**LAB 9-10 - NANOPROCESSOR DESIGN COMPETITION**

**GROUP 89**

* K.S. RANASINGHE - 210518H
* W.A.R.T FONSEKA - 210170G

**LAB TASK**

* Design a 4-bit arithmetic unit that can add and subtract signed integers.
* Design k-way b-bit multiplexers as necessary.
* Develop an assembly code that would simulate the necessary process.
* Hard code the Assembly code into the Program ROM.
* Design an Instruction Decoder that decodes the instructions in the Program ROM.
* Connect the individual components that were designed earlier together to create a Nano processor.

**ASSEMBLY PROGRAM**

MOVI R7, 0 // move zero to register R7

MOVI R1, 3 // move three to register R1

MOVI R2, -1 // move negative one to register R2

ADD R7, R1 // add values in R7 and R1 and move new value to R7

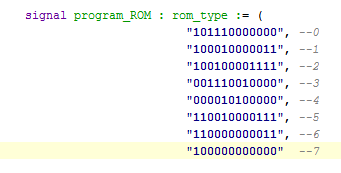
ADD R1, R2 // add values in R1 and R2 and move new value to R1

JZR R1, 7 // if value in R1 is equal to zero jump to instruction no. 7

JZR R0, 3 // if value in R0 is equal to zero jump to instruction no. 3

END

MACHINE CODE REPRESENTATION



**VHDL CODES**

* 4-bit Add/Subtract unit

Design Source File

entity AdderSubtracter is

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

C\_in, Ctrl : in STD\_LOGIC;

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

C\_out, Zero, Overflow : out STD\_LOGIC);

end AdderSubtracter;

architecture Behavioral of AdderSubtracter 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, FA\_S, T : std\_logic\_vector(3 downto 0);

SIGNAL N : std\_logic;

begin

process(Ctrl, B)

begin

case Ctrl is

when '1' => T <= NOT B;

when '0' => T <= B;

when others => T <= "UUUU";

end case;

end process;

FA\_0 :FA

PORT MAP(

A => A(0),

B => T(0),

C\_in => Ctrl,

S => FA\_S(0),

C\_Out => FA\_C(0));

FA\_1 :FA

PORT MAP(

A => A(1),

B => T(1),

C\_in => FA\_C(0),

S => FA\_S(1),

C\_Out => FA\_C(1));

FA\_2 :FA

PORT MAP(

A => A(2),

B => T(2),

C\_in => FA\_C(1),

S => FA\_S(2),

C\_Out => FA\_C(2));

FA\_3 :FA

PORT MAP(

A => A(3),

B => T(3),

C\_in => FA\_C(2),

S => FA\_S(3),

C\_Out => FA\_C(3));

S <= FA\_S;

C\_out <= FA\_C(3);

N <= NOT (A(3) XOR B(3)) AND (NOT Ctrl) AND FA\_C(3);

Overflow <= A(3) AND T(3);

process(FA\_S)

begin

if FA\_S = "0000" AND N = '0' then

Zero <= '1';

else

Zero <= '0';

end if;

end process;

end Behavioral;

Test Bench File

entity AdderSubtracterSim is

-- Port ( );

end AdderSubtracterSim;

architecture Behavioral of AdderSubtracterSim is

COMPONENT AdderSubtracter

PORT( A,B : IN STD\_LOGIC\_VECTOR(3 downto 0);

C\_in, Ctrl : IN STD\_LOGIC;

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

C\_out, Zero, Overflow : OUT STD\_LOGIC);

END COMPONENT;

SIGNAL A,B : std\_logic\_vector(3 downto 0);

SIGNAL C\_in, Ctrl : std\_logic;

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

SIGNAL C\_out,Zero,Overflow : std\_logic;

begin

UUT : AdderSubtracter PORT MAP(

A => A,

B => B,

Ctrl => Ctrl,

C\_in => C\_in,

C\_out => C\_out,

Zero => Zero,

Overflow => Overflow,

S => S

);

process

begin

A <= "0000";

B <= "0000";

C\_in <= '0';

Ctrl <= '0';

-- 0101 + 1011

WAIT FOR 100 ns;

A <= "0101";

B <= "1011";

Ctrl <= '1';

-- 0111 + 1111

WAIT FOR 100 ns;

A <= "0111";

B <= "1111";

Ctrl <= '0';

-- Index Number: 210518H => 11 0011 0110 0101 0110

-- Index Number: 210170G => 11 0011 0100 1111 1010

-- 0110 + 0101

WAIT FOR 100 ns;

A <= "0110";

B <= "0101";

Ctrl <= '0';

-- 0110 - 0011

WAIT FOR 100 ns;

A <= "0110";

B <= "0011";

Ctrl <= '1';

-- 1010 + 1111

WAIT FOR 100 ns;

A <= "1010";

B <= "1111";

Ctrl <= '0';

-- 0011 - 0100

WAIT FOR 100 ns;

A <= "0011";

B <= "0100";

Ctrl <= '1';

-- 0011 + 0011

WAIT FOR 100 ns;

A <= "0011";

B <= "0011";

Ctrl <= '0';

-- 0110 - 0100

WAIT FOR 100 ns;

A <= "0110";

B <= "0100";

Ctrl <= '1';

-- 0101 + 1111

WAIT FOR 100 ns;

A <= "0101";

B <= "1111";

Ctrl <= '0';

-- 0110 - 1010

WAIT FOR 100 ns;

A <= "0110";

B <= "1010";

Ctrl <= '1';

WAIT;

end process;

* 3-bit adder

Design Source File

entity Adder3Bit is

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

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

end Adder3Bit;

architecture Behavioral of Adder3Bit is

SIGNAL T : std\_logic\_vector(2 downto 0);

begin

T(0) <= '1';

S(0) <= A(0) XOR T(0);

T(1) <= A(0) AND T(0);

S(1) <= A(1) XOR T(1);

T(2) <= A(1) AND T(1);

S(2) <= A(2) XOR T(2);

end Behavioral;

Test Bench File

entity Adder3BitSim is

-- Port ( );

end Adder3BitSim;

architecture Behavioral of Adder3BitSim is

COMPONENT Adder3Bit

PORT( A : IN STD\_LOGIC\_VECTOR(2 downto 0);

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

END COMPONENT;

SIGNAL A : std\_logic\_vector(2 downto 0);

