Datapath – Vhdl – Part B

Niall Martin Ryan - 14315886

Datapath.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity DataPath is

port(

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

load\_enable : in std\_logic;

dataS : in std\_logic;

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

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

instruction\_MB : in std\_logic\_vector(15 downto 0);

INL : in std\_logic;

INR : in std\_logic;

Data\_in : in std\_logic\_vector(15 downto 0);

Clk : in std\_logic;

fselect : in std\_logic\_vector(4 downto 0);

MB\_sel : in std\_logic;

MD\_sel : in std\_logic;

v : out std\_logic;

c : out std\_logic;

n : out std\_logic;

z : out std\_logic;

Adrs\_out : out std\_logic\_vector(15 downto 0);

Data\_out\_bus\_b : out std\_logic\_vector(15 downto 0);

Control\_word : in std\_logic\_vector(16 downto 0)

);

end DataPath;

architecture Behavioral of DataPath is

-- signals

signal reg0\_q, reg1\_q, reg2\_q, reg3\_q, reg4\_q, reg5\_q, reg6\_q, reg7\_q : std\_logic\_vector(15 downto 0);

signal A\_output, B\_output, Mb\_output, Md\_output, functional\_unit\_output : std\_logic\_vector(15 downto 0);

signal A\_address\_out, B\_data\_out: std\_logic\_vector(15 downto 0);

-- control word signal

signal control\_word\_transfer : std\_logic\_vector (16 downto 0);

component Reg\_File port(

d0 : in STD\_LOGIC;

d1 : in STD\_LOGIC;

d2 : in STD\_LOGIC;

m0 : in STD\_LOGIC;

m1 : in STD\_LOGIC;

m2 : in STD\_LOGIC;

m3 : in STD\_LOGIC;

m4 : in STD\_LOGIC;

m5 : in STD\_LOGIC;

data : in STD\_LOGIC\_VECTOR(15 DOWNTO 0);

dataS : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Data\_into\_reg : in STD\_LOGIC\_VECTOR(15 downto 0);

src\_output\_a : out std\_logic\_vector(15 downto 0);

src\_output\_b : out std\_logic\_vector(15 downto 0);

R0 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R1 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R2 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R3 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R4 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R5 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R6 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R7 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0));

end component;

component Mux\_2\_1 port(

sel : in std\_logic;

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

g : out std\_logic\_vector(15 downto 0)

);

end component;

component Functional\_Unit port(

fsel : in std\_logic\_vector(4 downto 0);

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

IL : in std\_logic;

IR : in std\_logic;

f : out std\_logic\_vector(15 downto 0);

v : out std\_logic;

c : out std\_logic;

n : out std\_logic;

z : out std\_logic

);

end component;

begin

control\_word\_transfer <= Control\_word;

control\_word\_transfer(16) <= des\_sel(2);

control\_word\_transfer(15) <= des\_sel(1);

control\_word\_transfer(14) <= des\_sel(0);

control\_word\_transfer(13) <= A\_sel(2);

control\_word\_transfer(12) <= A\_sel(1);

control\_word\_transfer(11) <= A\_sel(0);

control\_word\_transfer(10) <= B\_sel(2);

control\_word\_transfer(9) <= B\_sel(1);

control\_word\_transfer(8) <= B\_sel(0);

control\_word\_transfer(7) <= MB\_sel;

control\_word\_transfer(6) <= fselect(4);

control\_word\_transfer(5) <= fselect(3);

control\_word\_transfer(4) <= fselect(2);

control\_word\_transfer(3) <= fselect(1);

control\_word\_transfer(2) <= fselect(0);

control\_word\_transfer(1) <= MD\_sel;

control\_word\_transfer(0) <= dataS;

functional : Functional\_Unit port map(

fsel => fselect,

a => A\_output,

b => Mb\_output,

f => functional\_unit\_output,

IR => INL,

IL => INR,

v => v,

c => c,

n => n,

z => z

);

reg : Reg\_File port map(

d0 => des\_sel(0),

d1 => des\_sel(1),

d2 => des\_sel(2),

m0 => A\_sel(0),

m1 => A\_sel(1),

m2 => A\_sel(2),

m3 => B\_sel(0),

m4 => B\_sel(1),

m5 => B\_sel(2),

Clk => Clk,

Data\_into\_reg => Md\_output,

src\_output\_a => A\_output ,

src\_output\_b => B\_output ,

R0 => reg0\_q,

R1 => reg1\_q,

R2 => reg2\_q,

R3 => reg3\_q,

R4 => reg4\_q,

R5 => reg5\_q,

R6 => reg6\_q,

R7 => reg7\_q

);

MB\_select : Mux\_2\_1 port map(

a => B\_output,

b => instruction\_MB,

sel => Mb\_sel,

g => Mb\_output

);

MD\_select : Mux\_2\_1 port map(

a => functional\_unit\_output,

b => data\_in,

sel => MD\_sel,

g => Md\_output

);

Adrs\_out <= A\_output;

Data\_out\_bus\_b <= B\_output;

end Behavioral;

Functional\_Unit.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Functional\_Unit is

port(

fsel : in std\_logic\_vector(4 downto 0);

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

f : out std\_logic\_vector(15 downto 0);

IL : in std\_logic;

IR : in std\_logic;

v : out std\_logic;

c : out std\_logic;

n : out std\_logic;

z : out std\_logic

);

end Functional\_Unit;

architecture Behavioral of Functional\_Unit is

-- signals

--input

signal A\_input, B\_input, B\_shifter, ALU\_output, Shifter\_output, zero\_detect\_output : std\_logic\_vector(15 downto 0);

signal Shift\_left\_input, Shift\_right\_input, Shift\_left\_output, Shift\_right\_output, z\_flag : std\_logic;

--output

signal OUTPUT : std\_logic\_vector(15 downto 0);

-- add components

component ALU port(

A, B : in std\_logic\_vector(15 downto 0);

S1 : in std\_logic;

S2 : in std\_logic;

S3 : in std\_logic;

Cin : in std\_logic;

G : out std\_logic\_vector(15 downto 0);

c : out std\_logic

);

end component;

component Shifter port(

IL : in std\_logic;

IR : in std\_logic;

s : in std\_logic\_vector(1 downto 0);

b : in std\_logic\_vector (15 downto 0);

O : out std\_logic\_vector(15 downto 0);

SL0 : out std\_logic;

SR0 : out std\_logic

);

end component;

component zeroDetect port(

input : in std\_logic\_vector(15 downto 0);

outputFlag : out std\_logic;

output : out std\_logic\_vector(15 downto 0)

);

end component;

component Mux\_2\_1 port(

sel : in std\_logic;

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

g : out std\_logic\_vector(15 downto 0)

);

end component;

begin

Shift\_left\_input <= IR;

