CS2022 PROJECT 1- DATAPATH DESIGN

PART A

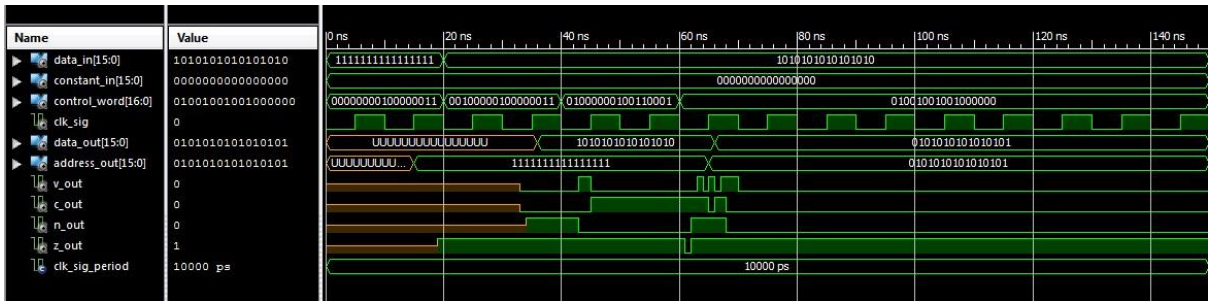
Aniket Agarwal
17317437

Contents:

1. Results of Test Benches

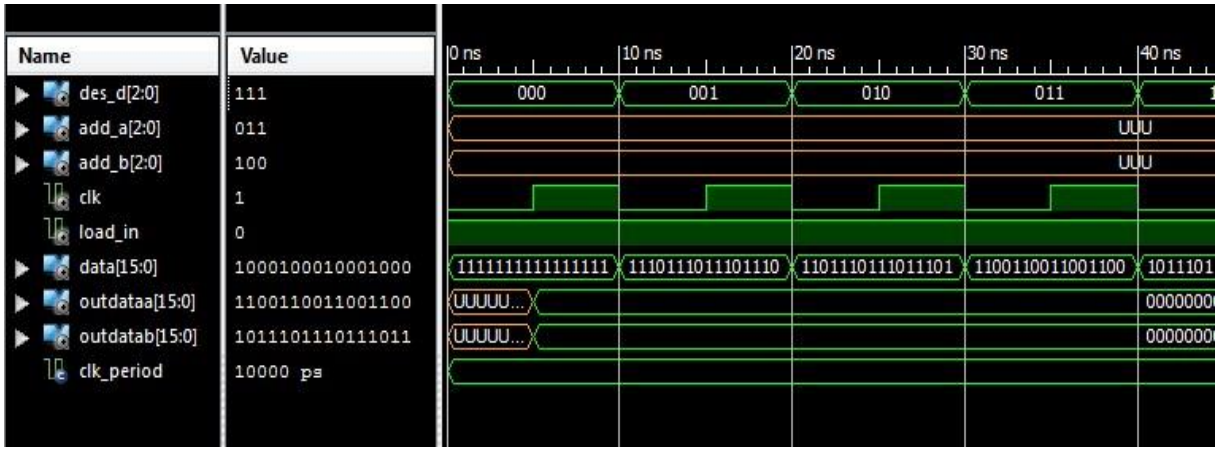
a) Project 1B Top Level

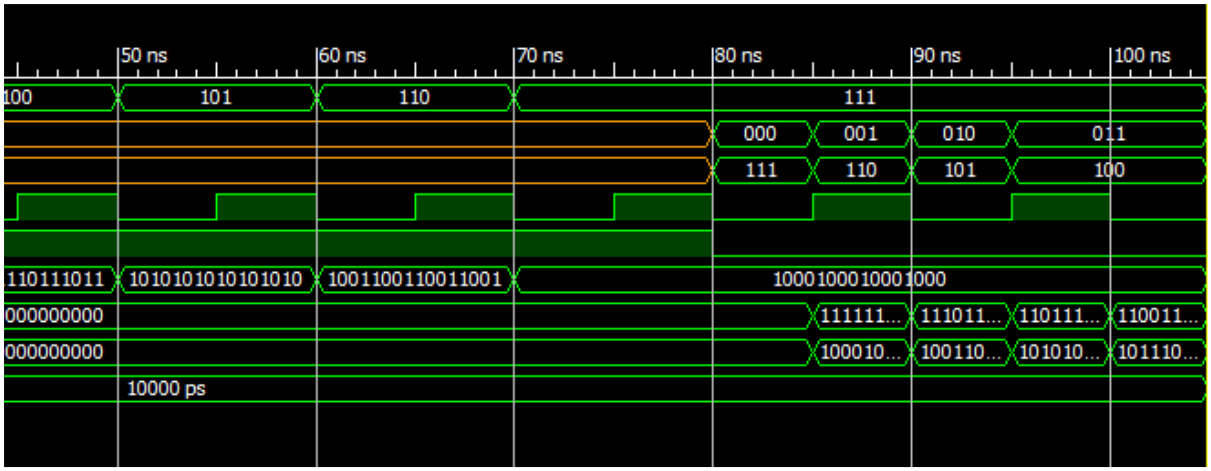
The results from this test-bench show that all the modules instantiated together with values are being loaded into the subsequent registers, and micro-operations are then being carried out appropriately with loading into a new register.