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

begin

UUT : Adder3Bit PORT MAP(

A => A,

S => S

);

process

begin

A <= "000";

WAIT FOR 100 ns;

A <= "001";

WAIT FOR 100 ns;

A <= "010";

WAIT FOR 100 ns;

A <= "011";

WAIT FOR 100 ns;

A <= "100";

WAIT FOR 100 ns;

A <= "101";

WAIT FOR 100 ns;

A <= "110";

WAIT FOR 100 ns;

A <= "111";

-- Index Number: 210518H => 110 011 011 001 010 110

-- Index Number: 210170G => 110 011 010 011 111 010

WAIT FOR 100 ns;

A <= "110";

WAIT FOR 100 ns;

A <= "010";

WAIT FOR 100 ns;

A <= "001";

WAIT FOR 100 ns;

A <= "011";

WAIT FOR 100 ns;

A <= "011";

WAIT FOR 100 ns;

A <= "110";

WAIT FOR 100 ns;

A <= "010";

WAIT FOR 100 ns;

A <= "111";

WAIT FOR 100 ns;

A <= "011";

WAIT FOR 100 ns;

A <= "010";

WAIT FOR 100 ns;

A <= "011";

WAIT FOR 100 ns;

A <= "110";

WAIT;

end process;

end Behavioral;

* 3-bit Program Counter (PC)

Design Source File

entity ProgramCounter is

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

Res, Clk : in STD\_LOGIC;

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

end ProgramCounter;

architecture Behavioral of ProgramCounter is

component D\_FF

port (

D : in STD\_LOGIC;

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Q : out STD\_LOGIC;

Qbar : out STD\_LOGIC );

end component;

component Slow\_Clk

port (

Clk\_in : in STD\_LOGIC;

Clk\_out : out STD\_LOGIC );

end component;

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

signal Clk\_slow : std\_logic;

begin

Slow\_Clk0 : Slow\_Clk

port map (

Clk\_in => Clk,

Clk\_out => Clk\_slow );

D\_FF0 : D\_FF

port map (

D => D(0),

Res => Res,

Clk => Clk\_slow,

Q => QD(0) );

D\_FF1 : D\_FF

port map (

D => D(1),

Res => Res,

Clk => Clk\_slow,

Q => QD(1) );

D\_FF2 : D\_FF

port map (

D => D(2),

Res => Res,

Clk => Clk\_slow,

Q => QD(2) );

Q <= QD;

end Behavioral;

Test Bench File

entity ProgramCounterSim is

-- Port ( );

end ProgramCounterSim;

architecture Behavioral of ProgramCounterSim is

COMPONENT ProgramCounter

PORT( Res, Clk : IN STD\_LOGIC;

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

Q : OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0));

END COMPONENT;

SIGNAL Res, Clk : std\_logic;

SIGNAL D, Q : std\_logic\_vector(2 downto 0);

begin

UUT : ProgramCounter PORT MAP(

D => D,

Res => Res,

Clk => Clk,

Q => Q );

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

--INPUT SIGNAL = 100MHz (10 ns)

--SLOW CLOCK COUNT = 50 000 000 == 50\_000\_000

--SLOW CLOCK SIM = WAIT FOR 1 ns

--OUTPUT SIGNAL = 0.2Hz (2 s)

process

begin

D <= "000";

Res <= '0';

WAIT FOR 500 ns;

-- Index Number: 210518H => 110 011 011 001 010 110

D <= "110";

Res <= '1';

WAIT FOR 5 ms;

D <= "010";

Res <= '0';

WAIT FOR 5 ms;

D <= "001";

Res <= '1';

WAIT FOR 5 ms;

D <= "011";

Res <= '0';

WAIT FOR 4 ms;

D <= "011";

Res <= '1';

WAIT FOR 5 ms;

-- Index Number: 210170G => 110 011 010 011 111 010

D <= "010";

Res <= '0';

WAIT FOR 7 ms;

D <= "111";

Res <= '1';

WAIT FOR 5 ms;

D <= "011";

Res <= '0';

WAIT FOR 3 ms;

D <= "010";

Res <= '1';

WAIT FOR 5 ms;

D <= "011";

Res <= '0';

WAIT FOR 2 ms;

end process;

end Behavioral;

* 2-way 3-bit multiplexer

Design Source File

entity Mux\_2way\_3bit is

Port ( Sel,Clk : in STD\_LOGIC;

A,B : 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

begin

process(Clk,Sel)

begin

if (rising\_edge(Clk)) then

case Sel is

when '0' => Y <= A;

when '1' => Y <= B;

when others => Y <= "UUU";

end case;

end if;

end process;

end Behavioral;

Test Bench File

entity Mux2Sim is

-- Port ( );

end Mux2Sim;

architecture Behavioral of Mux2Sim is

COMPONENT Mux\_2way\_3bit

PORT( Sel,Clk : IN STD\_LOGIC;

A,B : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

Y : OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0));

END COMPONENT;

SIGNAL Sel,Clk : std\_logic;

SIGNAL A,B,Y : std\_logic\_vector(2 downto 0);

begin

UUT : Mux\_2way\_3bit PORT MAP(

Sel => Sel,

A => A,

B => B,

Y => Y,

Clk => Clk);

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

process

begin

-- Index Number: 210518H => 110 011 011 001 010 110

-- Index Number: 210170G => 110 011 010 011 111 010

Sel <= '0';

A <= "110";

B <= "010";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "010";

B <= "111";

WAIT FOR 50 ns;

Sel <= '0';

A <= "001";

B <= "011";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "011";

B <= "010";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "011";

B <= "011";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "110";

B <= "110";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

end process;

end Behavioral;

* 2-way 4-bit multiplexer

Design Source File

entity Mux\_2way\_4bit is

Port ( Sel, Clk : in STD\_LOGIC;

A,B : 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

begin

process(Clk,Sel)

begin

if (rising\_edge(Clk)) then

case Sel is

when '0' => Y <= A;

when '1' => Y <= B;

when others => Y <= "UUUU";

end case;

end if;

end process;

end Behavioral;

Test Bench File

entity Mux1Sim is

-- Port ( );

end Mux1Sim;

architecture Behavioral of Mux1Sim is

COMPONENT Mux\_2way\_4bit

PORT( Sel, Clk : IN STD\_LOGIC;

A,B : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Y : OUT STD\_LOGIC\_VECTOR(3 DOWNTO 0));