Shift\_right\_input <= IL;

ariLogUnit : ALU

port map(

A => A,

B => A,

S3 => fsel(3),

S2 => fsel(2),

S1 => fsel(1),

Cin => fsel(0),

G => ALU\_output,

c => c

);

shift : Shifter

port map(

IL => Shift\_left\_input,

IR => Shift\_right\_input,

S(1) => fsel(3),

S(0) => fsel(2),

b => b,

O => Shifter\_output,

SL0 => Shift\_left\_output,

SR0 => Shift\_right\_output

);

detect\_Zero : zeroDetect

port map(

input => ALU\_output,

outputFlag => z,

output => zero\_detect\_output

);

MF\_Select : Mux\_2\_1

port map(

sel => fsel(4),

a => zero\_detect\_output,

b => Shifter\_output,

g => f

);

end Behavioral;

Reg\_File.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Reg\_File is

Port (

d0 : in STD\_LOGIC;

d1 : in STD\_LOGIC;

d2 : in STD\_LOGIC;

m0 : in STD\_LOGIC;

m1 : in STD\_LOGIC;

m2 : in STD\_LOGIC;

m3 : in STD\_LOGIC;

m4 : in STD\_LOGIC;

m5 : in STD\_LOGIC;

data : in STD\_LOGIC\_VECTOR(15 DOWNTO 0);

dataS : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Data\_into\_reg : in STD\_LOGIC\_VECTOR(15 downto 0);

src\_output\_a : out std\_logic\_vector(15 downto 0);

src\_output\_b : out std\_logic\_vector(15 downto 0);

R0 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R1 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R2 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R3 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R4 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R5 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R6 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R7 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0));

end Reg\_File;

architecture Behavioral of Reg\_File is

-- Components

-- reg16

COMPONENT reg16

PORT(

D: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

load: IN STD\_LOGIC;

Clk: IN STD\_LOGIC;

Q: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0)

);

END COMPONENT;

--3 to 8 decoder

COMPONENT Decoder38\_16bit

PORT(

S0: IN STD\_LOGIC;

S1: IN STD\_LOGIC;

S2: IN STD\_LOGIC;

dR0: OUT STD\_LOGIC;

dR1: OUT STD\_LOGIC;

dR2: OUT STD\_LOGIC;

dR3: OUT STD\_LOGIC;

dR4: OUT STD\_LOGIC;

dR5: OUT STD\_LOGIC;

dR6: OUT STD\_LOGIC;

dR7: OUT STD\_LOGIC

);

END COMPONENT;

--8in 16 bit mux

COMPONENT Mux8in\_16bit

PORT(

S0: IN STD\_LOGIC;

S1: IN STD\_LOGIC;

S2: IN STD\_LOGIC;

R0: IN STD\_LOGIC\_VECTOR(15 downto 0);

R1: IN STD\_LOGIC\_VECTOR(15 downto 0);

R2: IN STD\_LOGIC\_VECTOR(15 downto 0);

R3: IN STD\_LOGIC\_VECTOR(15 downto 0);

R4: IN STD\_LOGIC\_VECTOR(15 downto 0);

R5: IN STD\_LOGIC\_VECTOR(15 downto 0);

R6: IN STD\_LOGIC\_VECTOR(15 downto 0);

R7: IN STD\_LOGIC\_VECTOR(15 downto 0);

OUTPUT: OUT STD\_LOGIC\_VECTOR(15 downto 0)

);

END COMPONENT;

--2 to 1 line multiplexer 16 bit

COMPONENT Small\_Mux

PORT(

In0: IN STD\_LOGIC\_VECTOR(15 downto 0);

In1: IN STD\_LOGIC\_VECTOR(15 downto 0);

S: IN STD\_LOGIC;

output: OUT STD\_LOGIC\_VECTOR(15 downto 0)

);

END COMPONENT;

--signals

signal load\_reg0, load\_reg1, load\_reg2, load\_reg3, load\_reg4, load\_reg5,

load\_reg6, load\_reg7: STD\_LOGIC;

signal reg0\_q, reg1\_q, reg2\_q, reg3\_q, reg4\_q, reg5\_q, reg6\_q, reg7\_q,

data\_src\_mux\_out, src\_regA , src\_regB, Data\_into\_regput : STD\_LOGIC\_VECTOR(15 downto 0);

begin

--PORT MAPS

--register 0

reg00: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg0,

Clk => Clk,

Q => reg0\_q

);

--register 1

reg01: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg1,

Clk => Clk,

Q => reg1\_q

);

--register 2

reg02: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg2,

Clk => Clk,

Q => reg2\_q

);

--register 3

reg03: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg3,

Clk => Clk,

Q => reg3\_q

);

--register 4

reg04: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg4,

Clk => Clk,

Q => reg4\_q

);

--register 5

reg05: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg5,

Clk => Clk,

Q => reg5\_q

);

--register 6

reg06: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg6,

Clk => Clk,

Q => reg6\_q

);

--register 7

reg07: reg16 PORT MAP (

D => data\_src\_mux\_out,

load => load\_reg7,

Clk => Clk,

Q => reg7\_q

);

--Destination register decoder

-- will have to change this decoder..

des\_decoder\_3to8: Decoder38\_16bit PORT MAP (

S0 => d0,

S1 => d1,

S2 => d2,

dR0 => load\_reg0,

dR1 => load\_reg1,

dR2 => load\_reg2,

dR3 => load\_reg3,

dR4 => load\_reg4,

dR5 => load\_reg5,

dR6 => load\_reg6,

dR7 => load\_reg7

);

--2 to 1 Data Source Multiplexer

data\_src\_mux2: Small\_Mux PORT MAP (

In0 => data,

In1 => Data\_into\_reg,

S => dataS,

output => data\_src\_mux\_out

);

--8 to 1 source register multiplexer

src\_mux8: Mux8in\_16bit PORT MAP (

R0 => reg0\_q,

R1 => reg1\_q,

R2 => reg2\_q,

R3 => reg3\_q,

R4 => reg4\_q,

R5 => reg5\_q,

R6 => reg6\_q,

R7 => reg7\_q,

S0 => m0,

S1 => m1,

S2 => m2,

OUTPUT => src\_regA

);

src\_mux8\_B: Mux8in\_16bit PORT MAP (

R0 => reg0\_q,

R1 => reg1\_q,

R2 => reg2\_q,

R3 => reg3\_q,

R4 => reg4\_q,

R5 => reg5\_q,

R6 => reg6\_q,

R7 => reg7\_q,

S0 => m3,

S1 => m4,

S2 => m5,

OUTPUT => src\_regB

);

R0 <= reg0\_q;

R1 <= reg1\_q;

R2 <= reg2\_q;

R3 <= reg3\_q;

R4 <= reg4\_q;

R5 <= reg5\_q;

R6 <= reg6\_q;

R7 <= reg7\_q;

src\_output\_a <= src\_regA;

src\_output\_b <= src\_regB;

Data\_into\_regput <= Data\_into\_reg;

Data\_into\_regput <= data\_src\_mux\_out;

end Behavioral;

Mux\_2\_1.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Mux\_2\_1 is

port(

sel : in std\_logic;

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

g : out std\_logic\_vector(15 downto 0)

);

end Mux\_2\_1;

architecture Behavioral of Mux\_2\_1 is

begin

g <= a after 1 ns when sel='0' else

b after 1 ns when sel='1' else

"0000000000000000" after 1 ns;

end Behavioral;

ALU.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ALU is

port(

-- carry input??

