**Lab 9-10 – Nano-processor Design Competition**

CS1050 Computer Organization and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

|  |  |
| --- | --- |
| S.A.C.H.Gunapala  210190R | G.V.H.H.B.Jayarathna  210242F |

**Presented By** **–**

**Assigned Lab Task.**

This lab in particular is a team project. Teams of two are assigned to design and test a 4-bit processor on Basys3 FPGA which is capable of executing 4 instructions using a Programmable ROM.

**Assembly program and its machine code representation.**

Given a 12-bit instruction as follows,

Command

Activated Register A

Register B

Input Data

0 0 1 0 0 0 1 1 0 0 0 0

Let’s see some examples of machine code representation of some assembly language code,

|  |  |
| --- | --- |
| Machine Code | Assembly Language |
| 100010000011 | MOV R1, 3 |
| 101110000001 | MOV R7, 1 |
| 001110010000 | ADD R7, R1 |
| 011110000000 | NEG R7 |
| 110010000110 | JZR R1, 6 |

Possible commands and other useful commands,

|  |  |
| --- | --- |
| Command | Assembly Language Command |
| 00 | ADD (adds the data in two registers) |
| 10 | MOVI (moves data to a register) |
| 01 | NEG (gets the two’s compliment) |
| 11 | JZR (jump command if the given register has 0000 to an instruction) |

Register Selection,

|  |  |
| --- | --- |
| Command | Selected Register |
| 000 | R0 |
| 001 | R1 |
| 010 | R2 |
| 011 | R3 |
| 100 | R4 |
| 101 | R5 |
| 110 | R6 |
| 111 | R7 |

Last 4 digits display the immediate data input to the nanoprocessor.

**VHDL Codes.**

* **Instructions Decorder.**

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

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

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

Reg\_Sel0 : out STD\_LOGIC\_VECTOR (2 downto 0);

Reg\_Sel1 : out STD\_LOGIC\_VECTOR (2 downto 0);

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

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

Jump : out STD\_LOGIC;

Load\_Sel : out STD\_LOGIC;

Add\_Sub : out STD\_LOGIC);

end InstructionDecoder;

architecture Behavioral of InstructionDecoder is

signal Jump\_Temp : std\_logic;

begin

d\_value <= I(3 downto 0);

Reg\_EN <= I(9 downto 7);

Reg\_Sel1 <= I(9 downto 7);

Reg\_Sel0 <= I(6 downto 4);

Add\_Sub <= (not I(11)) and I(10);

Jump\_Temp <= I(11) and I(10);

Load\_Sel <= I(11) and (not I(10));

Address <= I(2 downto 0);

Jump <= Jump\_Temp and (not (Reg\_Check(0) or Reg\_Check(1) or Reg\_Check(2) or Reg\_Check(3)));

end Behavioral;

* **Mux 2way 4-bit.**

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\_2way\_4bit is

Port ( S : in STD\_LOGIC;

D0 : in STD\_LOGIC\_VECTOR (3 downto 0);

D1 : in STD\_LOGIC\_VECTOR (3 downto 0);

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

end Mux\_2way\_4bit;

architecture Behavioral of Mux\_2way\_4bit is

signal S\_vec :std\_logic\_vector(3 downto 0);

begin

S\_vec <= (others => S); -- Assigning S to every element of S\_vec

Y <= (D0 and not S\_vec) or (D1 and S\_vec);

end Behavioral;

* **LUT for 7-Segment Display.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity LUT\_16\_7 is

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

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

end LUT\_16\_7;

architecture Behavioral of LUT\_16\_7 is

type rom\_type is array (0 to 15) of std\_logic\_vector(6 downto 0);

signal sevenSegment\_ROM : rom\_type := (

"1000000", --0

"1111001", --1

"0100100", --2

"0110000", --3

"0011001", --4

"0010010", --5

"0000010", --6

"1111000", --7

"0000000", --8

"0010000", --9

"0001000", --a

"0000011", --b

"1000110", --c

"0100001", --d

"0000110", --e

"0001110" --f

);

begin

data <= sevenSegment\_ROM(to\_integer(unsigned(address)));

end Behavioral;

* **Programmable ROM.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity PROM is

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

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

end PROM;

architecture Behavioral of PROM is

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

signal instruction\_ROM : rom\_type := (

--CHANGE INSTRUCTIONS HERE

"101110000000", --MOV R7, 0

"100010000001", --MOV R1, 1

"100100000010", --MOV R2, 2

"100110000011", --MOV R3, 3

"001110010000", --ADD R7, R1

"001110100000", --ADD R7, R2

"001110110000", --ADD R7, R3

"110000000111" --JZR R0, 7

);

begin

I <= instruction\_ROM(to\_integer(unsigned(Mem\_Sel)));

end Behavioral;

* **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 PC is

Port ( Jump : in STD\_LOGIC;

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

Clk : in STD\_LOGIC;

Res : in STD\_LOGIC;

