**CS2022 PROJECT 1- DATAPATH DESIGN**

**PART A**

Aniket Agarwal

17317437

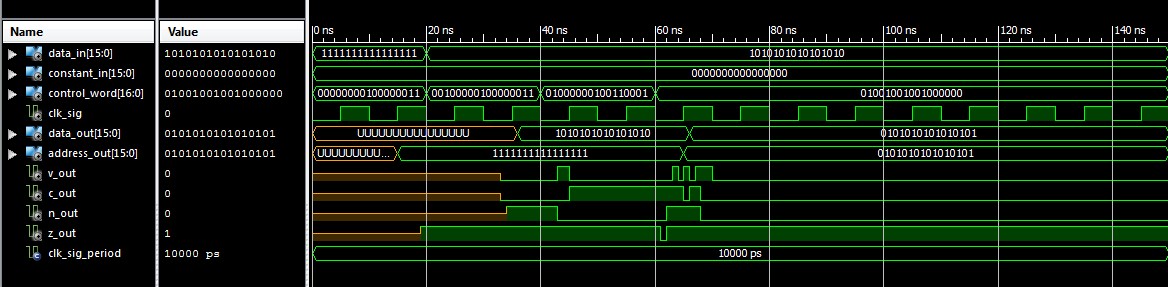
**Contents:**

1. Results Of Test Benches
2. VHDL Component Source Code (With Schematics)
3. Component Test Benches

**1. Results of Test Benches**

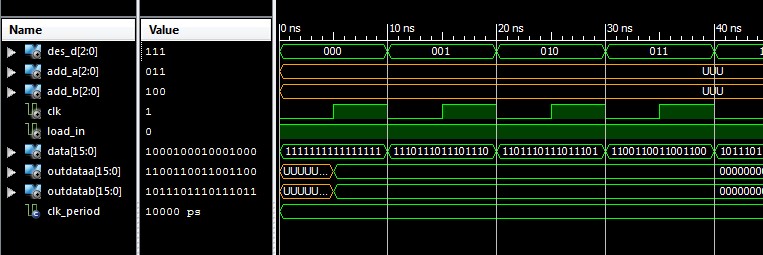
a) Project 1B Top Level

The results from this test-bench show that all the modules instantiated together with values are being loaded into the subsequent registers, and micro-operations are then being carried out appropriately with loading into a new register.



b) Register File

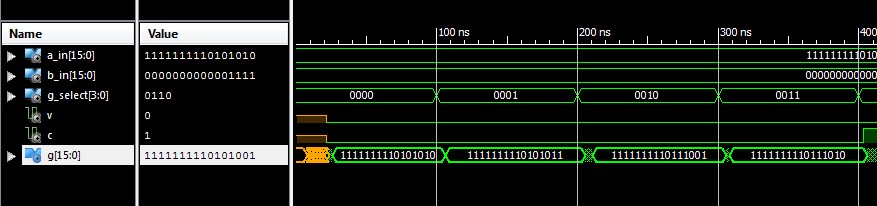
The register file test bench shows the register submodules working together through decoding, multiplexing, storing values and passing data in and out appropriately with respect to the clock period and the load.



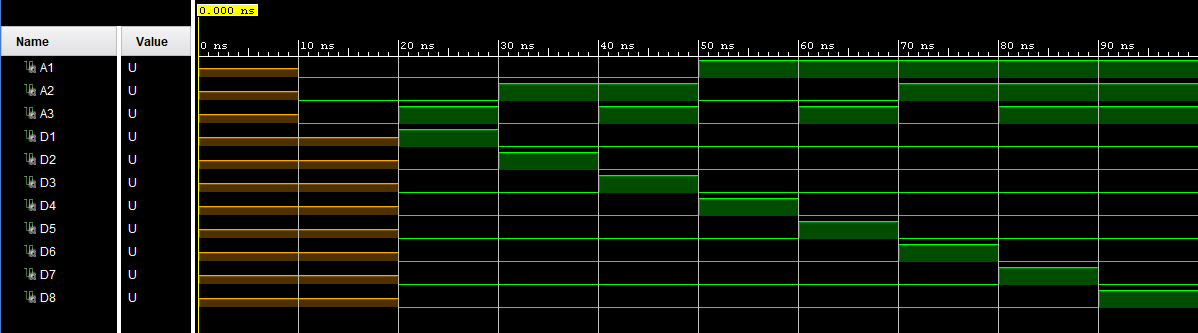


c) ALU Unit

The ALU component functions as intended by having correct outputs on its G pin where it’s a-in, b-in and g-select have data and operations supplied to them. The result is appropriate V and C flags being set, with a 16 bit output on G being output.

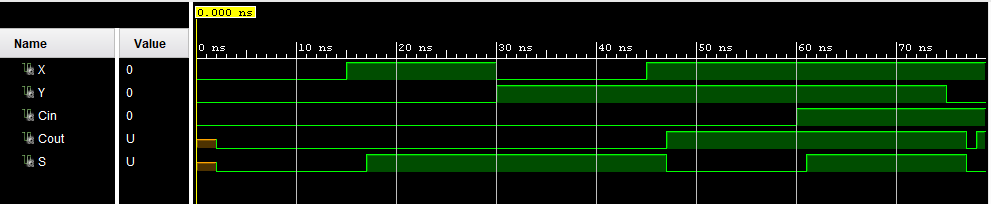


d) Decoder 3-8 bit

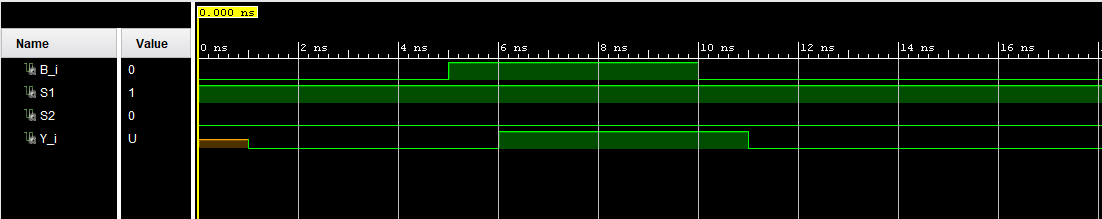


e) Full Adder

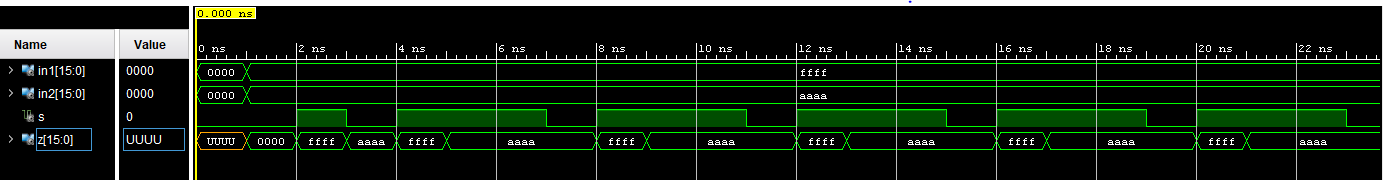
For each input, the appropriate carry is determined and correct operations are performed for the given select pin, which results in the intended output.



f) Mux 2-1 Bit

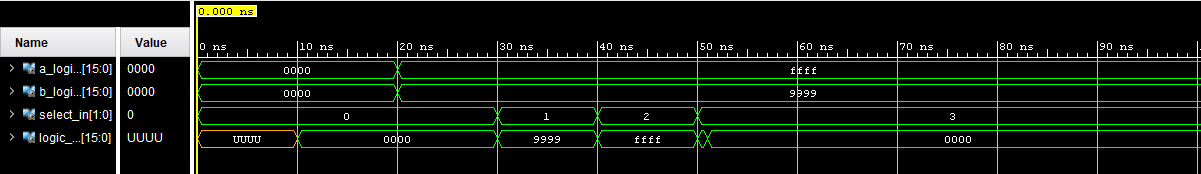


g) Mux 2-16 bit



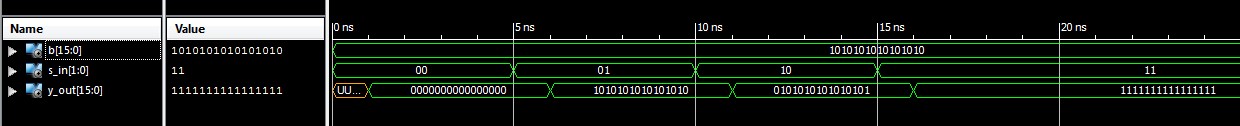
h) Logic Circuit A-B

It provides inputs of A and B logic as well as a select pin that determines the operations to be performed resulting in outputs via gate logic.