b) Register File

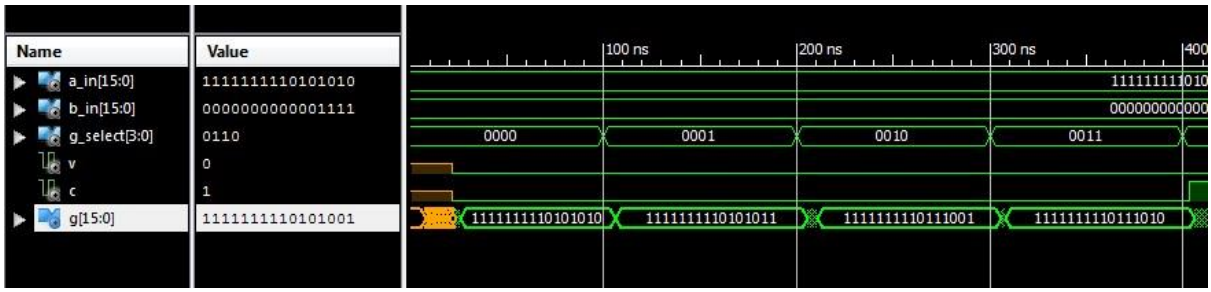
The register file test bench shows the register submodules working together through decoding, multiplexing, storing values and passing data in and out appropriately with respect to the clock period and the load.



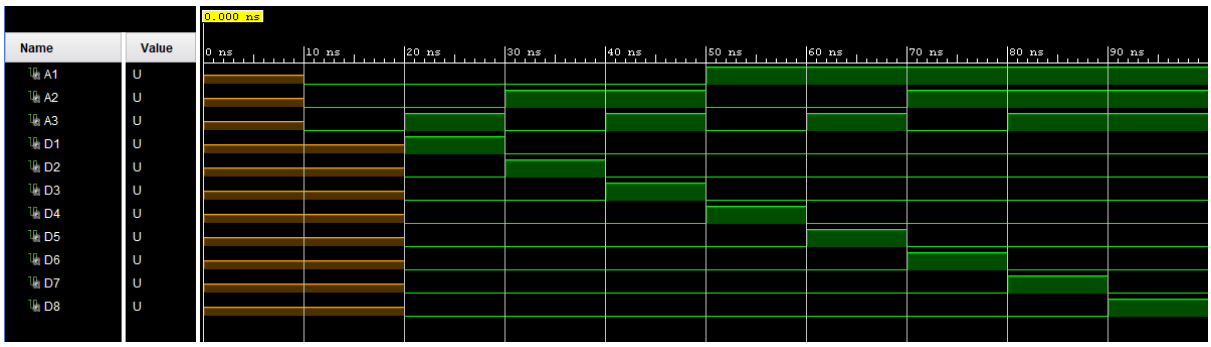


c) ALU Unit

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.

