University of Dublin,
Trinity College

Computer Architecture I (CS2022)

Project 1b - Datapath Design II

Author:
Edmond O'Flynn 12304742

Lecturer:
Dr. Michael Manzke

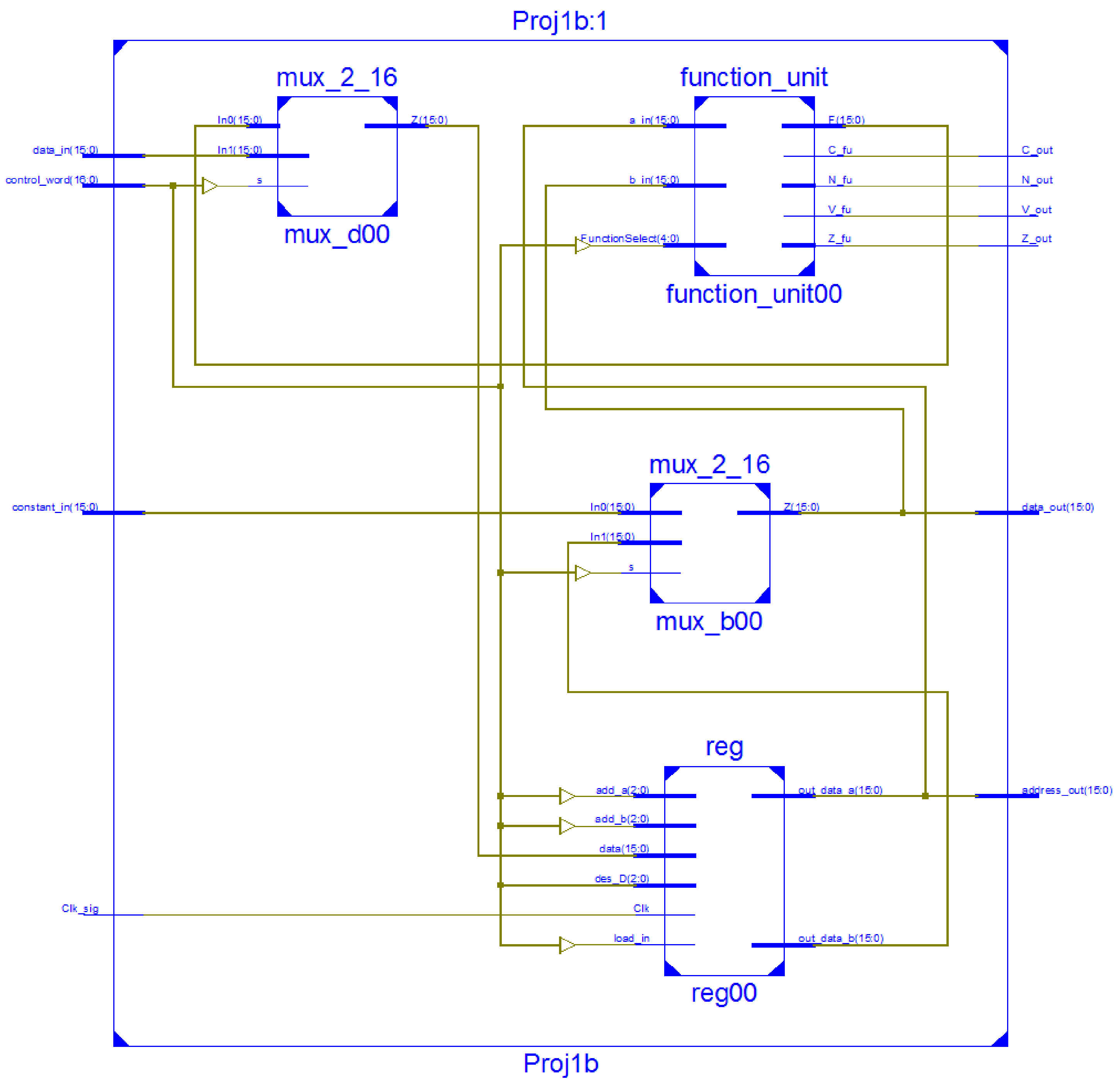
March 8, 2016

Contents

1 VHDL Component Source Code

1.1 Project 1b Top Level

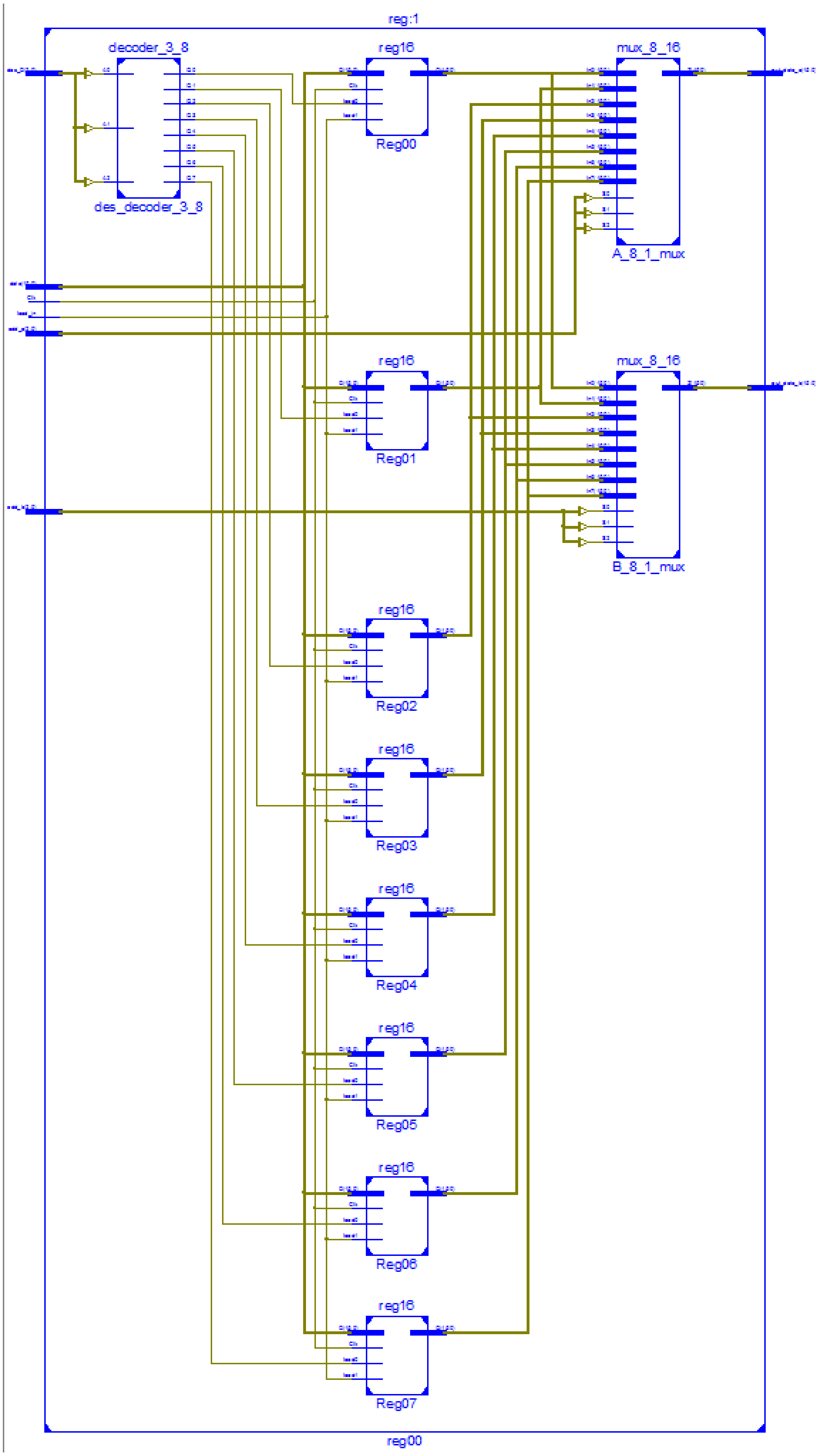
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

   --components
   --reg file
   Component reg
      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;
   --2-1 mux
   Component mux_2_16
      Port(
         In0, In1 : in STD_LOGIC_VECTOR(15 downto 0);
         s : in STD_LOGIC;
         Z : out STD_LOGIC_VECTOR(15 downto 0)
      );
   End Component;
   --function unit
   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_b00: mux_2_16 PORT MAP(
      In0 => constant_in,
      In1 => reg_file_out_b,
      s => control_word(7),
      Z => mux_b_out
   );

   mux_d00: mux_2_16 PORT MAP(
      In0 => function_unit_out,
      In1 => data_in,
      s => control_word(1),
      Z => mux_d_out
   );