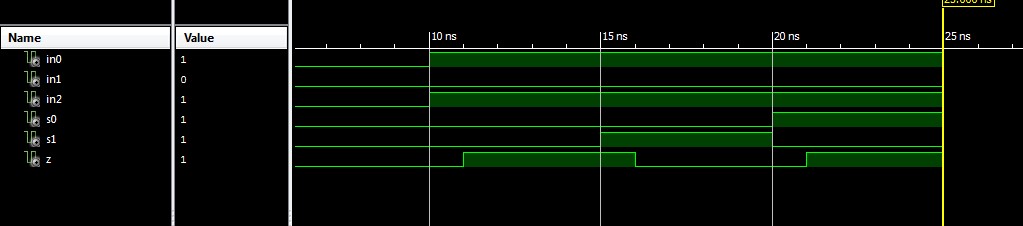


i) Logic Circuit B

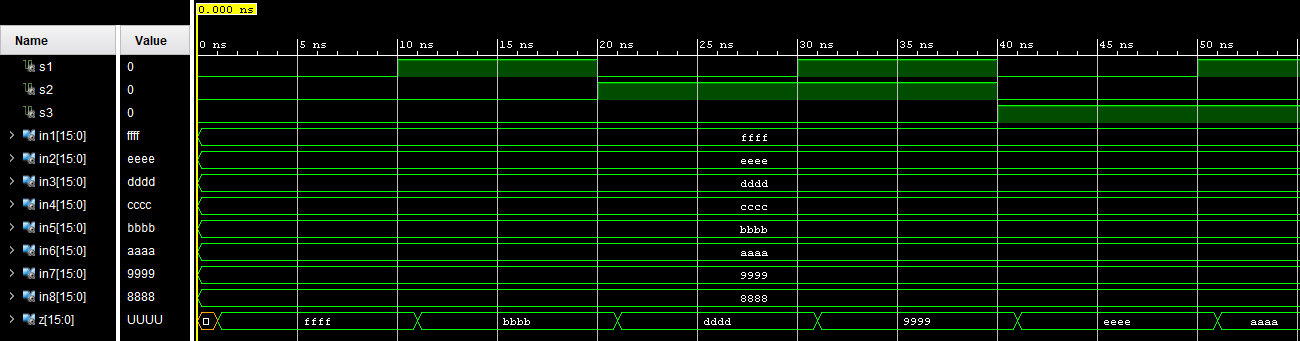
The B logic input results in correct outputs for each B and s value provided to the logic circuit.



j) Mux 3-1 bit

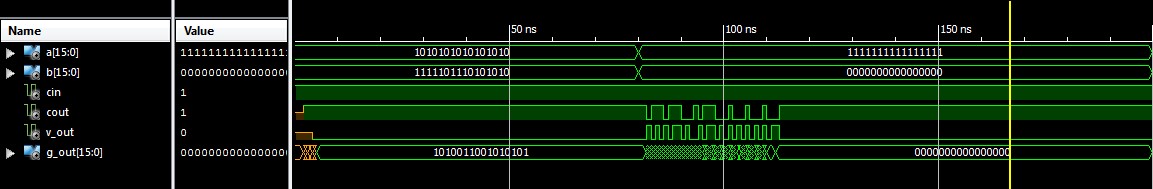


k) Mux 8-16 bit



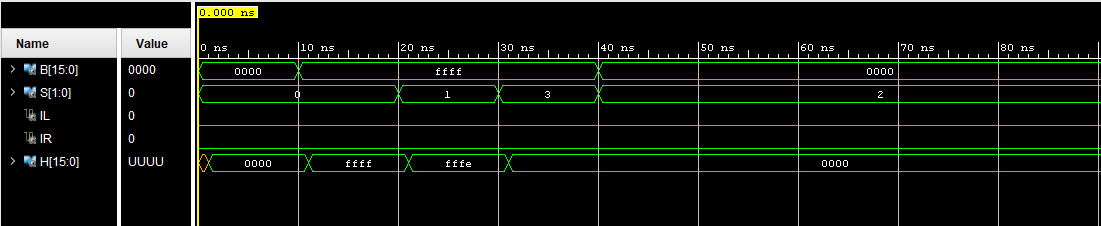
l) Ripple Adder

It appropriately determines the c-in, c-out, and v-out flags with oscillations on its G-out output to between 0 and 1, where the adder itself has two 16 bit inputs a and b.

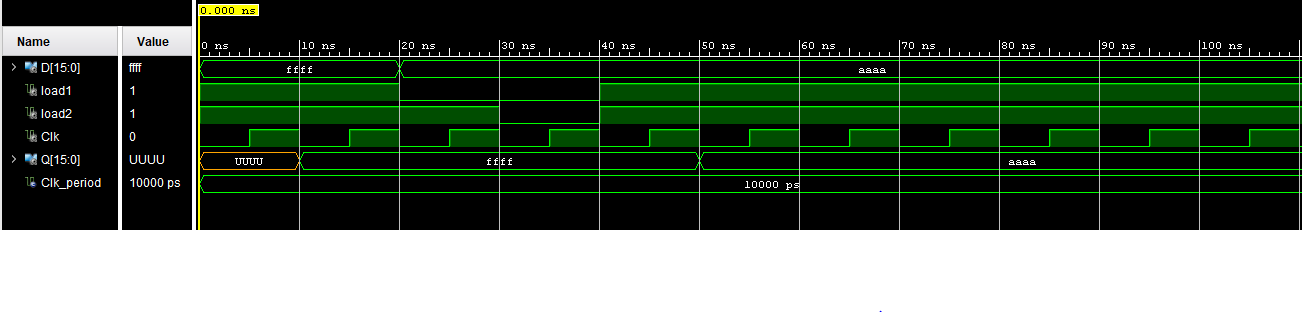


m) Shifter

This component works as intended by outputting correct left and right shifts on the H value, where the data and operations are input on b and s pins.