A, B : in std\_logic\_vector(15 downto 0);

S1 : in std\_logic;

S2 : in std\_logic;

S3 : in std\_logic;

Cin : in std\_logic; -- fS0

G : out std\_logic\_vector(15 downto 0);

c : out std\_logic;

v : out std\_logic;

n : out std\_logic

);

end ALU;

architecture Behavioral of ALU is

-- signals input

signal arith\_out : std\_logic\_vector(15 downto 0);

signal logic\_out : std\_logic\_vector(15 downto 0);

-- signals output

signal output\_mux : std\_logic\_vector(15 downto 0);

signal output\_n\_flag : std\_logic;

component Arithmetic\_Circuit

port(

A : in std\_logic\_vector(15 downto 0);

B : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

Cin : in std\_logic;

Cout : out std\_logic;

G : out std\_logic\_vector(15 downto 0)

);

end component;

component logic\_circuit

port(

A : in std\_logic\_vector(15 downto 0);

B : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

G : out std\_logic\_vector(15 downto 0)

);

end component;

component Mux\_2\_1

port (

sel : in std\_logic;

A : in std\_logic\_vector(15 downto 0);

B : in std\_logic\_vector(15 downto 0);

G : out std\_logic\_vector(15 downto 0)

);

end component;

begin

ArithmeticCircuitry : Arithmetic\_Circuit

port map(

A => A,

B => B,

S0 => S1,

S1 => S2,

Cin => Cin,

Cout => c,

G => arith\_out

);

LogicCircuit : logic\_circuit

port map(

A => A,

B => B,

S0 => S1,

S1 => S2,

G => logic\_out

);

Mux2\_1 : Mux\_2\_1

port map(

sel => S3,

A => arith\_out,

B => logic\_out,

G => output\_mux

);

G <= output\_mux;

output\_n\_flag <= '1' after 5 ns when (output\_mux AND "1000000000000000") = "1000000000000000"

else '0' after 5 ns ;

n <= output\_n\_flag;

end Behavioral;

Shifter.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Shifter is

port(

IL : in std\_logic;

IR : in std\_logic;

s : in std\_logic\_vector(1 downto 0);

b : in std\_logic\_vector (15 downto 0);

O : out std\_logic\_vector(15 downto 0);

SL0 : out std\_logic;

SR0 : out std\_logic

);

end Shifter;

architecture Behavioral of Shifter is

component shifter\_mux

port(

A : in std\_logic;

B : in std\_logic;

C : in std\_logic;

S : in std\_logic\_vector(1 downto 0);

H : out std\_logic

);

end component;

-- output signal

signal S\_0, S\_1, S\_2, S\_3,S\_4,S\_5,S\_6,S\_7,S\_8,S\_9,S\_10,

S\_11,S\_12,S\_13,S\_14,S\_15 : STD\_LOGIC;

begin

shifterMux00: shifter\_mux Port map(

A => b(0),

B => b(1),

C => IL,

S => s,

H => S\_0

);

shifterMux01: shifter\_mux Port map(

A => b(1),

B => b(2),

C => b(0),

S => s,

H => S\_1

);

shifterMux02: shifter\_mux Port map(

A => b(2),

B => b(3),

C => b(1),

S => s,

H => S\_2

);

shifterMux03: shifter\_mux Port map(

A => b(3),

B => b(4),

C => b(2),

S => s,

H => S\_3

);

shifterMux04: shifter\_mux Port map(

A => b(4),

B => b(5),

C => b(3),

S => s,

H => S\_4

);

shifterMux05: shifter\_mux Port map(

A => b(5),

B => b(6),

C => b(4),

S => s,

H => S\_5

);

shifterMux06: shifter\_mux Port map(

A => b(6),

B => b(7),

C => b(5),

S => s,

H => S\_6

);

shifterMux07: shifter\_mux Port map(

A => b(7),

B => b(8),

C => b(6),

S => s,

H => S\_7

);

shifterMux08: shifter\_mux Port map(

A => b(8),

B => b(9),

C => b(7),

S => s,

H => S\_8

);

shifterMux09: shifter\_mux Port map(

A => b(9),

B => b(10),

C => b(8),

S => s,

H => S\_9

);

shifterMux10: shifter\_mux Port map(

A => b(10),

B => b(11),

C => b(9),

S =>s,

H => S\_10

);

shifterMux11: shifter\_mux Port map(

A => b(11),

B => b(12),

C => b(10),

S => s,

H => S\_11

);

shifterMux12: shifter\_mux Port map(

A => b(12),

B => b(13),

C => b(11),

S => s,

H => S\_12

);

shifterMux13: shifter\_mux Port map(

A => b(13),

B => b(14),

C => b(12),

S => s,

H => S\_13

);

shifterMux14: shifter\_mux Port map(

A => b(14),

B => b(15),

C => b(13),

S => s,

H => S\_14

);

shifterMux15: shifter\_mux Port map(

A => b(15),

B => IR,

C => b(14),

S => s,

H => S\_15

);

SL0 <= b(15);

SR0 <= b(0);

O(0) <= s\_0;

O(1) <= s\_1;

O(2) <= s\_2;

O(3) <= s\_3;

O(4) <= s\_4;

O(5) <= s\_5;

O(6) <= s\_6;

O(7) <= s\_7;

O(8) <= s\_8;

O(9) <= s\_9;

O(10) <= s\_10;

O(11) <= s\_11;

O(12) <= s\_12;

O(13) <= s\_13;

O(14) <= s\_14;

O(15) <= s\_15;

end Behavioral;

Detect\_zero.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity zeroDetect is

port(

input : IN std\_logic\_vector(15 downto 0);

outputFlag : OUT std\_logic;

output : out std\_logic\_vector(15 downto 0)

);

end zeroDetect;

architecture Behavioral of zeroDetect is

begin

outputFlag <= '1' after 5 ns when (input OR "0000000000000000")=

"0000000000000000" else

'0' after 5 ns;

output <= input;

end Behavioral;

Arithmetic\_circuit.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Arithmetic\_Circuit is

port(

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

Cin : in std\_logic;

Cout : out std\_logic;

g : out std\_logic\_vector(15 downto 0)

);

end Arithmetic\_Circuit;

architecture Behavioral of Arithmetic\_Circuit is

-- signals

signal result\_y : std\_logic\_vector(15 downto 0);

component B\_Input\_Logic

port(

b : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

y : out std\_logic\_vector(15 downto 0)

);

end component;

component RippleAdder

port(

B, A : in std\_logic\_vector(15 downto 0);

C0 : in std\_logic;

S : out std\_logic\_vector(15 downto 0);

C15 : out std\_logic

);

end component;

begin

inputLogicB : B\_Input\_Logic

port map(

b => b,

S0 => S0,

S1 => S1,

y => result\_y

);

parallelAdd : RippleAdder port map(

A => a,

B => result\_y,

C0 => Cin,

S => g,

C15 => Cout

);

end Behavioral;