END COMPONENT;

SIGNAL Sel,Clk : std\_logic;

SIGNAL A,B,Y : std\_logic\_vector(3 downto 0);

begin

UUT : Mux\_2way\_4bit PORT MAP(

Sel => Sel,

A => A,

B => B,

Y => Y,

Clk => Clk);

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

process

begin

-- Index Number: 210518H => 11 0011 0110 0101 0110

-- Index Number: 210170G => 11 0011 0100 1111 1010

Sel <= '0';

A <= "0110";

B <= "1010";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "0101";

B <= "1111";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "0110";

B <= "0100";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

Sel <= '0';

A <= "0011";

B <= "0011";

WAIT FOR 50 ns;

Sel <= '1';

WAIT FOR 50 ns;

end process;

end Behavioral;

* 8-way 4-bit multiplexer

Design Source File

entity Mux\_8way\_4bit is

Port ( Clk : in STD\_LOGIC;

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

A,B,C,D,E,F,G,H : in STD\_LOGIC\_VECTOR (3 downto 0);

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

end Mux\_8way\_4bit;

architecture Behavioral of Mux\_8way\_4bit is

begin

process(Clk,Sel)

begin

if (rising\_edge(Clk)) then

case Sel is

when "000" => Y <= A;

when "001" => Y <= B;

when "010" => Y <= C;

when "011" => Y <= D;

when "100" => Y <= E;

when "101" => Y <= F;

when "110" => Y <= G;

when "111" => Y <= H;

when others => Y <= "UUUU";

end case;

end if;

end process;

end Behavioral;

Test Bench File

entity MuxSim is

-- Port ( );

end MuxSim;

architecture Behavioral of MuxSim is

COMPONENT Mux\_8way\_4bit

PORT( Clk : IN STD\_LOGIC;

Sel : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

A,B,C,D,E,F,G,H : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

Y : OUT STD\_LOGIC\_VECTOR(3 DOWNTO 0));

END COMPONENT;

SIGNAL Clk : std\_logic;

SIGNAL Sel : std\_logic\_vector(2 downto 0);

SIGNAL A,B,C,D,E,F,G,H,Y : std\_logic\_vector(3 downto 0);

begin

UUT : Mux\_8way\_4bit PORT MAP(

Sel => Sel,

Clk => Clk,

A => A,

B => B,

C => C,

D => D,

E => E,

F => F,

G => G,

H => H,

Y => Y );

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

process

begin

Sel <= "000";

A <= "0110";

B <= "1001";

C <= "1010";

D <= "0101";

E <= "0110";

F <= "1101";

G <= "1110";

H <= "1111";

WAIT FOR 50 ns;

-- Index Number: 210518H => 110 011 011 001 010 110

Sel <= "110";

WAIT FOR 50 ns;

Sel <= "010";

WAIT FOR 50 ns;

Sel <= "001";

WAIT FOR 50 ns;

Sel <= "011";

WAIT FOR 50 ns;

Sel <= "011";

WAIT FOR 50 ns;

Sel <= "110";

WAIT FOR 50 ns;

-- Index Number: 210170G => 110 011 010 011 111 010

Sel <= "010";

WAIT FOR 50 ns;

Sel <= "111";

WAIT FOR 50 ns;

Sel <= "011";

WAIT FOR 50 ns;

Sel <= "010";

WAIT FOR 50 ns;

Sel <= "011";

WAIT FOR 50 ns;

Sel <= "110";

WAIT FOR 50 ns;

end process;

end Behavioral;

* Register Bank

Design Source File

entity RegisterBank is

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

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end RegisterBank;

architecture Behavioral of RegisterBank is

component D\_FF\_4bit

port (

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

Res : in STD\_LOGIC;

En : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end component;

component Slow\_Clk is

Port ( Clk\_in : in STD\_LOGIC;

Clk\_out : out STD\_LOGIC);

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 T : STD\_LOGIC\_VECTOR(7 downto 0);

signal Clk\_slow : STD\_LOGIC;

begin

Slow\_Clk\_0 : Slow\_Clk

port map(

Clk\_in => Clk,

Clk\_out => Clk\_slow );

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

port map(

I => RegSel,

EN => '1',

Y => T );

R0 <= "0000";

R\_1 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(1),

Clk => Clk\_slow,

Q => R1 );

R\_2 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(2),

Clk => Clk\_slow,

Q => R2 );

R\_3 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(3),

Clk => Clk\_slow,

Q => R3 );

R\_4 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(4),

Clk => Clk\_slow,

Q => R4 );

R\_5 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(5),

Clk => Clk\_slow,

Q => R5 );

R\_6 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(6),

Clk => Clk\_slow,

Q => R6 );

R\_7 : D\_FF\_4bit

port map (

D => D,

Res => Res,

En => T(7),

Clk => Clk\_slow,

Q => R7 );

end Behavioral;

Test Bench File

entity RegisterBankSim is

-- Port ( );

end RegisterBankSim;

architecture Behavioral of RegisterBankSim is

COMPONENT RegisterBank

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

RegSel : IN STD\_LOGIC\_VECTOR(2 downto 0);

Res, Clk : IN STD\_LOGIC;

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

END COMPONENT;

SIGNAL D : std\_logic\_vector(3 downto 0);

SIGNAL RegSel : std\_logic\_vector(2 downto 0);

SIGNAL Res, Clk : std\_logic;

SIGNAL R0, R1, R2, R3, R4, R5, R6, R7 :std\_logic\_vector(3 downto 0) := "0000";

begin

UUT : RegisterBank PORT MAP(

D => D,

RegSel => RegSel,

Res => Res,

Clk => Clk,

R0 => R0,

R1 => R1,

R2 => R2,

R3 => R3,

R4 => R4,

R5 => R5,

R6 => R6,

R7 => R7

);

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

--INPUT SIGNAL = 100MHz (10 ns)

--SLOW CLOCK COUNT = 500 000 000 == 50\_000\_000

--SLOW CLOCK SIM = WAIT FOR 1 ns

--OUTPUT SIGNAL = 0.2Hz (2 s)

process

begin

Res <= '1';

WAIT FOR 5 ms;

Res <= '0';

D <= "0001";

RegSel <= "000";

WAIT FOR 5 ms;

RegSel <= "001";

WAIT FOR 5 ms;

RegSel <= "010";

WAIT FOR 5 ms;

RegSel <= "011";