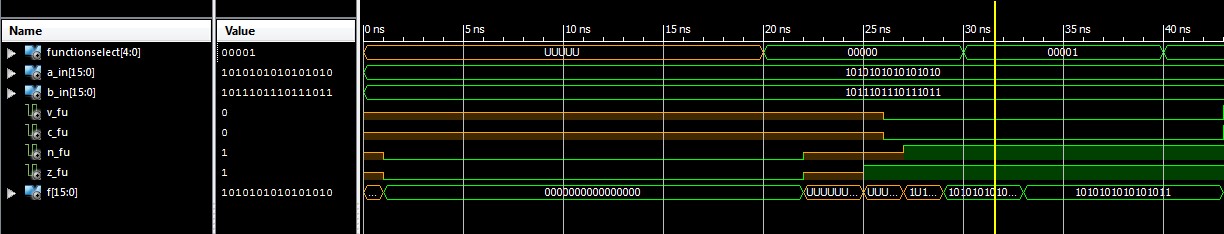


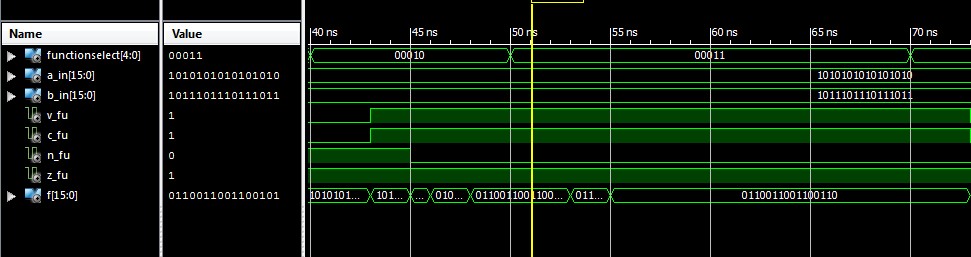
n) Register

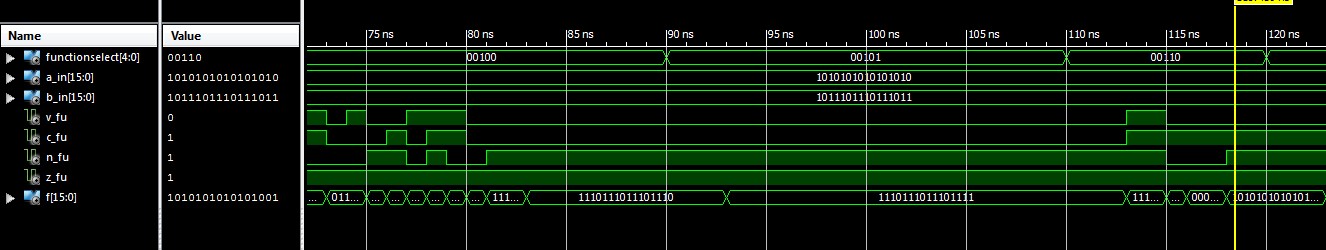


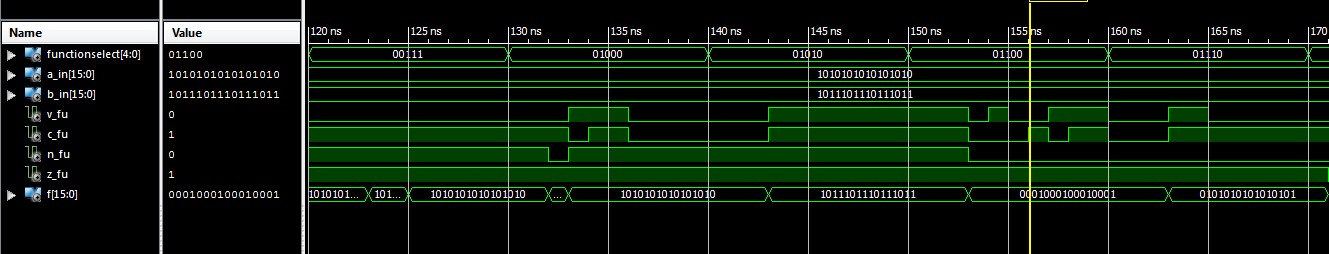
o) Functional Unit

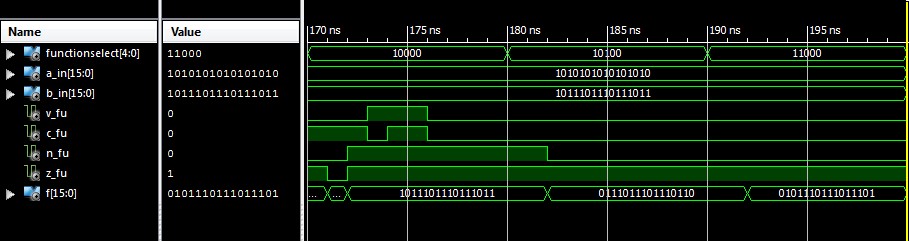
The function unit works as it should be with appropriate values being handled and raising correct N, Z, C and V fags in the unit itself to the word passed in for function selection.











**2. VHDL Component Source Code**

a) Project 1B Top level

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Proj1b is

Port(

data\_in, constant\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

control\_word : in STD\_LOGIC\_VECTOR(16 downto 0);

Clk\_sig : in STD\_LOGIC;

data\_out, address\_out : out STD\_LOGIC\_VECTOR(15 downto 0);

N\_out, Z\_out, C\_out, V\_out : out STD\_LOGIC

);

end Proj1b;

architecture Behavioral of Proj1b is

Component reg2

Port(

des\_D, add\_a, add\_b : in STD\_LOGIC\_VECTOR(2 downto 0);

Clk, load\_in : in STD\_LOGIC;

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

out\_data\_a, out\_data\_b : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component mux

Port(

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

s : in STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component function\_unit

Port(

FunctionSelect : in STD\_LOGIC\_VECTOR(4 downto 0); -- 5 input

a\_in, b\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

N\_fu, Z\_fu, V\_fu, C\_fu : out STD\_LOGIC;

F : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

signal mux\_b\_out, mux\_d\_out, reg\_file\_out\_a, reg\_file\_out\_b, function\_unit\_out : STD\_LOGIC\_VECTOR(15 downto 0);

begin

mux\_b: mux PORT MAP(

in1 => constant\_in,

in2 => reg\_file\_out\_b,

s => control\_word(7),

z => mux\_b\_out

);

mux\_d: mux PORT MAP(

in1 => function\_unit\_out,

in2 => data\_in,

s => control\_word(1),

z => mux\_d\_out

);

reg1: reg2 PORT MAP(

des\_D => control\_word(16 downto 14),

add\_a => control\_word(13 downto 11),

add\_b => control\_word(10 downto 8),

Clk => Clk\_sig,

load\_in => control\_word(0),

data => mux\_d\_out,

out\_data\_a => reg\_file\_out\_a,

out\_data\_b => reg\_file\_out\_b

);

data\_out <= mux\_b\_out;

address\_out <= reg\_file\_out\_a;

function\_unit1: function\_unit PORT MAP(

FunctionSelect => control\_word(6 downto 2),

A\_in => reg\_file\_out\_a,

B\_in => mux\_b\_out,

N\_fu => N\_out,

Z\_fu => Z\_out,

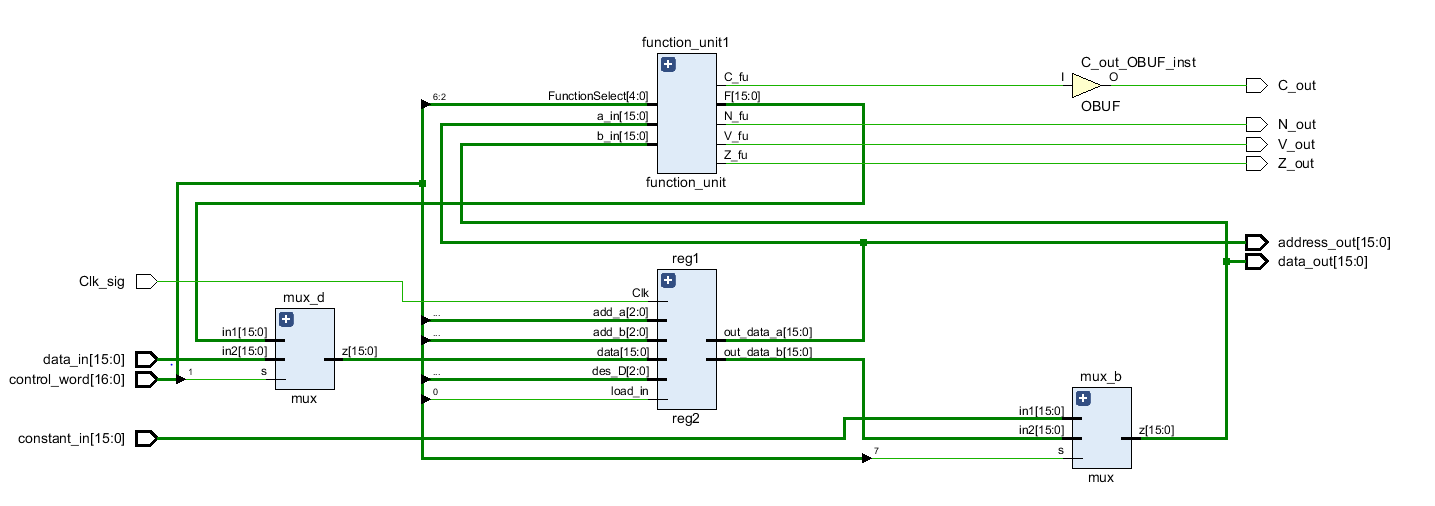
C\_fu => C\_out,

V\_fu => V\_out,

F => function\_unit\_out

);

end Be havioral;



b) Register File

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity reg2 is

Port(

des\_D, add\_a, add\_b : in STD\_LOGIC\_VECTOR(2 downto 0);

Clk, load\_in : in STD\_LOGIC;

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

out\_data\_a, out\_data\_b : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end reg2;

architecture Behavioral of reg2 is

Component reg16

Port(

D : in STD\_LOGIC\_VECTOR(15 downto 0);

load1, load2, Clk : in STD\_LOGIC;

Q : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end Component;

Component decoder

Port(

A1, A2, A3 : in STD\_LOGIC;

D1, D2, D3, D4, D5, D6, D7, D8 : out STD\_LOGIC

);

End Component;

Component mux

Port(

in1, in2 : STD\_LOGIC\_VECTOR(15 downto 0);

s : STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component multiplexer

Port(

in1, in2, in3, in4, in5, in6, in7, in8 : in STD\_LOGIC\_VECTOR(15 downto 0);

s1, s2, s3 : in STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

signal load\_r1, load\_r2, load\_r3, load\_r4, load\_r5, load\_r6, load\_r7, load\_r8 : STD\_LOGIC;

signal r1\_q, r2\_q, r3\_q, r4\_q, r5\_q, r6\_q, r7\_q, r8\_q, data\_src\_mux\_out, src\_reg, out\_sig\_a, out\_sig\_b : STD\_LOGIC\_VECTOR(15 downto 0);

begin

Reg1: reg16 PORT MAP(

D => data,

load1 => load\_r1,

load2 => load\_in,

Clk => Clk,

Q => r1\_q

);

