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,
```

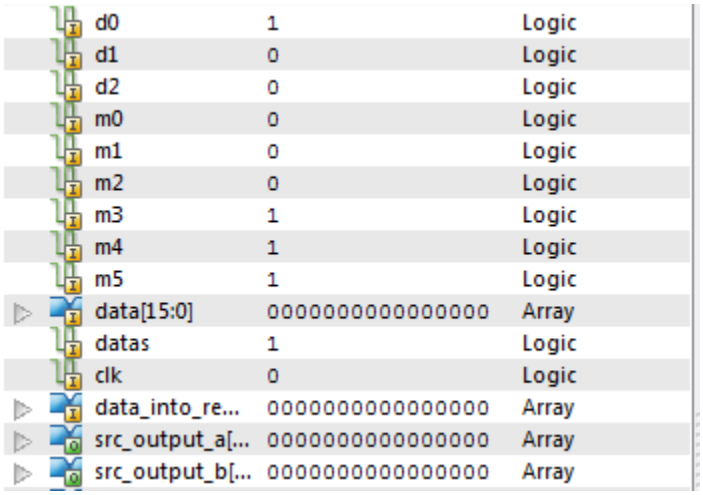
```
        z => z,
        Adrs_out => Adrs_out,
        Data_out_bus_b => Data_out_bus_b,

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

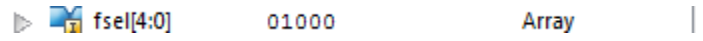
Shows the values of the control word being added into selects for registers

Control word = "00100011100100001"



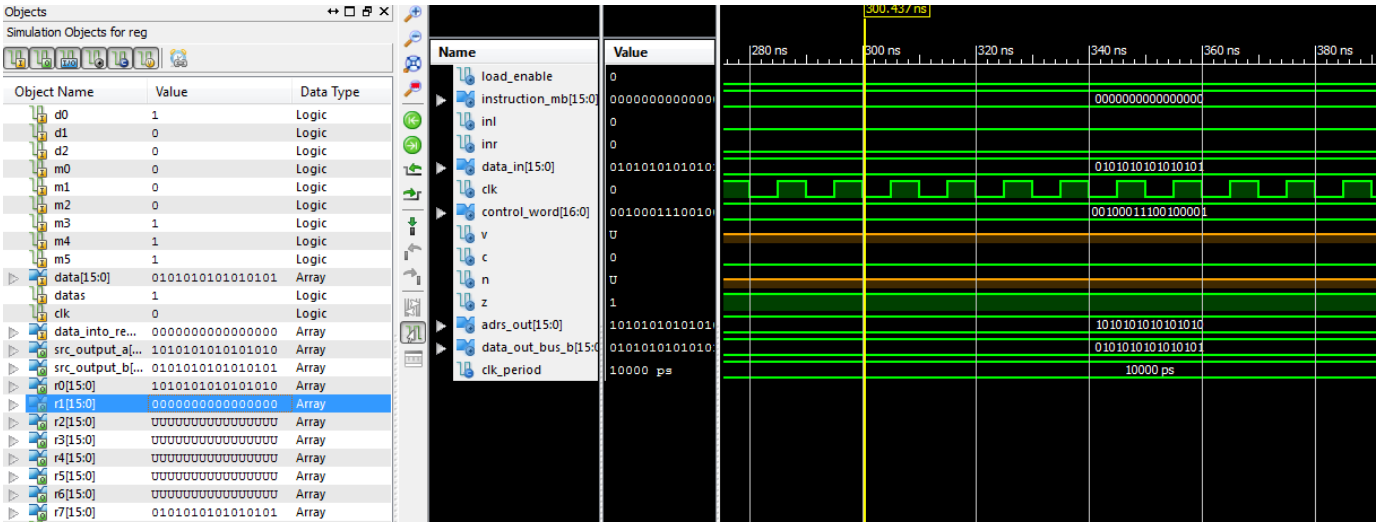
d0	1	Logic
d1	0	Logic
d2	0	Logic
m0	0	Logic
m1	0	Logic
m2	0	Logic
m3	1	Logic
m4	1	Logic
m5	1	Logic
data[15:0]	0000000000000000	Array
datas	1	Logic
clk	0	Logic
data_into_re...	0000000000000000	Array
src_output_a[...	0000000000000000	Array
src_output_b[...	0000000000000000	Array