WAIT FOR 5 ms;

RegSel <= "100";

WAIT FOR 5 ms;

RegSel <= "101";

WAIT FOR 5 ms;

RegSel <= "110";

WAIT FOR 5 ms;

RegSel <= "111";

WAIT FOR 5 ms;

Res <= '1';

RegSel <= "000";

WAIT FOR 5 ms;

RegSel <= "001";

WAIT FOR 5 ms;

RegSel <= "010";

WAIT FOR 5 ms;

RegSel <= "011";

WAIT FOR 5 ms;

RegSel <= "100";

WAIT FOR 5 ms;

RegSel <= "101";

WAIT FOR 5 ms;

RegSel <= "110";

WAIT FOR 5 ms;

RegSel <= "111";

WAIT FOR 5 ms;

-- Index Number: 210518H => 11 0011 0110 0101 0110

-- Index Number: 210170G => 11 0011 0100 1111 1010

Res <= '0';

D <= "0110";

RegSel <= "000";

WAIT FOR 5 ms;

D <= "0101";

RegSel <= "001";

WAIT FOR 5 ms;

D <= "0110";

RegSel <= "010";

WAIT FOR 5 ms;

D <= "0011";

RegSel <= "011";

WAIT FOR 5 ms;

Res <= '0';

D <= "1010";

RegSel <= "100";

WAIT FOR 5 ms;

D <= "1111";

RegSel <= "101";

WAIT FOR 5 ms;

D <= "0100";

RegSel <= "110";

WAIT FOR 5 ms;

D <= "0011";

RegSel <= "111";

WAIT;

end process;

end Behavioral;

* Program ROM

Design Source File

entity ProgramROM is

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

I : 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 program\_ROM : rom\_type := (

"101110000000", --0

"100010000011", --1

"100100001111", --2

"001110010000", --3

"000010100000", --4

"110010000111", --5

"110000000011", --6

"100000000000" --7

);

begin

I <= program\_ROM(to\_integer(unsigned(Sel)));

end Behavioral;

Test Bench File

entity ProgramROMSim is

-- Port ( );

end ProgramROMSim;

architecture Behavioral of ProgramROMSim is

COMPONENT ProgramROM

PORT( Sel : IN STD\_LOGIC\_VECTOR(2 downto 0);

I : OUT STD\_LOGIC\_VECTOR(11 DOWNTO 0));

END COMPONENT;

SIGNAL Sel : std\_logic\_vector(2 downto 0);

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

begin

UUT : ProgramROM PORT MAP(

Sel => Sel,

I => I );

process

begin

-- Index Number: 210518H => 110 011 011 001 010 110

Sel <= "110";

WAIT FOR 5 ms;

Sel <= "010";

WAIT FOR 5 ms;

Sel <= "001";

WAIT FOR 5 ms;

Sel <= "011";

WAIT FOR 5 ms;

-- Index Number: 210170G => 110 011 010 011 111 010

Sel <= "010";

WAIT FOR 5 ms;

Sel <= "111";

WAIT FOR 5 ms;

Sel <= "011";

WAIT FOR 5 ms;

Sel <= "010";

WAIT FOR 5 ms;

end process;

end Behavioral;

* Instruction Decoder

Design Source File

entity InstructionDecoder is

Port ( JumpCheck : in STD\_LOGIC;

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

Reg1, Reg2, Jump : out STD\_LOGIC\_VECTOR(2 downto 0);

LoadSel, OpSel, Jflag : out STD\_LOGIC;

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

end InstructionDecoder;

architecture Behavioral of InstructionDecoder is

signal Op : std\_logic\_vector(1 downto 0);

begin

Op <= A(11 downto 10);

Reg1 <= A(9 downto 7);

Reg2 <= A(6 downto 4);

Val <= A(3 downto 0);

Jump <= A(2 downto 0);

process (Op,JumpCheck)

begin

case Op is

when "00" => OpSel <= '0'; LoadSel <= '0'; Jflag <= '0';

when "01" => OpSel <= '1'; LoadSel <= '0'; Jflag <= '0';

when "10" => LoadSel <= '1'; OpSel <= '0'; Jflag <= '0';

when "11" =>

case JumpCheck is

when '1' => Jflag <= '1'; OpSel <= '0'; LoadSel <= '0';

when others => Jflag <= '0'; OpSel <= '0'; LoadSel <= '0';

end case;

when others => Jflag <= '0'; OpSel <= '0'; LoadSel <= '0';

end case;

end process;

end Behavioral;

Test Bench File

entity InstructionDecoderSim is

-- Port ( );

end InstructionDecoderSim;

architecture Behavioral of InstructionDecoderSim is

COMPONENT InstructionDecoder

PORT( JumpCheck : in STD\_LOGIC;

A : IN STD\_LOGIC\_VECTOR(11 downto 0);

Reg1, Reg2, Jump : OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0);

LoadSel, OpSel, Jflag : out STD\_LOGIC;

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

END COMPONENT;

SIGNAL JumpCheck : std\_logic;

SIGNAL A : std\_logic\_vector(11 downto 0);

SIGNAL Reg1, Reg2, Jump : std\_logic\_vector(2 downto 0);

SIGNAL LoadSel, OpSel, Jflag : std\_logic;

SIGNAL Val : std\_logic\_vector(3 downto 0);

begin

UUT : InstructionDecoder PORT MAP(

JumpCheck => JumpCheck,

A => A,

Reg1 => Reg1,

Reg2 => Reg2,

Jump => Jump,

LoadSel => LoadSel,

OpSel => OpSel,

Jflag => Jflag,

Val => Val );

process

begin

-- Index Number: 210518H => 11 0011 0110 0101 0110

-- Index Number: 210170G => 11 0011 0100 1111 1010

JumpCheck <= '0';

A <= "011001010110";

WAIT FOR 5 ms;

A <= "001101100101";

WAIT FOR 5 ms;

A <= "110011011001";

WAIT FOR 5 ms;

JumpCheck <= '1';

WAIT FOR 5 ms;

A <= "010011111010";

WAIT FOR 5 ms;

A <= "001101001111";

WAIT FOR 5 ms;

A <= "110011010011";

WAIT FOR 5 ms;

JumpCheck <= '0';

WAIT FOR 5 ms;

end process;

end Behavioral;

* Seven-Segment ROM (LUT)

Design Source File

entity LUT is

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

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

end LUT;

architecture Behavioral of LUT 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;