Logic\_Circuit.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity logic\_Circuit is

port(

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

S1 : in std\_logic;

S0 : in std\_logic;

g : out std\_logic\_vector(15 downto 0)

);

end logic\_Circuit;

--output signal

architecture Behavioral of logic\_Circuit is

begin

g <= (a AND b) after 1 ns when S1 ='0' and S0 ='0' else

(a OR b) after 1 ns when S1 ='0' and S0 ='1' else

(a XOR b) after 1 ns when S1 ='1' and S0 ='0' else

(NOT a) after 1 ns when S1 ='1' and S0='1' else

"0000000000000000" after 1 ns ;

end Behavioral;

Shifter\_mux.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity shifter\_mux is

port(

A : in std\_logic;

B : in std\_logic;

C : in std\_logic;

S : in std\_logic\_vector(1 downto 0);

H : out std\_logic

);

end shifter\_mux;

architecture Behavioral of shifter\_mux is

begin

H <= A after 5 ns when S="00" else

B after 5 ns when S= "01" else

C after 5 ns when S= "10" else

'0' after 5 ns;

end Behavioral;

B\_input\_logic.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity B\_Input\_Logic is

port(

b : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

y : out std\_logic\_vector(15 downto 0)

);

end B\_Input\_Logic;

architecture Behavioral of B\_Input\_Logic is

begin

y <= "0000000000000000" after 1 ns when S1='0' and S0='0' else

b after 1 ns when S1='0' and S0='1' else

(Not b) after 1 ns when S1='1' and S0='0' else

"1111111111111111" after 1 ns when S1='1' and S0='1' else

"0000000000000000" after 1 ns;

end Behavioral;  
RippleAdder.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity RippleAdder is

port(

B, A : in std\_logic\_vector(15 downto 0);

C0 : in std\_logic;

S : out std\_logic\_vector(15 downto 0);

C15 : out std\_logic);

end RippleAdder;

architecture Behavioral of RippleAdder is

component Full\_adder

port(

x, y , z: in std\_logic;

s, c : out std\_logic);

end component;

--Carry signals

signal C\_1, C\_2, C\_3, C\_4, C\_5, C\_6, C\_7, C\_8, C\_9, C\_10,

C\_11, C\_12, C\_13, C\_14, C\_15: STD\_LOGIC;

--Output bit signals

signal S\_0, S\_1, S\_2, S\_3, S\_4, S\_5, S\_6, S\_7, S\_8, S\_9, S\_10,

S\_11, S\_12, S\_13, S\_14, S\_15: STD\_LOGIC;

signal C : std\_logic\_vector (16 downto 0);

--signal output : std\_logic\_vector(15 downto 0);

begin

C(0) <= C0;

Adder16:

for i in 0 to 15 generate

RAdd: Full\_adder Port Map(

x => A(i),

y => B(i),

z => C(i),

s => S(i),

c => C(i+1));

end generate;

C15 <= C(16);

end Behavioral;

Test\_Benches.vhd

DataPath\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Datapath\_test IS

END Datapath\_test;

ARCHITECTURE behavior OF Datapath\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT DataPath

PORT(

load\_enable : in std\_logic;

instruction\_MB : in std\_logic\_vector(15 downto 0);

INL : in std\_logic;

INR : in std\_logic;

Data\_in : in std\_logic\_vector(15 downto 0);

Clk : in std\_logic;

v : out std\_logic;

c : out std\_logic;

n : out std\_logic;

z : out std\_logic;

Adrs\_out : out std\_logic\_vector(15 downto 0);

Data\_out\_bus\_b : out std\_logic\_vector(15 downto 0);

Control\_word : in std\_logic\_vector(16 downto 0)

);

END COMPONENT;

--Inputs

signal load\_enable : std\_logic := '0';

signal instruction\_MB : std\_logic\_vector(15 downto 0) := (others => '0');

signal INL, INR : std\_logic := '0';

signal Data\_in : std\_logic\_vector(15 downto 0) := (others => '0');

signal Clk : std\_logic := '0';

signal Control\_word : std\_logic\_vector(16 downto 0) := (others => '0');

--Outputs

signal v : std\_logic;

signal c : std\_logic;

signal n : std\_logic;

signal z : std\_logic;

signal Adrs\_out : std\_logic\_vector(15 downto 0);

signal Data\_out\_bus\_b : std\_logic\_vector(15 downto 0);

-- Clock period definitions

constant Clk\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: DataPath PORT MAP (

load\_enable => load\_enable,

instruction\_MB => instruction\_MB,

INL => INL,

INR => INR,

Data\_in => Data\_in,

Clk => Clk,

v => v,

c => c,

n => n,

z => z,

Adrs\_out => Adrs\_out,

Data\_out\_bus\_b => Data\_out\_bus\_b,

Control\_word => Control\_word

);

-- Clock process definitions

Clk\_process :process

begin

Clk <= '0';

wait for Clk\_period/2;

Clk <= '1';

wait for Clk\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

-- first store values in registers

Control\_word <= "00001000000000011"; -- write data in to register 0

Data\_in <= "1010101010101010";

wait for 50 ns;

Control\_word <= "11101000000000011"; -- write data in to register 8

Data\_in <= "0101010101010101";

-- Can see the results working throught the register file..

wait for 50 ns;

-- Control\_word <= "11100000000000011"; -- write date in to register 1

wait for 50 ns;

Control\_word <= "00100011100100001"; -- And instruction, dest 1, r0 0, r1 8, fs 01000, write

wait for 50 ns;

Control\_word <= "00001000000000011"; -- write data in to register 0

Data\_in <= "0000000011111111"; -- data in

wait for 50 ns;

Control\_word <= "11001000000000011"; -- write data in to register 6

Data\_in <= "1111111100000000"; -- data in

wait for 50 ns;

Control\_word <= "10000011000001001"; -- Add instruction , dest R4, R0 0, R1 6, fs 00010, write

wait for 50 ns;

wait for Clk\_period\*10;

-- insert stimulus here

wait;

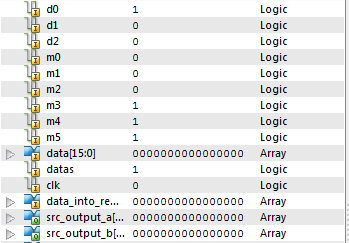
end process;

END;

