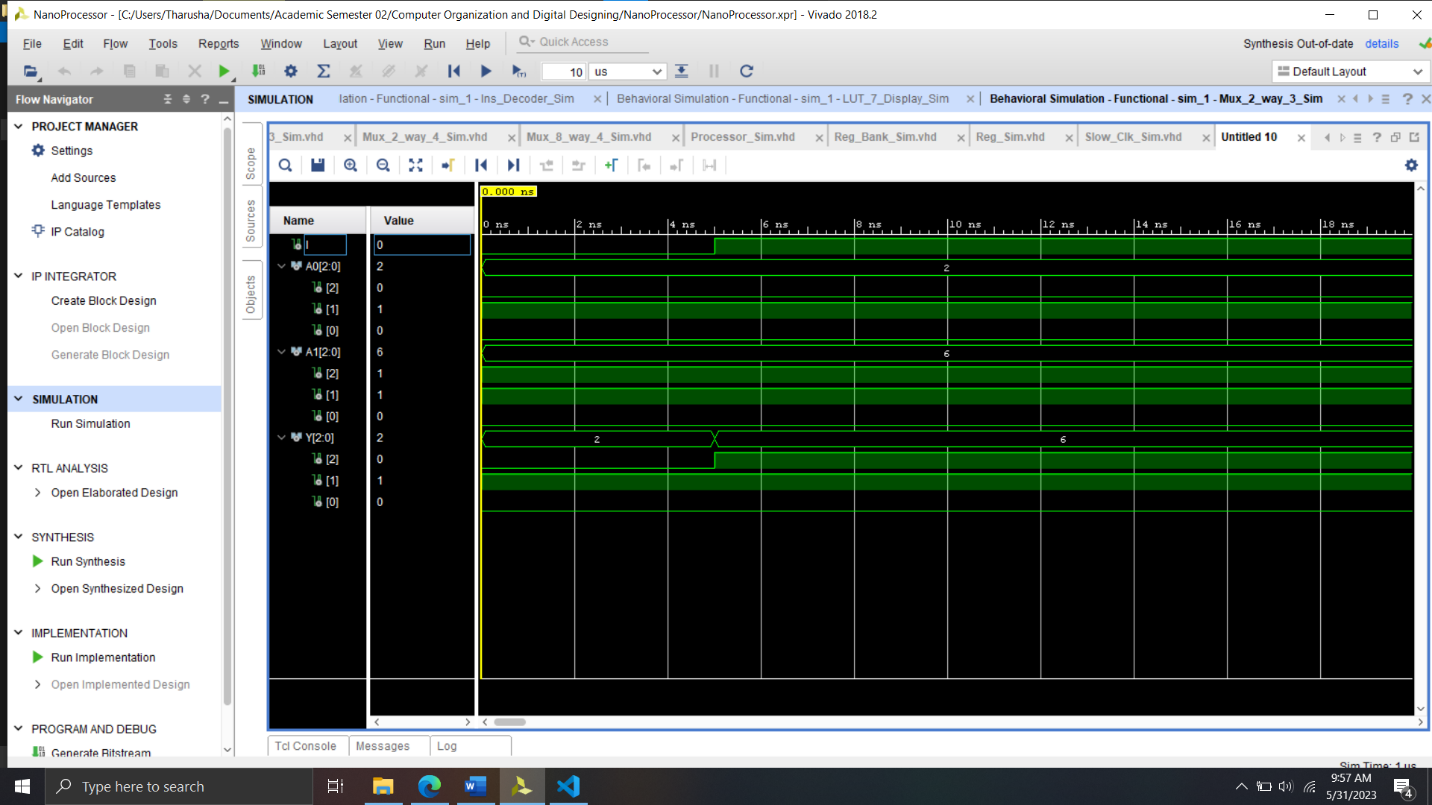
***Sim Instruction Decoder***

******

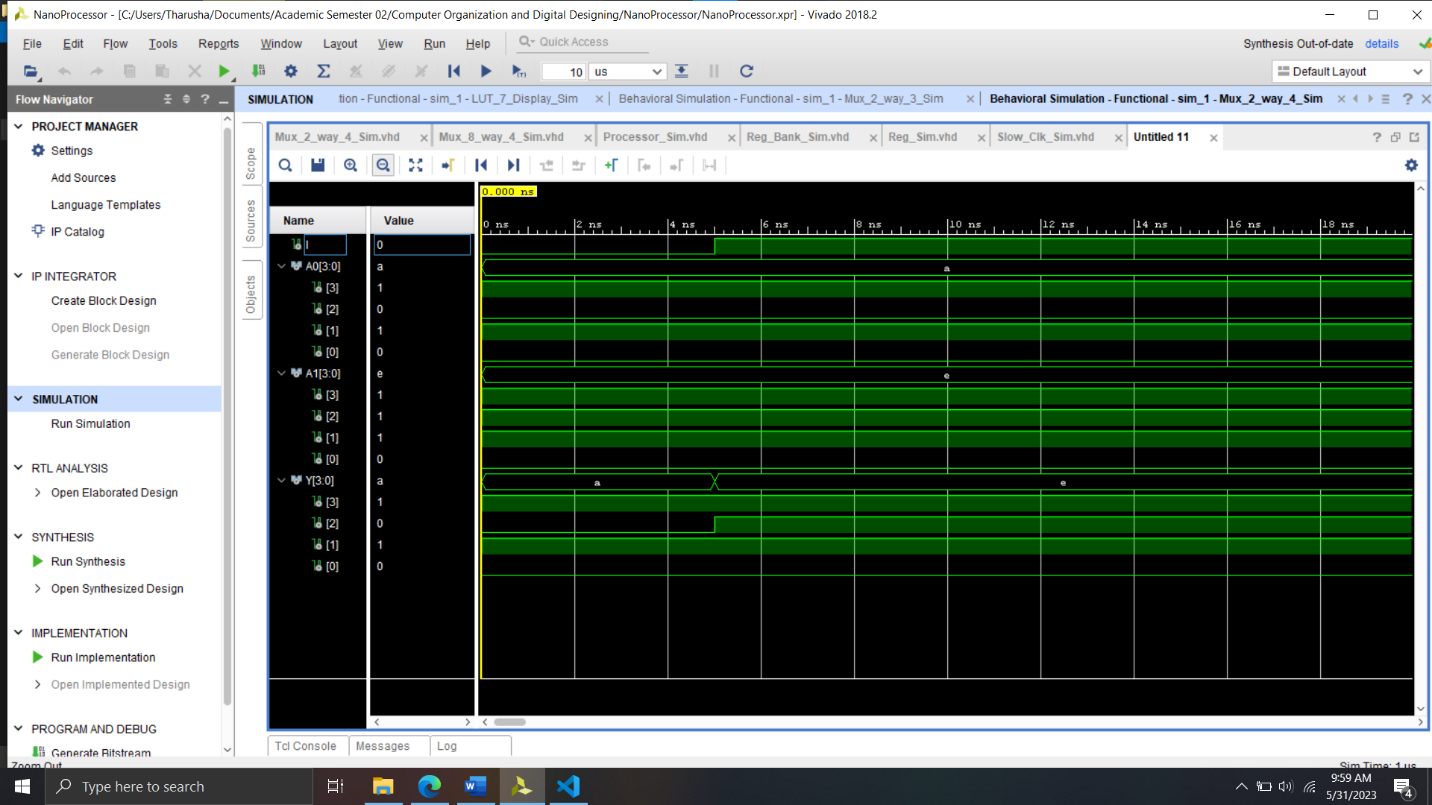
***Sim 7 Segment Lookup Table***

******

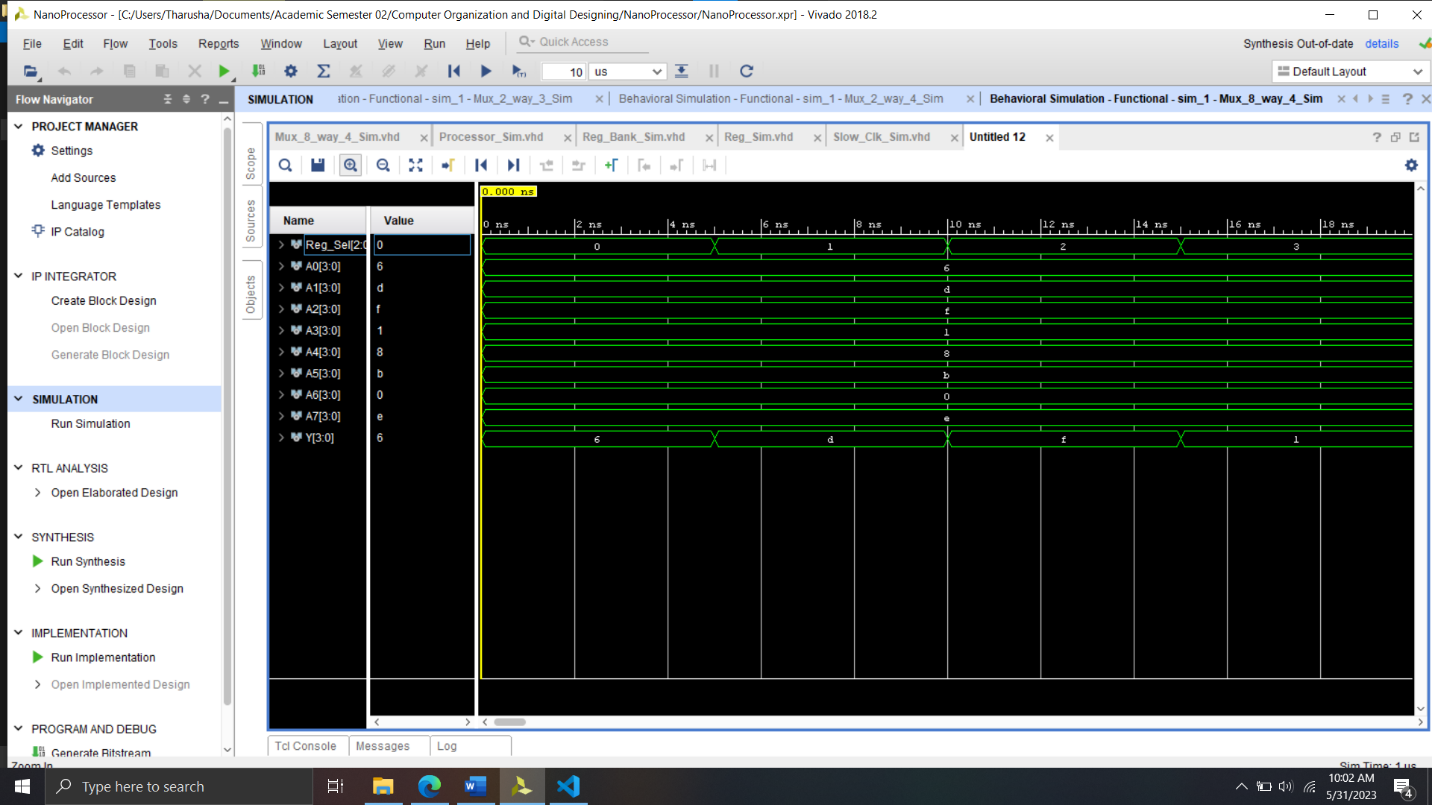