* Seven-Segment Display

Design Source File

entity Display is

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

S\_7Seg : out STD\_LOGIC\_VECTOR (6 downto 0));

end Display;

architecture Behavioral of Display is

component LUT is

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

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

end component;

begin

LUT\_1\_0 : LUT

port map(

address => A,

data => S\_7Seg );

end Behavioral;

Test Bench File

entity DisplaySim is

-- Port ( );

end DisplaySim;

architecture Behavioral of DisplaySim is

component Display is

port (

A : in std\_logic\_vector(3 downto 0);

S\_7Seg : out std\_logic\_vector(6 downto 0));

end component;

signal A : std\_logic\_vector(3 downto 0) := "0000";

signal S\_7Seg : std\_logic\_vector(6 downto 0);

begin

UUT : Display PORT MAP (

A => A,

S\_7Seg => S\_7Seg

);

process

begin

A <= "0000";

wait for 5 ns;

A <= "0001";

wait for 5 ns;

A <= "0010";

wait for 5 ns;

A <= "0011";

wait for 5 ns;

A <= "0100";

wait for 5 ns;

A <= "0101";

wait for 5 ns;

A <= "0110";

wait for 5 ns;

A <= "0111";

wait for 5 ns;

A <= "1000";

wait for 5 ns;

A <= "1001";

wait for 5 ns;

A <= "1010";

wait for 5 ns;

A <= "1011";

wait for 5 ns;

A <= "1100";

wait for 5 ns;

A <= "1101";

wait for 5 ns;

A <= "1110";

wait for 5 ns;

A <= "1111";

wait;

wait;

end process;

end Behavioral;

* Slow Clock

Design Source File

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 = 50\_000\_000) then

clk\_status <= not clk\_status;

Clk\_out <= clk\_status;

count <= 1;

end if;

end if;

end process;

end Behavioral;

* Micro Processor

Design Source File

entity MicroProcessor is

Port ( Res, Clk : in STD\_LOGIC;

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

Zero, Overflow : out STD\_LOGIC;

S\_7Seg : out STD\_LOGIC\_VECTOR (6 downto 0));

end MicroProcessor;

architecture Behavioral of MicroProcessor is

component InstructionDecoder

Port ( JumpCheck : in STD\_LOGIC;

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

Reg1, Reg2, Jump : out STD\_LOGIC\_VECTOR(2 downto 0);

LoadSel, OpSel, Jflag : out STD\_LOGIC;

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

end component;

component ProgramROM

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

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

end component;

component ProgramCounter

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

Res, Clk : in STD\_LOGIC;

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

end component;

component Adder3Bit

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

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

end component;

component Mux\_2way\_3bit

Port ( Sel,Clk : in STD\_LOGIC;

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

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

end component;

component Mux\_2way\_4bit

Port ( Sel, Clk : in STD\_LOGIC;

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

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

end component;

component Mux\_8way\_4bit

Port ( Clk : in STD\_LOGIC;

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

A,B,C,D,E,F,G,H : in STD\_LOGIC\_VECTOR (3 downto 0);

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

end component;

component RegisterBank

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

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

Res : in STD\_LOGIC;

Clk : in STD\_LOGIC;

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

end component;

component AdderSubtracter

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

C\_in, Ctrl : in STD\_LOGIC;

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

C\_out, Zero, Overflow : out STD\_LOGIC);

end component;

component Display

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

S\_7Seg : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

SIGNAL Jflag, JumpCheck, LoadSel, OpSel : std\_logic;

-- Jflag - this signal indicates whether a jump is called

-- JumpCheck - this signal checks whether register is empty to jump

-- LoadSel - this signal is sent to the 2 way 4 bit mux as the selecter

-- OpSel - this signal is sent to the Adder Subtracter unit to select the operation

SIGNAL C\_out : std\_logic;

-- C\_out - this is a internal signal generated by the Adder Subtracter unit indicating whether there is carry out

SIGNAL T, R, R0, R1, R2, R3, R4, R5, R6, R7, M1, M2, D : std\_logic\_vector(3 downto 0);

-- T - this is 4 bit bus connecting Instruction Decoder to 2 way 4 bit mux

-- R - this is 4 bit bus connecting 2 way 4 bit mux to Register Bank

-- R(0-7) - these are 4 bit busses connecting the individual registers to the 8 way 4 bit mux

-- M1,M2 - these are 4 bit buses connecting the two 8 way 4 bit mux to the Adder Subtracter unit

-- D - this is a 4 bit bus connecting the output of Adder Subtracter unit to 2 way 4 bit mux

SIGNAL Sel,P, A, J, RS1, RS2 : std\_logic\_vector(2 downto 0);

-- Sel - this is a 3 bit bus connecting Program Counter to Program ROM and 3 Bit Adder

-- P - this is a 3 bit bus connecting 2 way 3 bit mux to the Program Counter

-- A - this is a 3 bit bus connecting 3 bit Adder to 2 way 3 bit mux

-- J - this is a 3 bit bus connecting Instruction Decoder to 2 way 3 bit mux

-- RS1, RS2 - these are signals sent by Instruction Decoder to the two 8 way 4 bit mux as selecters

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

-- I - this is a 12 bit bus connecting Program ROM to Instruction Decoder

begin

IntructionDecoder\_0 : InstructionDecoder

PORT MAP(

JumpCheck => JumpCheck,

A => I,

Reg1 => RS1,

Reg2 => RS2,

Jump => J,

LoadSel => LoadSel,

OpSel => OpSel,

Jflag => Jflag,

Val => T);

ProgramROM\_0 : ProgramROM

PORT MAP(

Sel => Sel,

I => I);

ProgramCounter\_0 : ProgramCounter

PORT MAP(

D => P,

Res => Res,

Clk => Clk,

Q => Sel);

Adder3Bit\_0 : Adder3Bit

PORT MAP(

A => Sel,

S => A);

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

PORT MAP(

Sel => Jflag,

Clk => Clk,

A => A,

B => J,

Y => P);

Mux\_2way\_4bit\_0 : Mux\_2way\_4bit

PORT MAP(

Sel => LoadSel,

Clk => Clk,

A => D,

B => T,

Y => R);

RegisterBank\_0 : RegisterBank

PORT MAP(

D => R,

RegSel => RS1,

Res => Res,

Clk => Clk,

R0 => R0,

R1 => R1,

R2 => R2,

R3 => R3,

R4 => R4,

R5 => R5,

R6 => R6,

R7 => R7 );