   reg00: reg 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_unit00: 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 Behavioral;
```



1.2 Register File

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg 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 reg;

architecture Behavioral of reg is
```

```
   --reg16
   Component reg16
      Port(
         D : in STD_LOGIC_VECTOR(15 downto 0);
         load0, load1, Clk : in STD_LOGIC;
         Q : out STD_LOGIC_VECTOR(15 downto 0)
      );
   end Component;
   --decoder 3-8
   Component decoder_3_8
      Port(
         A0, A1, A2 : in STD_LOGIC;
         Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 : out STD_LOGIC
      );
   End Component;
   --mux 2-1
   Component mux_2_16
      Port(
         In0, In1 : STD_LOGIC_VECTOR(15 downto 0);
         s : STD_LOGIC;
         Z : out STD_LOGIC_VECTOR(15 downto 0)
      );
   End Component;
   --mux 8-16
   Component mux_8_16
      Port(
         In0, In1, In2, In3, In4, In5, In6, In7 : in STD_LOGIC_VECTOR(15 downto 0);
         S0, S1, S2 : in STD_LOGIC;
         Z : out STD_LOGIC_VECTOR(15 downto 0)
      );
   End Component;

   --internal 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_reg, out_sig_a, out_sig_b : STD_LOGIC_VECTOR(15 downto 0);

begin
   --reg0
   Reg00: reg16 PORT MAP(
         D => data,
         load0 => load_reg0,
         load1 => load_in,
         Clk => Clk,
         Q => reg0_q
      );
   --reg1
   Reg01: reg16 PORT MAP(
         D => data,
         load0 => load_reg1,
         load1 => load_in,
         Clk => Clk,
         Q => reg1_q
      );
   --reg2
   Reg02: reg16 PORT MAP(
         D => data,
         load0 => load_reg2,
         load1 => load_in,
         Clk => Clk,
         Q => reg2_q
      );
   --reg3
   Reg03: reg16 PORT MAP(
         D => data,
         load0 => load_reg3,
         load1 => load_in,
         Clk => Clk,
         Q => reg3_q
      );
   --reg4
   Reg04: reg16 PORT MAP(
         D => data,
         load0 => load_reg4,
         load1 => load_in,
         Clk => Clk,
         Q => reg4_q
      );
   --reg5
   Reg05: reg16 PORT MAP(
         D => data,
         load0 => load_reg5,
         load1 => load_in,
         Clk => Clk,
         Q => reg5_q
      );
   --reg6
   Reg06: reg16 PORT MAP(
         D => data,
         load0 => load_reg6,
         load1 => load_in,
         Clk => Clk,
         Q => reg6_q
      );
   --reg7
   Reg07: reg16 PORT MAP(
         D => data,
         load0 => load_reg7,
         load1 => load_in,
         Clk => Clk,
         Q => reg7_q
      );
   --decoder for destination registers
   des_decoder_3_8: decoder_3_8 PORT MAP(
         A0 => des_D(0),
         A1 => des_D(1),
         A2 => des_D(2),
         Q0 => load_reg0,
           Q1 => load_reg1,
           Q2 => load_reg2,
           Q3 => load_reg3,
           Q4 => load_reg4,
           Q5 => load_reg5,
           Q6 => load_reg6,
           Q7 => load_reg7
       );
     --8 to 1 mux for A
     A_8_1_mux: mux_8_16 PORT MAP(
           In0 => reg0_q,
           In1 => reg1_q,
           In2 => reg2_q,
           In3 => reg3_q,
           In4 => reg4_q,
           In5 => reg5_q,
           In6 => reg6_q,
           In7 => reg7_q,
           S0 => add_a(0),
           S1 => add_a(1),
           S2 => add_a(2),
           Z => out_sig_a
       );
     --8 to 1 mux for B
     B_8_1_mux: mux_8_16 PORT MAP(
           In0 => reg0_q,
           In1 => reg1_q,
           In2 => reg2_q,
           In3 => reg3_q,
           In4 => reg4_q,
           In5 => reg5_q,
           In6 => reg6_q,
           In7 => reg7_q,
           S0 => add_b(0),
           S1 => add_b(1),
           S2 => add_b(2),
           Z => out_sig_b
       );

       out_data_a <= out_sig_a;
       out_data_b <= out_sig_b;

   end Behavioral;
```


reg:1
decoder_3_8
des_decoder_3_8
reg16
Reg00
mux_8_16
A_8_1_mux
reg16
Reg01
mux_8_16
B_8_1_mux
reg16
Reg02
reg16
Reg03
reg16
Reg04
reg16
Reg05
reg16
Reg06
reg16
Reg07
reg00

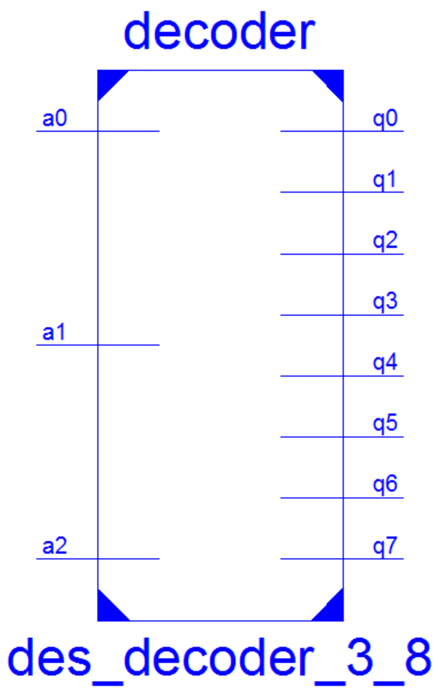
1.3 ALU Unit

```
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

   --components in ALU
   --ripple adder
   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;
   --a b logic for and or xor not
   Component logic_circuit_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;
   --b logic circuit
   Component logic_circuit_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;
   --2-1 mux
   Component mux_2_16
      Port(
         In0, In1 : 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
```

```
   --instantiation of components
   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_b00: logic_circuit_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_b00 : logic_circuit_b PORT MAP(
         B => b_in,
         S_in => G_select(2 downto 1),
         Y_out => logic_out
   );

   mux_2_1600: mux_2_16 PORT MAP(
         In0 => ripple_out,
         In1 => logic_output_a_b,
         s => G_select(3),
         Z => G
   );

end Behavioral;
```

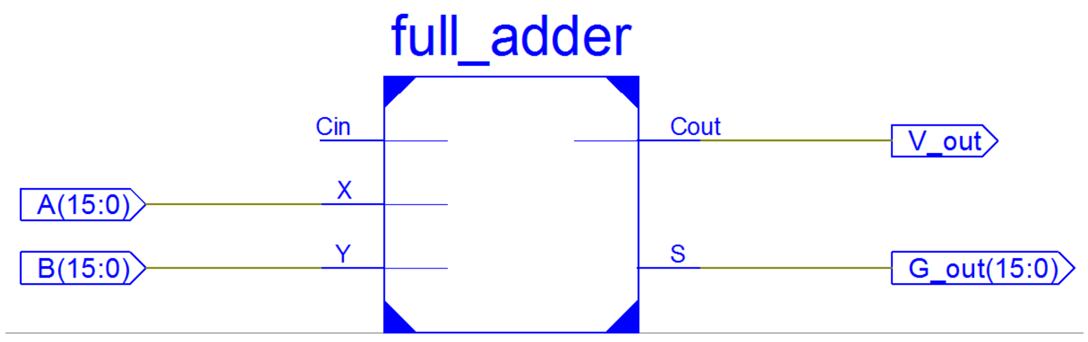
1.4 Decoder 3-8 Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder_3_8 is
   Port(
      A0, A1, A2 : in STD_LOGIC;
      Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 : out STD_LOGIC
      );
end decoder_3_8;

architecture Behavioral of decoder_3_8 is
begin
      Q0 <= ((not A0) and (not A1) and (not A2)) after 1ns; --000
      Q1 <= ((A0) and (not A1) and (not A2)) after 1ns; --001
      Q2 <= ((not A0) and (A1) and (not A2)) after 1ns; --010
      Q3 <= ((A0) and (A1) and (not A2)) after 1ns; --011
      Q4 <= ((not A0) and (not A1) and (A2)) after 1ns; --100
      Q5 <= ((A0) and (not A1) and (A2)) after 1ns; --101
      Q6 <= ((not A0) and (A1) and (A2)) after 1ns; --110
      Q7 <= ((A0) and (A1) and (A2)) after 1ns; --111
end Behavioral;
```



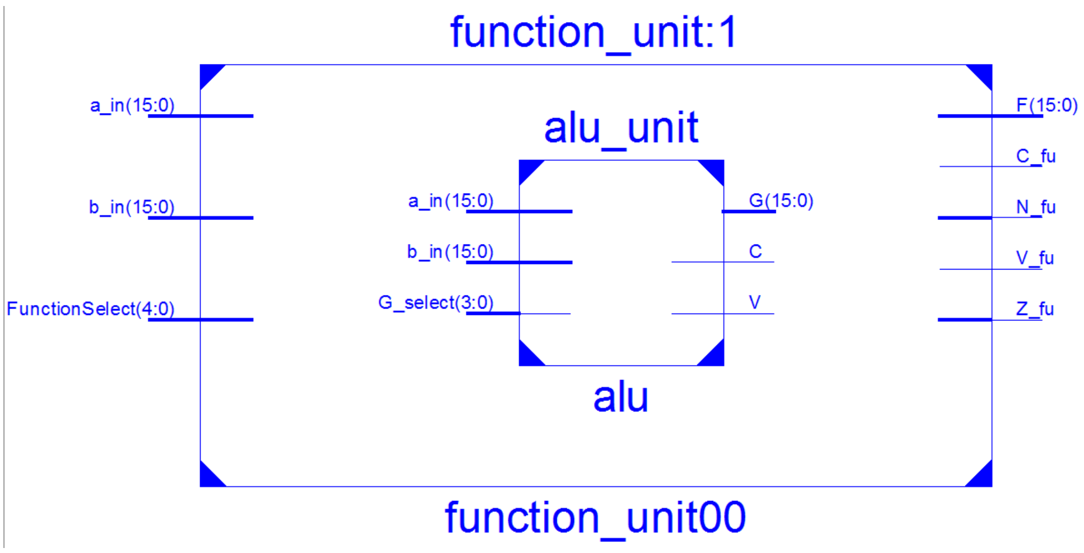
1.5 Full Adder

```
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 S0, S1, S2 : STD_LOGIC;
begin
   S0 <= (X xor Y) after 1ns;
   S1 <= (Cin and S0) after 1ns;
   S2 <= (X and Y) after 1ns;
   S <= (S0 xor Cin) after 1ns;
   Cout <= (S1 or S2) after 1ns;

end Behavioral;
```



1.6 Function 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

   --2 to 1 mux
   Component mux_2_16
      Port(
         In0, In1 : in STD_LOGIC_VECTOR(15 downto 0);
         s : in STD_LOGIC;
         Z : out STD_LOGIC_VECTOR(15 downto 0)
      );
   End Component;
   --shifter
   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;
   --alu
   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
   shifter00: shifter PORT MAP(
      B => b_in,
      S => FunctionSelect(3 downto 2),
      IL => '0',
      IR => '0',
      H => H_out
   );

   mux_2_1600: mux_2_16 PORT MAP(
      In0 => ALU_out,
      In1 => 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;
```