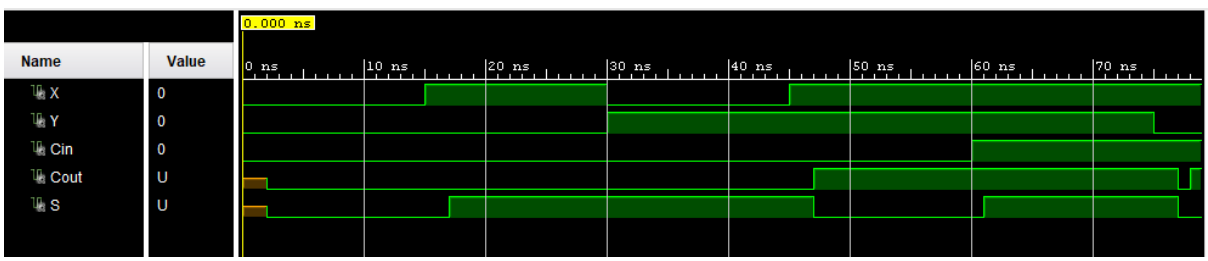


d) Decoder 3-8 bit

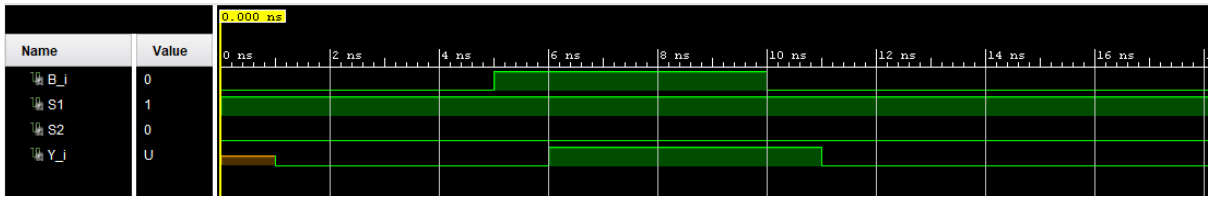


e) Full Adder

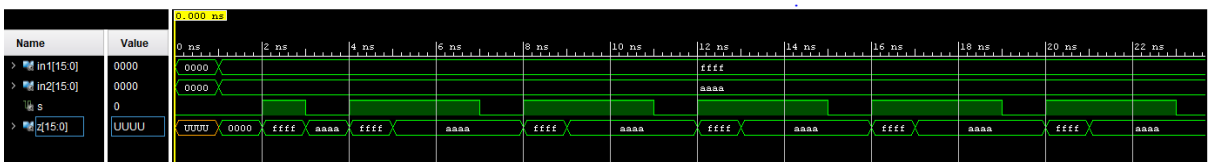
For each input, the appropriate carry is determined and correct operations are performed for the given select pin, which results in the intended output.



f) Mux 2-1 Bit

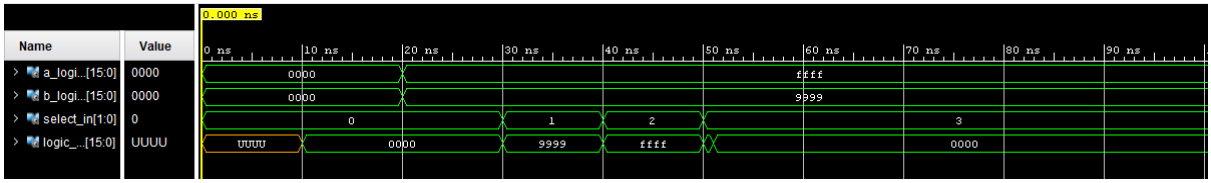


g) Mux 2-16 bit



h) Logic Circuit A-B

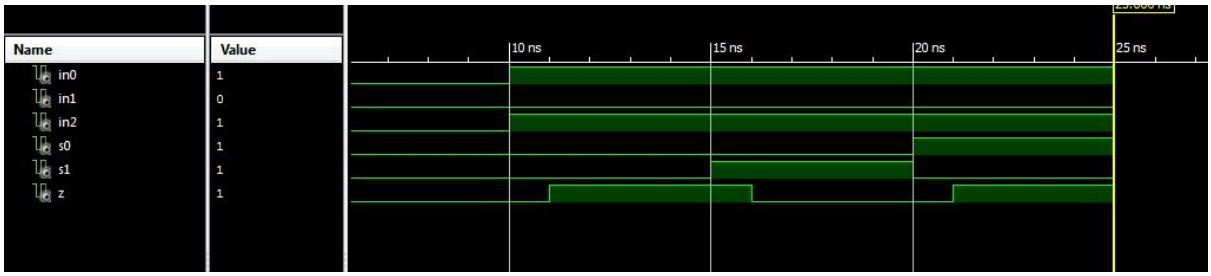
It provides inputs of A and B logic as well as a select pin that determines the operations to be performed resulting in outputs via gate logic.