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

end PC;

architecture Behavioral of PC is

component RCA\_3

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

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

C\_out : out STD\_LOGIC);

end component;

component Mux\_2way\_3bit

Port ( S : in STD\_LOGIC;

D0 : in STD\_LOGIC\_VECTOR (2 downto 0);

D1 : in STD\_LOGIC\_VECTOR (2 downto 0);

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

end component;

component PCReg

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end component;

SIGNAL PC\_out, adder\_out,Mux\_out : STD\_LOGIC\_VECTOR(2 downto 0);

SIGNAL C\_out\_adder: STD\_LOGIC;

begin

adder\_3\_bit: RCA\_3

port map(

A => PC\_out,

S => adder\_out,

C\_out => C\_out\_adder);

Mux\_2way\_3bit\_0: Mux\_2way\_3bit

port map(

S => Jump,

D0 => adder\_out,

D1 => Address,

Y => Mux\_out);

ProgramCounter: PCReg

port map(

D => Mux\_out,

Res => Res,

Clk => Clk,

Q => PC\_out);

M <= PC\_out;

end Behavioral;

* **Program Counter Register(PC\_reg).**

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 PCReg is

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end PCReg;

architecture Behavioral of PCReg 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(0),

Res => Res,

Clk => Clk,

Q => Q(0));

D\_FF1 : D\_FF

port map (

D => D(1),

Res => Res,

Clk => Clk,

Q => Q(1));

D\_FF2 : D\_FF

port map (

D => D(2),

Res => Res,

Clk => Clk,

Q => Q(2));

end Behavioral;

* **D\_FF**

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 D\_FF is

Port ( D : in STD\_LOGIC;

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Q : out STD\_LOGIC);

end D\_FF;

architecture Behavioral of D\_FF is

signal Q\_reg : std\_logic := '0';

begin

process (Clk) begin

if (rising\_edge(Clk)) then

if Res = '1' then

Q\_reg <= '0';

else

Q\_reg <= D;

end if;

end if;

Q <= Q\_reg;

end process;

end Behavioral;

* **Mux\_2way\_3bit**

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\_2way\_3bit is

Port ( S : in STD\_LOGIC;

D0 : in STD\_LOGIC\_VECTOR (2 downto 0);

D1 : in STD\_LOGIC\_VECTOR (2 downto 0);

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

end Mux\_2way\_3bit;

architecture Behavioral of Mux\_2way\_3bit is

signal S\_vec :std\_logic\_vector(2 downto 0);

begin

S\_vec <= (others => S); -- Assigning S to every element of S\_vec

Y <= (D0 and not S\_vec) or (D1 and S\_vec);

end Behavioral;

* **Adder\_3\_bit**

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 RCA\_3 is

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

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

C\_out : out STD\_LOGIC);

end RCA\_3;

architecture Behavioral of RCA\_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 FA0\_C,FA1\_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\_out);

end Behavioral;

* **Add/Sub Unit.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity Add\_Sub\_unit is

Port ( D : in STD\_LOGIC\_VECTOR (31 downto 0);

RegSel0 : in STD\_LOGIC\_VECTOR (2 downto 0);

RegSel1 : in STD\_LOGIC\_VECTOR (2 downto 0);

AddSub : in STD\_LOGIC; -- 0 is adder, 1 is substracter

Overflow : out STD\_LOGIC;

Zero : out STD\_LOGIC;

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

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

end Add\_Sub\_unit;

architecture Behavioral of Add\_Sub\_unit is

component RCAS\_4

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

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

Func : in STD\_LOGIC; -- 0 is adder, 1 is substracter

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

C\_out : out STD\_LOGIC);

end component;

component Mux\_8way\_4bit

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

D : in STD\_LOGIC\_VECTOR (31 downto 0);

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

end component;

SIGNAL Mux0\_out, Mux1\_out, result: STD\_LOGIC\_VECTOR(3 downto 0);

SIGNAL Overflow\_flag: STD\_LOGIC;

begin

Mux\_8way\_4bit\_0 : Mux\_8way\_4bit

port map(

S => RegSel0,

D => D,

Y => Mux0\_out);

Mux\_8way\_4bit\_1 : Mux\_8way\_4bit

port map(

S => RegSel1,

D => D,

Y => Mux1\_out);

RCAS\_0 : RCAS\_4

port map(

A => Mux0\_out,

B => Mux1\_out,

Func => AddSub,

S => result,

C\_out => Overflow\_flag);

Overflow <= Overflow\_flag;

Output <= result;

RegCheck <= Mux0\_out;

Zero <= NOT( result(0) OR result(1) OR result(2) OR result(3) OR Overflow\_flag);

end Behavioral;

* **Mux\_8way\_4bit.**

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\_8way\_4bit is

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

D : in STD\_LOGIC\_VECTOR (31 downto 0);