1.7 Logic Circuit A-B

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity logic_circuit_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 logic_circuit_a_b;

architecture Behavioral of logic_circuit_a_b is
```

```
begin
   logic_output_a_b <=  (a_logic_in and b_logic_in) after 1ns when select_in = "00" else
                       (a_logic_in or b_logic_in) after 1ns when select_in = "01" else
                       (a_logic_in xor b_logic_in) after 1ns when select_in = "10" else
                       (not (a_logic_in)) after 1ns;

end Behavioral;
```

1.8 Logic Circuit B

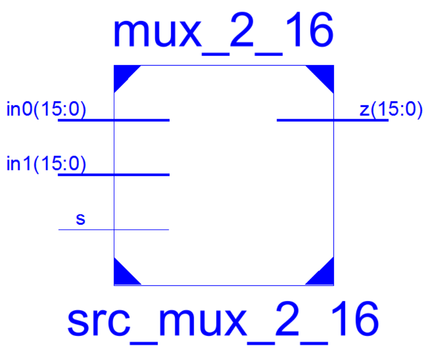
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity logic_circuit_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 logic_circuit_b;

architecture Behavioral of logic_circuit_b is

   --mux 2-1 component
   Component mux_2_1
   Port(
      B_i, S0, S1 : in STD_LOGIC;
      Y_i : out STD_LOGIC
   );
   End Component;

begin
   mux00: mux_2_1 PORT MAP(
      B_i => B(0),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(0)
   );

   mux01: mux_2_1 PORT MAP(
      B_i => B(1),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(1)
   );

   mux02: mux_2_1 PORT MAP(
      B_i => B(2),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(2)
   );

   mux03: mux_2_1 PORT MAP(
      B_i => B(3),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(3)
   );

   mux04: mux_2_1 PORT MAP(
      B_i => B(4),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(4)
   );

   mux05: mux_2_1 PORT MAP(
      B_i => B(5),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(5)
   );

   mux06: mux_2_1 PORT MAP(
      B_i => B(6),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(6)
   );

   mux07: mux_2_1 PORT MAP(
      B_i => B(7),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(7)
   );

   mux08: mux_2_1 PORT MAP(
      B_i => B(8),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(8)
   );

   mux09: mux_2_1 PORT MAP(
      B_i => B(9),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(9)
   );

   mux10: mux_2_1 PORT MAP(
      B_i => B(10),
      S0 => S_in(0),
      S1 => S_in(1),
      Y_i => Y_out(10)
   );

end Behavioral;
```

1.9 Mux 2-1 Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_2_1 is
   Port(
         B_i, S0, S1 : in STD_LOGIC;
         Y_i : out STD_LOGIC
      );
end mux_2_1;

architecture Behavioral of mux_2_1 is

begin
   Y_i <=  S0 after 1ns when B_i = '1' else
           S1 after 1ns when B_i = '0' else
           '0' after 1ns;

end Behavioral;
```

1.10 Mux 2-16 Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_2_16 is
   Port(
      In0, In1 : in STD_LOGIC_VECTOR(15 downto 0);
      s : in STD_LOGIC;
      Z : out STD_LOGIC_VECTOR(15 downto 0)
   );
end mux_2_16;

architecture Behavioral of mux_2_16 is

begin
   Z <=  In0 after 1ns when s='0' else
         In1 after 1ns when s='1' else
         x"0000" after 1ns;

end Behavioral;
```

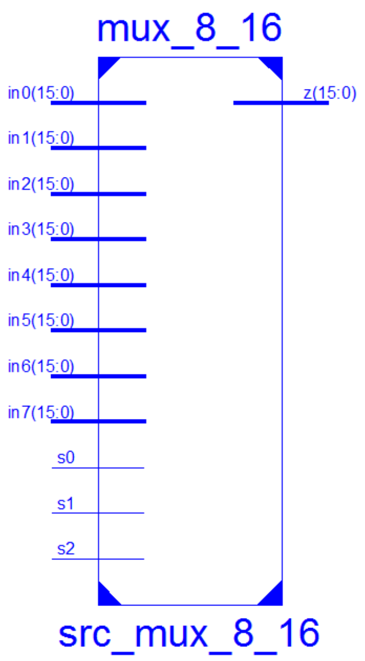


1.11 Mux 3-1 Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_3_1 is
   Port(
      In0, In1, In2 : in STD_LOGIC;
      S0, S1 : in STD_LOGIC;
      Z : out STD_LOGIC
      );
end mux_3_1;

architecture Behavioral of mux_3_1 is

begin
   Z <=  In0 after 1ns when S0 = '0' and S1 = '0' else
        In1 after 1ns when S0 = '0' and S1 = '1' else
        In2 after 1ns when S0 = '1' and S1 = '0' else
        '0' after 1ns;

end Behavioral;
```

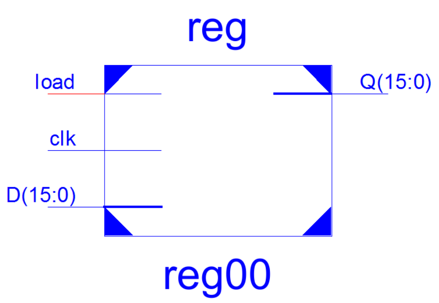
1.12 Mux 8-16 Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_8_16 is
   Port(
      In0, In1, In2, In3, In4, In5, In6, In7 : in STD_LOGIC_VECTOR(15 downto 0);
      S0, S1, S2 : in STD_LOGIC;
      Z : out STD_LOGIC_VECTOR(15 downto 0)
      );
end mux_8_16;

architecture Behavioral of mux_8_16 is

begin
   Z <= In0 after 1ns when S0='0' and S1='0' and S2='1' else
        In1 after 1ns when S0='1' and S1='0' and S2='0' else
        In2 after 1ns when S0='0' and S1='1' and S2='0' else
        In3 after 1ns when S0='1' and S1='1' and S2='0' else
        In4 after 1ns when S0='0' and S1='0' and S2='1' else
        In5 after 1ns when S0='1' and S1='0' and S2='1' else
        In6 after 1ns when S0='0' and S1='1' and S2='1' else
        In7 after 1ns when S0='1' and S1='1' and S2='1' else
        x"0000" after 1ns;

end Behavioral;
```



1.13 Register

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg16 is
   Port(
      D : in STD_LOGIC_VECTOR(15 downto 0);
      load0, load1, 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((load0 = '1') and (load1 = '1')) then
               Q <= D after 5ns;
            end if;
         end if;
   end process;

end Behavioral;
```

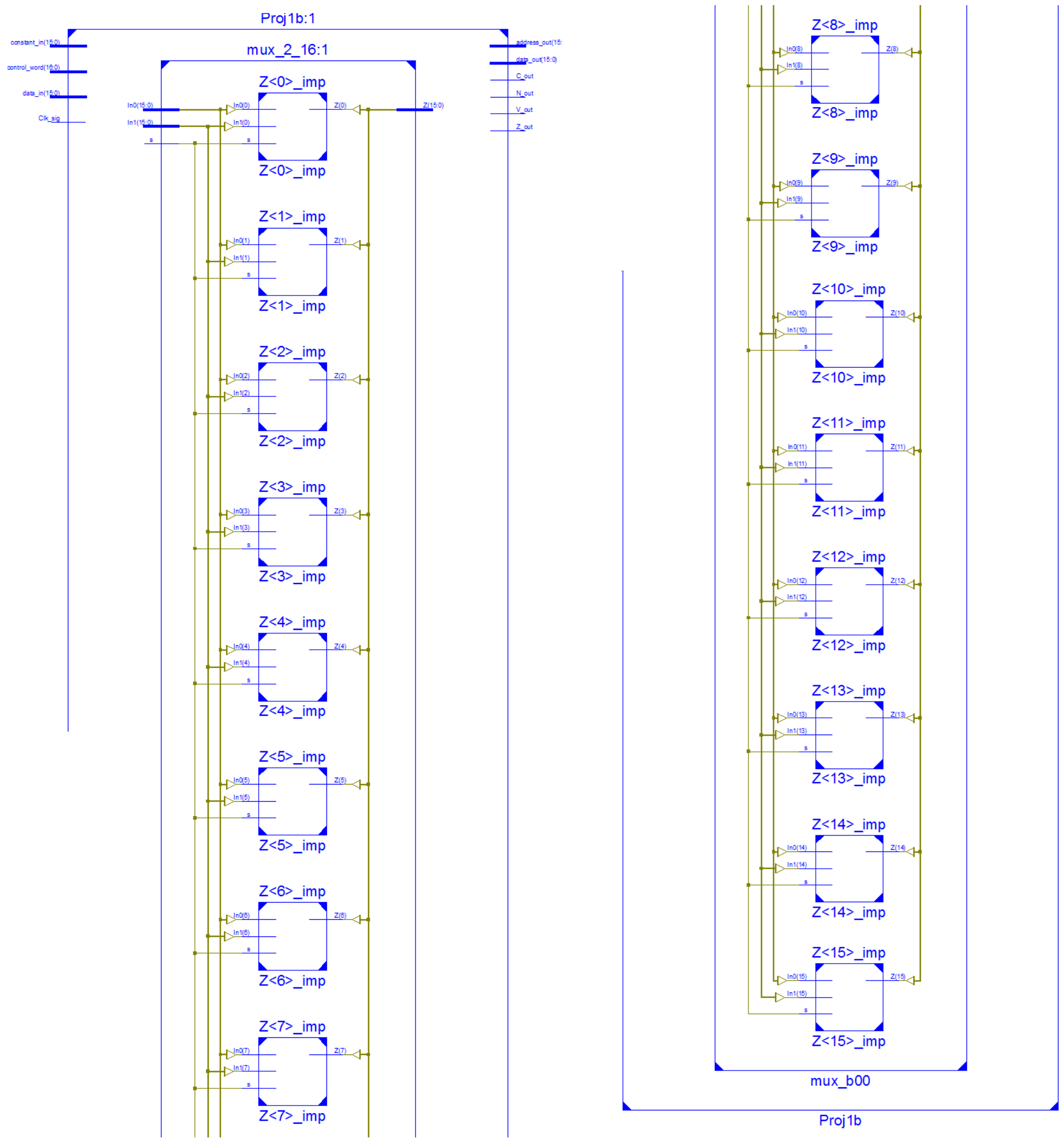


1.14 Ripple Adder

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ripple_adder is
   Port(
         A, B : in STD_LOGIC_VECTOR(15 downto 0);
         Cin : STD_LOGIC;
         Cout, V_out : out STD_LOGIC;
         G_out : out STD_LOGIC_VECTOR(15 downto 0)
      );

end ripple_adder;

architecture Behavioral of ripple_adder is

   Component full_adder
      Port(
         X, Y, Cin : in STD_LOGIC;
         Cout, S : out STD_LOGIC
         );
   End Component;

   --signals - 16 carry bits and 1 output
   signal C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C_out :
       STD_LOGIC;

begin
   full_adder_00: full_adder PORT MAP(
      X => A(0),
      Y => B(0),
      Cin => Cin,
      Cout => C0,
      S => G_out(0)
   );

   full_adder_01: full_adder PORT MAP(
      X => A(1),
      Y => B(1),
      Cin => C0,
      Cout => C1,
      S => G_out(1)
   );

   full_adder_02: full_adder PORT MAP(
      X => A(2),
      Y => B(2),
      Cin => C1,
      Cout => C2,
      S => G_out(2)
   );

   full_adder_03: full_adder PORT MAP(
      X => A(3),
      Y => B(3),
      Cin => C2,
      Cout => C3,
      S => G_out(3)
   );

   full_adder_04: full_adder PORT MAP(
      X => A(4),
      Y => B(4),
      Cin => C3,
      Cout => C4,
      S => G_out(4)
   );

   full_adder_05: full_adder PORT MAP(
      X => A(5),
      Y => B(5),
      Cin => C4,
      Cout => C5,
      S => G_out(5)
   );

   full_adder_06: full_adder PORT MAP(
      X => A(6),
      Y => B(6),
      Cin => C5,
      Cout => C6,
      S => G_out(6)
   );

   full_adder_07: full_adder PORT MAP(
      X => A(7),
      Y => B(7),
      Cin => C6,
      Cout => C7,
      S => G_out(7)
   );

   full_adder_08: full_adder PORT MAP(
      X => A(8),
      Y => B(8),
      Cin => C7,
      Cout => C8,
      S => G_out(8)
   );

   full_adder_09: full_adder PORT MAP(
      X => A(9),
      Y => B(9),
      Cin => C8,
      Cout => C9,
      S => G_out(9)
   );

   full_adder_10: full_adder PORT MAP(
      X => A(10),
      Y => B(10),
      Cin => C9,
      Cout => C10,
      S => G_out(10)
   );

   full_adder_11: full_adder PORT MAP(
      X => A(11),
      Y => B(11),
      Cin => C10,
      Cout => C11,
      S => G_out(11)
   );

   full_adder_12: full_adder PORT MAP(
      X => A(12),
      Y => B(12),
      Cin => C11,
      Cout => C12,
      S => G_out(12)
   );

   full_adder_13: full_adder PORT MAP(
      X => A(13),
      Y => B(13),
      Cin => C12,
      Cout => C13,
      S => G_out(13)
   );

   full_adder_14: full_adder PORT MAP(
      X => A(14),
      Y => B(14),
      Cin => C13,
      Cout => C14,
      S => G_out(14)
   );

   full_adder_15: full_adder PORT MAP(
      X => A(15),
      Y => B(15),
      Cin => C14,
      Cout => C15,
      S => G_out(15)
   );

   --carry
   Cout <= C_out;
   --overflow
   V_out <= (C_out xor C15);

end Behavioral;
```