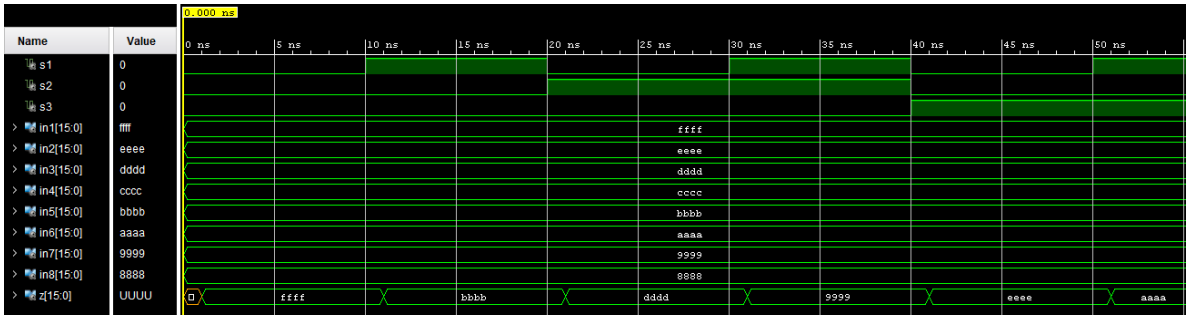
i) Logic Circuit B

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

j) Mux 3-1 bit

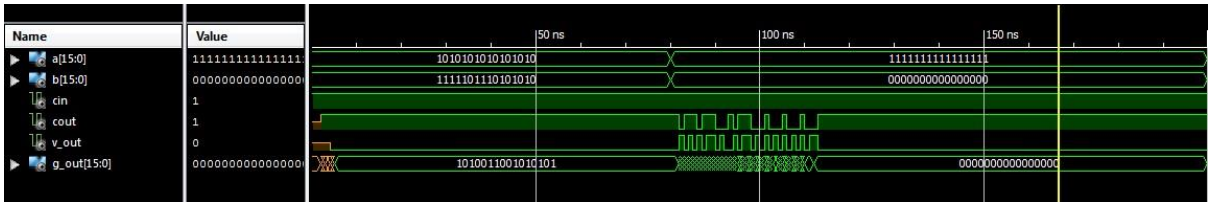


k) Mux 8-16 bit



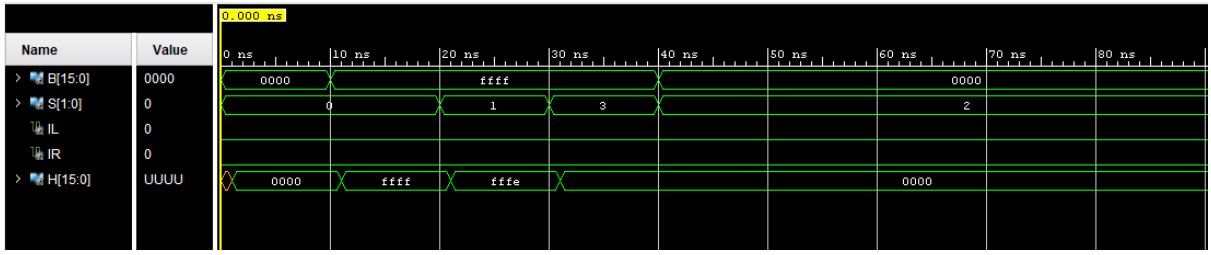
l) Ripple Adder

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.

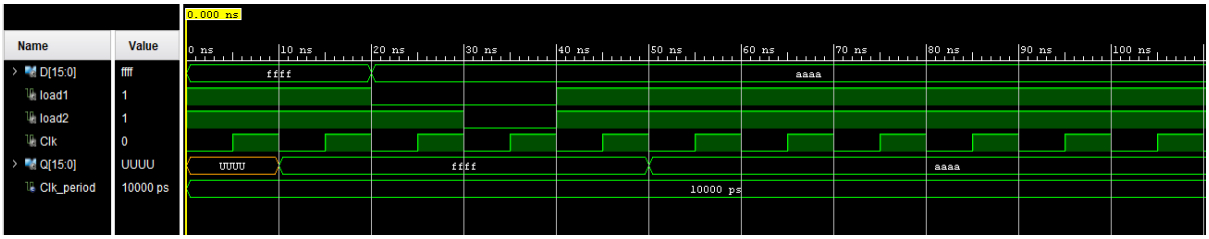


m) Shifter

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.