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

end Mux\_8way\_4bit;

architecture Behavioral of Mux\_8way\_4bit is

component Decoder\_3\_to\_8

port (

address: in std\_logic\_vector(2 downto 0);

data: out std\_logic\_vector(7 downto 0));

end component;

signal decoded\_S :std\_logic\_vector(7 downto 0);

signal S0\_vec :std\_logic\_vector(3 downto 0);

signal S1\_vec :std\_logic\_vector(3 downto 0);

signal S2\_vec :std\_logic\_vector(3 downto 0);

signal S3\_vec :std\_logic\_vector(3 downto 0);

signal S4\_vec :std\_logic\_vector(3 downto 0);

signal S5\_vec :std\_logic\_vector(3 downto 0);

signal S6\_vec :std\_logic\_vector(3 downto 0);

signal S7\_vec :std\_logic\_vector(3 downto 0);

begin

LUT\_Decoder : Decoder\_3\_to\_8

port map(

address => S,

data => decoded\_S);

S0\_vec <= (others => decoded\_S(0)); -- Assigning decoded\_s(0) to every element of S0\_vec

S1\_vec <= (others => decoded\_S(1));

S2\_vec <= (others => decoded\_S(2));

S3\_vec <= (others => decoded\_S(3));

S4\_vec <= (others => decoded\_S(4));

S5\_vec <= (others => decoded\_S(5));

S6\_vec <= (others => decoded\_S(6));

S7\_vec <= (others => decoded\_S(7));

Y <= (D(3 downto 0) and S0\_vec) or (D(7 downto 4) and S1\_vec) or (D(11 downto 8) and S2\_vec) or (D(15 downto 12) and S3\_vec) or (D(19 downto 16) and S4\_vec) or (D(23 downto 20) and S5\_vec) or (D(27 downto 24) and S6\_vec) or (D(31 downto 28) and S7\_vec);

end Behavioral;

* **RCAS (Ripple carry 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 RCAS\_4 is

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

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

Func : in STD\_LOGIC; -- 0 is adder, 1 is substracter

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

C\_out : out STD\_LOGIC);

end RCAS\_4;

architecture Behavioral of RCAS\_4 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 : STD\_LOGIC;

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

begin

FA\_0 : FA

port map(

A => A(0),

B => BS(0),

C\_in => Func,

S => S(0),

C\_out => FA0\_C);

FA\_1 : FA

port map(

A => A(1),

B => BS(1),

C\_in => FA0\_C,

S => S(1),

C\_out => FA1\_C);

FA\_2 : FA

port map(

A => A(2),

B => BS(2),

C\_in => FA1\_C,

S => S(2),

C\_out => FA2\_C);

FA\_3 : FA

port map(

A => A(3),

B => BS(3),

C\_in => FA2\_C,

S => S(3),

C\_out => C\_out);

BS(0) <= B(0) xor Func;

BS(1) <= B(1) xor Func;

BS(2) <= B(2) xor Func;

BS(3) <= B(3) xor Func;

end Behavioral;

* **FA (Full Adder).**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity FA is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C\_in : in STD\_LOGIC;

S : out STD\_LOGIC;

C\_out : out STD\_LOGIC);

end FA;

architecture Behavioral of FA is

component HA

port (

A : in STD\_LOGIC;

B : in STD\_LOGIC;

S : out STD\_LOGIC;

C : out STD\_LOGIC);

end component;

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

begin

HA\_0 : HA

port map(

A => A,

B => B,

S => HA0\_S,

C => HA0\_C);

HA\_1 : HA

port map(

A => HA0\_S,

B => C\_in,

S => HA1\_S,

C => HA1\_C);

S <= HA1\_S;

C\_out <= HA0\_C OR HA1\_C;

end Behavioral;

* **HA (Half Adder).**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity 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;

* **Reg Bank.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity RegisterBank is

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

D : out STD\_LOGIC\_VECTOR (31 downto 0));

end RegisterBank;

architecture Behavioral of RegisterBank is

component Reg

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

En : in STD\_LOGIC;

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end component;

component Decoder\_3\_to\_8

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

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

end component;

signal EN\_vec : std\_logic\_vector(7 downto 0);

begin

LUT\_Decoder : Decoder\_3\_to\_8

PORT MAP(

address => R\_EN,

data => EN\_vec

);

Reg\_0 : Reg

PORT MAP(

D => "0000",

En => '1' ,

Res => '0',

Clk => Clk,

Q => D(3 downto 0)

);

Reg\_1 : Reg

PORT MAP(

D => Input,

En => EN\_vec(1) ,

Res => Res,

Clk => Clk,

Q => D(7 downto 4)

);

Reg\_2 : Reg

PORT MAP(

D => Input,

En => EN\_vec(2) ,

Res => Res,

Clk => Clk,

Q => D(11 downto 8)

);

Reg\_3 : Reg

PORT MAP(

D => Input,

En => EN\_vec(3) ,

Res => Res,

Clk => Clk,

Q => D(15 downto 12)

);