1.15 Shifter

```vhdl
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

   --2 to 1 mux
   Component mux_3_1
      Port(
         In0, In1, In2, S0, S1 : in STD_LOGIC;
         Z : out STD_LOGIC
      );
   End Component;

begin
   mux00: mux_3_1 PORT MAP(
      In0 => B(0),
      In1 => B(1),
      In2 => IL,
      S0 => S(0),
      S1 => S(1),
      Z => H(0)
   );

   mux01: mux_3_1 PORT MAP(
      In0 => B(1),
      In1 => B(2),
      In2 => B(0),
      S0 => S(0),
      S1 => S(1),
      Z => H(1)
   );

   mux02: mux_3_1 PORT MAP(
      In0 => B(2),
      In1 => B(3),
      In2 => B(1),
      S0 => S(0),
      S1 => S(1),
      Z => H(2)
   );

   mux03: mux_3_1 PORT MAP(
      In0 => B(3),
      In1 => B(4),
      In2 => B(2),
      S0 => S(0),
      S1 => S(1),
      Z => H(3)
   );

   mux04: mux_3_1 PORT MAP(
      In0 => B(4),
      In1 => B(5),
      In2 => B(3),
      S0 => S(0),
      S1 => S(1),
      Z => H(4)
   );

   mux05: mux_3_1 PORT MAP(
      In0 => B(5),
      In1 => B(6),
      In2 => B(4),
      S0 => S(0),
      S1 => S(1),
      Z => H(5)
   );

   mux06: mux_3_1 PORT MAP(
      In0 => B(6),
      In1 => B(7),
      In2 => B(5),
      S0 => S(0),
      S1 => S(1),
      Z => H(6)
   );

   mux07: mux_3_1 PORT MAP(
      In0 => B(7),
      In1 => B(8),
      In2 => B(6),
      S0 => S(0),
      S1 => S(1),
      Z => H(7)
   );

   mux08: mux_3_1 PORT MAP(
      In0 => B(8),
      In1 => B(9),
      In2 => B(7),
      S0 => S(0),
      S1 => S(1),
      Z => H(8)
   );

   mux09: mux_3_1 PORT MAP(
      In0 => B(9),
      In1 => B(10),
      In2 => B(8),
      S0 => S(0),
      S1 => S(1),
      Z => H(9)
   );

   mux10: mux_3_1 PORT MAP(
      In0 => B(10),
      In1 => B(11),
      In2 => B(9),
      S0 => S(0),
      S1 => S(1),
      Z => H(10)
   );

   mux11: mux_3_1 PORT MAP(
      In0 => B(11),
      In1 => B(12),
      In2 => B(10),
      S0 => S(0),
      S1 => S(1),
      Z => H(11)
   );

   mux12: mux_3_1 PORT MAP(
      In0 => B(12),
      In1 => B(13),
      In2 => B(11),
      S0 => S(0),
      S1 => S(1),
      Z => H(12)
   );

   mux13: mux_3_1 PORT MAP(
      In0 => B(13),
      In1 => B(14),
      In2 => B(12),
      S0 => S(0),
      S1 => S(1),
      Z => H(13)
   );

   mux14: mux_3_1 PORT MAP(
      In0 => B(14),
      In1 => B(15),
      In2 => B(13),
      S0 => S(0),
      S1 => S(1),
      Z => H(14)
   );

   mux15: mux_3_1 PORT MAP(
      In0 => B(15),
      In1 => IR,
      In2 => B(14),
      S0 => S(0),
      S1 => S(1),
      Z => H(15)
    );

end Behavioral;
```



2 Component Test Benches

2.1 Project 1b Top Level

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

ENTITY Proj1b_TB IS
END Proj1b_TB;

ARCHITECTURE behavior OF Proj1b_TB IS

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

    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;


   --Inputs
   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';

   --Outputs
   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;

   -- Clock period definitions
   constant Clk_sig_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   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
        );

   -- Clock process definitions
   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;


   -- Stimulus 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;
```

2.2 Register File

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

ENTITY RegFile_TB IS
END RegFile_TB;

ARCHITECTURE behavior OF RegFile_TB IS

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

    COMPONENT reg
    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;


   --Inputs
   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');

   --Outputs
   signal out_data_a : std_logic_vector(15 downto 0);
   signal out_data_b : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant Clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: reg 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
        );

   -- 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

   end process;
```

```
END;
```

2.3 ALU Unit

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

ENTITY ALU_TB IS
END ALU_TB;

ARCHITECTURE behavior OF ALU_TB IS

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

    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;


   --Inputs
   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');

   --Outputs
   signal V : std_logic;
   signal C : std_logic;
   signal G : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: alu_unit PORT MAP (
          a_in => a_in,
          b_in => b_in,
          G_select => G_select,
          V => V,
          C => C,
          G => G
        );

   -- Stimulus process
   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;
```

2.4 Decoder 3-8 Bit

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

ENTITY decoder_TB IS
END decoder_TB;

ARCHITECTURE behavior OF decoder_TB IS

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

    COMPONENT decoder_3_8
    PORT(
         A0 : IN std_logic;
         A1 : IN std_logic;
         A2 : IN std_logic;
         Q0 : OUT std_logic;
         Q1 : OUT std_logic;
         Q2 : OUT std_logic;
         Q3 : OUT std_logic;
         Q4 : OUT std_logic;
         Q5 : OUT std_logic;
         Q6 : OUT std_logic;
         Q7 : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal A0 : std_logic := '0';
   signal A1 : std_logic := '0';
   signal A2 : std_logic := '0';

   --Outputs
   signal Q0 : std_logic;
   signal Q1 : std_logic;
   signal Q2 : std_logic;
   signal Q3 : std_logic;
   signal Q4 : std_logic;
   signal Q5 : std_logic;
   signal Q6 : std_logic;
   signal Q7 : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: decoder_3_8 PORT MAP (
          A0 => A0,
          A1 => A1,
          A2 => A2,
          Q0 => Q0,
          Q1 => Q1,
          Q2 => Q2,
          Q3 => Q3,
          Q4 => Q4,
          Q5 => Q5,
          Q6 => Q6,
          Q7 => Q7
        );

   -- Stimulus process
   stim_proc: process
   begin
      wait for 5ns;
      A0 <= '0';
      A1 <= '0';
      A2 <= '0';

      wait for 5ns;
      A0 <= '1';
      A1 <= '0';
      A2 <= '0';

      wait for 5ns;
      A0 <= '0';
      A1 <= '1';
      A2 <= '0';

      wait for 5ns;
      A0 <= '1';
      A1 <= '1';
      A2 <= '0';

      wait for 5ns;
      A0 <= '0';
      A1 <= '0';
      A2 <= '1';

      wait for 5ns;
      A0 <= '1';
      A1 <= '0';
      A2 <= '1';

      wait for 5ns;
      A0 <= '0';
      A1 <= '1';
      A2 <= '1';

      wait for 5ns;
      A0 <= '1';
      A1 <= '1';
      A2 <= '1';
      wait;
   end process;

END;
```

2.5 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 Declaration for the Unit Under Test (UUT)

    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;


   --Inputs
   signal X : std_logic := '0';
   signal Y : std_logic := '0';
   signal Cin : std_logic := '0';
   --Outputs
   signal Cout : std_logic;
   signal S : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: full_adder PORT MAP (
          X => X,
          Y => Y,
          Cin => Cin,
          Cout => Cout,
          S => S
        );

   -- Stimulus process
   stim_proc: process
   begin
      wait for 5ns;
      X <= '1';

      wait for 5ns;
      X <= '0';
      Y <= '1';

      wait for 5ns;
      X <= '1';

      wait for 5ns;
      Cin <= '1';

      wait for 5ns;
      Y <= '0';

      wait for 5ns;
      X <= '0';

      wait;
   end process;

END;
```

2.6 Function Unit

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

ENTITY FuncUnit_TB IS
END FuncUnit_TB;

ARCHITECTURE behavior OF FuncUnit_TB IS

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

    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;


   --Inputs
   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');

   --Outputs
   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);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name


BEGIN

   -- Instantiate the Unit Under Test (UUT)
   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
        );

   -- Stimulus process
   stim_proc: process
   begin
      a_in <= x"AAAA";
      b_in <= x"BBBB";

      wait for 20ns;
      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;
```

2.7 Logic Circuit A-B

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

ENTITY logic_circuit_a_b_TB IS
END logic_circuit_a_b_TB;