Reg2: reg16 PORT MAP(

D => data,

load1 => load\_r2,

load2 => load\_in,

Clk => Clk,

Q => r2\_q

);

Reg3: reg16 PORT MAP(

D => data,

load1 => load\_r3,

load2 => load\_in,

Clk => Clk,

Q => r3\_q

);

Reg4: reg16 PORT MAP(

D => data,

load1 => load\_r4,

load2 => load\_in,

Clk => Clk,

Q => r4\_q

);

Reg5: reg16 PORT MAP(

D => data,

load1 => load\_r5,

load2 => load\_in,

Clk => Clk,

Q => r5\_q

);

Reg6: reg16 PORT MAP(

D => data,

load1 => load\_r6,

load2 => load\_in,

Clk => Clk,

Q => r6\_q

);

Reg7: reg16 PORT MAP(

D => data,

load1 => load\_r7,

load2 => load\_in,

Clk => Clk,

Q => r7\_q

);

Reg8: reg16 PORT MAP(

D => data,

load1 => load\_r8,

load2 => load\_in,

Clk => Clk,

Q => r8\_q

);

des\_decoder\_3\_8: decoder PORT MAP(

A1 => des\_D(0),

A2 => des\_D(1),

A3 => des\_D(2),

D1 => load\_r1,

D2 => load\_r2,

D3 => load\_r3,

D4 => load\_r4,

D5 => load\_r5,

D6 => load\_r6,

D7 => load\_r7,

D8 => load\_r8

);

A\_8\_1\_mux: multiplexer PORT MAP(

in1 => r1\_q,

in2 => r2\_q,

in3 => r3\_q,

in4 => r4\_q,

in5 => r5\_q,

in6 => r6\_q,

in7 => r7\_q,

in8 => r8\_q,

s1 => add\_a(0),

s2 => add\_a(1),

s3 => add\_a(2),

z => out\_sig\_a

);

B\_8\_1\_mux: multiplexer PORT MAP(

in1 => r1\_q,

in2 => r2\_q,

in3 => r3\_q,

in4 => r4\_q,

in5 => r5\_q,

in6 => r6\_q,

in7 => r7\_q,

in8 => r8\_q,

s1 => add\_b(0),

s2 => add\_b(1),

s3 => add\_b(2),

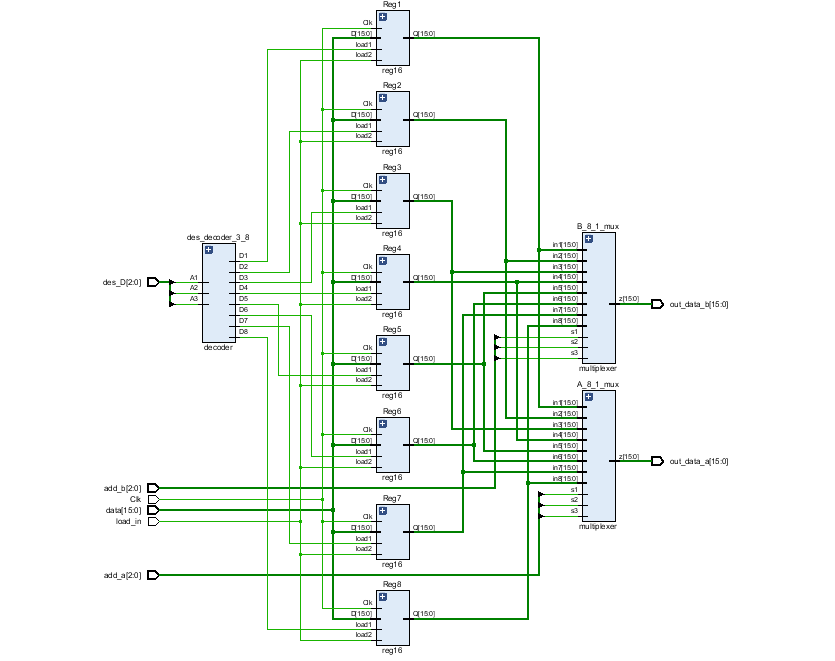
z => out\_sig\_b

);

out\_data\_a <= out\_sig\_a;

out\_data\_b <= out\_sig\_b;

end Behavioral;



c) ALU Unit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity alu\_unit is

Port(

a\_in, b\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

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

V, C : out STD\_LOGIC; -- flags

G : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end alu\_unit;

architecture Behavioral of alu\_unit is

Component ripple\_adder

Port(

A, B : in STD\_LOGIC\_VECTOR(15 downto 0);

Cin : in STD\_LOGIC;

Cout, V\_out : out STD\_LOGIC;

G\_out : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component bus\_a\_b

Port(

a\_logic\_in, b\_logic\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

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

logic\_output\_a\_b : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component bus\_b

Port(

B : in STD\_LOGIC\_VECTOR(15 downto 0);

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

Y\_out : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

Component mux

Port(

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

s : in STD\_LOGIC;

z : out STD\_LOGIC\_VECTOR(15 downto 0)

);

End Component;

signal logic\_out, logic\_output\_a\_b, ripple\_out : STD\_LOGIC\_VECTOR(15 downto 0);

begin

r\_adder: ripple\_adder PORT MAP(

A => a\_in,

B => b\_in,

Cin => G\_select(0),

Cout => C,

V\_out => V,

G\_out => ripple\_out

);

logic\_circuit\_a\_b: bus\_a\_b PORT MAP(

a\_logic\_in => a\_in,

b\_logic\_in => b\_in,

select\_in => G\_select(2 downto 1),

logic\_output\_a\_b => logic\_output\_a\_b

);

logic\_circuit\_b : bus\_b PORT MAP(

B => b\_in,

S\_in => G\_select(2 downto 1),

Y\_out => logic\_out

);

mux\_2\_16: mux PORT MAP(

in1 => ripple\_out,

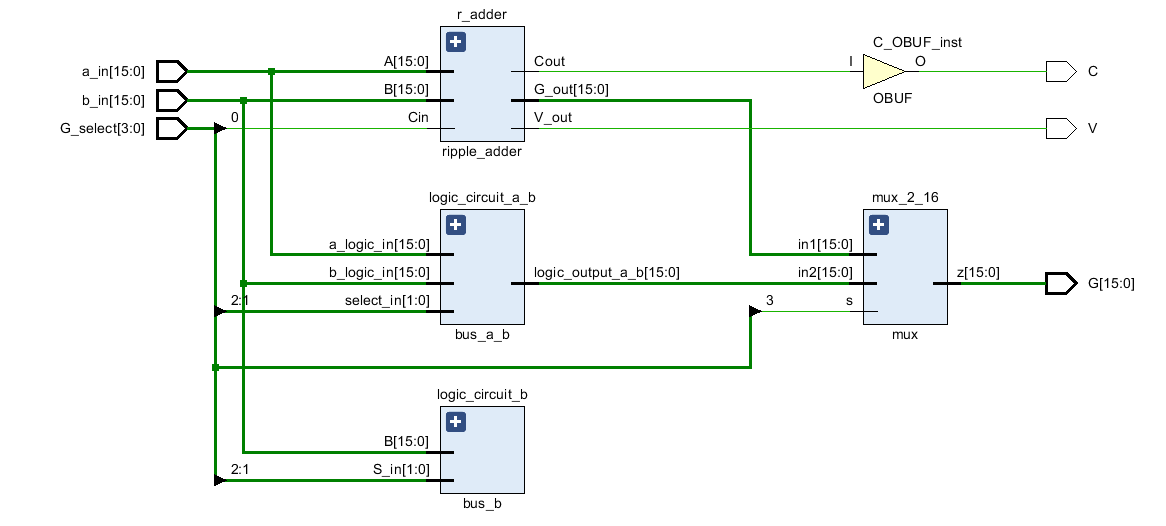
in2 => logic\_output\_a\_b,

s => G\_select(3),

z => G

);

end Behavioral;



d) Decoder 3-8 Bit

library IEEE;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity decoder is

port(A1, A2, A3: in std\_logic;

D1, D2, D3, D4, D5, D6, D7, D8: out std\_logic);

end decoder;

architecture Behavioral of decoder is

begin

D1<=((not A1) and (not A2) and (not A3)) after 10ns ;