DataPath\_test.vhd -WaveForm

Shows the values of the control word being added into selects for registers

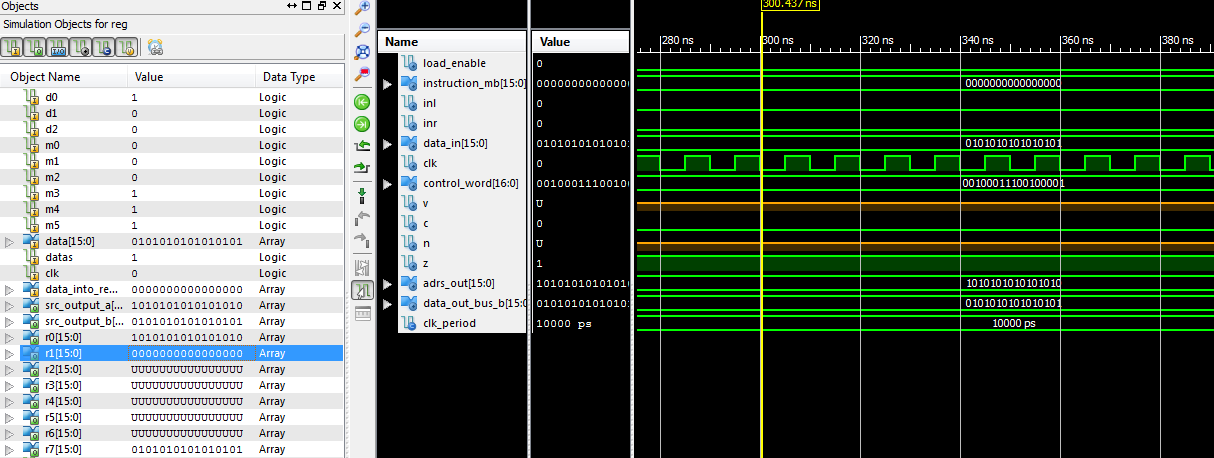
Control word = "00100011100100001"



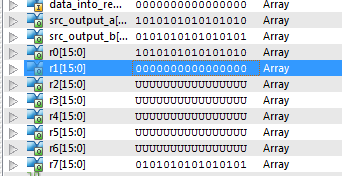
Fs select being updated



After inserting the values into registers 0 and 7. The control word is updated to and registers 0 and 7 and store the result in R1. This is the case in the example below.



Close up of register results



Functional\_unit\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Functional\_unit\_test IS

END Functional\_unit\_test;

ARCHITECTURE behavior OF Functional\_unit\_test IS

-- Component Declaration

COMPONENT Functional\_Unit

PORT(

fsel : in std\_logic\_vector(4 downto 0);

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

f : out std\_logic\_vector(15 downto 0);

IL : in std\_logic;

IR : in std\_logic;

v : out std\_logic;

c : out std\_logic;

n : out std\_logic;

z : out std\_logic );

END COMPONENT;

--output

signal OUTPUT : std\_logic\_vector(15 downto 0) := (others => '0');

signal v , c , n , z , IL, IR : std\_logic := '0';

signal fsel : std\_logic\_vector(4 downto 0) := (others => '0');

signal a, b, f : std\_logic\_vector(15 downto 0) := (others => '0');

BEGIN

-- Component Instantiation

uut: Functional\_Unit PORT MAP(

fsel => fsel,

a => a,

b => b,

f => f,

IL => IL,

IR => IR,

v => v,

c => c,

n => n,

z => z

);

-- Test Bench Statements

tb : PROCESS

BEGIN

wait for 20 ns; -- wait until global set/reset completes

IL <= '0';

IR <= '0';

a <= "0000111100001111";

b <= "0000101010101010";

wait for 20 ns; -- F = A;

fsel <= "00000";

wait for 20 ns; -- F = A+1;

fsel <= "00001";

wait for 20 ns; -- F = A + B;

fsel <= "00010";

wait for 20 ns; -- F = A+ B +1;

fsel <= "00011";

wait for 20 ns; -- F = A+ NOTB

fsel <= "00100";

wait for 20 ns; -- F = A+ notB +1 -- subtract

fsel <= "00101";

wait for 20 ns; -- F = A -1

fsel <= "00110";

wait for 20 ns; -- F = A

fsel <= "00111";

wait for 20 ns; -- F = A AND B

fsel <= "01000";

wait for 20 ns; -- F = A OR B

fsel <= "01010";

wait for 20 ns; -- F = A XOR B

fsel <= "01100";

wait for 20 ns; -- F = NOT A

fsel <= "01110";

wait for 20 ns; -- F = B

fsel <= "10000";

wait for 20 ns; -- F = sr B

fsel <= "10100";

wait for 20 ns; -- F = sl B

fsel <= "11000";

wait for 20 ns; -- F = sr B

wait; -- will wait forever

END PROCESS tb;

-- End Test Bench

END;

Functional\_unit - Waveforms



Reg\_file\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Reg\_File\_Test IS

END Reg\_File\_Test;

ARCHITECTURE behavior OF Reg\_File\_Test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Reg\_File

Port (

d0 : in STD\_LOGIC;

d1 : in STD\_LOGIC;

d2 : in STD\_LOGIC;

m0 : in STD\_LOGIC;

m1 : in STD\_LOGIC;

m2 : in STD\_LOGIC;

m3 : in STD\_LOGIC;

m4 : in STD\_LOGIC;

m5 : in STD\_LOGIC;

data : in STD\_LOGIC\_VECTOR(15 DOWNTO 0);

dataS : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Data\_into\_reg : in STD\_LOGIC\_VECTOR(15 downto 0);

src\_output\_a : out std\_logic\_vector(15 downto 0);

src\_output\_b : out std\_logic\_vector(15 downto 0);

R0 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R1 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R2 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R3 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R4 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R5 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R6 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0);

R7 : out STD\_LOGIC\_VECTOR(15 DOWNTO 0));

END COMPONENT;

--Inputs

signal d0 : std\_logic := '0';

signal d1 : std\_logic := '0';

signal d2 : std\_logic := '0';

signal m0 : std\_logic := '0';

signal m1 : std\_logic := '0';

signal m2 : std\_logic := '0';

signal m3 : std\_logic := '0';

signal m4 : std\_logic := '0';

signal m5 : std\_logic := '0';

signal Data\_into\_reg : std\_logic\_vector(15 downto 0) := (others => '0');

signal Clk : std\_logic := '0';

signal dataS : std\_logic := '0';

signal data : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal src\_output\_a : std\_logic\_vector (15 downto 0);

signal src\_output\_b : std\_logic\_vector (15 downto 0);

signal R0 : std\_logic\_vector(15 downto 0);

signal R1 : std\_logic\_vector(15 downto 0);

signal R2 : std\_logic\_vector(15 downto 0);

signal R3 : std\_logic\_vector(15 downto 0);

signal R4 : std\_logic\_vector(15 downto 0);

signal R5 : std\_logic\_vector(15 downto 0);

signal R6 : std\_logic\_vector(15 downto 0);

signal R7 : std\_logic\_vector(15 downto 0);

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Reg\_File PORT MAP (

d0 => d0,

d1 => d1,

d2 => d2,

m0 => m0,

m1 => m1,

m2 => m2,

m3 => m3,

m4 => m4,

m5 => m5,

Data\_into\_reg => Data\_into\_reg,

dataS => dataS,

src\_output\_a => src\_output\_a,

src\_output\_b => src\_output\_b,

data => data,

CLK => CLK,

R0 => R0,

R1 => R1,

R2 => R2,

R3 => R3,

R4 => R4,

R5 => R5,

R6 => R6,

R7 => R7

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for CLK\_period;