***Sim Mux 2 way 3***

******

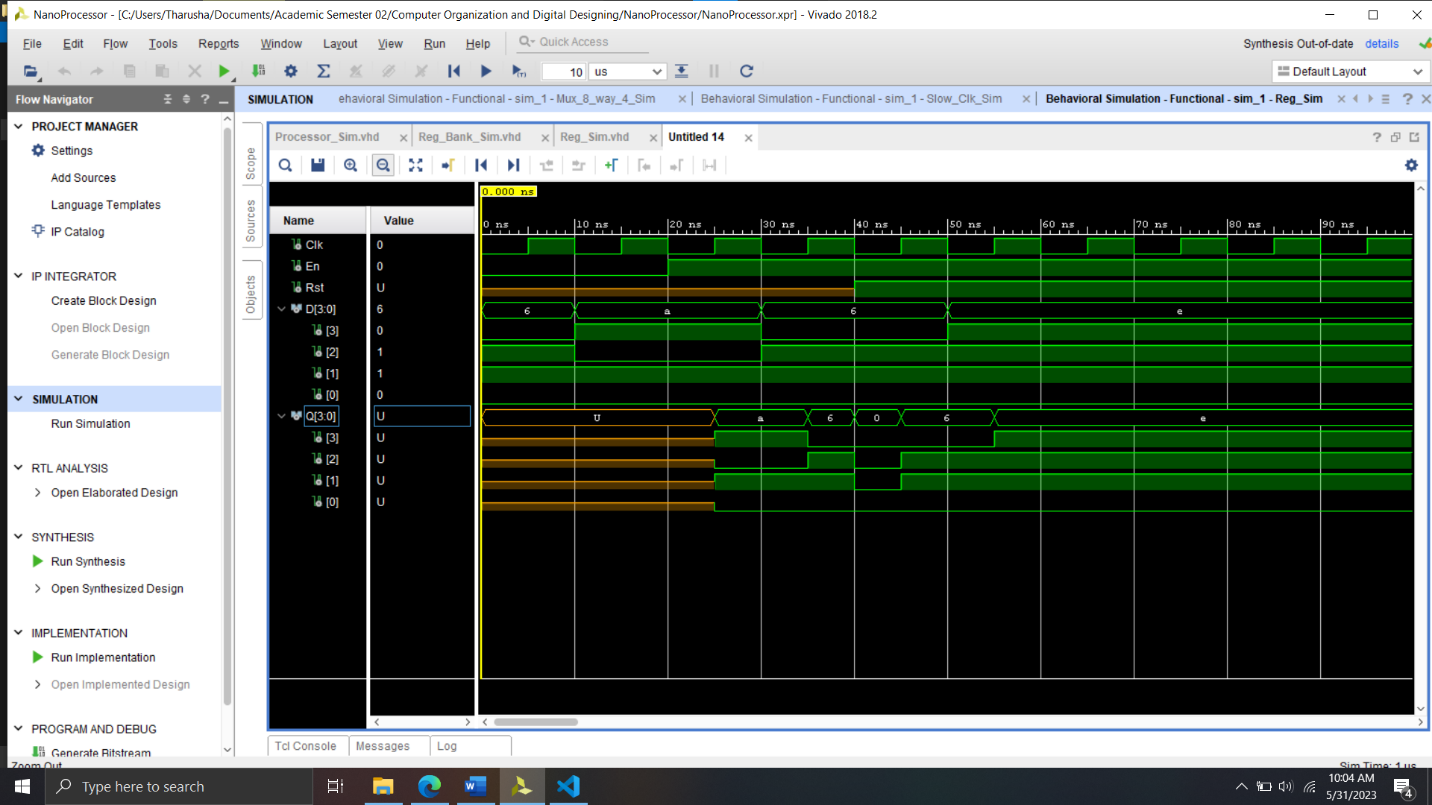
***Sim Mux 2 way 4***

******

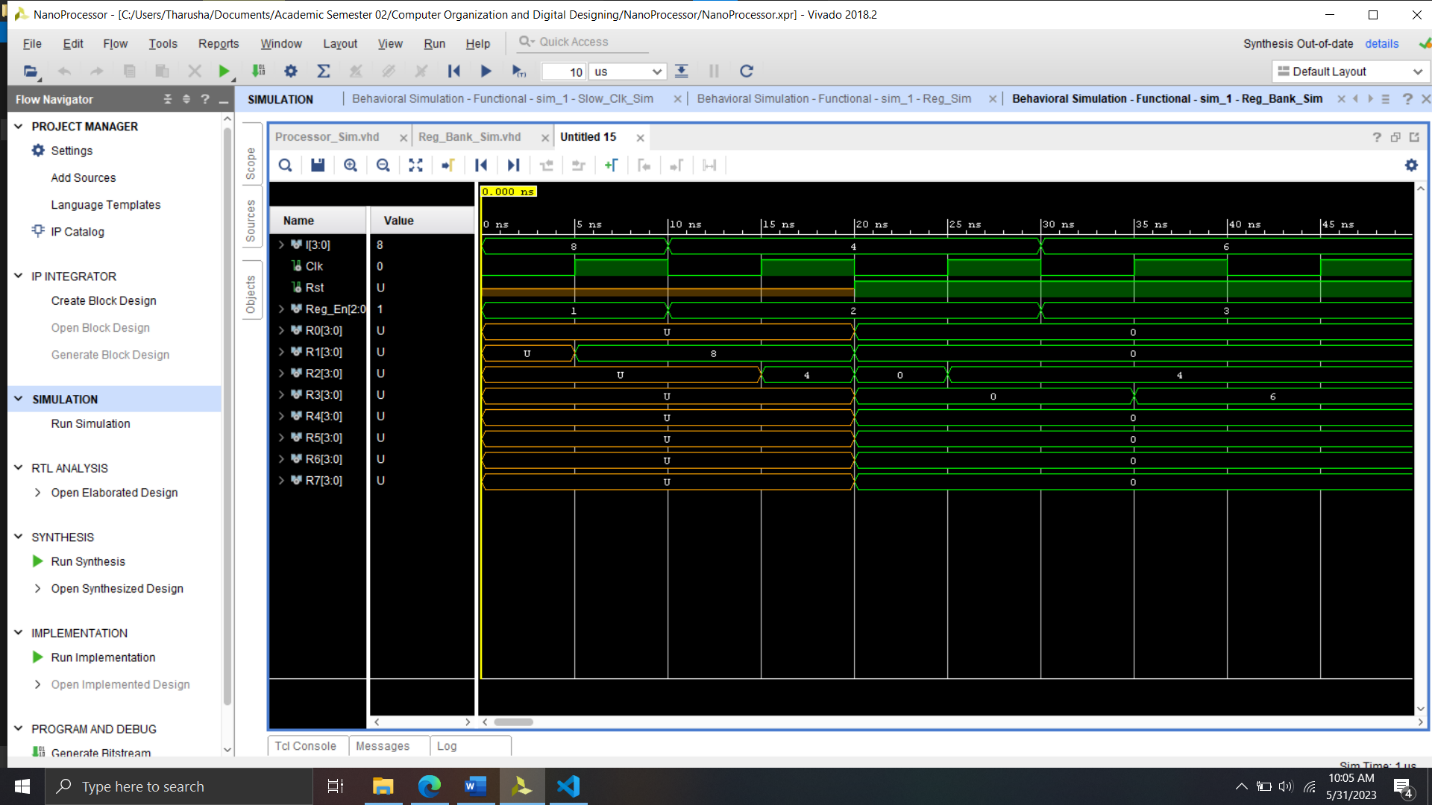
***Sim Mux 8 way 4***

******

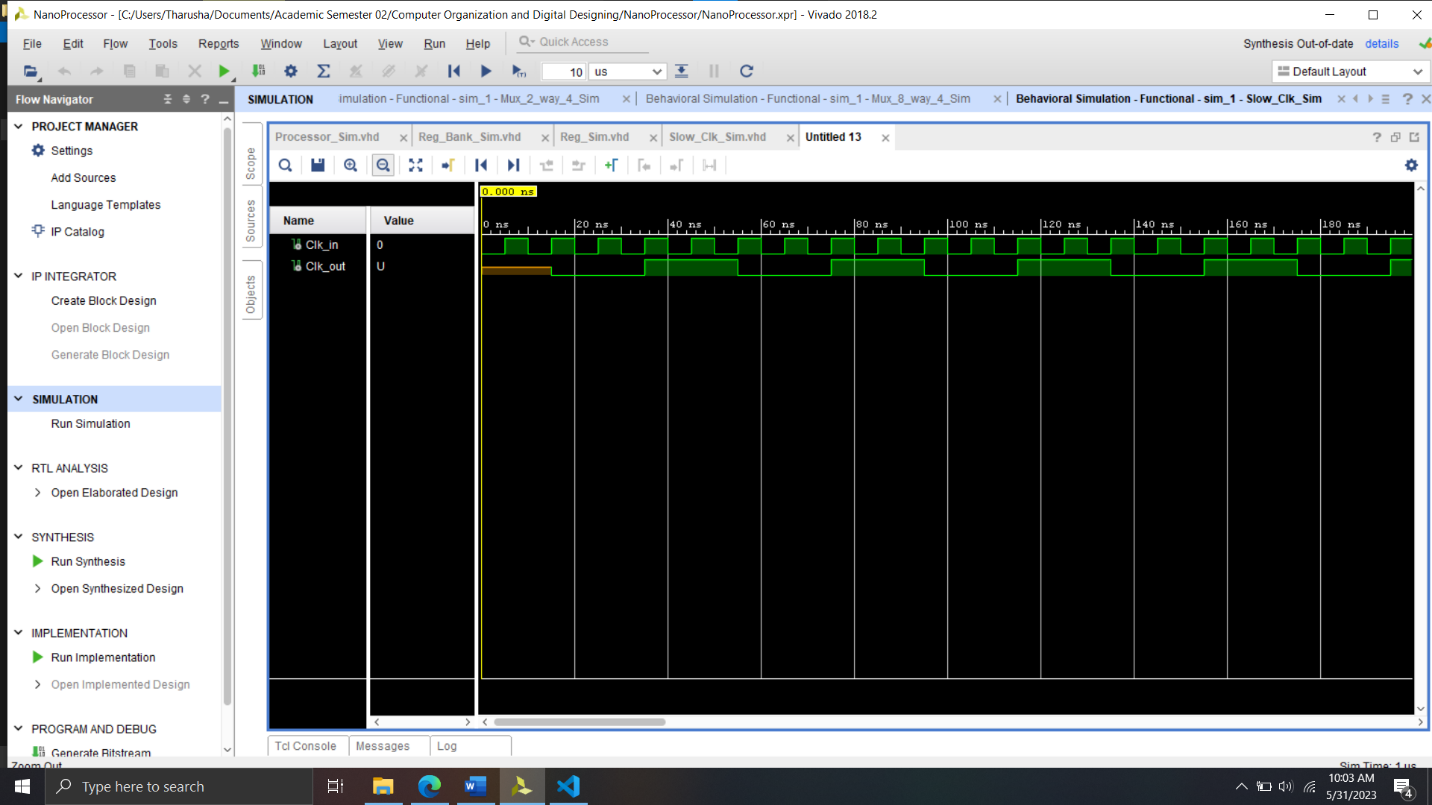