Fs select being updated

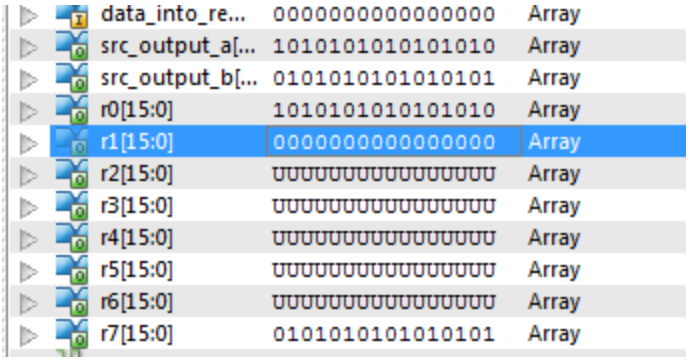


fsel[4:0]	01000	Array

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;
```

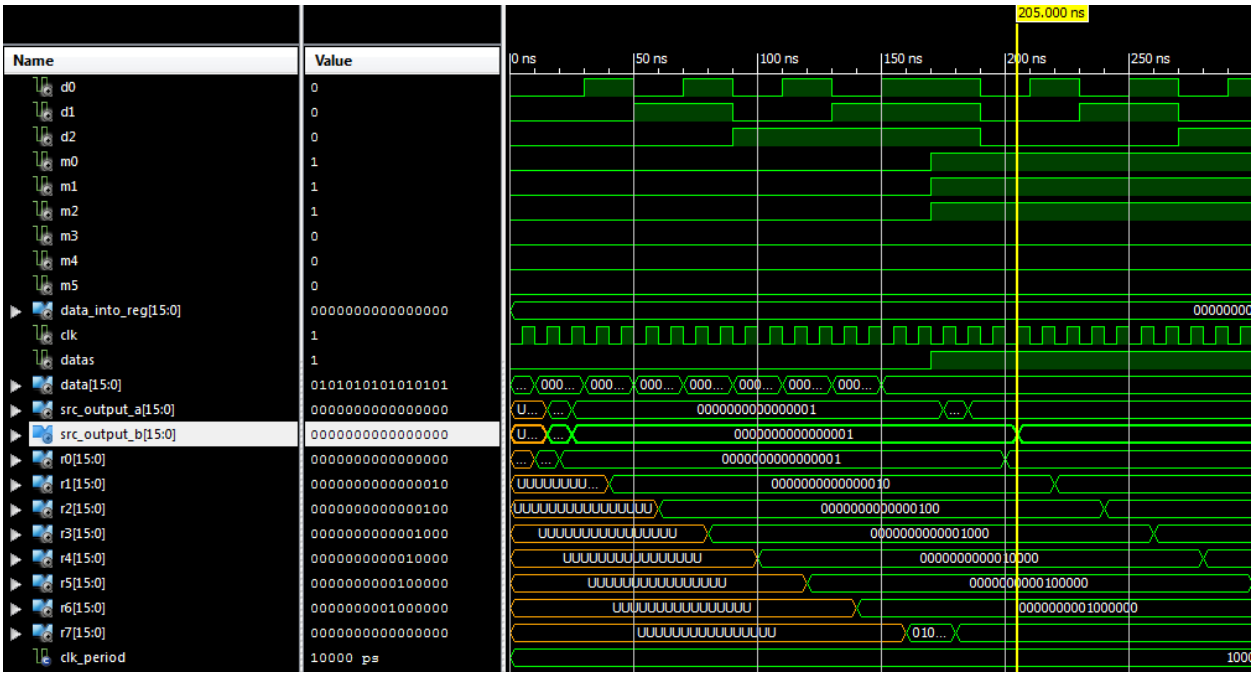
```
BEGIN