D2<=((not A1) and (not A2) and (A3)) after 10ns ;

D3<=((not A1) and (A2) and (not A3)) after 10ns ;

D4<=((not A1) and (A2) and (A3)) after 10ns ;

D5<=((A1) and (not A2) and (not A3)) after 10ns ;

D6<=((A1) and (not A2) and (A3)) after 10ns ;

D7<=((A1) and (A2) and (not A3)) after 10ns ;

D8<=((A1) and (A2) and (A3)) after 10ns ;

end Behavioral;

e) Full Adder

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity full\_adder is

Port(

X, Y, Cin : in STD\_LOGIC;

Cout, S : out STD\_LOGIC

);

end full\_adder;

architecture Behavioral of full\_adder is

signal S1, S2, S3 : STD\_LOGIC;

begin

S1 <= (X xor Y) after 1ns;

S2 <= (Cin and S1) after 1ns;

S3 <= (X and Y) after 1ns;

S <= (S1 xor Cin) after 1ns;

Cout <= (S2 or S3) after 1ns;

end Behavioral;

f) Functional Unit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity function\_unit is

Port(

FunctionSelect : in STD\_LOGIC\_VECTOR(4 downto 0); -- 5 input

a\_in, b\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

N\_fu, Z\_fu, V\_fu, C\_fu : out STD\_LOGIC;

F : out STD\_LOGIC\_VECTOR(15 downto 0));

end function\_unit;

architecture Behavioral of function\_unit is

Component mux

Port(

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

s : in STD\_LOGIC; z : out STD\_LOGIC\_VECTOR(15 downto 0));

End Component;

Component shifter

Port(

B : in STD\_LOGIC\_VECTOR(15 downto 0);

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

IL, IR : in STD\_LOGIC;

H : out STD\_LOGIC\_VECTOR(15 downto 0));

End Component;

Component alu\_unit

Port(

a\_in, b\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

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

V, C : out STD\_LOGIC; -- flags

G : out STD\_LOGIC\_VECTOR(15 downto 0) );

End Component;

signal H\_out, ALU\_out, mux\_out : STD\_LOGIC\_VECTOR(15 downto 0);

begin

shifter0: shifter PORT MAP(

B => b\_in, S => FunctionSelect(3 downto 2),

IL => '0', IR => '0', H => H\_out );

mux\_2\_16: mux PORT MAP(

in1 => ALU\_out, in2 => H\_out,

s => FunctionSelect(4), z => mux\_out );

alu: alu\_unit PORT MAP(

a\_in => a\_in, b\_in => b\_in,

G\_select => FunctionSelect(3 downto 0),

V => V\_fu, C => C\_fu, G => ALU\_out );

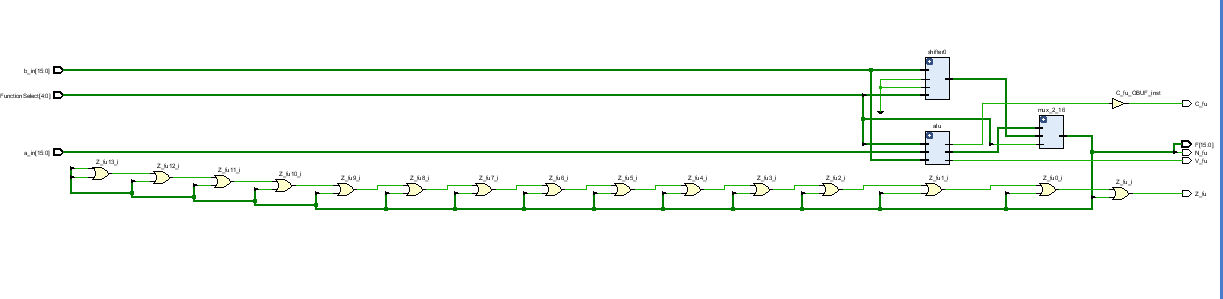
F <= mux\_out; N\_fu <= mux\_out(15);

Z\_fu <= (mux\_out(15) or mux\_out(14) or mux\_out(13) or mux\_out(12) or mux\_out(11)

or mux\_out(10) or mux\_out(9) or mux\_out(8) or mux\_out(7) or mux\_out(6)

or mux\_out(5) or mux\_out(4) or mux\_out(3) or mux\_out(2) or mux\_out(1) or mux\_out(0));

end Behavioral;



g) Logic Circuit A-B

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity bus\_a\_b is

Port(

a\_logic\_in, b\_logic\_in : in STD\_LOGIC\_VECTOR(15 downto 0);

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

logic\_output\_a\_b : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end bus\_a\_b;

architecture Behavioral of bus\_a\_b is

begin

logic\_output\_a\_b <= (a\_logic\_in and b\_logic\_in) after 10ns when select\_in = "00" else (a\_logic\_in or b\_logic\_in) after 10ns when select\_in = "01" else

(a\_logic\_in xor b\_logic\_in) after 10ns when select\_in = "10" else

(not (a\_logic\_in)) after 1ns;

end Behavioral;

h) Logic Circuit B

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity bus\_b is

Port(

B : in STD\_LOGIC\_VECTOR(15 downto 0);

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

Y\_out : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end bus\_b;

architecture Behavioral of bus\_b is

Component mux\_2\_1

Port(

B\_i, S1, S2 : in STD\_LOGIC;

Y\_i : out STD\_LOGIC

);

End Component;

begin

mux1: mux\_2\_1 PORT MAP(

B\_i => B(0),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(0)

);

mux2: mux\_2\_1 PORT MAP(

B\_i => B(1),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(1)

);

mux3: mux\_2\_1 PORT MAP(

B\_i => B(2),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(2)

);

mux4: mux\_2\_1 PORT MAP(

B\_i => B(3),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(3)

);

mux5: mux\_2\_1 PORT MAP(

B\_i => B(4),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(4)

);

mux6: mux\_2\_1 PORT MAP(

B\_i => B(5),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(5)

);

mux7: mux\_2\_1 PORT MAP(

B\_i => B(6),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(6)

);

mux8: mux\_2\_1 PORT MAP(

B\_i => B(7),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(7)

);

mux9: mux\_2\_1 PORT MAP(

B\_i => B(8),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(8)

);

mux10: mux\_2\_1 PORT MAP(

B\_i => B(9),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(9)

);

mux11: mux\_2\_1 PORT MAP(

B\_i => B(10),

S1 => S\_in(0),

S2 => S\_in(1),

Y\_i => Y\_out(10)

);

end Behavioral;

i) Mux 2-1 Bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux\_2\_1 is

Port(

B\_i, S1, S2 : in STD\_LOGIC;

Y\_i : out STD\_LOGIC );

end mux\_2\_1;

architecture Behavioral of mux\_2\_1 is

begin

Y\_i <= S1 after 1ns when B\_i = '1' else

S2 after 1ns when B\_i = '0' else

'0' after 1ns;

end Behavioral;

j) Mux 2-16 Bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux is

Port(

s : in STD\_LOGIC;

in1, in2 : in STD\_LOGIC\_VECTOR(15 downto 0);

z : out STD\_LOGIC\_VECTOR(15 downto 0) );

end mux;

architecture Behavioral of mux is

begin

z <= in1 after 1ns when s = '0' else

in2 after 1ns when s = '1' else

x"0000" after 1ns;

end Behavioral;

k) Mux 3-1 Bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux\_3\_1 is

Port(

in1, in2, in3 : in STD\_LOGIC; s1, s2 : in STD\_LOGIC;

z : out STD\_LOGIC );

end mux\_3\_1;

architecture Behavioral of mux\_3\_1 is

begin

z <= in1 after 1ns when s1 = '0' and s2 = '0' else

in2 after 1ns when s1 = '0' and s2 = '1' else

in3 after 1ns when s1 = '1' and s2 = '0' else

'0' after 1ns;

end Behavioral;

l) Mux 8-16 Bit

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity multiplexer is

Port ( s1, s2, s3 : in STD\_LOGIC;

in1, in2, in3, in4, in5, in6, in7, in8 : in STD\_LOGIC\_VECTOR (15 downto 0);

z : out STD\_LOGIC\_VECTOR (15 downto 0));

end multiplexer;

architecture Behavioral of multiplexer is

begin

z <= in1 after 1ns when s1 = '0' and s2 = '0' and s3 = '0' else

in2 after 1ns when s1 = '0' and s2 = '0' and s3 = '1' else

in3 after 1ns when s1 = '0' and s2 = '1' and s3 = '0' else

in4 after 1ns when s1 = '0' and s2 = '1' and s3 = '1' else

in5 after 1ns when s1 = '1' and s2 = '0' and s3 = '0' else

in6 after 1ns when s1 = '1' and s2 = '0' and s3 = '1' else

in7 after 1ns when s1 = '1' and s2 = '1' and s3 = '0' else

in8 after 1ns when s1 = '1' and s2 = '1' and s3 = '1' else

x"0000" after 10ns;

end Behavioral;

m) Register

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity reg16 is

Port(

D : in STD\_LOGIC\_VECTOR(15 downto 0);

load1, load2, Clk : in STD\_LOGIC;