Reg\_4 : Reg

PORT MAP(

D => Input,

En => EN\_vec(4) ,

Res => Res,

Clk => Clk,

Q => D(19 downto 16)

);

Reg\_5 : Reg

PORT MAP(

D => Input,

En => EN\_vec(5) ,

Res => Res,

Clk => Clk,

Q => D(23 downto 20)

);

Reg\_6 : Reg

PORT MAP(

D => Input,

En => EN\_vec(6) ,

Res => Res,

Clk => Clk,

Q => D(27 downto 24)

);

Reg\_7 : Reg

PORT MAP(

D => Input,

En => EN\_vec(7) ,

Res => Res,

Clk => Clk,

Q => D(31 downto 28)

);

end Behavioral;

* **Decoder\_3\_to\_8(LUT).**

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 Decoder\_3\_to\_8 is

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

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

end Decoder\_3\_to\_8;

architecture Behavioral of Decoder\_3\_to\_8 is

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

signal decoder\_ROM : rom\_type := (

"00000001", --0

"00000010", --1

"00000100", --2

"00001000", --3

"00010000", --4

"00100000", --5

"01000000", --6

"10000000" --7

);

begin

data <= decoder\_ROM(to\_integer(unsigned(address)));

end Behavioral;

* **Register.**

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 is

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

En : in STD\_LOGIC;

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end Reg;

architecture Behavioral of Reg is

begin

process (Clk,Res)

begin

if (rising\_edge(Clk)) then -- respond when clock rises

if En = '1' then -- Enable should be set

Q <= D;

end if;

if (Res = '1') then

Q <= "0000";

end if;

end if;

end process;

end Behavioral;

* **Slow Clock.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity Slow\_Clk is

Port ( Clk\_in : in STD\_LOGIC;

Clk\_out : out STD\_LOGIC);

end Slow\_Clk;

architecture Behavioral of Slow\_Clk is

signal count: integer := 1;

signal clk\_status: std\_logic := '0';

begin

process(Clk\_in) begin

if (rising\_edge(Clk\_in)) then

count <= count + 1;

--if (count = 5) then

if(count = 50000000) then

clk\_status <= not clk\_status;

Clk\_out <= clk\_status;

count <= 1;

end if;

end if;

end process;

end Behavioral;

* **NanoProcessor.**

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;

Seg7input : out std\_logic\_vector (3 downto 0);

Reset : in STD\_LOGIC;

OverFlow : out STD\_LOGIC;

Zero : out STD\_LOGIC;

Seg7output : out STD\_LOGIC\_VECTOR (6 downto 0);

Anode : out std\_logic\_vector (3 downto 0));

end NanoProcessor;

architecture Behavioral of NanoProcessor is

component Slow\_Clk

Port ( Clk\_in : in STD\_LOGIC;

Clk\_out : out STD\_LOGIC);

end component;

component LUT\_16\_7

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

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

end component;

component RegisterBank

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

D : out STD\_LOGIC\_VECTOR (31 downto 0));

end component;

component Add\_Sub\_unit

Port ( D : in STD\_LOGIC\_VECTOR (31 downto 0);

RegSel0 : in STD\_LOGIC\_VECTOR (2 downto 0);

RegSel1 : in STD\_LOGIC\_VECTOR (2 downto 0);

AddSub : in STD\_LOGIC; -- 0 is adder, 1 is substracter

Overflow : out STD\_LOGIC;

Zero : out STD\_LOGIC;

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

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

end component;

component PC

Port ( Jump : in STD\_LOGIC;

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

Clk : in STD\_LOGIC;

Res : in STD\_LOGIC;

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

end component;

component PROM

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

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

end component;

component Mux\_2way\_4bit

Port ( S : in STD\_LOGIC;

D0 : in STD\_LOGIC\_VECTOR (3 downto 0);

D1 : in STD\_LOGIC\_VECTOR (3 downto 0);

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

end component;

component InstructionDecoder

Port ( I : in STD\_LOGIC\_VECTOR (11 downto 0);

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

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

Reg\_Sel0 : out STD\_LOGIC\_VECTOR (2 downto 0);

Reg\_Sel1 : out STD\_LOGIC\_VECTOR (2 downto 0);

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

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

Jump : out STD\_LOGIC;

Load\_Sel : out STD\_LOGIC;

Add\_Sub : out STD\_LOGIC);

end component;

signal slow\_clk\_sig, AddSub,Jump, Load\_sel : std\_logic;

signal Reg\_En,RegSel0, RegSel1, JumpAddress,M : std\_logic\_vector(2 downto 0);

signal Regbank\_input, AddSub\_out, RegCheck, d\_value : std\_logic\_vector(3 downto 0);

signal I : std\_logic\_vector(11 downto 0);

signal D : std\_logic\_vector(31 downto 0);

begin

Slow\_Clk\_0: Slow\_Clk

port map(

Clk\_in => Clk,

Clk\_out => slow\_clk\_sig

);