ARCHITECTURE behavior OF logic_circuit_a_b_TB IS

    -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT logic_circuit_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;


  --Inputs
  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');

  --Outputs
  signal logic_output_a_b : std_logic_vector(15 downto 0);

BEGIN

  -- Instantiate the Unit Under Test (UUT)
  uut: logic_circuit_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
       );

  -- Stimulus process
  stim_proc: process
  begin
     wait for 5ns;
     a_login_in <= x"FFFF";
     b_logic_in <= x"9999";
     select_in <= "00";

     wait for 5ns;
     select_in <= "01";

     wait for 5ns;
     select_in <= "10";

     wait for 5ns;
     select_in <= "11";

     wait;
  end process;

END;
```

2.8 Logic Circuit B

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

```
ENTITY logic_circuit_b_TB IS
END logic_circuit_b_TB;

ARCHITECTURE behavior OF logic_circuit_b_TB IS

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

    COMPONENT logic_circuit_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;


   --Inputs
   signal B : std_logic_vector(15 downto 0) := (others => '0');
   signal S_in : std_logic_vector(1 downto 0) := (others => '0');

   --Outputs
   signal Y_out : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: logic_circuit_b PORT MAP (
          B => B,
          S_in => S_in,
          Y_out => Y_out
        );

   -- Stimulus process
   stim_proc: process
   begin
      B <= x"AAAA";
      S_in <= "00";

      wait for 5ns;
      S_in <= "01";

      wait for 5ns;
      S_in <= "10";

      wait for 5ns;
      S_in <= "11";

      wait;
   end process;

END;
```

2.9 Mux 2-1 Bit

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;

ENTITY mux_2_1_TB IS
END mux_2_1_TB;

ARCHITECTURE behavior OF mux_2_1_TB IS

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

    COMPONENT mux_2_1
    PORT(
         B_i : IN std_logic;
         S0 : IN std_logic;
         S1 : IN std_logic;
         Y_i : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal B_i : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';

   --Outputs
   signal Y_i : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux_2_1 PORT MAP (
          B_i => B_i,
          S0 => S0,
          S1 => S1,
          Y_i => Y_i
        );

   -- Stimulus process
   stim_proc: process
   begin
      S0 <= '1';
      S1 <= '0';

      wait for 5ns;
      B_i <= '1';

      wait for 5ns;
      B_i <= '0';

      wait;
   end process;

END;
```

2.10 Mux 2-16 Bit

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

ENTITY mux_2_16_TB IS
END mux_2_16_TB;

ARCHITECTURE behavior OF mux_2_16_TB IS

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

    COMPONENT mux_2_16
    PORT(
         In0 : IN std_logic_vector(15 downto 0);
         In1 : IN std_logic_vector(15 downto 0);
         s : IN std_logic;
         Z : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal In0 : std_logic_vector(15 downto 0) := (others => '0');
   signal In1 : std_logic_vector(15 downto 0) := (others => '0');
   signal s : std_logic := '0';

   --Outputs
   signal Z : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux_2_16 PORT MAP (
          In0 => In0,
          In1 => In1,
          s => s,
          Z => Z
        );

   -- Stimulus process
   stim_proc: process
   begin
      In0 <= x"FFFF";
      In1 <= x"AAAA";

      wait for 10ns;
      s <= '1';

      wait for 10ns;
      s <= '0';

      wait for 10ns;
      s <= '1';

      wait;
   end process;
```

```
END;
```

2.11 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 Declaration for the Unit Under Test (UUT)

    COMPONENT mux_3_1
    PORT(
         In0 : IN std_logic;
         In1 : IN std_logic;
         In2 : IN std_logic;
         S0 : IN std_logic;
         S1 : IN std_logic;
         Z : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal In0 : std_logic := '0';
   signal In1 : std_logic := '0';
   signal In2 : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';

   --Outputs
   signal Z : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux_3_1 PORT MAP (
          In0 => In0,
          In1 => In1,
          In2 => In2,
          S0 => S0,
          S1 => S1,
          Z => Z
        );

   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
      In0 <= '1';
      In1 <= '0';
      In2 <= '1';

      wait for 5ns;
      S0 <= '0';
      S1 <= '1';

      wait for 5ns;
      S0 <= '1';
      S1 <= '0';

      wait for 5ns;
      S0 <= '1';
      S1 <= '1';

      wait;
   end process;

END;
```

2.12 Mux 8-16 Bit

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

ENTITY mux_8_16_TB IS
END mux_8_16_TB;

ARCHITECTURE behavior OF mux_8_16_TB IS

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

    COMPONENT mux_8_16
    PORT(
         In0 : IN std_logic_vector(15 downto 0);
         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);
         S0 : IN std_logic;
         S1 : IN std_logic;
         S2 : IN std_logic;
         Z : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal In0 : std_logic_vector(15 downto 0) := (others => '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 S0 : std_logic := '0';
   signal S1 : std_logic := '0';
   signal S2 : std_logic := '0';

   --Outputs
   signal Z : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux_8_16 PORT MAP (
          In0 => In0,
          In1 => In1,
          In2 => In2,
          In3 => In3,
          In4 => In4,
          In5 => In5,
          In6 => In6,
          In7 => In7,
          S0 => S0,
          S1 => S1,
          S2 => S2,
          Z => Z
        );

   -- Stimulus process
   stim_proc: process
   begin
      In0 <= x"FFFF";
      In1 <= x"EEEE";
      In2 <= x"DDDD";
      In3 <= x"CCCC";
      In4 <= x"BBBB";
      In5 <= x"AAAA";
      In6 <= x"9999";
      In7 <= x"8888";

      wait for 10ns;
      S0 <= '1';
      S1 <= '0';
      S2 <= '0';

      wait for 10ns;
      S0 <= '0';
      S1 <= '1';
      S2 <= '0';

      wait for 10ns;
      S0 <= '1';
      S1 <= '1';
      S2 <= '0';
      wait for 10ns;
      S0 <= '0';
      S1 <= '0';
      S2 <= '1';

      wait for 10ns;
      S0 <= '1';
      S1 <= '0';
      S2 <= '1';

      wait for 10ns;
      S0 <= '0';
      S1 <= '1';
      S2 <= '1';

      wait for 10ns;
      S0 <= '1';
      S1 <= '1';
      S2 <= '1';

      wait;
   end process;

END;
```

2.13 Register

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

ENTITY Reg_TB IS
END Reg_TB;

ARCHITECTURE behavior OF Reg_TB IS

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

    COMPONENT reg16
    PORT(
         D : IN std_logic_vector(15 downto 0);
         load0 : IN std_logic;
         load1 : IN std_logic;
         Clk : IN std_logic;
         Q : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal D : std_logic_vector(15 downto 0) := (others => '0');
   signal load0 : std_logic := '0';
   signal load1 : std_logic := '0';
   signal Clk : std_logic := '0';

   --Outputs
   signal Q : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant Clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: reg16 PORT MAP (
          D => D,
          load0 => load0,
          load1 => load1,
          Clk => Clk,
          Q => Q
        );

   -- 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
      D <= x"FFFF";
      load0 <= '1';
      load1 <= '1';

      wait for 15ns;
      D <= x"AAAA";
      load0 <= '0';

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

      wait for 10ns;
      load0 <= '1';
      load1 <= '1';

      wait;
   end process;

END;
```

2.14 Ripple Adder

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

ENTITY ripple_adder_TB IS
```

```vhdl
END ripple_adder_TB;

ARCHITECTURE behavior OF ripple_adder_TB IS

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

    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;


   --Inputs
   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';

   --Outputs
   signal Cout : std_logic;
   signal V_out : std_logic;
   signal G_out : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: ripple_adder PORT MAP (
          A => A,
          B => B,
          Cin => Cin,
          Cout => Cout,
          V_out => V_out,
          G_out => G_out
        );

   -- Stimulus process
   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;
```

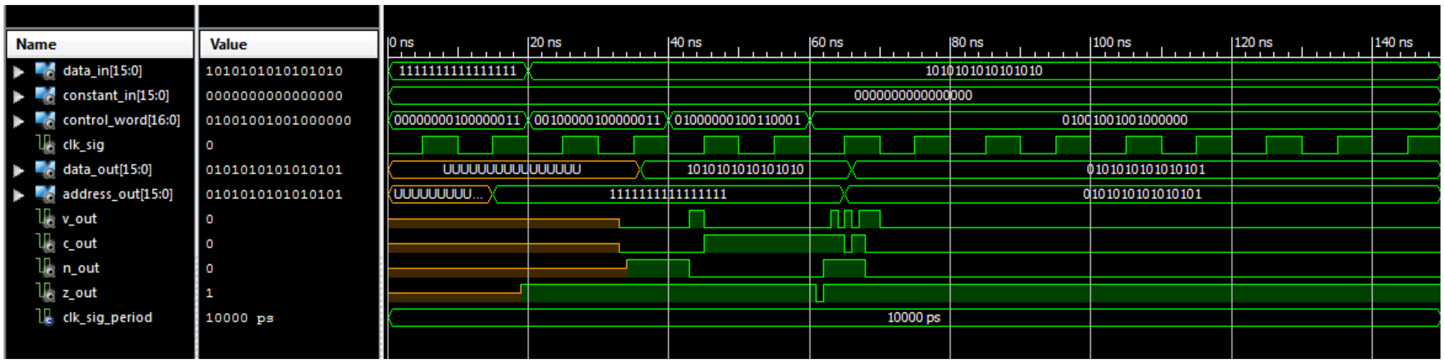
2.15 Shifter

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

ENTITY shifter_TB IS
END shifter_TB;

ARCHITECTURE behavior OF shifter_TB IS

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

    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;


   --Inputs
   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';

   --Outputs
   signal H : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: shifter PORT MAP (
          B => B,
          S => S,
          IL => IL,
          IR => IR,
          H => H
        );

   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
      B <= x"FFFF";
      S <= "00";

      wait for 16ns;
      S <= "01";
      wait for 16ns;
      S <= H;

      wait for 16ns;

      B <= H;
      S <= "10";

      wait;
   end process;

END;
```