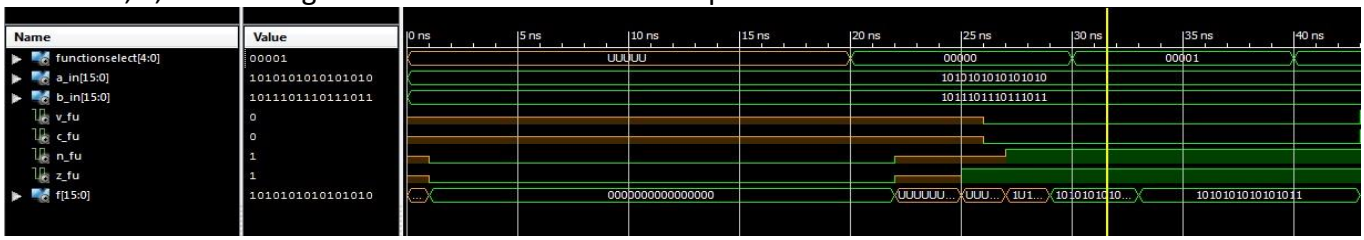


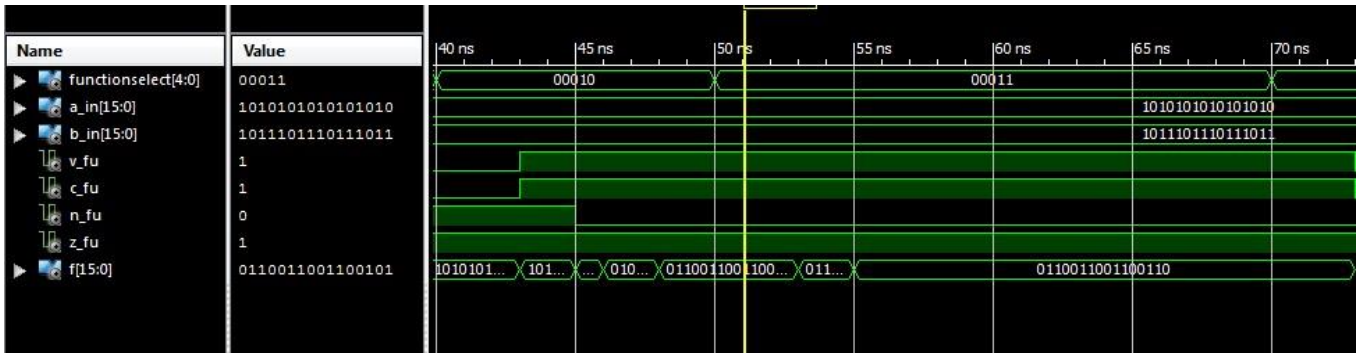
n) Register

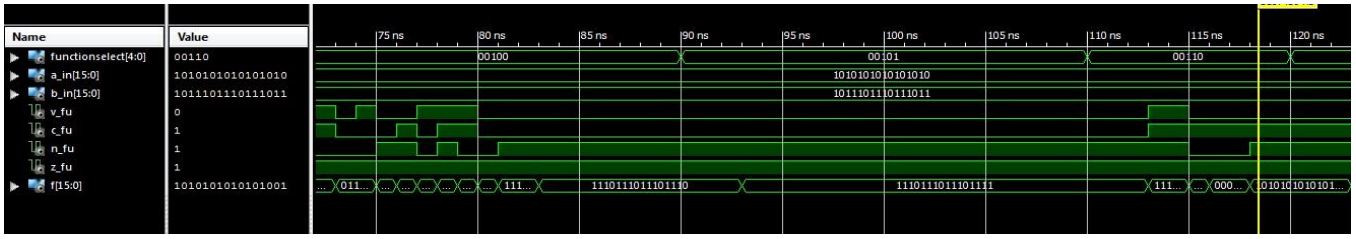


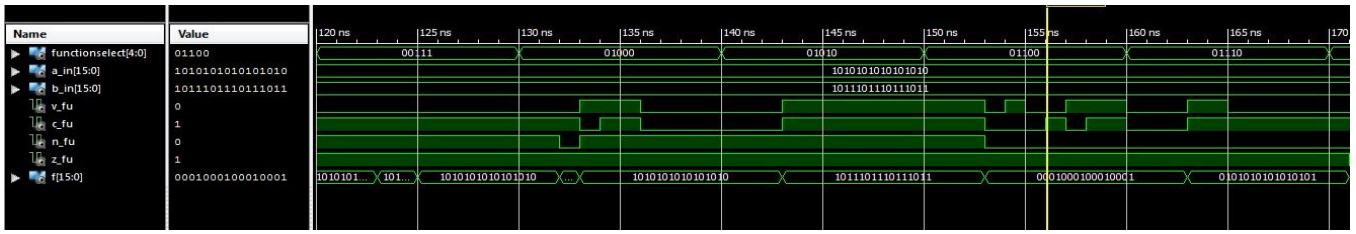
o) Functional Unit

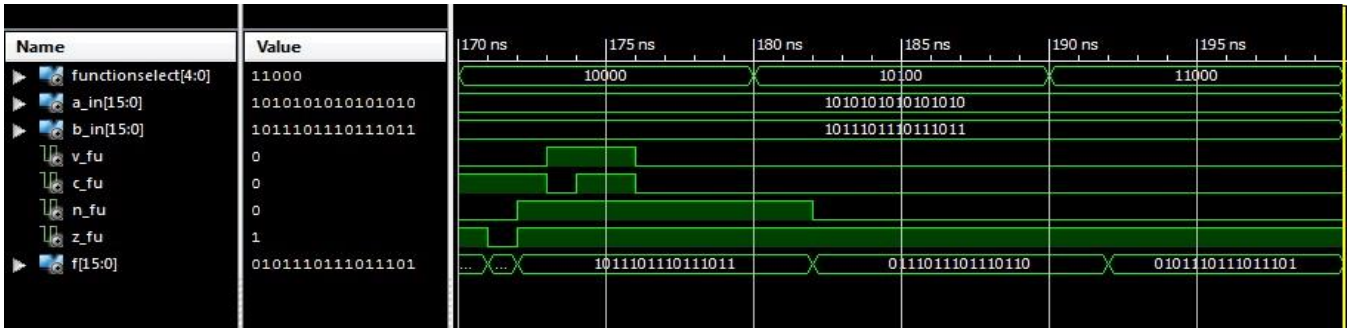
The function unit works as it should be with appropriate values being handled and raising correct N, Z, C and V fags in the unit itself to the word passed in for function selection.