Q : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end reg16;

architecture Behavioral of reg16 is

begin

process (Clk)

begin

if(rising\_edge(Clk)) then

if((load1 = '1') and (load2 = '1')) then

Q <= D after 5ns;

end if;

end if;

end process;

end Behavioral;

n) Ripple Adder

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity shifter is

Port(

B : in STD\_LOGIC\_VECTOR(15 downto 0);

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

IL, IR : in STD\_LOGIC;

H : out STD\_LOGIC\_VECTOR(15 downto 0) );

end shifter;

architecture Behavioral of shifter is

Component mux\_3\_1

Port(

in1, in2, in3, s1, s2 : in STD\_LOGIC;

z : out STD\_LOGIC );

End Component;

begin

mux1: mux\_3\_1 PORT MAP(

in1 => B(0),

in2 => B(1),

in3 => IL,

s1 => S(0),

s2 => S(1),

z => H(0)

);

mux2: mux\_3\_1 PORT MAP(

in1 => B(1),

in2 => B(2),

in3 => B(0),

s1 => S(0),

s2 => S(1),

z => H(1) );

mux3: mux\_3\_1 PORT MAP(

in1 => B(2),

in2 => B(3),

in3 => B(1),

s1 => S(0),

s2 => S(1),

z => H(2) );

mux4: mux\_3\_1 PORT MAP(

in1 => B(3),

in2 => B(4),

in3 => B(2),

s1 => S(0),

s2 => S(1),

z => H(3) );

mux5: mux\_3\_1 PORT MAP(

in1 => B(4),

in2 => B(5),

in3 => B(3),

s1 => S(0),

s2 => S(1),

z => H(4) );

mux6: mux\_3\_1 PORT MAP(

in1 => B(5),

in2 => B(6),

in3 => B(4),

s1 => S(0),

s2 => S(1),

z => H(5) );

mux7: mux\_3\_1 PORT MAP(

in1 => B(6),

in2 => B(7),

in3 => B(5),

s1 => S(0),

s2 => S(1),

z => H(6) );

mux8: mux\_3\_1 PORT MAP(

in1 => B(7),

in2 => B(8),

in3 => B(6),

s1 => S(0),

s2 => S(1),

z => H(7) );

mux9: mux\_3\_1 PORT MAP(

in1 => B(8),

in2 => B(9),

in3 => B(7),

s1 => S(0),

s2 => S(1),

z => H(8) );

mux10: mux\_3\_1 PORT MAP(

in1 => B(9),

in2 => B(10),

in3 => B(8),

s1 => S(0),

s2 => S(1),

z => H(9) );

mux11: mux\_3\_1 PORT MAP(

in1 => B(10),

in2 => B(11),

in3 => B(9),

s1 => S(0),

s2 => S(1),

z => H(10) );

mux12: mux\_3\_1 PORT MAP(

in1 => B(11),

in2 => B(12),

in3 => B(10),

s1 => S(0),

s2 => S(1),

z => H(11) );

mux13: mux\_3\_1 PORT MAP(

in1 => B(12),

in2 => B(13),

in3 => B(11),

s1 => S(0),

s2 => S(1),

z => H(12) );

mux14: mux\_3\_1 PORT MAP(

in1 => B(13),

in2 => B(14),

in3 => B(12),

s1 => S(0),

s2 => S(1),

z => H(13));

mux15: mux\_3\_1 PORT MAP(

in1 => B(14),

in2 => B(15),

in3 => B(13),

s1 => S(0),

s2 => S(1),

z => H(14) );

mux16: mux\_3\_1 PORT MAP(

in1 => B(15),

in2 => IR,

in3 => B(14),

s1 => S(0),

s2 => S(1),

z => H(15) );

end Behavioral;

o) Shifter

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity shifter is

Port(

B : in STD\_LOGIC\_VECTOR(15 downto 0);

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

IL, IR : in STD\_LOGIC;

H : out STD\_LOGIC\_VECTOR(15 downto 0) );

end shifter;

architecture Behavioral of shifter is

Component mux\_3\_1

Port(

in1, in2, in3, s1, s2 : in STD\_LOGIC;

z : out STD\_LOGIC );

End Component;

begin

mux1: mux\_3\_1 PORT MAP(

in1 => B(0),

in2 => B(1),

in3 => IL,

s1 => S(0),

s2 => S(1),

z => H(0) );

mux2: mux\_3\_1 PORT MAP(

in1 => B(1),

in2 => B(2),

in3 => B(0),

s1 => S(0),

s2 => S(1),

z => H(1)

);

mux3: mux\_3\_1 PORT MAP(

in1 => B(2),

in2 => B(3),

in3 => B(1),

s1 => S(0),

s2 => S(1),

z => H(2)

);

mux4: mux\_3\_1 PORT MAP(

in1 => B(3),

in2 => B(4),

in3 => B(2),

s1 => S(0),

s2 => S(1),

z => H(3)

);

mux5: mux\_3\_1 PORT MAP(

in1 => B(4),

in2 => B(5),

in3 => B(3),

s1 => S(0),

s2 => S(1),

z => H(4)

);

mux6: mux\_3\_1 PORT MAP(

in1 => B(5),

in2 => B(6),

in3 => B(4),

s1 => S(0),

s2 => S(1),

z => H(5)

);

mux7: mux\_3\_1 PORT MAP(

in1 => B(6),

in2 => B(7),

in3 => B(5),

s1 => S(0),

s2 => S(1),

z => H(6)

);

mux8: mux\_3\_1 PORT MAP(

in1 => B(7),

in2 => B(8),

in3 => B(6),

s1 => S(0),

s2 => S(1),

z => H(7)

);

mux9: mux\_3\_1 PORT MAP(

in1 => B(8),

in2 => B(9),

in3 => B(7),

s1 => S(0),

s2 => S(1),

z => H(8)

);

mux10: mux\_3\_1 PORT MAP(

in1 => B(9),

in2 => B(10),

in3 => B(8),

s1 => S(0),

s2 => S(1),

z => H(9)

);

mux11: mux\_3\_1 PORT MAP(

in1 => B(10),

in2 => B(11),

in3 => B(9),

s1 => S(0),

s2 => S(1),

z => H(10)

);

mux12: mux\_3\_1 PORT MAP(

in1 => B(11),

in2 => B(12),

in3 => B(10),

s1 => S(0),

s2 => S(1),

z => H(11)

);

mux13: mux\_3\_1 PORT MAP(

in1 => B(12),

in2 => B(13),

in3 => B(11),

s1 => S(0),

s2 => S(1),

z => H(12)

);

mux14: mux\_3\_1 PORT MAP(

in1 => B(13),

in2 => B(14),

in3 => B(12),

s1 => S(0),

s2 => S(1),

z => H(13)

);

mux15: mux\_3\_1 PORT MAP(

in1 => B(14),

in2 => B(15),

in3 => B(13),

s1 => S(0),

s2 => S(1),

z => H(14)

);

mux16: mux\_3\_1 PORT MAP(

in1 => B(15),

in2 => IR,

in3 => B(14),

s1 => S(0),

s2 => S(1),

z => H(15)

);

end Behavioral;

**3. Component Test Benches**

a) Project 1B Top Level

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Proj\_1b\_TB IS

END Proj\_1b\_TB;

ARCHITECTURE behavior OF Proj\_1b\_TB IS

COMPONENT Proj1b

PORT(

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

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

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

Clk\_sig : IN std\_logic;

data\_out : OUT std\_logic\_vector(15 downto 0);

address\_out : OUT std\_logic\_vector(15 downto 0);

N\_out : OUT std\_logic;

Z\_out : OUT std\_logic;

C\_out : OUT std\_logic;

V\_out : OUT std\_logic

);

END COMPONENT;