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

PORT MAP(

Sel => RS1,

Clk => Clk,

A => R0,

B => R1,

C => R2,

D => R3,

E => R4,

F => R5,

G => R6,

H => R7,

Y => M1);

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

PORT MAP(

Sel => RS2,

Clk => Clk,

A => R0,

B => R1,

C => R2,

D => R3,

E => R4,

F => R5,

G => R6,

H => R7,

Y => M2);

AdderSubtracter\_0 : AdderSubtracter

PORT MAP(

A => M2,

B => M1,

C\_in => '0',

Ctrl => OpSel,

S => D,

C\_out => C\_out,

Zero => Zero,

Overflow => Overflow);

Display\_0 : Display

PORT MAP(

A => R7,

S\_7Seg => S\_7Seg);

process(Clk,Sel)

begin

if (rising\_edge(Clk)) then

case M1 is

when "0000" => JumpCheck <= '1';

when others => Jumpcheck <= '0';

end case;

end if;

end process;

S <= R7;

end Behavioral;

Test Bench File

entity MicroProcessorSim is

-- Port ( );

end MicroProcessorSim;

architecture Behavioral of MicroProcessorSim is

COMPONENT MicroProcessor

PORT( Res,Clk : in STD\_LOGIC;

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

Zero, Overflow : out STD\_LOGIC;

S\_7Seg : out STD\_LOGIC\_VECTOR (6 downto 0));

END COMPONENT;

SIGNAL Res, Clk : std\_logic;

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

SIGNAL Zero, Overflow : std\_logic;

SIGNAL S\_7Seg : std\_logic\_vector(6 downto 0);

begin

UUT : MicroProcessor PORT MAP(

Res => Res,

Clk => Clk,

S => S,

Zero => Zero,

Overflow =>Overflow,

S\_7Seg => S\_7Seg);

Clock : process

begin

Clk <= '0';

WAIT FOR 1 ns;

Clk <= '1';

WAIT FOR 1 ns;

end process;

process

begin

Res <= '1';

WAIT FOR 2 ms;

Res <= '0';

WAIT FOR 40 ms;

Res <= '1';

WAIT FOR 5 ms;

Res <= '0';

WAIT FOR 40 ms;

WAIT;

end process;

end Behavioral;

Design Constraint File

## Clock signal

set\_property PACKAGE\_PIN W5 [get\_ports Clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports Clk]

create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports Clk]

##Buttons

set\_property PACKAGE\_PIN U18 [get\_ports Res]

set\_property IOSTANDARD LVCMOS33 [get\_ports Res]

## LEDs

set\_property PACKAGE\_PIN U16 [get\_ports {S[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S[0]}]

set\_property PACKAGE\_PIN E19 [get\_ports {S[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S[1]}]

set\_property PACKAGE\_PIN U19 [get\_ports {S[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S[2]}]

set\_property PACKAGE\_PIN V19 [get\_ports {S[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S[3]}]

set\_property PACKAGE\_PIN P1 [get\_ports {Zero}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Zero}]

set\_property PACKAGE\_PIN L1 [get\_ports {Overflow}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {Overflow}]

##7 segment display

set\_property PACKAGE\_PIN W7 [get\_ports {S\_7Seg[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[0]}]

set\_property PACKAGE\_PIN W6 [get\_ports {S\_7Seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[1]}]

set\_property PACKAGE\_PIN U8 [get\_ports {S\_7Seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[2]}]

set\_property PACKAGE\_PIN V8 [get\_ports {S\_7Seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[3]}]

set\_property PACKAGE\_PIN U5 [get\_ports {S\_7Seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[4]}]

set\_property PACKAGE\_PIN V5 [get\_ports {S\_7Seg[5]}]

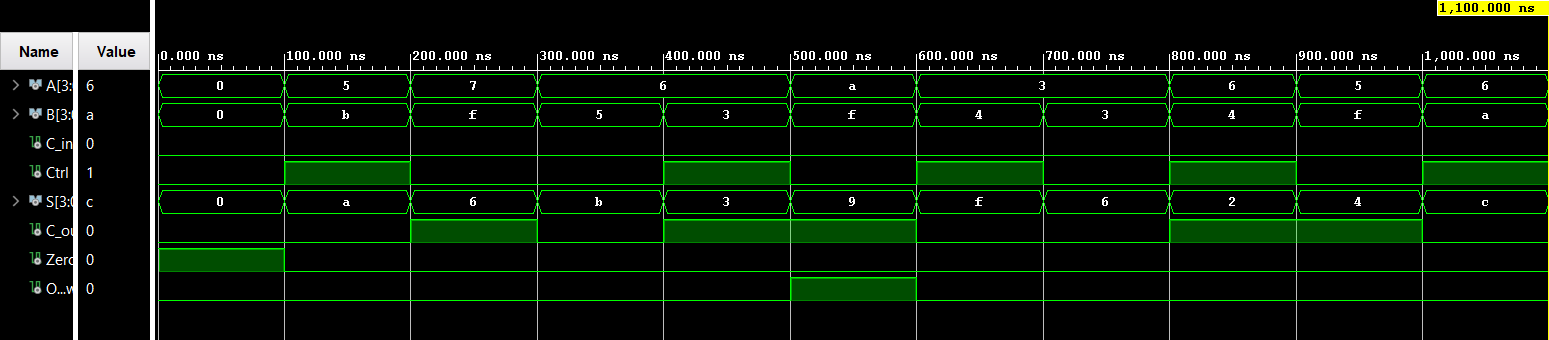
set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[5]}]

set\_property PACKAGE\_PIN U7 [get\_ports {S\_7Seg[6]}]

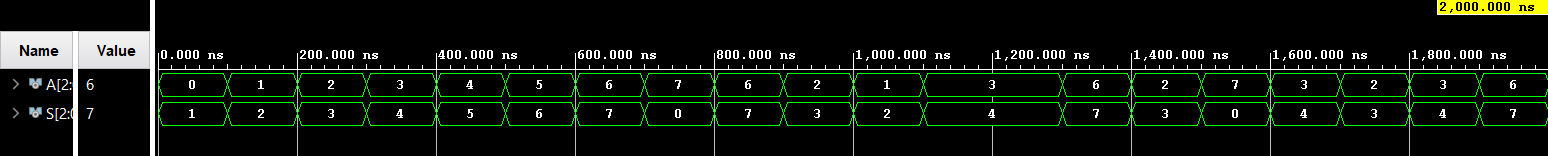
set\_property IOSTANDARD LVCMOS33 [get\_ports {S\_7Seg[6]}]