-- insert stimulus here

--Allow writing to registers

dataS <= '0';

--Select R0

d0 <= '0';

d1 <= '0';

d2 <='0';

data <= "0000000000000001";

wait for CLK\_period\*2;

--Select R1

d0 <= '1';

d1 <= '0';

d2 <='0';

data <= "0000000000000010";

wait for CLK\_period\*2;

--Select R2

d0 <= '0';

d1 <= '1';

d2 <='0';

data <= "0000000000000100";

wait for CLK\_period\*2;

--Select R3

d0 <= '1';

d1 <= '1';

d2 <='0';

data <= "0000000000001000";

wait for CLK\_period\*2;

--Select R4

d0 <= '0';

d1 <= '0';

d2 <='1';

data <= "0000000000010000";

wait for CLK\_period\*2;

--Select R5

d0 <= '1';

d1 <= '0';

d2 <='1';

data <= "0000000000100000";

wait for CLK\_period\*2;

--Select R6

d0 <= '0';

d1 <= '1';

d2 <='1';

data <= "0000000001000000";

wait for CLK\_period\*2;

--Select R7

d0 <= '1';

d1 <= '1';

d2 <='1';

data <= "0101010101010101";

wait for CLK\_period\*2;

--Now load into different registers

dataS <= '1';

--Move R7 to other registers

--Select R7 with mux

m0 <= '1';

m1 <= '1';

m2 <= '1';

wait for CLK\_period\*2;

--Select R0 with decoder

d0 <= '0';

d1 <= '0';

d2 <= '0';

wait for CLK\_period\*2;

--Select R1 with decoder

d0 <= '1';

d1 <= '0';

d2 <= '0';

wait for CLK\_period\*2;

--Select R2 with decoder

d0 <= '0';

d1 <= '1';

d2 <= '0';

wait for CLK\_period\*2;

--Select R3 with decoder

d0 <= '1';

d1 <= '1';

d2 <= '0';

wait for CLK\_period\*2;

--Select R4 with decoder

d0 <= '0';

d1 <= '0';

d2 <= '1';

wait for CLK\_period\*2;

--Select R5 with decoder

d0 <= '1';

d1 <= '0';

d2 <= '1';

wait for CLK\_period\*2;

--Select R6 with decoder

d0 <= '0';

d1 <= '1';

d2 <= '1';

wait for CLK\_period\*2;

--Select R7 with decoder

d0 <= '1';

d1 <= '1';

d2 <= '1';

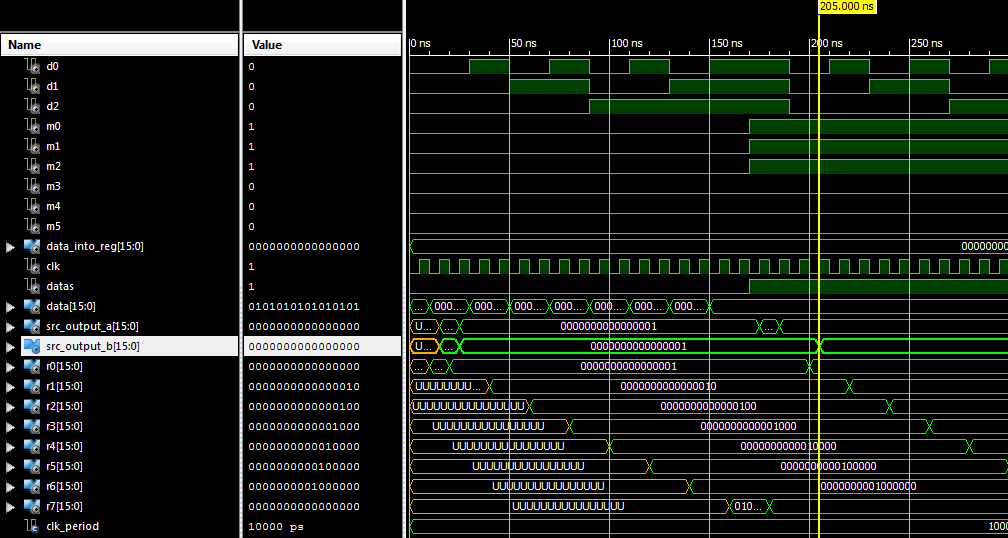
wait for CLK\_period\*2;

wait;

end process;

END;

Reg\_file\_test – Waveforms



ALU\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY ALU\_test IS

END ALU\_test;

ARCHITECTURE behavior OF ALU\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT ALU

PORT(

A, B : in std\_logic\_vector(15 downto 0);

S1 : in std\_logic;

S2 : in std\_logic;

S3 : in std\_logic;

Cin : in std\_logic; -- fS0

G : out std\_logic\_vector(15 downto 0);

c : out std\_logic;

v : out std\_logic;

n : out std\_logic

);

END COMPONENT;

--Inputs

signal S1 : std\_logic := '0';

signal S2 : std\_logic := '0';

signal S3 : std\_logic := '0';

signal Cin : std\_logic := '0'; --fS1

signal A : std\_logic\_vector(15 downto 0) := (others => '0');

signal B : std\_logic\_vector(15 downto 0) := (others =>'0');

--Outputs

signal G : std\_logic\_vector(15 downto 0) := (others => '0');

signal cOut : std\_logic := '0';

signal v : std\_logic := '0';

signal c : std\_logic := '0';

signal n : std\_logic := '0';

signal z : std\_logic := '0';

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: ALU PORT MAP (

S1 => S1,

S2 => S2,

S3 => S3,

A => A,

B => B,

G => G,

Cin => Cin,

v => v,

c => c,

n => n

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

A <= "0000000000110000";

B <= "0000000011000000";

-- g == A

Cin <= '0';

S3 <= '0';

S2 <= '0';

S1 <= '0';

wait for 100 ns; -- g == A+1

Cin <= '1';

wait for 100 ns; -- g ==A + B

Cin <= '0';

S3 <= '0';

S2 <= '0';

S1 <= '1';

wait for 100 ns; -- g =A + B +1

Cin <= '1';

wait for 100 ns; -- g = A + B .. not sure

Cin <= '0';

S3 <= '0';

S2 <= '1';

S1 <= '0';

wait for 100 ns; -- g = A + B + 1

Cin <= '1';

wait for 30 ns; -- g = A - 1

Cin <= '0';

S3 <= '0';

S2 <= '1';

S1 <= '1';

wait for 30 ns; -- g = A

Cin <= '1';

wait for 30 ns; -- g = A and B

S3 <= '1';

S2 <= '0';

S1 <= '0';

wait for 30 ns; -- g = A OR B

-- dont need to continously change s2 but Just to make it clear

S3 <= '1';

S2 <= '0';

S1 <= '1';

wait for 30 ns; -- g =A XOR B

S3 <= '1';

S2 <= '1';

S1 <= '0';

wait for 30 ns; -- g = NOT A

S3 <= '1';

S2 <= '1';