-- 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

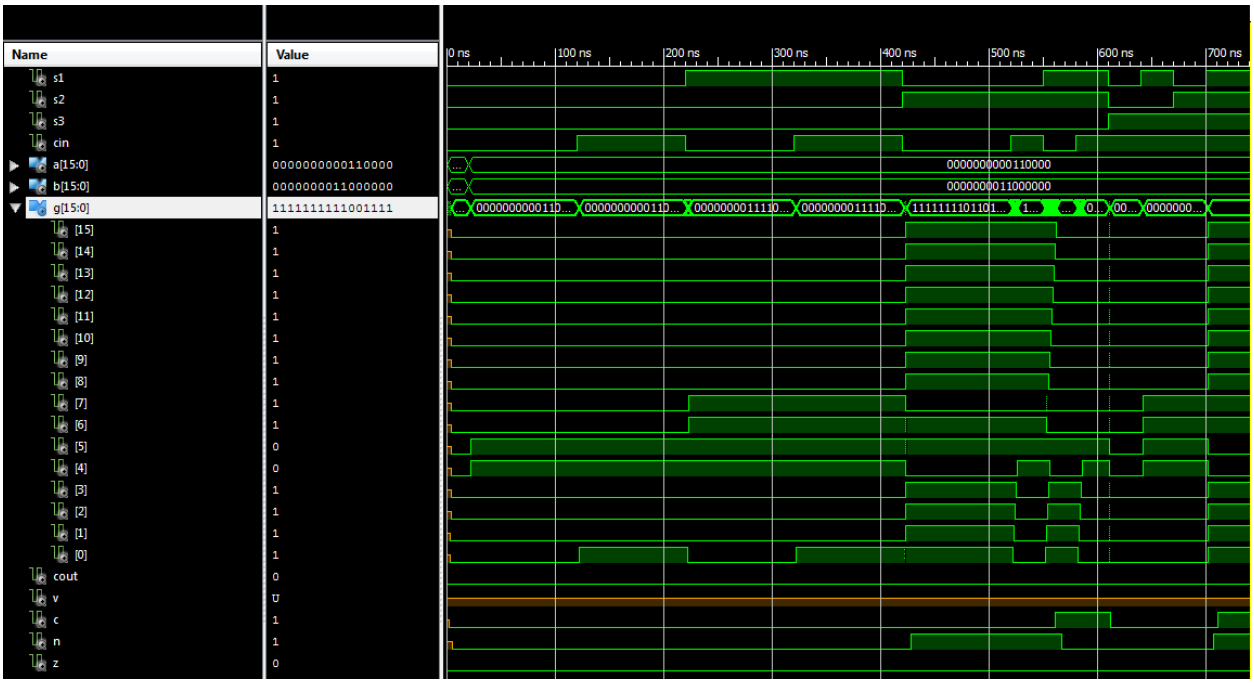
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

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



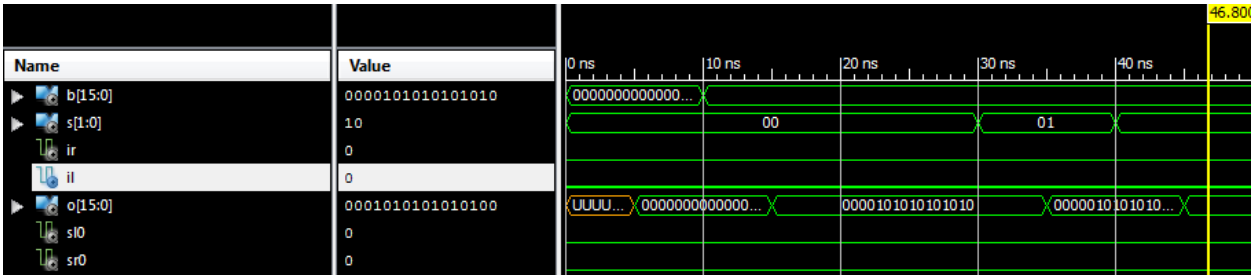
Shifter_test.vhd

```
-- 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

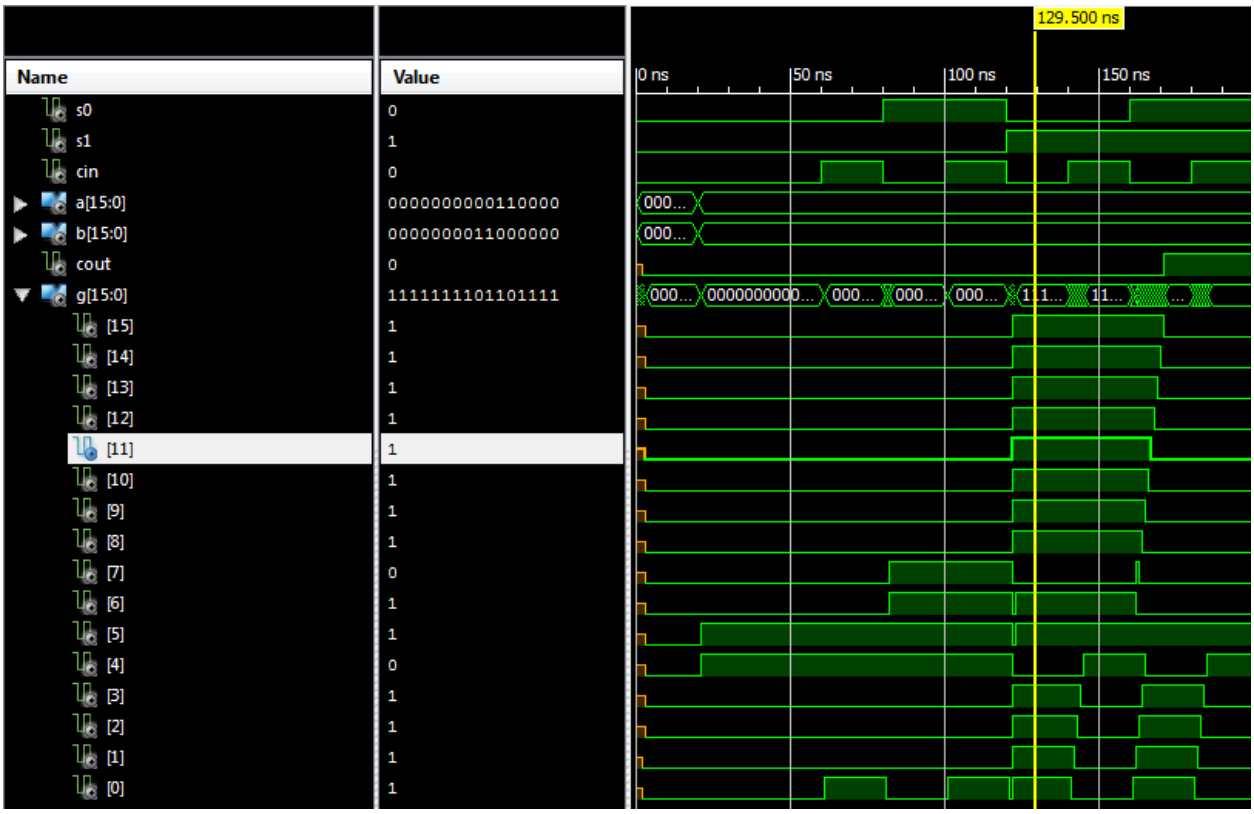


Arithmetic_circuit_test.vhd