**TIMING DIAGRAMS**

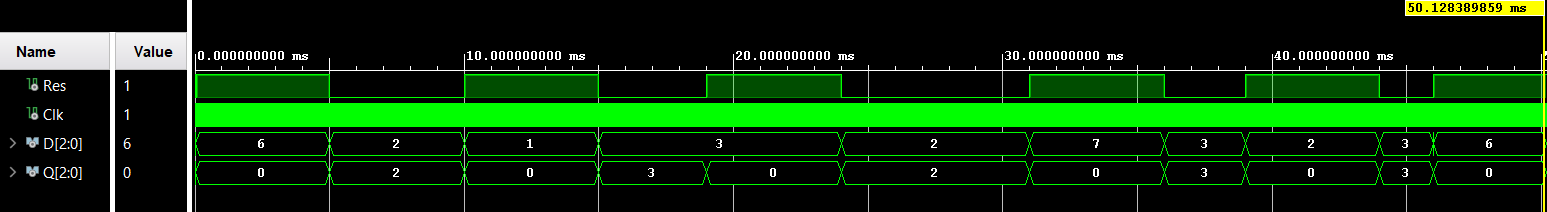
* 4-bit Add/Subtract unit



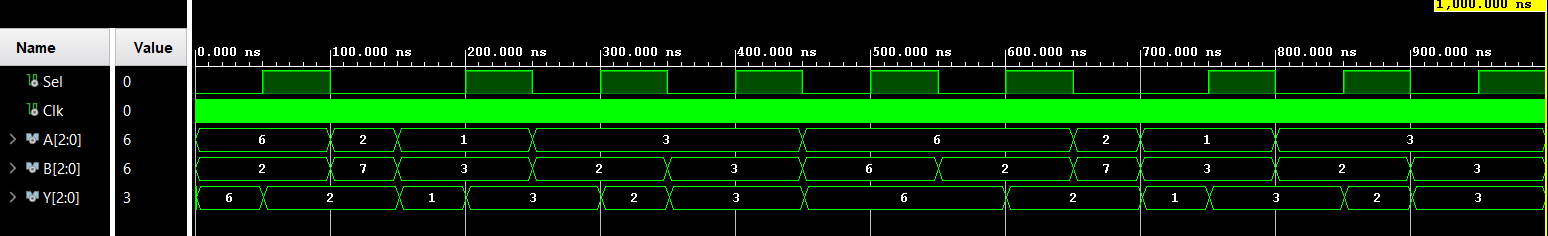
* 3-bit Adder



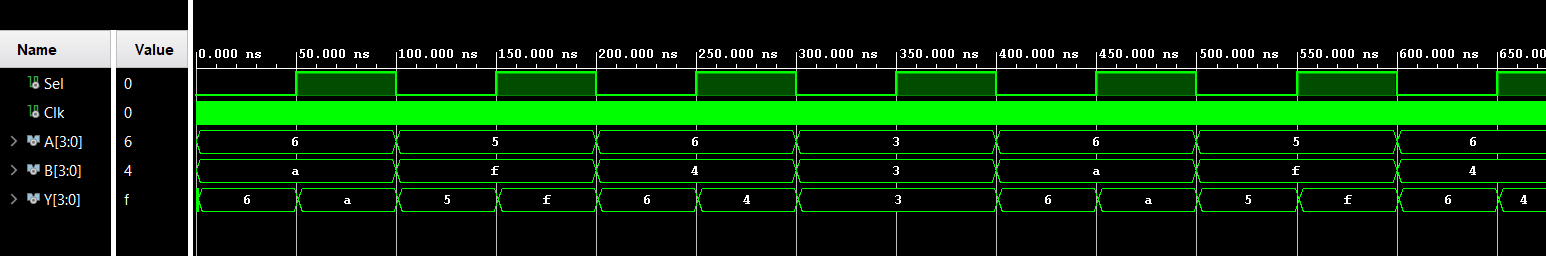
* 3-bit Program Counter (PC)