3 Results of Test Benches

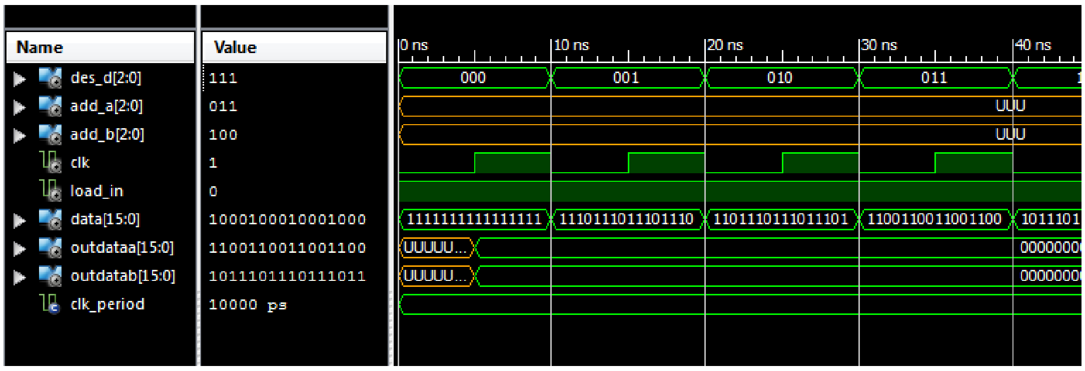
3.1 Project 1b Top Level

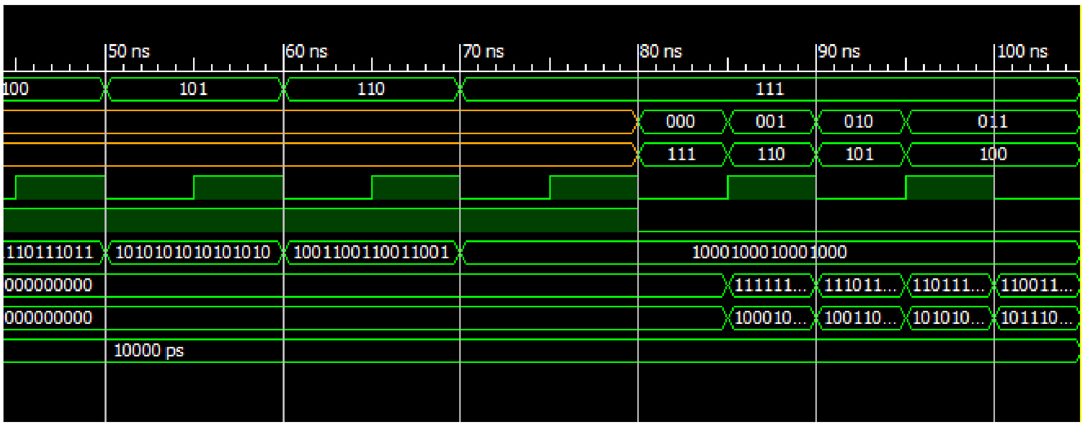
The results of the testbench show all of the modules instantiated together with values being loaded into subsequent registers, and micro-operations being carried out appropriately with loading into a new register.



3.2 Register File

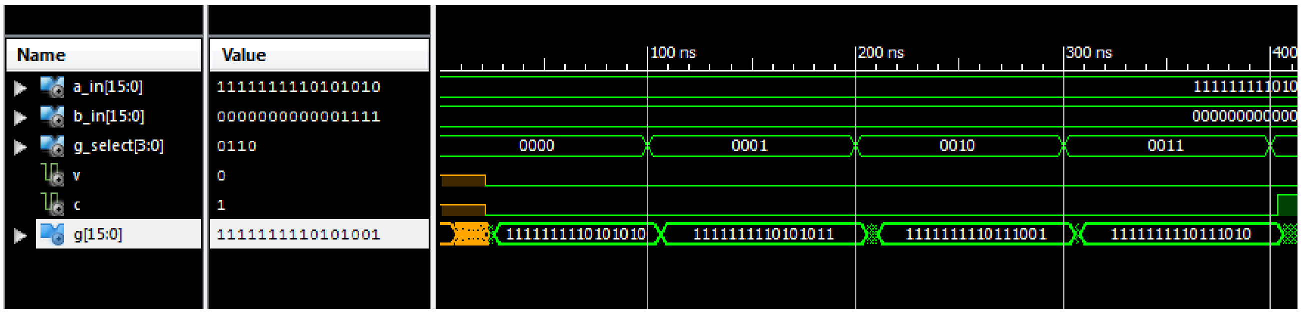
The register file testbench shows the register submodules working together through decoding, multiplexing, storing values, and passing data in and out appropriately with respect to the clock period and load.



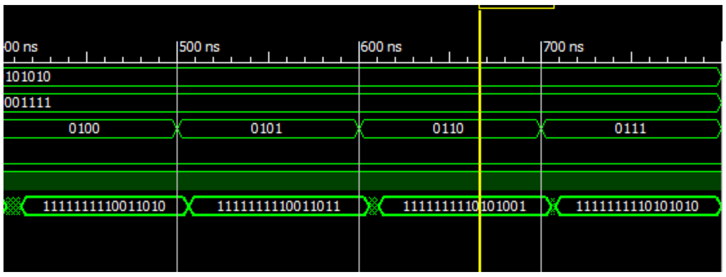


3.3 ALU Unit

The ALU component functions as intended by having correct outputs on its G pin where its 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.

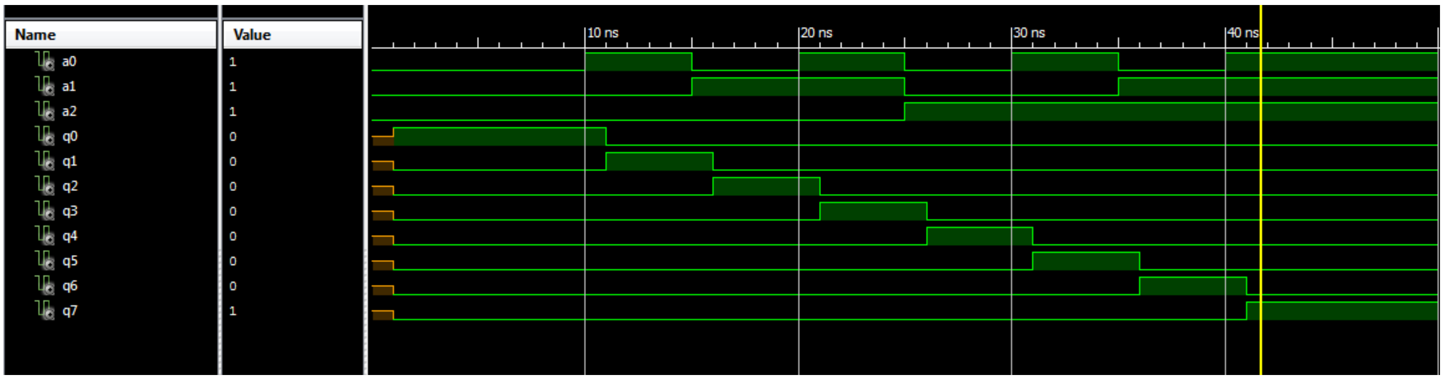


Name	Value
a_in[15:0]	1111111110101010
b_in[15:0]	0000000000001111
g_select[3:0]	0110
v	0
c	1
g[15:0]	1111111110101001



3.4 Decoder 3-8 Bit

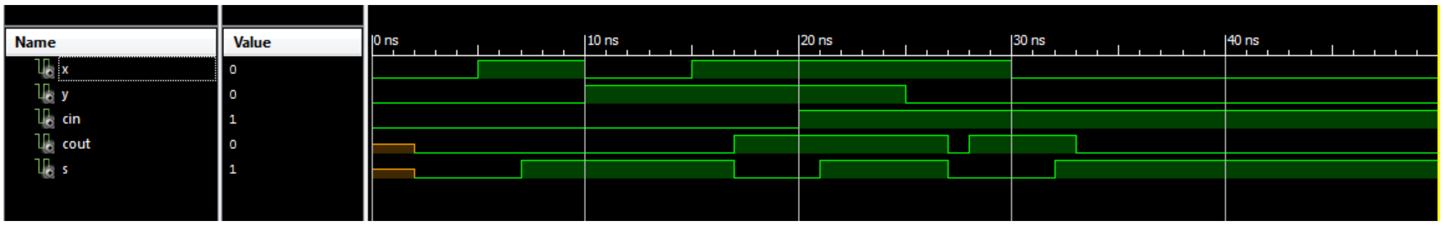
The decoder unit cycles through each individual combination of input values and making the appropriate pin turn to high after the elapsing of 5ns, which then corresponds to the selection of the individual register associated with the combination generated.



Name	Value
a0	1
a1	1
a2	1
q0	0
q1	0
q2	0
q3	0
q4	0
q5	0
q6	0
q7	1

3.5 Full Adder

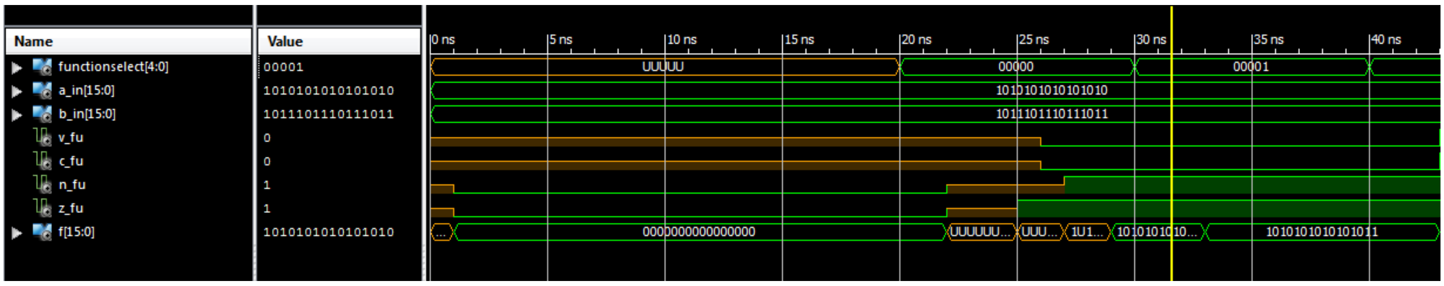
For each input, the appropriate carry is determined and correct operations are performed for the given select pin, resulting in the intended output.