```
-- 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;
```

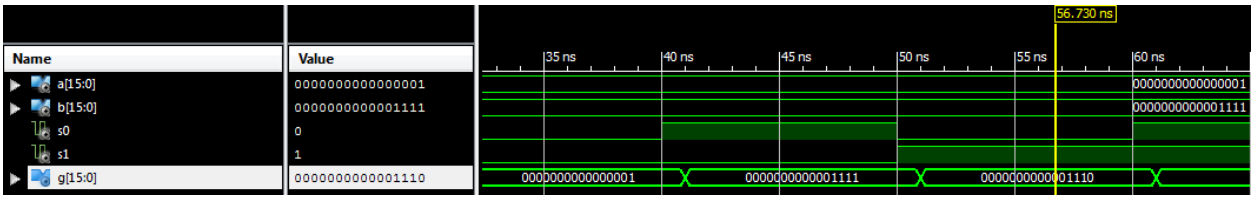
```
S1 <= '0';
S0 <= '1';
Cin <= '0';
wait for 20 ns;
Cin <= '1';
wait for 20 ns;
S1 <= '1';
S0 <= '0';
Cin <= '0';
wait for 20 ns;
Cin <= '1';
wait for 20 ns;
S1 <= '1';
S0 <= '1';
Cin <= '0';
wait for 20 ns;
Cin <= '1';
wait for 20 ns;
      -- Add user defined stimulus here


      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';
```

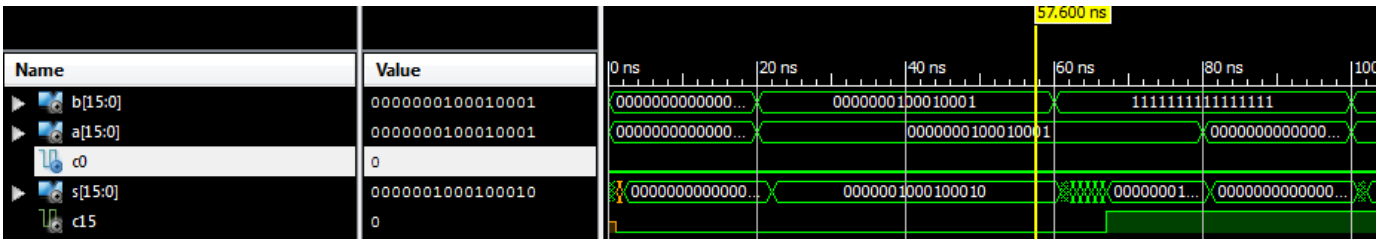
```
  signal S1 : std_logic := '0';