* 2-way 3-bit multiplexer



* 2-way 4-bit multiplexer

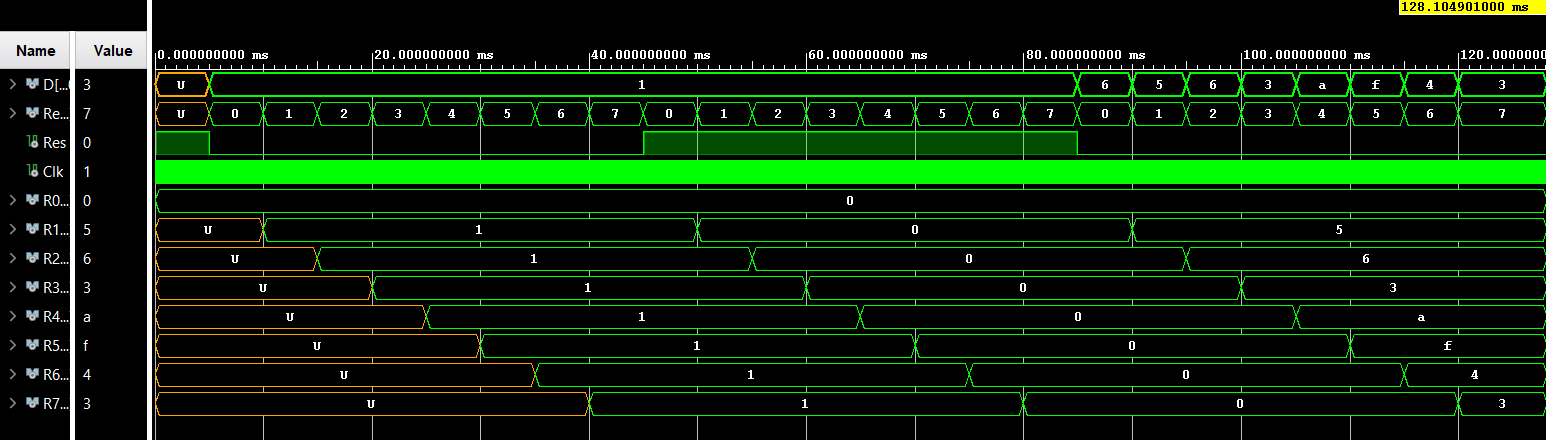
****

* 8-way 4-bit multiplexer

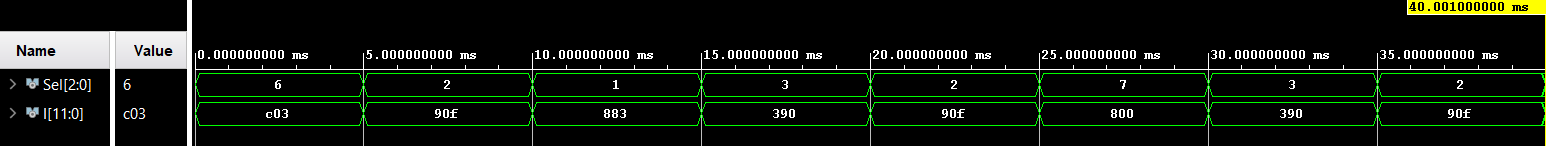
A screenshot of a computer

Description automatically generated with low confidence

* Register Bank



* Program ROM

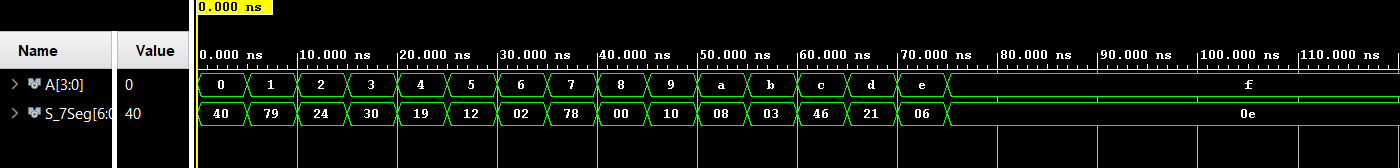


* Instruction Decoder

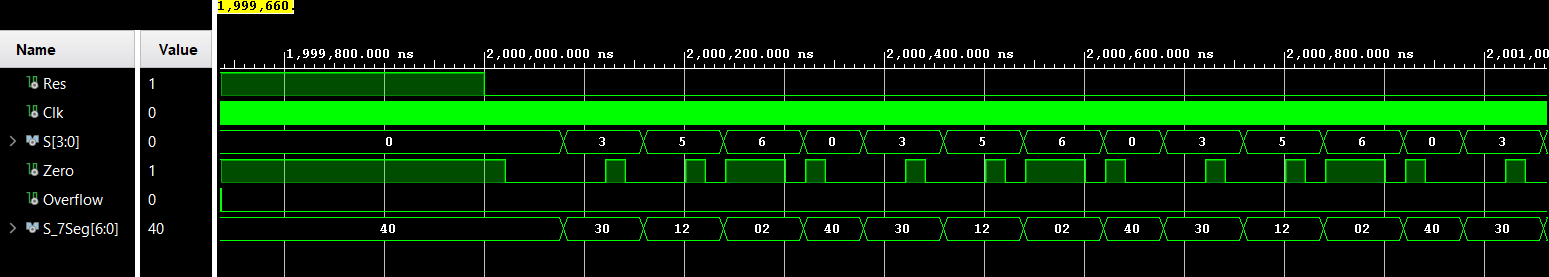
A picture containing screenshot, colorfulness

Description automatically generated

* Seven-Segment Display



* Micro Processor



**CONCLUSION**

* In the implementation of the 4 bit Add/Subtract unit, we can use an RCA with the input in two’s complement form.
* When the subtract operation is called, invert the input that is to be subtracted and make the carry in signal into the first full adder ‘1’.
* 3 bit adder is implemented with only one 3 bit input because the purpose of the 3 bit adder is to increment the input by one. So the other input is set to ‘1’.
* 3 bit program counter has the duty of slowing the input signal and sending it out. In this case the signal is slowed by a factor of 50 million. It waits for 50 million clock cycles before sending the output signal.
* The three different multiplexers had to be clock synchronized, otherwise the output signal won’t change during the instruction cycle.
* Unlike the other components without a clock connection like the instruction decoder or the adders where the inputs constantly change from one instruction to another, mux will have the same inputs until it passes through the signals. Hence they have to be clock synchronized.
* The register bank is implemented using d flip flops and R0 is hard coded to zero. The zero signal stored in R0 is used for the jump instructions.
* Program ROM was implemented by hardcoding the set of instructions. The ROM is limited to 8 instructions. The final instruction is kept empty to jump out of the internal loop when the conditions are met.
* A variety of buses had to be used when connecting the individual components to create the nano processor.
* The instruction decoder has two input signals and a lot of output signals. The inputs are the instruction from the ROM and the jump check signal.
* The jump check signal becomes one when the output of the first 8 way 4 bit mux becomes zero.
* This coupled with the instruction calling a jump, would raise the jump flag which is connected to the 2 way 3 bit mux which would then jump to the number specified in the instruction.
* The instruction decoder decides which registers to be selected when add/subtract operations are called.
* Similarly it decides which register to move a particular value to.
* Finally the nano processor component which is the combination of all the other components has two inputs as clock and the reset button.
* The reset button sets the program counter as well as all the registers in the register bank to zero.
* There are two outputs from the add/subtract unit denoting zero output and negative overflow.
* During the addition process in the add/subtract unit if two negative numbers are added which sums to more than negative eight, then the overflow bit becomes ‘1’.
* There is a output connected to the register R7. Which has a seven segment display also connected to it. 7 segment uses hexa-decimal representation.

**CONTRIBUTION BY MEMBERS**

**210518H - K.S. RANASINGHE**

* 4 -bit Add/Subtract unit, 3-bit Adder, Program Counter, Instruction Decoder.

**210170G - W.A.R.T FONSEKA**

* All 3 mux components, Assembly code and Program ROM, Register Bank, 7-segment display.

Assembling of the components together and creating the nano processor was done by both of us together.

Basys simulation and corrections were also done together.

Formation of this lab report was also a joint work.

No. of hours spent was not measured. Given below is an estimate.

* K.S. RANASINGHE ( 210518H ) = 5-6 hours
* W.A.R.T FONSEKA ( 210170G ) = 5-6 hours