Reg\_bank: RegisterBank

port map(

R\_EN => Reg\_En,

Res => Reset,

Clk => slow\_clk\_sig,

Input => Regbank\_input,

D => D

);

AddSubUnit: Add\_Sub\_unit

port map(

D => D,

RegSel0 => RegSel0,

RegSel1 => RegSel1,

AddSub => AddSub,

OverFlow => overflow,

Zero => Zero,

Output => AddSub\_out,

RegCheck => RegCheck

);

ProgrammeCounter: PC

port map(

Jump => Jump,

Address => JumpAddress,

Clk => slow\_clk\_sig,

Res => Reset,

M => M

);

ProgramROM : PROM

port map(

Mem\_Sel => M,

I => I

);

LUT\_16\_7\_0: LUT\_16\_7

port map(

result => D(31 downto 28),

data => Seg7output

);

Mux: Mux\_2way\_4bit

port map(

S => Load\_sel,

D0 => AddSub\_out,

D1 => d\_value,

Y => Regbank\_input

);

Instruction\_decoder: InstructionDecoder

port map(

I => I,

Reg\_Check => RegCheck,

Reg\_EN => Reg\_En,

Reg\_sel0 => RegSel0,

Reg\_sel1 => RegSel1,

Address => JumpAddress,

d\_value => d\_value,

Jump => Jump,

Load\_Sel => Load\_sel,

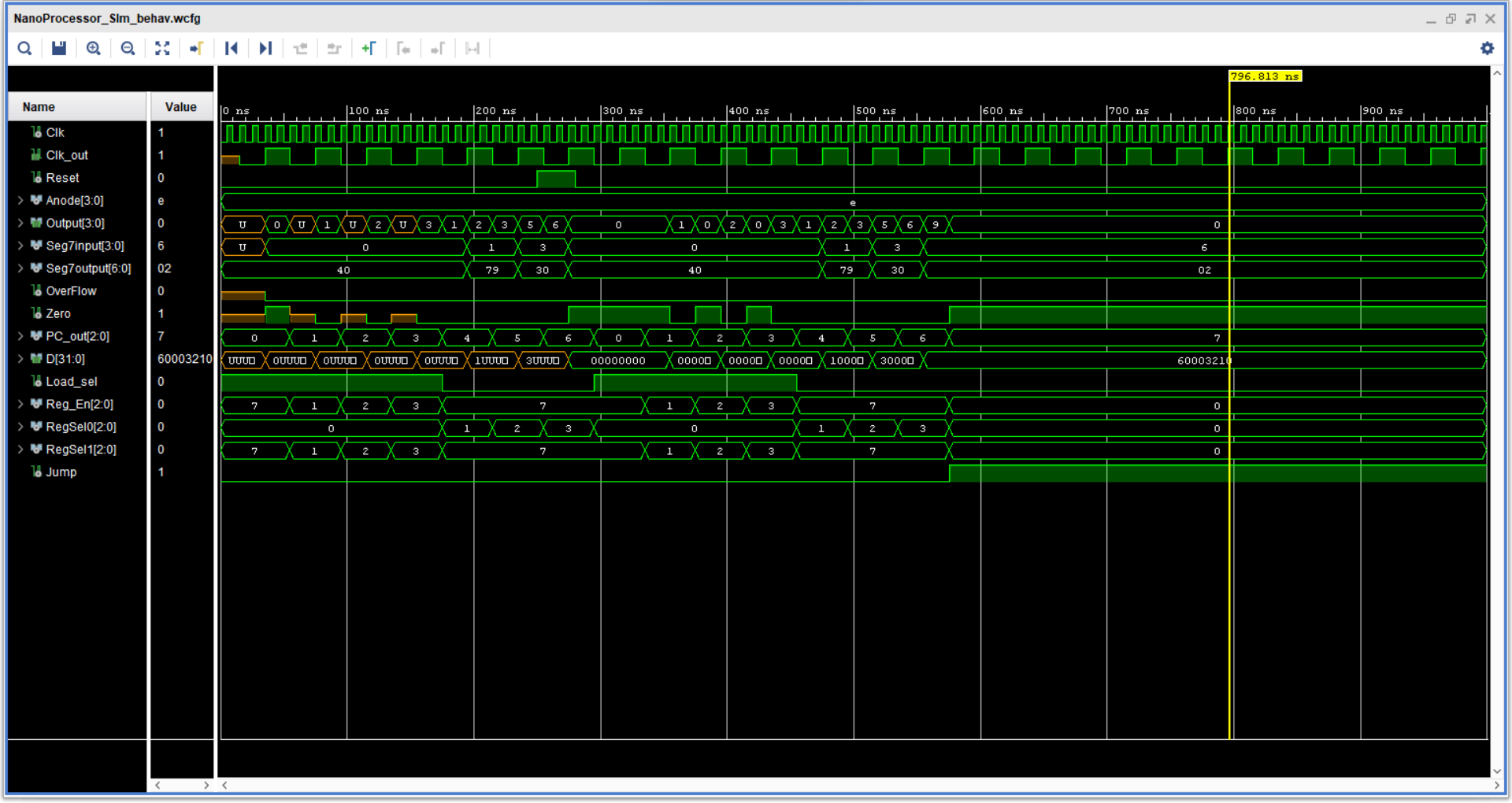
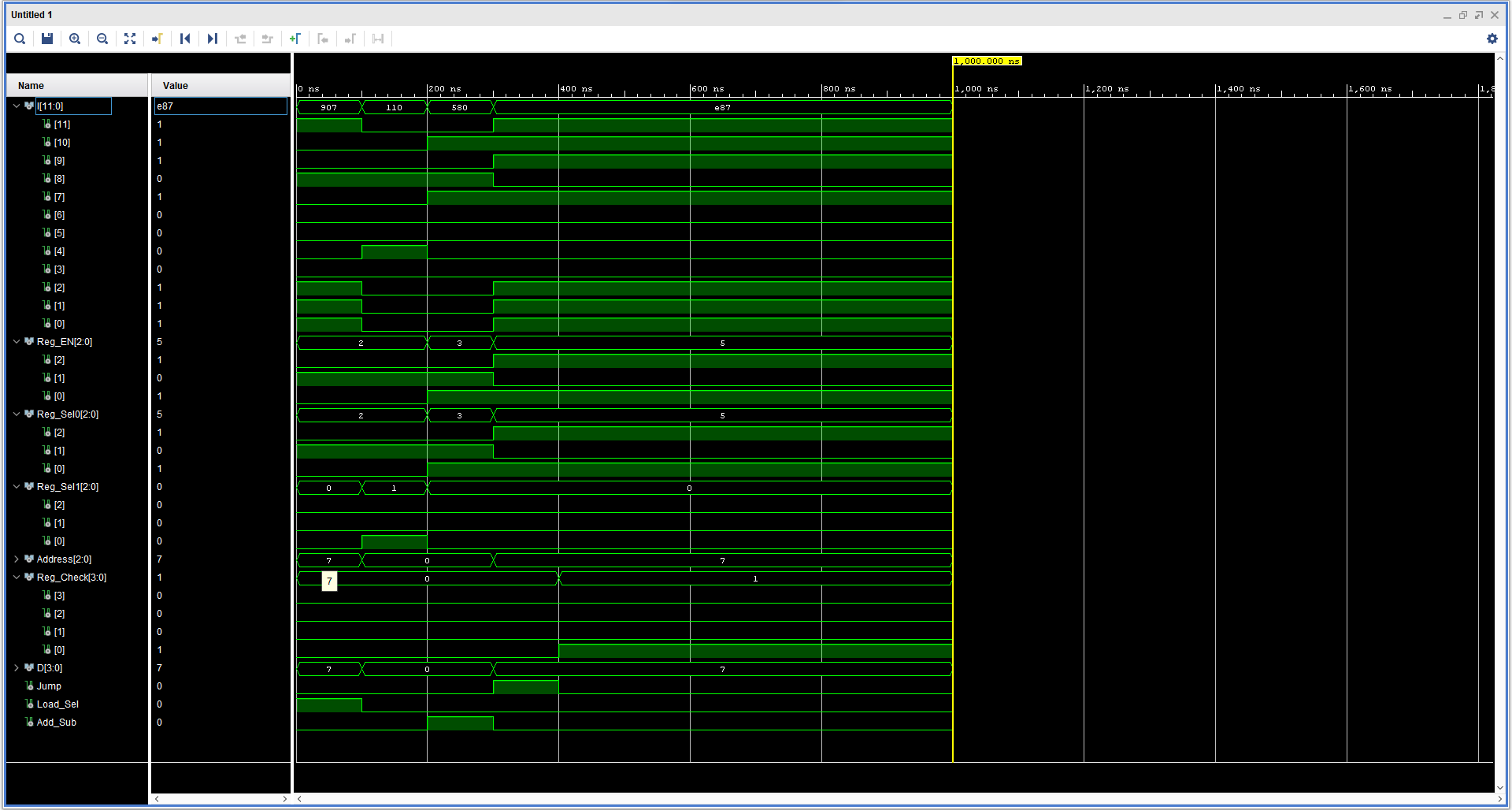
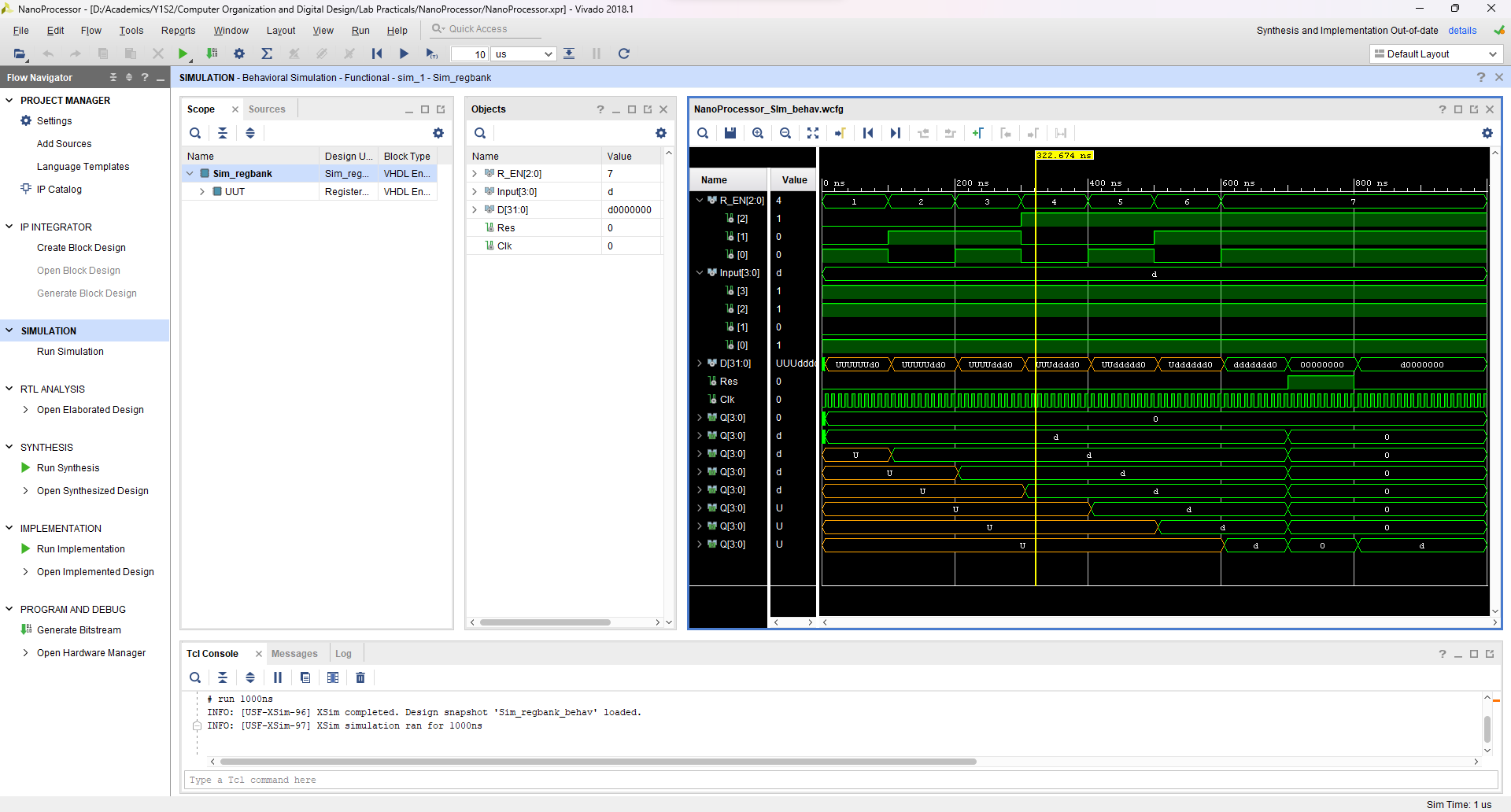
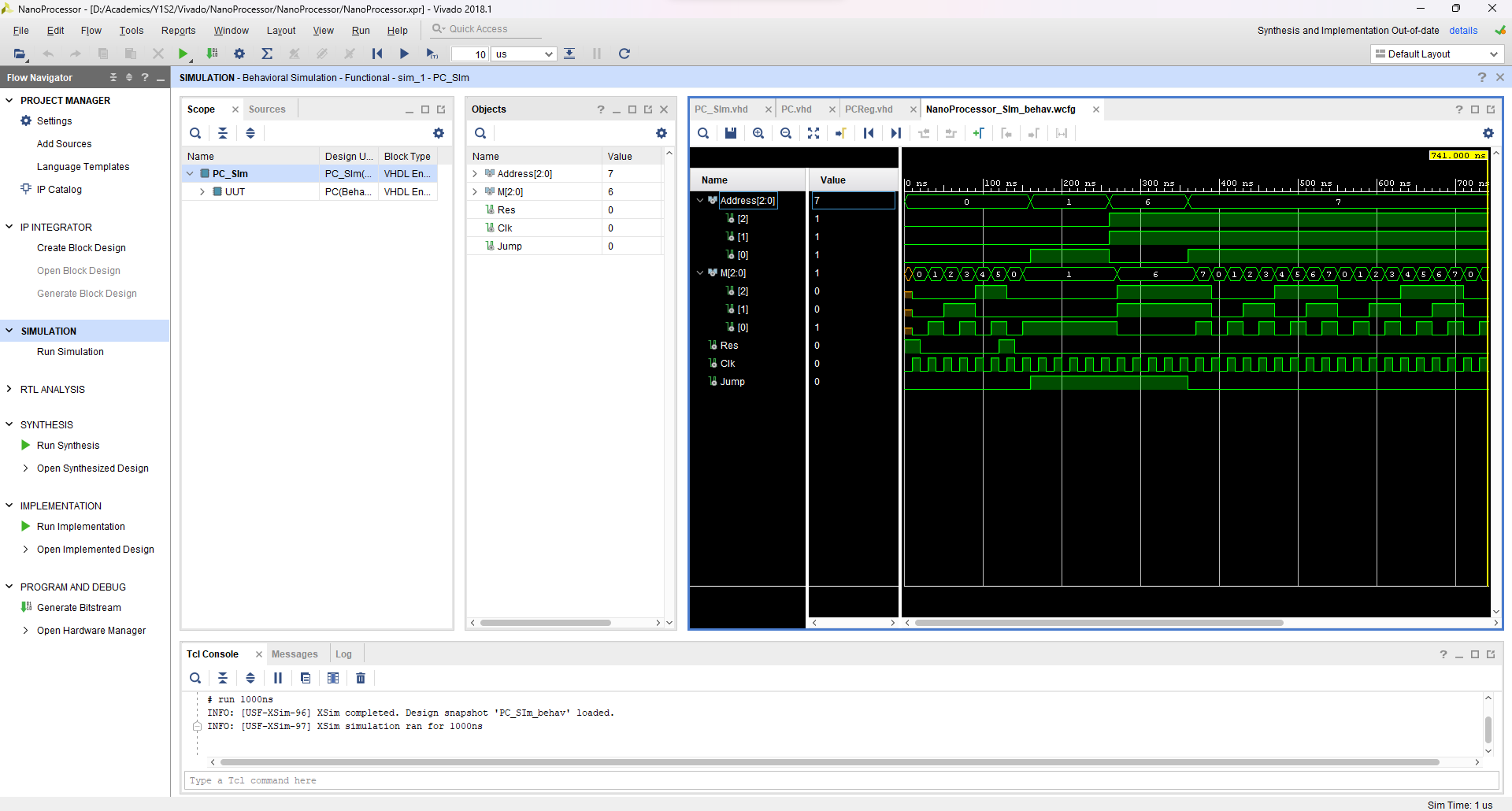
Add\_Sub => AddSub