***Sim Register***

******

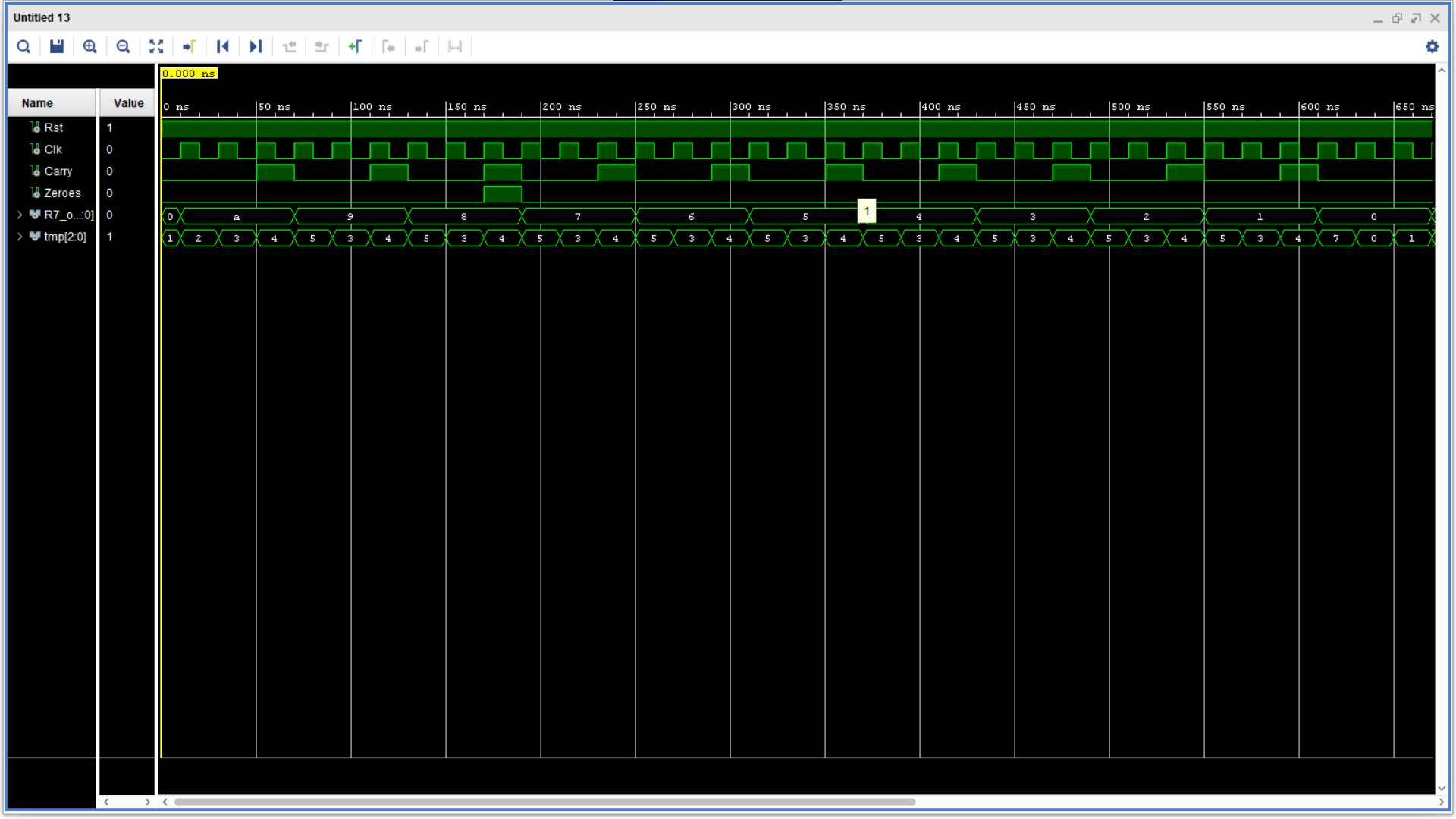
***Sim Register Bank***

******

***Sim Slow Clock***

******

***Sim Processor***

******

**Conclusion from the Lab**

We can use Basys 3 board to implement a processor and using that we can identify and get some idea about the functionalities of each units in Processor.

**Contribution for the project**

*Chandrawansha J.P.S.A*

* *Created Multiplexers.*
* *Created Decoders.*
* *Created Instruction Decoder.*
* *Created Registers and Register Bank.*
* *Designed the ALU.*
* *Created Program ROM.*
* *Created Simulation files for above files.*
* *Designed and constructed the Processor.*
* *Wrote the assembly program and hardcoded associated machine instructions into the ROM.*

*Dinujaya P.H.T*

* *Created Adder.*
* *Created Program Counter.*
* *Created Adder and Subtractor.*
* *Created Lookup Table for Seven Segment Display.*
* *Mapped LED with Basys 3 Board.*
* *Modified the constrain file.*
* *Created simulation for above files.*
* *Simulated the processor.*
* *Created Lab report.*