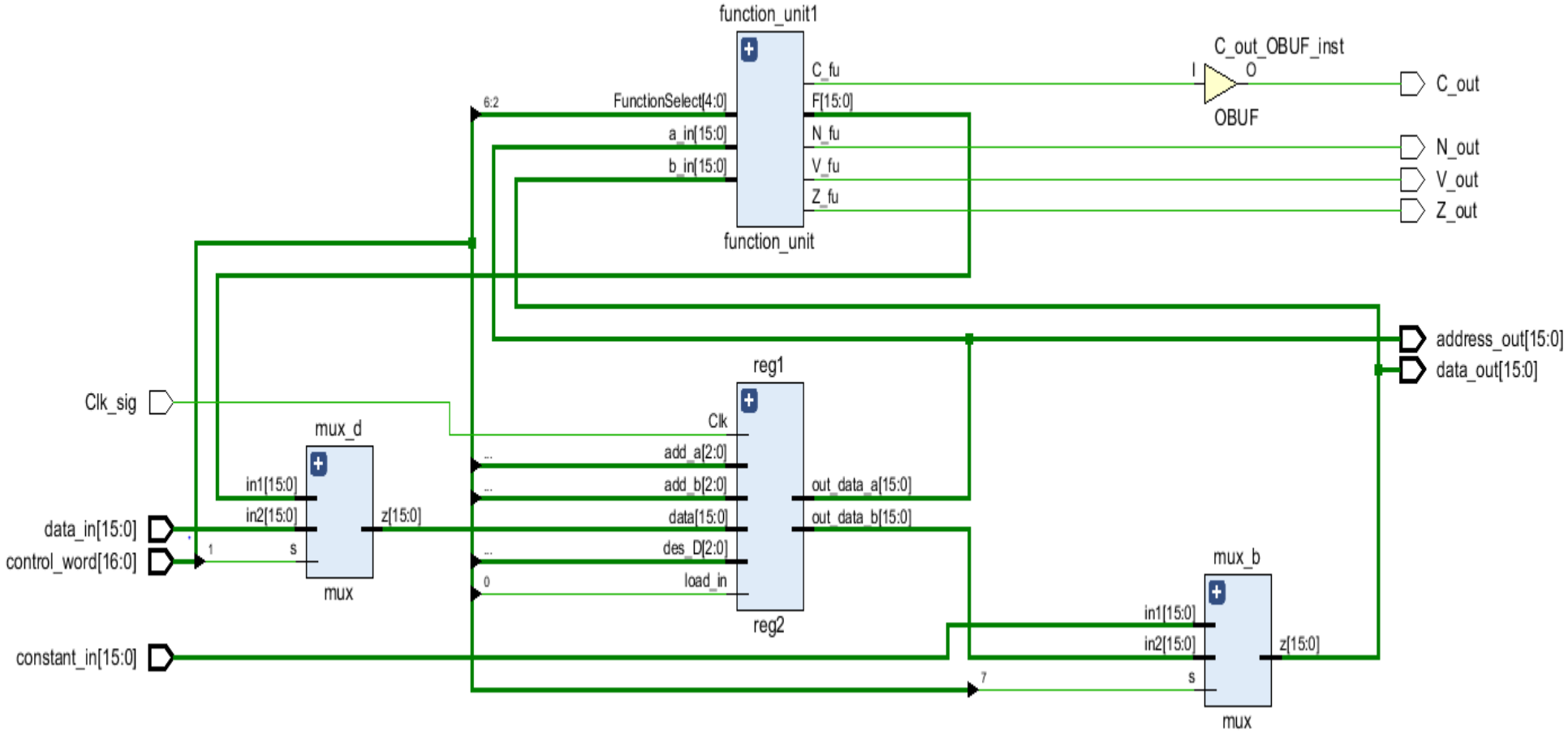
2. VHDL Component Source Code

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;
```


function_unit1
C_fu
F[15:0]
N_fu
V_fu
Z_fu
FunctionSelect[4:0]
a_in[15:0]
b_in[15:0]
6:2
function_unit
C_out_OBUF_inst
I
O
OBUF
C_out
N_out
V_out
Z_out
address_out[15:0]
data_out[15:0]
reg1
Clk
add_a[2:0]
add_b[2:0]
data[15:0]
des_D[2:0]
load_in
out_data_a[15:0]
out_data_b[15:0]
reg2
Clk_sig
mux_d
in1[15:0]
in2[15:0]
z[15:0]
s
mux
data_in[15:0]
control_word[16:0]