);

Anode <= "1110";

Seg7input <= D(31 downto 28);

**Timing Diagrams.**

* **Nano-Processor.**
* **Instructions Decoder.**
* **Reg Bank.**
* **Program Counter.**

**Conclusion.**

We were able to practice team-working skills and improve communication, coordination, sharing responsibilities and integrating components developed by different team members. We were able to design our very own 4-bit arithmetic unit that can add and subtract signed integers, decode instructions to activate different components of the processor and to verify the functionality of the nano-processor.

**Contribution.**

|  |  |
| --- | --- |
| S.A.C.H.Gunapala  210190R | G.V.H.H.B.Jayarathna  210242F |
| Designed the Register Bank | Designed the 4-bit Add/Sub unit |
| Designed all the Multiplexers inside the nano-processor | Designed 3-bit adder |
| Designed the Programmable ROM | Designed the Instruction Decoder |
| Designed part of Program Counter | Designed part of Program Counter. |
| Simulated the related components | Simulated the related components |

Designing each component and connecting them took about 12 hours. Creating Test Bench files and simulating each component separately took about 6 hours. Connecting each component and simulating the Nano-Processor took 2 hours. Fixing bugs and optimizing the components for less logic gates took about another 10 – 12 hours. Spent about 30 – 32 hours completing the Nano-Processor.

**NanoProcessor simulation VHDL.**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity NanoProcessor\_SIm is

-- Port ( );

end NanoProcessor\_SIm;

architecture Behavioral of NanoProcessor\_SIm is

component NanoProcessor

Port ( Clk : in STD\_LOGIC;

Reset : in STD\_LOGIC;

Seg7input : out std\_logic\_vector (3 downto 0);

OverFlow : out STD\_LOGIC;

Zero : out STD\_LOGIC;

Seg7output : out STD\_LOGIC\_VECTOR (6 downto 0);

Anode : out std\_logic\_vector (3 downto 0));

end component;

signal Clk, Reset, OverFlow, Zero : std\_logic;

signal Seg7output : std\_logic\_vector (6 downto 0);

signal Seg7input,Anode : std\_logic\_vector (3 downto 0);

begin

UUT: NanoProcessor port map (

Clk => Clk,

Reset => Reset,

Seg7input => Seg7input,

OverFlow => OverFlow,

Zero => Zero,

Seg7output => Seg7output,

Anode => Anode);

Clk\_process : process

begin

Clk <= '0';

wait for 5 ns;

Clk <= '1';

wait for 5 ns;

end process;

processor : process

begin

Reset <= '0';

wait for 250 ns;

Reset <= '1';

wait for 30 ns;

Reset <= '0';

wait;

end process;

end Behavioral;