 --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;
```

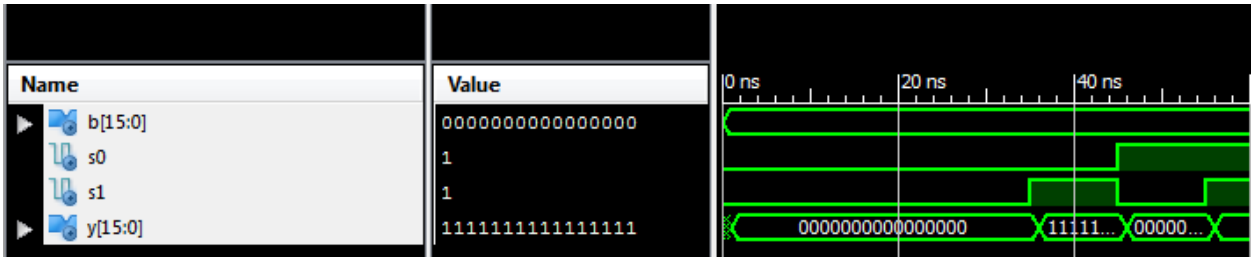


BInput_logic_test.vhd

```
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;
```

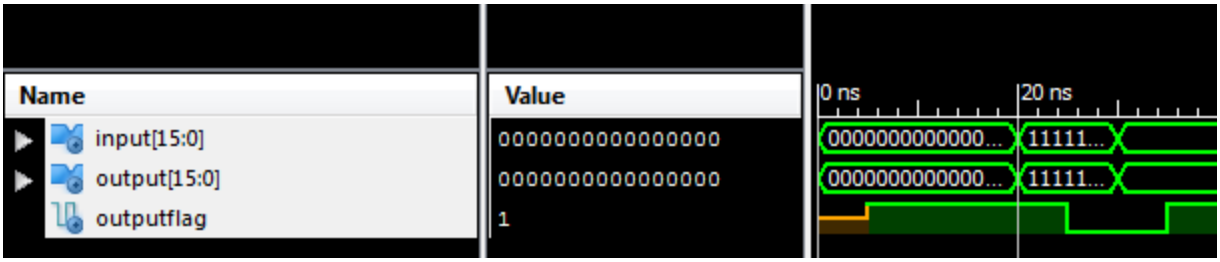
```
S0 <= '1';
S1 <= '0';
wait for 10 ns;
S0 <= '1';
S1 <= '1';
wait for 10 ns;
    -- insert stimulus here
    wait;
  end process;
END;
```



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)
```

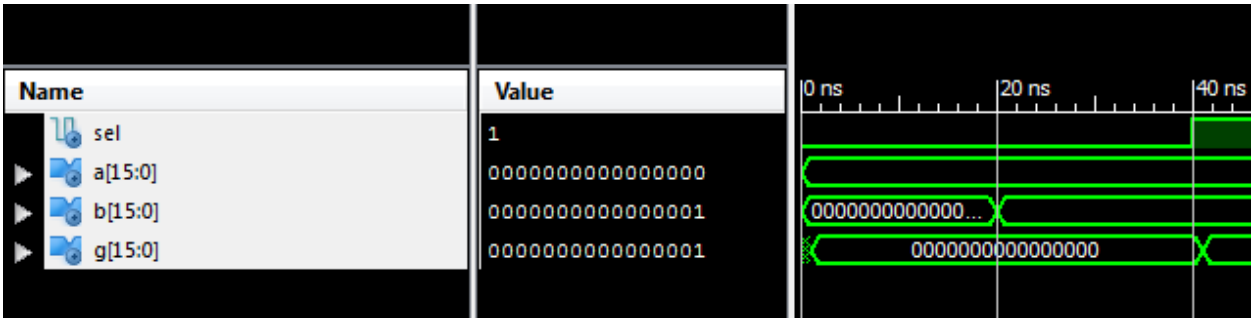
```
  uut: zeroDetect PORT MAP (
        input => input,
outputFlag => outputFlag,
        output => output
       );
  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 20 ns;
input <= "1111111111111111";
wait for 10 ns;

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;
```