S1 <= '1';

wait for 30 ns; --

-- insert stimulus here

wait;

end process;

END;

ALU\_test - Waveforms



Shifter\_test.vhd

-- TestBench Template

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

-- TestBench Template

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Shifter\_test IS

END Shifter\_test;

ARCHITECTURE behavior OF Shifter\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Shifter

PORT(

IL : in std\_logic;

IR : in std\_logic;

s : in std\_logic\_vector(1 downto 0);

b : in std\_logic\_vector (15 downto 0);

O : out std\_logic\_vector(15 downto 0);

SL0 : out std\_logic;

SR0 : out std\_logic

);

END COMPONENT;

--Inputs

signal b : std\_logic\_vector(15 downto 0) := (others => '0');

signal s : std\_logic\_vector(1 downto 0 ) := (others => '0');

signal IR : std\_logic := '0';

signal IL : std\_logic := '0';

--outputs

signal O : std\_logic\_vector(15 downto 0) := (others => '0');

signal SL0 : std\_logic := '0';

signal SR0 : std\_logic := '0';

--Outputs

signal output : std\_logic\_vector(15 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Shifter PORT MAP (

IL => IL,

IR => IR,

s => s,

b => b,

O => O,

SL0 => SL0,

SR0 => SR0

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 10 ns;

-- insert stimulus here

b <= "0000101010101010" ;

--Set for In0

s <= "00";

wait for 20 ns;

--Set for In1

s <= "01";

wait for 10 ns;

s <= "10";

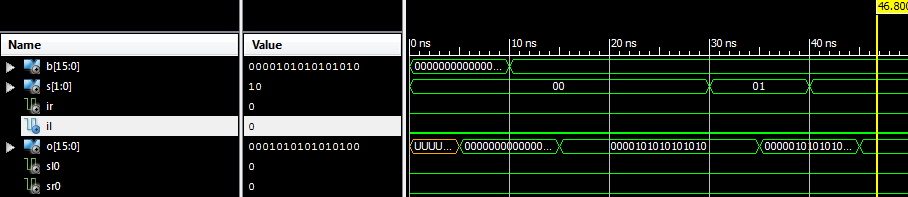
wait for 10 ns;

wait;

end process;

END;

Shifter\_test - Waveforms



Arithmetic\_circuit\_test.vhd

-- TestBench Template

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Arithmetic\_circuit\_test IS

END Arithmetic\_circuit\_test;

ARCHITECTURE behavior OF Arithmetic\_circuit\_test IS

-- Component Declaration

COMPONENT Arithmetic\_Circuit

PORT(

a : in std\_logic\_vector(15 downto 0);

b : in std\_logic\_vector(15 downto 0);

S0 : in std\_logic;

S1 : in std\_logic;

Cin : in std\_logic;

Cout : out std\_logic;

g : out std\_logic\_vector(15 downto 0)

);

END COMPONENT;

-- inputs

SIGNAL S0 : std\_logic := '0';

signal S1 : std\_logic := '0';

signal Cin : std\_logic := '0';

SIGNAL a : std\_logic\_vector(15 downto 0) := (others => '0');

signal b : std\_logic\_vector(15 downto 0) := (others => '0');

signal Cout : std\_logic := '0';

-- outputs

signal g : std\_logic\_vector(15 downto 0) := (others => '0');

BEGIN

-- Component Instantiation

uut: Arithmetic\_Circuit PORT MAP(

a => a,

b => b,

S0 => S0,

S1 => S1,

Cin => Cin,

Cout => Cout,

g => g

);

-- Test Bench Statements

tb : PROCESS

BEGIN

wait for 20 ns; -- wait until global set/reset completes

A <= "0000000000110000";

B <= "0000000011000000";

wait for 20 ns;

S1 <= '0';

S0 <= '0';

Cin <= '0';

wait for 20 ns;

Cin <= '1';

wait for 20 ns;

S1 <= '0';

S0 <= '1';

Cin <= '0';

wait for 20 ns;

Cin <= '1';

wait for 20 ns;

S1 <= '1';

S0 <= '0';

Cin <= '0';

wait for 20 ns;

Cin <= '1';

wait for 20 ns;

S1 <= '1';

S0 <= '1';

Cin <= '0';

wait for 20 ns;

Cin <= '1';

wait for 20 ns;

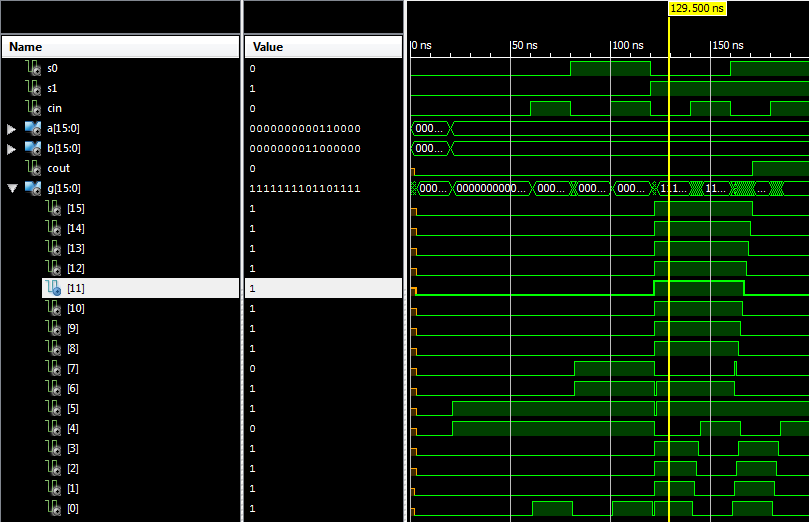
-- Add user defined stimulus here

wait; -- will wait forever

END PROCESS tb;

-- End Test Bench

END;



Logic\_circuit\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY logic\_circuit\_test IS

END logic\_circuit\_test;

ARCHITECTURE behavior OF logic\_circuit\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT logic\_Circuit

PORT(

a : IN std\_logic\_vector(15 downto 0);

b : IN std\_logic\_vector(15 downto 0);

S0 : IN std\_logic;

S1 : IN std\_logic;

g : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal a : std\_logic\_vector(15 downto 0) := (others => '0');

signal b : std\_logic\_vector(15 downto 0) := (others => '0');

signal S0 : std\_logic := '0';

signal S1 : std\_logic := '0';

--Outputs

signal g : std\_logic\_vector(15 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: logic\_Circuit PORT MAP (

a => a,

b => b,

S0 => S0,

S1 => S1,

g => g

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

a <= "0000000000000001";

b <= "0000000000001111";

wait for 10 ns;

S0 <= '0';

S1 <= '0';

wait for 10 ns;

S0 <= '1';

S1 <= '0';

wait for 10 ns;

S0 <= '0';

S1 <= '1';

wait for 10 ns;

S0 <= '1';