constant_in[15:0]
mux_b
in1[15:0]
in2[15:0]
z[15:0]
s
mux

b) Register File

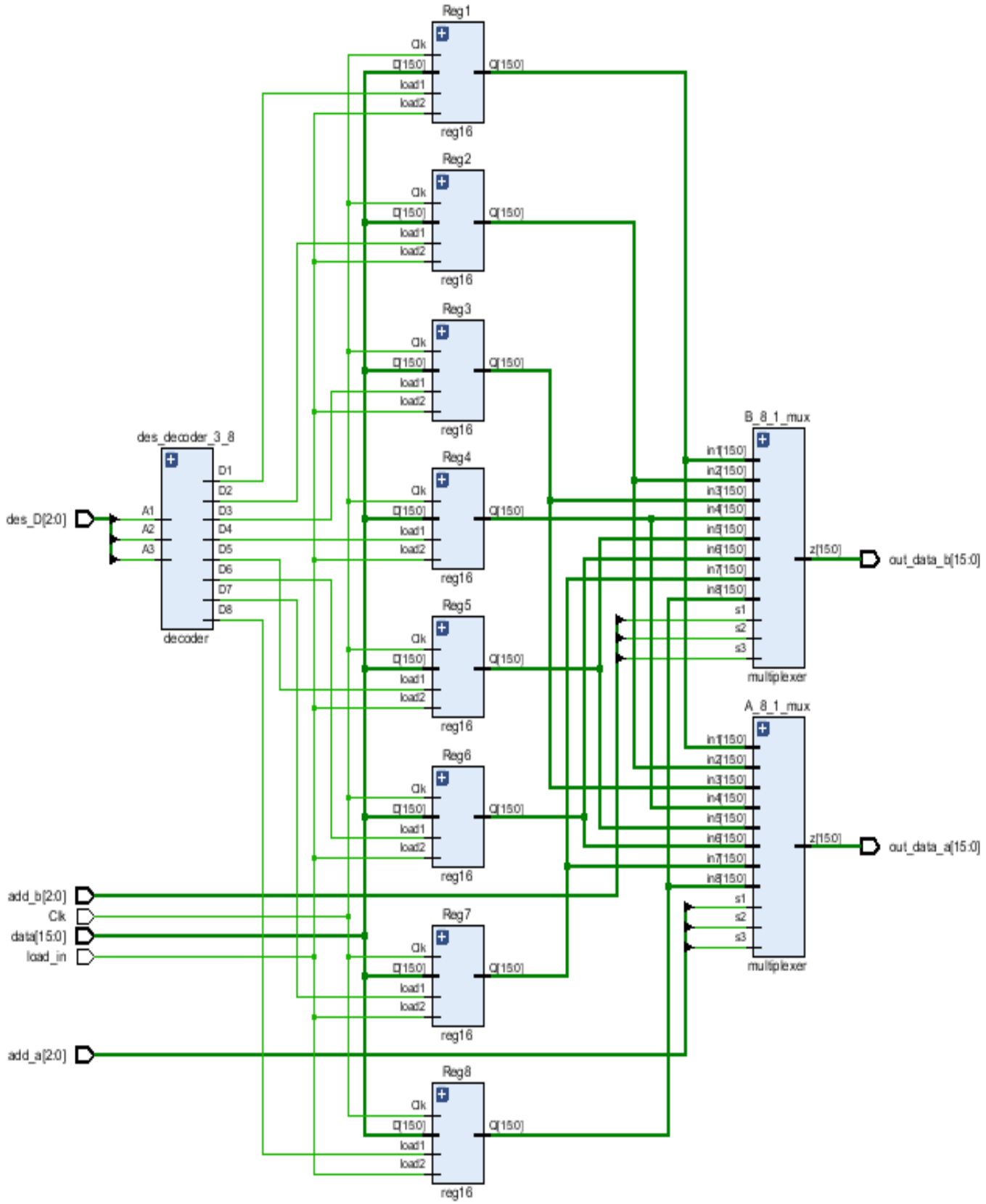
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity reg2 is
	Port(
			des_D, add_a, add_b : in STD_LOGIC_VECTOR(2 downto 0);
			Clk, load_in : in STD_LOGIC;
			data : in STD_LOGIC_VECTOR(15 downto 0);
			out_data_a, out_data_b : out STD_LOGIC_VECTOR(15 downto 0)
		);
end reg2;
architecture Behavioral 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;
```