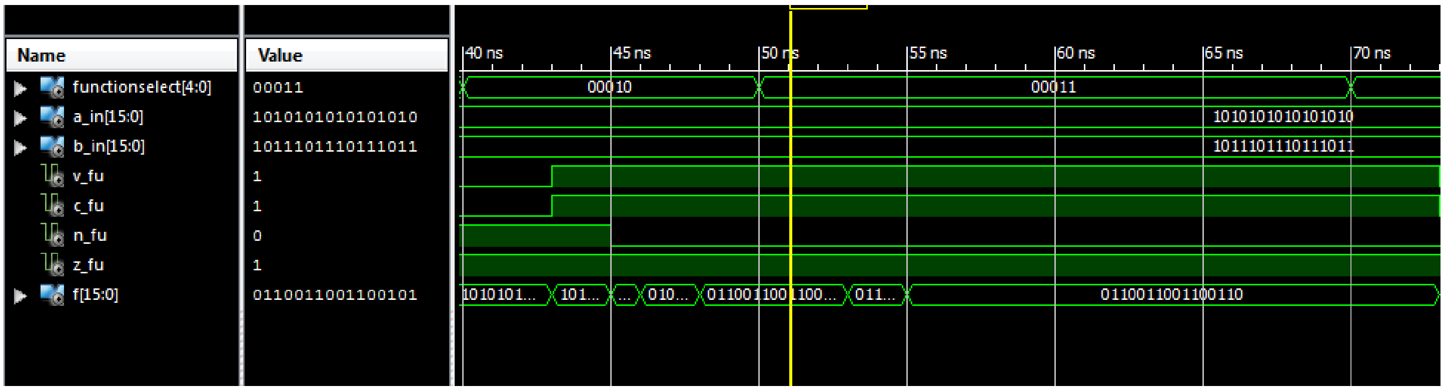
Name	Value
x	0
y	0
cin	1
cout	0
s	1

3.6 Function Unit

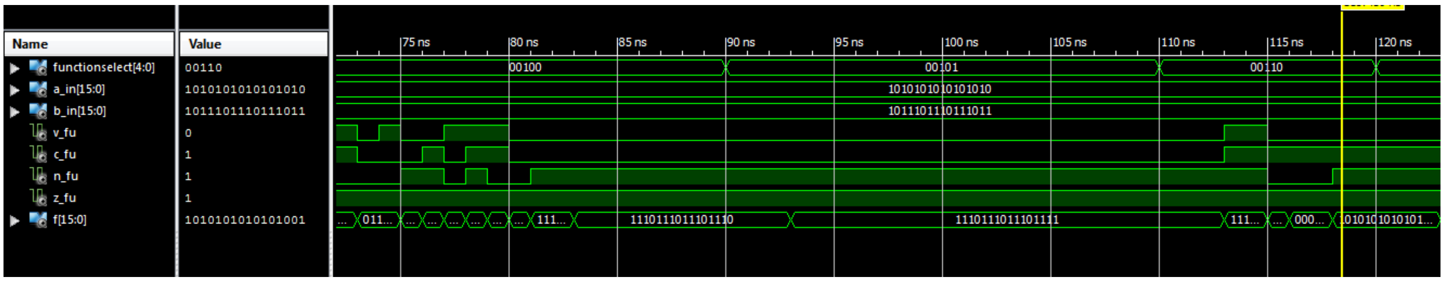
The function unit works as it should be with appropriate values being handled and raising correct N, Z, C and V flags in the unit itself to the word passed in for function selection.



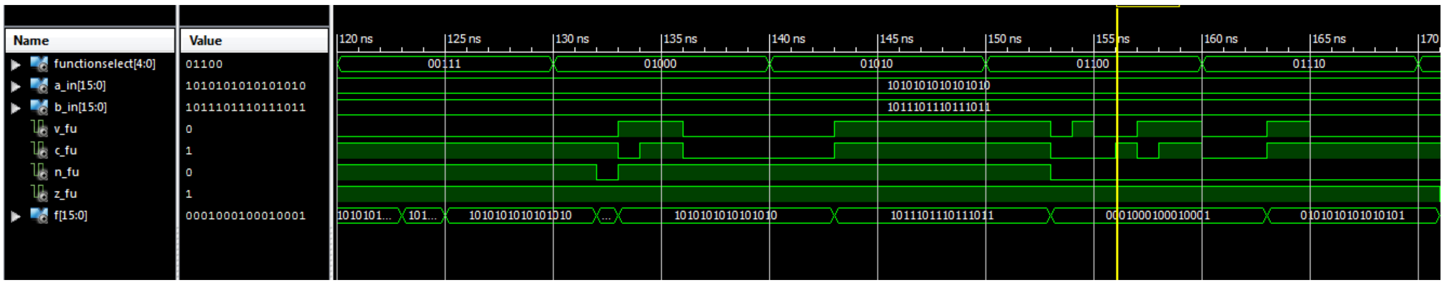
Name	Value
functionselect[4:0]	00001
a_in[15:0]	1010101010101010
b_in[15:0]	1011101110111011
v_fu	0
c_fu	0
n_fu	1
z_fu	1
f[15:0]	1010101010101010



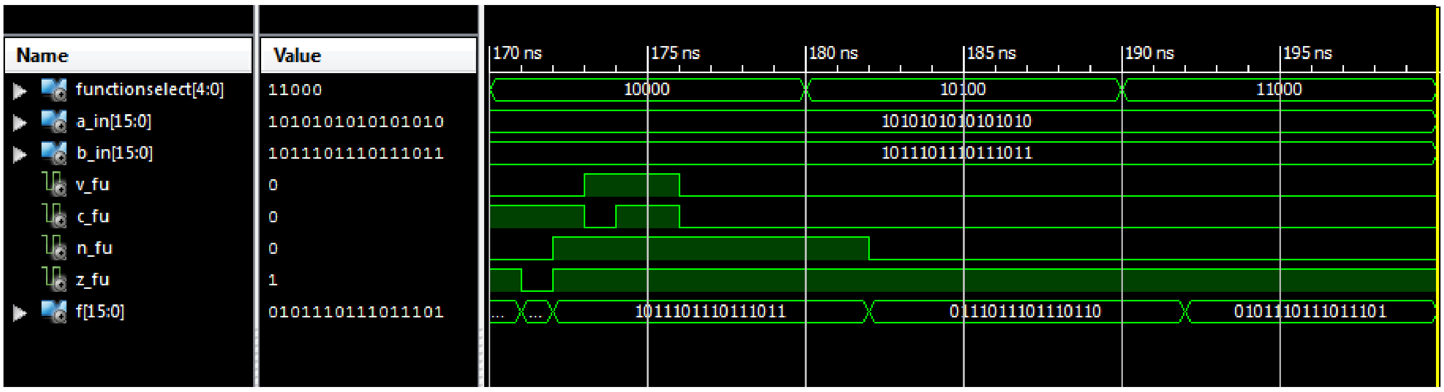
Name	Value
functionselect[4:0]	00011
a_in[15:0]	1010101010101010
b_in[15:0]	1011101110111011
v_fu	1
c_fu	1
n_fu	0
z_fu	1
f[15:0]	0110011001100101



Name	Value
functionselect[4:0]	00110
a_in[15:0]	1010101010101010
b_in[15:0]	1011101110111011
v_fu	0
c_fu	1
n_fu	1
z_fu	1
f[15:0]	1010101010101001



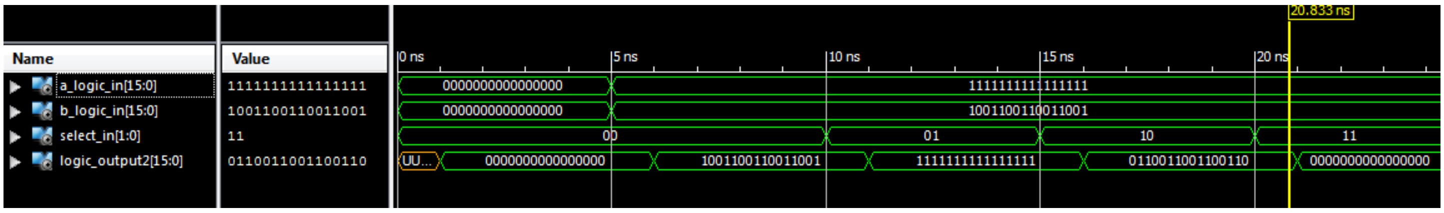
Name	Value
functionselect[4:0]	01100
a_in[15:0]	1010101010101010
b_in[15:0]	1011101110111011
v_fu	0
c_fu	1
n_fu	0
z_fu	1
f[15:0]	0001000100010001



Name	Value
functionselect[4:0]	11000
a_in[15:0]	1010101010101010
b_in[15:0]	1011101110111011
v_fu	0
c_fu	0
n_fu	0
z_fu	1
f[15:0]	0101110111011101

3.7 Logic Circuit A-B

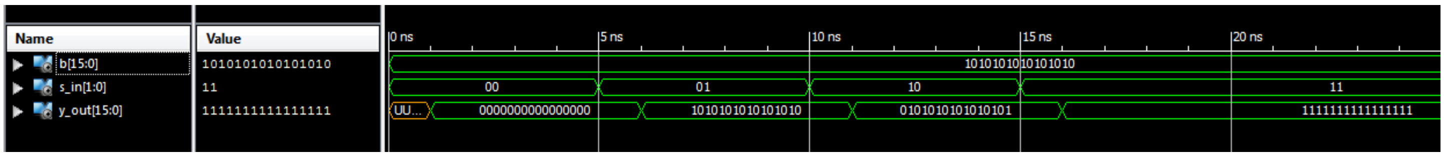
The logic circuit here provides two inputs of a and b logic provided, as well as a select pin that determines the operations to be performed resulting in outputs via gate logic.



```
Name                 Value
a_logic_in[15:0]     1111111111111111
b_logic_in[15:0]     1001100110011001
select_in[1:0]       11
logic_output2[15:0]  0110011001100110
```

3.8 Logic Circuit B

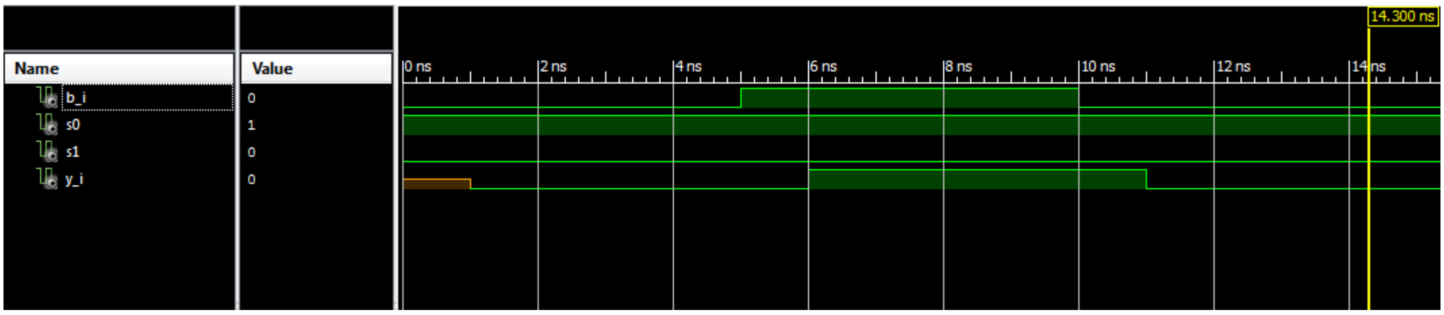
The B-logic input results in correct outputs for each B and s value provided to the logic circuit.



```
Name          Value
b[15:0]       1010101010101010
s_in[1:0]     11
y_out[15:0]   1111111111111111
```