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

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

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

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

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

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

signal N\_out : std\_logic;

signal Z\_out : std\_logic;

signal C\_out : std\_logic;

signal V\_out : std\_logic;

constant Clk\_sig\_period : time := 10 ns;

BEGIN

uut: Proj1b PORT MAP (

data\_in => data\_in, constant\_in => constant\_in,

control\_word => control\_word, Clk\_sig => Clk\_sig,

data\_out => data\_out, address\_out => address\_out,

N\_out => N\_out, Z\_out => Z\_out, C\_out => C\_out,

V\_out => V\_out );

Clk\_sig\_process :process

begin

Clk\_sig <= '0';

wait for Clk\_sig\_period/2;

Clk\_sig <= '1';

wait for Clk\_sig\_period/2;

end process;

stim\_proc: process

begin

data\_in <= x"FFFF";

constant\_in <= x"0000";

control\_word <= "00000000100000011"; wait for 40ns;

data\_in <= x"AAAA";

control\_word <= "00100000100000011"; wait for 40ns;

control\_word <= "01000000100110001"; wait for 40ns;

control\_word <= "01001001001000000";

wait;

end process;

END;

b) Register File

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY reg2\_TB IS

END reg2\_TB;

ARCHITECTURE behavior OF reg2\_TB IS

COMPONENT reg2

PORT(

des\_D : IN std\_logic\_vector(2 downto 0);

add\_a : IN std\_logic\_vector(2 downto 0);

add\_b : IN std\_logic\_vector(2 downto 0);

Clk : IN std\_logic;

load\_in : IN std\_logic;

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

out\_data\_a : OUT std\_logic\_vector(15 downto 0);

out\_data\_b : OUT std\_logic\_vector(15 downto 0)

);

END COMPONENT;

signal des\_D : std\_logic\_vector(2 downto 0) := (others => '0');

signal add\_a : std\_logic\_vector(2 downto 0) := (others => '0');

signal add\_b : std\_logic\_vector(2 downto 0) := (others => '0');

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

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

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

signal out\_data\_a : std\_logic\_vector(15 downto 0);

signal out\_data\_b : std\_logic\_vector(15 downto 0);

constant Clk\_period : time := 10ns;

BEGIN

uut: reg2 PORT MAP (

des\_D => des\_D,

add\_a => add\_a,

add\_b => add\_b,

Clk => Clk,

load\_in => load\_in,

data => data,

out\_data\_a => out\_data\_a,

out\_data\_b => out\_data\_b

);

Clk\_process :process

begin

Clk <= '0';

wait for Clk\_period/2;

Clk <= '1';

wait for Clk\_period/2;

end process;

stim\_proc: process

begin

wait;

end process;

END;

c) ALU Unit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY ALU\_TB IS

END ALU\_TB;

ARCHITECTURE behavior OF ALU\_TB IS

COMPONENT alu\_unit

PORT(

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

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

G\_select : IN std\_logic\_vector(3 downto 0);

V : OUT std\_logic;

C : OUT std\_logic;

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

);

END COMPONENT;

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

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

signal G\_select : std\_logic\_vector(3 downto 0) := (others => '0');

signal V : std\_logic;

signal C : std\_logic;

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

BEGIN

uut: alu\_unit PORT MAP (

a\_in => a\_in,

b\_in => b\_in,

G\_select => G\_select,

V => V,

C => C,

G => G

);

stim\_proc: process

begin

a\_in <= x"FFAA";

b\_in <= x"000F";

G\_select <= "0000";

wait for 100ns;

G\_select <= "0001";

wait for 100ns;

G\_select <= "0010";

wait for 100ns;

G\_select <= "0010";

wait for 100ns;

G\_select <= "0011";

wait for 100ns;

G\_select <= "0100";

wait for 100ns;

G\_select <= "0101";

wait for 100ns;

G\_select <= "0110";

wait for 100ns;

G\_select <= "0111";

wait;

end process;

END;

d) Decoder 3-8 Bit

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity decoder\_tb is

end;

architecture behavior of decoder\_tb is

component decoder

port(A1, A2, A3: in std\_logic;

D1, D2, D3, D4, D5, D6, D7, D8: out std\_logic);

end component;

signal A1, A2, A3: std\_logic;

signal D1, D2, D3, D4, D5, D6, D7, D8: std\_logic;

begin

uut: decoder port map ( A1 => A1,A2 => A2, A3 => A3,

D1 => D1,D2 => D2,D3 => D3,D4 => D4,D5 => D5,D6 => D6,D7 => D7,D8 => D8);

stimulus: process

begin

wait for 10ns; A1 <= '0';A2 <= '0';A3 <= '0';

wait for 10ns; A1 <= '0';A2 <= '0';A3 <= '1';

wait for 10ns; A1 <= '0';A2 <= '1';A3 <= '0';

wait for 10ns; A1 <= '0';A2 <= '1';A3 <= '1';

wait for 10ns; A1 <= '1';A2 <= '0';A3 <= '0';

wait for 10ns; A1 <= '1';A2 <= '0';A3 <= '1';

wait for 10ns; A1 <= '1';A2 <= '1';A3 <= '0';

wait for 10ns; A1 <= '1';A2 <= '1';A3 <= '1';

wait;

end process;

end;

e) Full Adder

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY full\_adder\_TB IS

END full\_adder\_TB;

ARCHITECTURE behavior OF full\_adder\_TB IS

COMPONENT full\_adder

PORT(

X : IN std\_logic; Y : IN std\_logic; Cin : IN std\_logic; Cout : OUT std\_logic;

S : OUT std\_logic );

END COMPONENT;

signal X : std\_logic := '0'; signal Y : std\_logic := '0';

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

BEGIN

uut: full\_adder PORT MAP (

X => X, Y => Y,

Cin => Cin, Cout => Cout,

S => S );

stim\_proc: process

begin

wait for 15ns; X <= '1';

wait for 15ns; X <= '0'; Y <= '1';

wait for 15ns; X <= '1';

wait for 15ns; Cin <= '1';

wait for 15ns; Y <= '0';

wait for 15ns; X <= '0';

wait;

end process;

END;

f) Function Unit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY function\_unit\_tb IS

END function\_unit\_tb;

ARCHITECTURE behavior OF function\_unit\_tb IS

COMPONENT function\_unit

PORT(

FunctionSelect : IN std\_logic\_vector(4 downto 0);

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

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

N\_fu : OUT std\_logic;

Z\_fu : OUT std\_logic;

V\_fu : OUT std\_logic;

C\_fu : OUT std\_logic;

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

);

END COMPONENT;

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

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

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

signal N\_fu : std\_logic;

signal Z\_fu : std\_logic;

signal V\_fu : std\_logic;

signal C\_fu : std\_logic;

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

BEGIN

uut: function\_unit PORT MAP (

FunctionSelect => FunctionSelect,

a\_in => a\_in,

b\_in => b\_in,

N\_fu => N\_fu,

Z\_fu => Z\_fu,

V\_fu => V\_fu,

C\_fu => C\_fu,

F => F

);

stim\_proc: process

begin

a\_in <= x"AAAA"; b\_in <= x"BBBB";

wait for 10ns; FunctionSelect <= "00000";

wait for 10ns; FunctionSelect <= "00001";

wait for 10ns; FunctionSelect <= "00010";

wait for 10ns; FunctionSelect <= "00011";

wait for 10ns; FunctionSelect <= "00100";

wait for 10ns; FunctionSelect <= "00101";

wait for 10ns; FunctionSelect <= "00110";

wait for 10ns; FunctionSelect <= "00111";

wait for 10ns; FunctionSelect <= "01000";

wait for 10ns; FunctionSelect <= "01010";

wait for 10ns; FunctionSelect <= "01100";

wait for 10ns; FunctionSelect <= "01110";

wait for 10ns; FunctionSelect <= "10000";

wait for 10ns; FunctionSelect <= "10100";

wait for 10ns; FunctionSelect <= "11000";

wait;

end process;

END;

g) Logic Circuit A-B

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY bus\_a\_b\_tb IS

END bus\_a\_b\_tb;

ARCHITECTURE behavior OF bus\_a\_b\_tb IS

COMPONENT bus\_a\_b

PORT(

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

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

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

logic\_output\_a\_b : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

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

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

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

BEGIN

uut: bus\_a\_b PORT MAP (

a\_logic\_in => a\_logic\_in, b\_logic\_in => b\_logic\_in,

select\_in => select\_in,

logic\_output\_a\_b => logic\_output\_a\_b );

stim\_proc: process

begin

wait for 20ns; a\_logic\_in <= x"FFFF"; b\_logic\_in <= x"9999";select\_in <= "00"; wait for 10ns; select\_in <= "01"; wait for 10ns; select\_in <= "10";

wait for 10ns;select\_in <= "11";

wait;

end process;