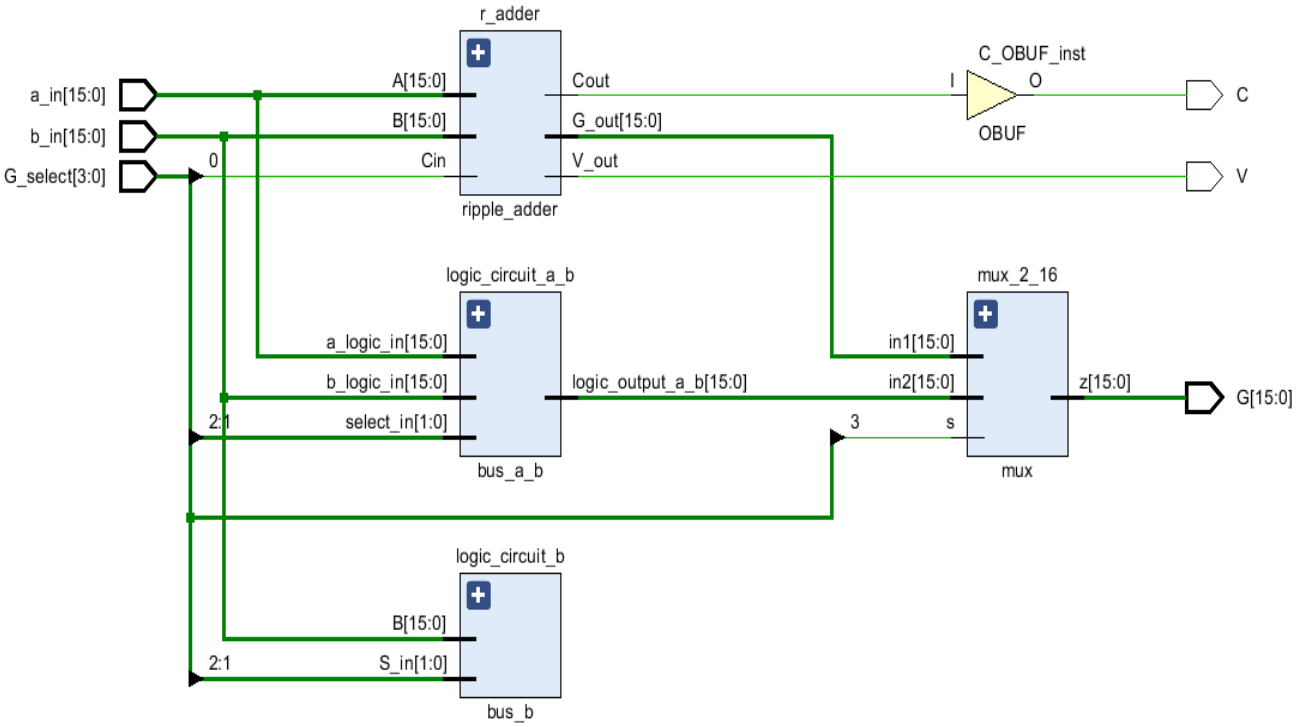

Reg1
Clk
D[15:0]
load1
load2
Q[15:0]
reg16
Reg2
Reg3
Reg4
Reg5
Reg6
Reg7
Reg8
des_decoder_3_8
A1
A2
A3
D1
D2
D3
D4
D5
D6
D7
D8
decoder
des_D[2:0]
B_8_1_mux
A_8_1_mux
in1[15:0]
in2[15:0]
in3[15:0]
in4[15:0]
in5[15:0]
in6[15:0]
in7[15:0]
in8[15:0]
s1
s2
s3
z[15:0]
multiplexer
out_data_b[15:0]
out_data_a[15:0]
add_b[2:0]
Clk
data[15:0]
load_in
add_a[2:0]

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;
```

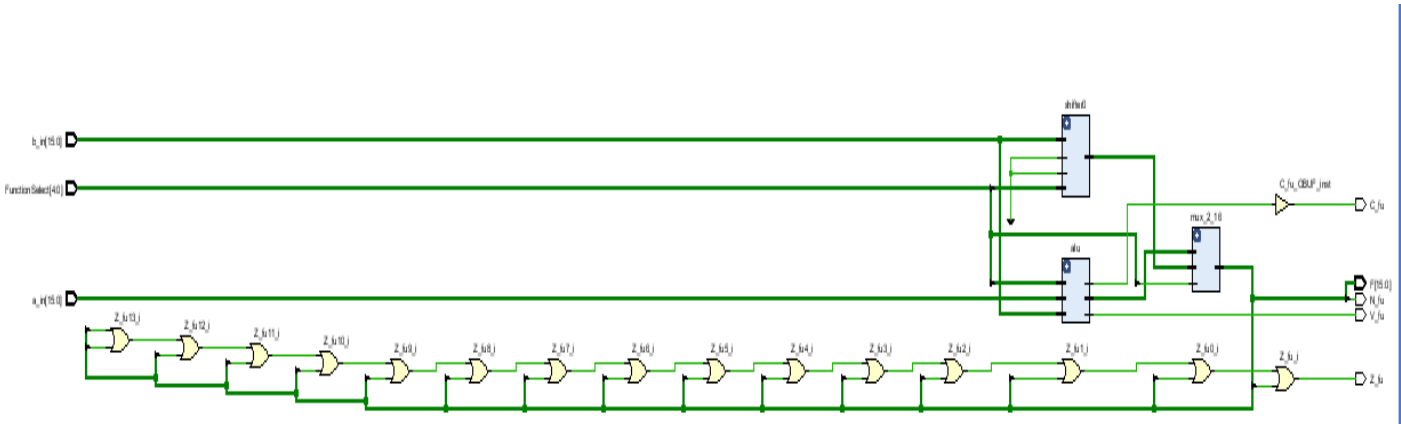
e) Full Adder

```
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;
```