3.9 Mux 2-1 Bit

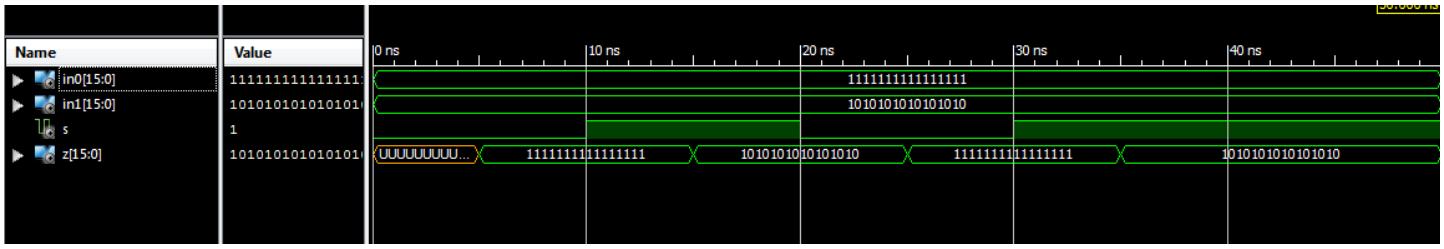
The 2-1 mux works correctly where the s value changes regularly from between 0 to 1 with correct resulting changes in In0 and In1.



```
Name   Value
b_i    0
s0     1
s1     0
y_i    0
```

3.10 Mux 2-16 Bit

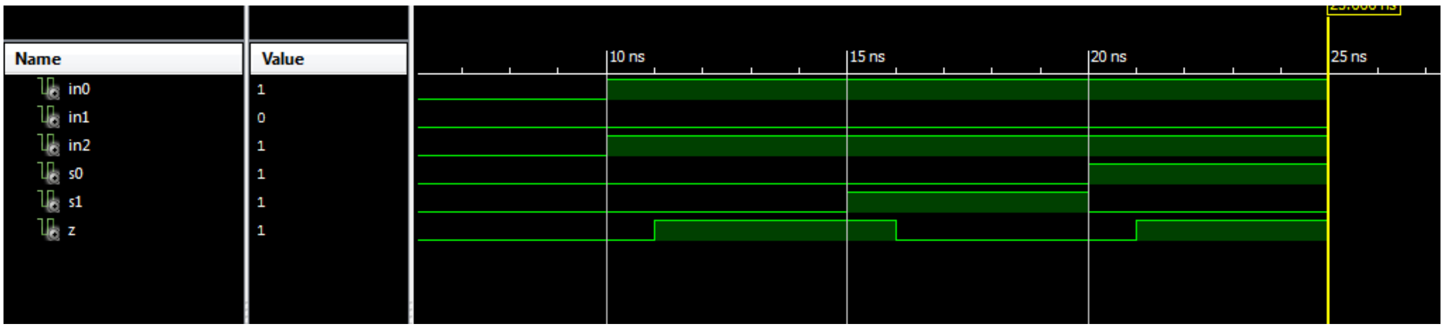
For the input s, which can be either 0 or 1, this drives a corresponding output on z via the two values for in0 and in1.



```
Name       Value
in0[15:0]  111111111111111:
in1[15:0]  101010101010101
s          1
z[15:0]    101010101010101
```

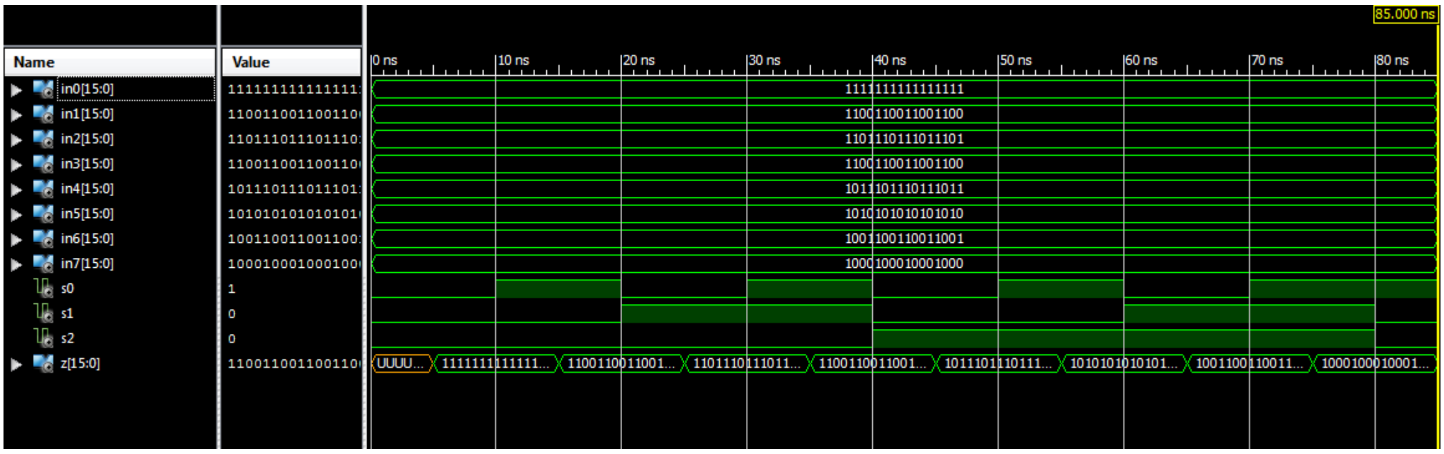
3.11 Mux 3-1 Bit

This multiplexer works correctly by giving out correct values on z through cycling between results acquired from in0, in1 and in2; which correspond to inputs s0 and s1 fed into the mux.



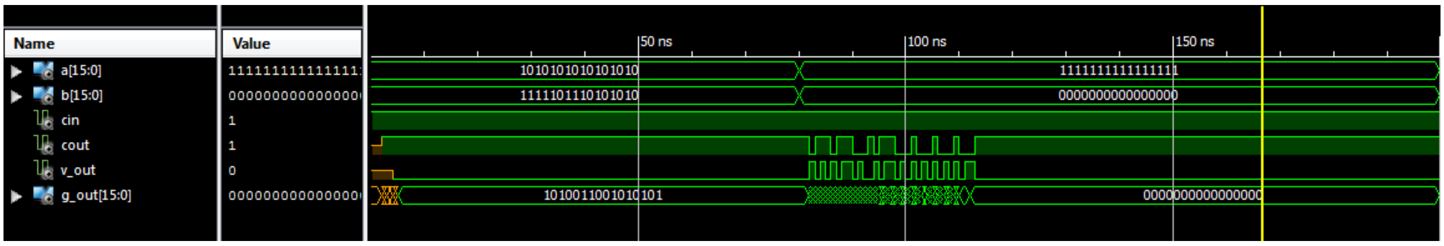
3.12 Mux 8-16 Bit

The multiplexer correctly takes all combinations of its inputs s0, s1, and s2 in its cycle, and outputs values on its z output by cycling through in0 to in7.



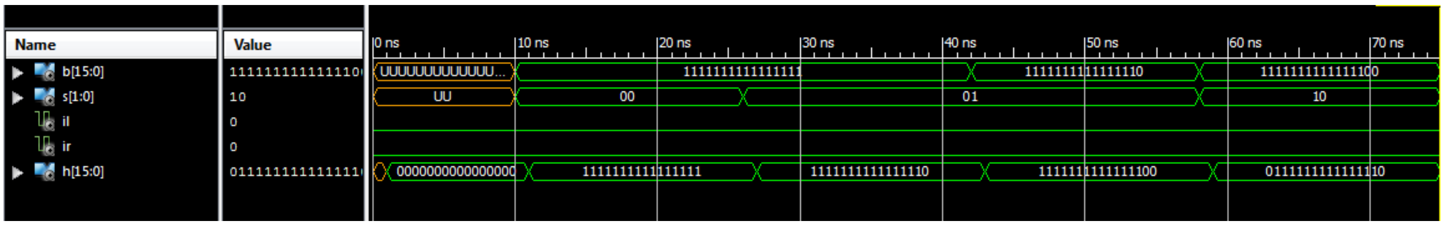
3.13 Ripple Adder

The rippled adder 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.



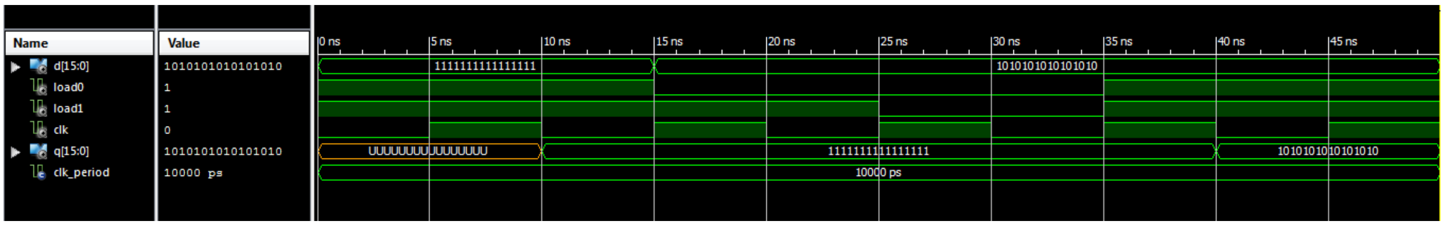
3.14 Shifter

The shifter 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.



3.15 Register

The register component shows correct behaviour of loading in new values if and only if a rising edge is detected on 5ns elapsing, given that the loads applied are both high.



Name	Value
d[15:0]	1010101010101010
load0	1
load1	1
clk	0
q[15:0]	1010101010101010
clk_period	10000 ps

```
d[15:0]     1111111111111111 | 1010101010101010
q[15:0]     UUUUUUUUUUUUUUUU | 1111111111111111 | 1010101010101010
clk_period  10000 ps
```