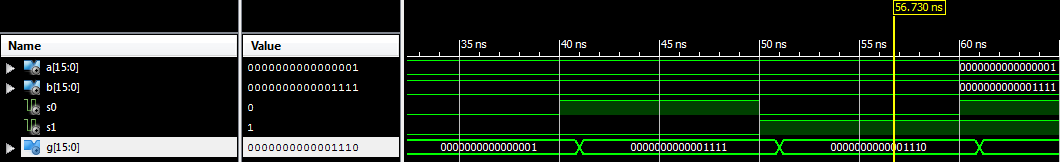
S1 <= '1';

-- insert stimulus here

wait;

end process;

END;



RippleAdder\_test.vhd

-- TestBench Template

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY RippleAdder\_test IS

END RippleAdder\_test;

ARCHITECTURE behavior OF RippleAdder\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT RippleAdder

PORT(

B, A : in std\_logic\_vector(15 downto 0);

C0 : in std\_logic; -- input

S : out std\_logic\_vector(15 downto 0);

C15 : out std\_logic -- output

);

END COMPONENT;

--Inputs

signal B : std\_logic\_vector(15 downto 0) := (others => '0');

signal A : std\_logic\_vector(15 downto 0) := (others => '0');

signal C0 : std\_logic := '0';

--Outputs

signal s : std\_logic\_vector(15 downto 0);

signal C15 : std\_logic := '0';

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: RippleAdder PORT MAP (

A => A,

B => B,

S => S,

C0 => C0,

C15 => C15

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

-- insert stimulus here

A <= "0000000100010001";

B <= "0000000100010001";

--Set for a

C0 <= '0';

wait for 20 ns;

--Set for b

wait for 20 ns;

B <= x"FFFF";

wait for 20 ns;

A <= x"0001";

wait for 20 ns;

A<= x"8000";

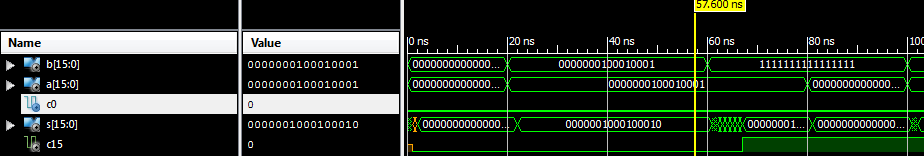
B<= x"8000";

wait for 20 ns;

wait;

end process;

END;



BInput\_logic\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY B\_input\_logic\_test IS

END B\_input\_logic\_test;

ARCHITECTURE behavior OF B\_input\_logic\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT B\_Input\_Logic

PORT(

b : IN std\_logic\_vector(15 downto 0);

S0 : IN std\_logic;

S1 : IN std\_logic;

y : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal b : std\_logic\_vector(15 downto 0) := (others => '0');

signal S0 : std\_logic := '0';

signal S1 : std\_logic := '0';

--Outputs

signal y : std\_logic\_vector(15 downto 0) := (others => '0');

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: B\_Input\_Logic PORT MAP (

b => b,

S0 => S0,

S1 => S1,

y => y

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

b <= "0000000000000000";

wait for 5 ns;

S0 <= '0';

S1 <= '0';

wait for 10 ns;

S0 <= '0';

S1 <= '1';

wait for 10 ns;

S0 <= '1';

S1 <= '0';

wait for 10 ns;

S0 <= '1';

S1 <= '1';

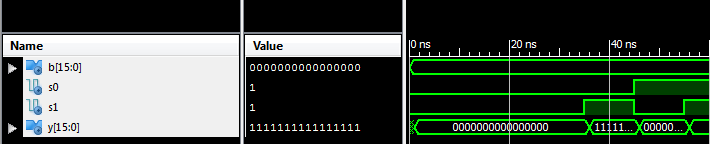
wait for 10 ns;

-- insert stimulus here

wait;

end process;

END;



Detect\_zero\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY zeroDetect\_test IS

END zeroDetect\_test;

ARCHITECTURE behavior OF zeroDetect\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT zeroDetect

PORT(

input : IN std\_logic\_vector(15 downto 0);

outputFlag : OUT std\_logic;

output : out std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal input : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal output : std\_logic\_vector(15 downto 0) := (others => '0');

signal outputFlag : std\_logic;

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: zeroDetect PORT MAP (

input => input,

outputFlag => outputFlag,

output => output

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

input <= "1111111111111111";

wait for 10 ns;

input <= "0000000000000000";

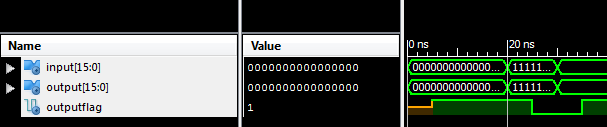
wait for 10 ns;

-- insert stimulus here

wait;

end process;

END;



Mux\_2\_1\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Mux2\_1\_test IS

END Mux2\_1\_test;

ARCHITECTURE behavior OF Mux2\_1\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Mux\_2\_1

PORT(

sel : IN std\_logic;

a : IN std\_logic\_vector(15 downto 0);

b : IN std\_logic\_vector(15 downto 0);

g : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

--Inputs

signal sel : std\_logic := '0';

signal a : std\_logic\_vector(15 downto 0) := (others => '0');

signal b : std\_logic\_vector(15 downto 0) := (others => '0');

--Outputs

signal g : std\_logic\_vector(15 downto 0);

-- No clocks detected in port list. Replace <clock> below with

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: Mux\_2\_1 PORT MAP (

sel => sel,

a => a,

b => b,

g => g

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

a <= "0000000000000000";

b <= "0000000000000001";

wait for 10 ns;

sel <= '0';

wait for 10 ns;

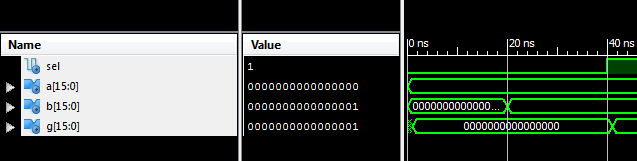
sel <= '1';

-- insert stimulus here

wait;

end process;

END;



Shifter\_mux\_test.vhd

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY ShifterMux\_test IS

END ShifterMux\_test;

ARCHITECTURE behavior OF ShifterMux\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT shifter\_mux

PORT(

A : IN std\_logic;

B : IN std\_logic;

C : IN std\_logic;

S : IN std\_logic\_vector(1 downto 0);

H : OUT std\_logic

);

END COMPONENT;

--Inputs

signal A : std\_logic := '0';

signal B : std\_logic := '0';

signal C : std\_logic := '0';

signal S : std\_logic\_vector(1 downto 0) := (others => '0');

--Outputs

signal H : std\_logic;

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: shifter\_mux PORT MAP (

A => A,

B => B,

C => C,

S => S,

H => H

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 20 ns;

S <= "00";

wait for 10 ns;

A <= '1';

wait for 10 ns;

S<= "01";

A <= '0';

wait for 10 ns;

B <= '1';

wait for 10 ns;

S<= "10";

wait for 10 ns;

C <= '1';

B <= '0';

wait for 20 ns;

-- insert stimulus here

wait;

end process;

END;