END;

h) Logic Circuit B

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY bus\_b\_tb IS

END bus\_b\_tb;

ARCHITECTURE behavior OF bus\_b\_tb IS

COMPONENT bus\_b

PORT(

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

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

Y\_out : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

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

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

BEGIN

uut: bus\_b PORT MAP (

B => B,

S\_in => S\_in,

Y\_out => Y\_out );

stim\_proc: process

begin

B <= x"AAAA"; S\_in <= "00";

wait for 10ns; S\_in <= "01";

wait for 10ns; S\_in <= "10";

wait for 10ns; S\_in <= "11";

wait;

end process;

END;

i) Mux 2-1 Bit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY mux\_n\_TB IS

END mux\_n\_TB;

ARCHITECTURE behavior OF mux\_n\_TB IS

COMPONENT mux\_2\_1

PORT(

B\_i : IN std\_logic;

S1 : IN std\_logic;

S2 : IN std\_logic;

Y\_i : OUT std\_logic );

END COMPONENT;

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

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

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

signal Y\_i : std\_logic;

BEGIN

uut: mux\_2\_1 PORT MAP (

B\_i => B\_i,

S1 => S1,

S2 => S2,

Y\_i => Y\_i );

stim\_proc: process

begin

S1 <= '1'; S2 <= '0';

wait for 5ns; B\_i <= '1';

wait for 5ns; B\_i <= '0';

wait;

end process;

END;

j) Mux 2-16 bit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY mux\_tb IS

END mux\_tb;

ARCHITECTURE behavior OF mux\_tb IS

COMPONENT mux

PORT(

in1 : IN std\_logic\_vector(15 downto 0);

in2 : IN std\_logic\_vector(15 downto 0);

s : IN std\_logic;

z : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

signal in1 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in2 : std\_logic\_vector(15 downto 0) := (others => '0');

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

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

BEGIN

mux\_2\_16: mux PORT MAP (

in1 => in1, in2 => in2,

s => s, z => z );

stim\_proc: process

begin

wait for 1ns; in1 <= x"FFFF"; in2 <= x"AAAA";

wait for 1ns; s <= '1';

wait for 1ns; s <= '0';

wait for 1ns; s <= '1';

end process;

END;

k) Mux 3-1 Bit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY mux\_3\_1\_TB IS

END mux\_3\_1\_TB;

ARCHITECTURE behavior OF mux\_3\_1\_TB IS

COMPONENT mux\_3\_1

PORT(

In1 : IN std\_logic; In2 : IN std\_logic; In3 : IN std\_logic;

S1 : IN std\_logic; S2 : IN std\_logic;

Z : OUT std\_logic );

END COMPONENT;

signal In1 : std\_logic := '0'; signal In2 : std\_logic := '0'; signal In3 : std\_logic := '0';

signal S1 : std\_logic := '0'; signal S2 : std\_logic := '0';

signal Z : std\_logic;

BEGIN

uut: mux\_3\_1 PORT MAP (

In1 => In1, In2 => In2, In3 => In3,

S1 => S1, S2 => S2,

Z => Z );

stim\_proc: process

begin

wait for 5ns; In1 <= '1'; In2 <= '0'; In3 <= '1';

wait for 5ns; S1 <= '0'; S2 <= '1';

wait for 5ns; S1 <= '1'; S2 <= '0';

wait for 5ns; S1 <= '1'; S2 <= '1';

wait;

end process;

END;

l) Mux 8-16 Bit

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY multiplexer\_tb IS

END multiplexer\_tb;

ARCHITECTURE behavior OF multiplexer\_tb IS

COMPONENT multiplexer

PORT(

s1: IN std\_logic;

s2: IN std\_logic;

s3: IN std\_logic;

in1 : IN std\_logic\_vector(15 downto 0);

in2 : IN std\_logic\_vector(15 downto 0);

in3 : IN std\_logic\_vector(15 downto 0);

in4 : IN std\_logic\_vector(15 downto 0);

in5 : IN std\_logic\_vector(15 downto 0);

in6 : IN std\_logic\_vector(15 downto 0);

in7 : IN std\_logic\_vector(15 downto 0);

in8 : IN std\_logic\_vector(15 downto 0);

z : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

signal s1 : std\_logic := '0';

signal s2 : std\_logic := '0';

signal s3 : std\_logic := '0';

signal in1 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in2 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in3 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in4 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in5 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in6 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in7 : std\_logic\_vector(15 downto 0) := (others => '0');

signal in8 : std\_logic\_vector(15 downto 0) := (others => '0');

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

BEGIN

uut: multiplexer PORT MAP (

s1 => s1,s2 => s2,s3 => s3,

in1 => in1,in2 => in2,in3 => in3,in4 => in4,in5 => in5,in6 => in6,in7 => in7,in8 => in8,

z => z

);

stim\_proc: process

begin

in1 <= x"FFFF";

in2 <= x"EEEE";

in3 <= x"DDDD";

in4 <= x"CCCC";

in5 <= x"BBBB";

in6 <= x"AAAA";

in7 <= x"9999";

in8 <= x"8888";

wait for 10ns; s1 <= '1';s2 <= '0';s3 <= '0';

wait for 10ns; s1 <= '0';s2 <= '1';s3 <= '0';

wait for 10ns; s1 <= '1';s2 <= '1';s3 <= '0';

wait for 10ns; s1 <= '0';s2 <= '0';s3 <= '1';

wait for 10ns; s1 <= '1';s2 <= '0';s3 <= '1';

wait for 10ns; s1 <= '0';s2 <= '1';s3 <= '1';

wait for 10ns; s1 <= '1';s2 <= '1';s3 <= '1';

end process;

END;

m) Register

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY reg16\_TB IS

END reg16\_TB;

ARCHITECTURE behavior OF reg16\_TB IS

COMPONENT reg16

PORT(

D : IN std\_logic\_vector(15 downto 0); load1 : IN std\_logic; load2 : IN std\_logic;

Clk : IN std\_logic; Q : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

signal load1 : std\_logic := '0'; signal load2 : std\_logic := '0';

signal Clk : std\_logic := '0'; signal Q : std\_logic\_vector(15 downto 0);

constant Clk\_period : time := 10 ns;

BEGIN

uut: reg16 PORT MAP (

D => D, load1 => load1, load2 => load2, Clk => Clk, Q => Q );

Clk\_process :process

begin

Clk <= '0'; wait for Clk\_period/2; Clk <= '1'; wait for Clk\_period/2;

end process;

stim\_proc: process

begin

D <= x"FFFF"; load1 <= '1'; load2 <= '1'; wait for 20ns;D <= x"AAAA"; load1 <= '0';

wait for 10ns; load2 <= '0'; wait for 10ns; load1 <= '1';load2 <= '1';

wait;

end process;

END;

n) Ripple Adder

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY ripple\_adder\_TB IS

END ripple\_adder\_TB;

ARCHITECTURE behavior OF ripple\_adder\_TB IS

COMPONENT ripple\_adder

PORT(

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

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

Cin : IN std\_logic; Cout : OUT std\_logic;

V\_out : OUT std\_logic; G\_out : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

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

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

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

BEGIN

uut: ripple\_adder PORT MAP (

A => A, B => B,

Cin => Cin, Cout => Cout,

V\_out => V\_out, G\_out => G\_out );

stim\_proc: process

begin

A <= x"AAAA"; B <= x"FBAA"; Cin <= '1';

wait for 80ns; A <= x"FFFF"; B <= x"0000"; Cin <= '1';

wait;

end process;

END;

o) Shifter

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY shifter\_tb IS

END shifter\_tb;

ARCHITECTURE behavior OF shifter\_tb IS

COMPONENT shifter

PORT(

B : IN std\_logic\_vector(15 downto 0); S : IN std\_logic\_vector(1 downto 0);

IL : IN std\_logic; IR : IN std\_logic;

H : OUT std\_logic\_vector(15 downto 0) );

END COMPONENT;

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

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

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

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

BEGIN

uut: shifter PORT MAP (

B => B, S => S, IL => IL, IR => IR,

H => H );

stim\_proc: process

begin

wait for 10ns; B <= x"FFFF"; S <= "00";

wait for 10ns; S <= "01";

wait for 10ns; S <= "11";

wait for 10ns; B <= H; S <= "10";

wait;

end process;

END;