32 bit ALU and Testbench

Submitted by: Piyusha Jahagirdar

In this lab, we designed a 32-bit Arithmetic Logic Unit (ALU) by writing code VHDL which is then tested using the testbench and test vector.

Time taken to design and implement was five and half hours.

Table 1 is the Complete Table of Arithmetic Operations

Test	F[2:0]	A	B	Y	Zero
ADD 0+0	2	00000000	00000000	00000000	1
ADD 0+(-1)	2	00000000	FFFFFFFF	FFFFFFFF	0
ADD 1+(-1)	2	00000001	FFFFFFFF	00000000	1
ADD FF+1	2	000000FF	00000001	00000100	0
SUB 0-0	6	00000000	00000000	00000000	1
SUB 0-(-1)	6	00000000	FFFFFFFF	00000001	0
SUB 1-1	6	00000001	00000001	00000000	1
SUB 100-1	6	00000100	00000001	000000FF	0
SLT 0,0	7	00000000	00000000	00000000	1
SLT 0,1	7	00000000	00000001	00000001	0
SLT 0,-1	7	00000000	FFFFFFFF	00000000	1
SLT 1,0	7	00000001	00000000	00000001	0
SLT -1,0	7	FFFFFFFF	00000000	00000000	1
AND FFFFFFFF,FFFFFFFF	0	FFFFFFFF	FFFFFFFF	FFFFFFFF	0
AND FFFFFFFF, 12345678	0	FFFFFFFF	12345678	12345678	0
AND 12345678, 87654321	0	12345678	87654321	02244220	0
AND 00000000, FFFFFFFF	0	00000000	FFFFFFFF	00000000	1
OR FFFFFFFF, FFFFFFFF	1	FFFFFFFF	FFFFFFFF	FFFFFFFF	0
OR 12345678, 87654321	1	12345678	87654321	97755779	0
OR 00000000, FFFFFFFF	1	00000000	FFFFFFFF	FFFFFFFF	0
OR 00000000, 00000000	1	00000000	00000000	00000000	1

Table 1

VHDL CODE

ALU.VHD

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use IEEE.NUMERIC_STD.ALL;
use STD.TEXTIO.all;

entity alu is
port(a,b: in std_logic_vector(31 downto 0);
      f:  in std_logic_vector(2 downto 0);
```

```
       y:   out std_logic_vector(31 downto 0);
       zero: out std_logic
      );
end entity;

architecture sim of alu is
signal y1: std_logic_vector(31 downto 0);
 begin
   y<= y1;
process(f, a, b) begin
  case f is
    when "000" =>
      y1 <= a and b;
    when "001" =>
      y1 <= a or b;
    when "010" =>
       y1 <= std_logic_vector(signed(a) + signed(b));
    when "100" =>
       y1 <= a and not b;
    when "101" =>
       y1 <= a or not b;
    when "110" =>
       y1 <= std_logic_vector(signed(a) - signed(b));
    when "111"=>
         if (signed(a) < signed(b)) then
            y1 <= "00000000000000000000000000000001";
         else
                   y1 <= "00000000000000000000000000000000";
        end if;
   when others =>
         y1 <= a;

end case;

if(y1 = "00000000000000000000000000000000")then
 zero<= '0';
else
zero<= '1';
end if;
end process;
end;
```

ALU.TV

```
2_00000000_00000000_00000000_1
2_00000000_ FFFFFFFF_ FFFFFFFF_0
2_00000001_FFFFFFFF_00000000_1
2_000000FF_00000001_00000100_0
6_00000000_00000000_00000000_1
6_00000000_FFFFFFFF_00000001_0
6_00000001_00000001_00000000_1
6_00000100_00000001_000000FF_0
7_00000000_00000000_00000000_1
7_00000000_00000001_00000001_0
7_00000000_FFFFFFFF_00000000_1
7_00000001_00000000_00000001_0
7_FFFFFFFF_00000000_00000000_1
0_FFFFFFFF_FFFFFFFF_FFFFFFFF_0
0_FFFFFFFF_12345678_12345678_0
0_12345678_87654321_02244220_0
0_00000000_FFFFFFFF_00000000_1
1_FFFFFFFF_FFFFFFFF_FFFFFFFF_0
1_12345678_87654321_97755779_0
```

TESTBENCH.VHD

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.all;

entity testbench3 is
end;

architecture sim of testbench3 is
 component alu
   port (a,b: in std_logic_vector(31 downto 0);
       f:   in std_logic_vector(2 downto 0);
       y:   out std_logic_vector(31 downto 0);
       zero:      out std_logic
      );
      end component;
  signal a,b,y: std_logic_vector(31 downto 0);
  signal f: std_logic_vector(2 downto 0);
  signal zero: std_logic;
  signal y_expected:   std_logic_vector(31 downto 0);
  signal zero_expected:      std_logic;
  signal clk, reset:   std_logic;

begin
-- instantiate device under test
  dut:      alu port map(a,b,f,y,zero);
```

```
-- generate clock
  process begin
     clk <= '1' ; wait for 5 ns;
     clk <= '0' ; wait for 5 ns;
  end process;

  -- at start of test, pulse reset
  process begin
     reset <= '1';  wait for 27 ns; reset <= '0';
     wait;
  end process;

  -- run tests
  process is
    file tv: text;
    variable L: line;
    variable vector_a: std_logic_vector(31 downto 0);
    variable dummy1:   character;
    variable vector_b: std_logic_vector(31 downto 0);
    variable dummy2:   character;
    variable vector_f: std_logic_vector(3 downto 0);
    variable dummy3:   character;
    variable vector_y_ex: std_logic_vector(31 downto 0);
    variable dummy4:   character;
    variable vector_z_ex: std_logic_vector(3 downto 0);
    variable vectornum: integer := 0;
    variable errors: integer := 0;
  begin
    FILE_OPEN(tv, "testbench.tv", READ_MODE);
    while not endfile(tv) loop

      -- change vectors on rising edge
      wait until rising_edge(clk);

      -- read the next line of testvectors split it up
      readline(tv, L);
      hread(L,vector_f);
      read(L, dummy3);
      hread(L, vector_a);
      read(L, dummy1);
      hread(L,vector_b);
      read(L, dummy2);
      hread(L,vector_y_ex);
      read(L, dummy4);
      hread(L,vector_z_ex);
      a <= vector_a after 1 ns;
      b <= vector_b;
      f <= vector_f(2 downto 0);
      y_expected <= vector_y_ex;
      zero_expected <= vector_z_ex(0);

      wait until falling_edge(clk);
      if y /= y_expected then
        report("Error with line: " & integer'image(vectornum));
```

```
       -- report string'image(L);
        report("zero = " & std_logic'image(zero));
       -- report("Error: y=" & std_logic_vector'image(y));
       -- report("Error: a=" & std_logic_vector'image(a));
       -- report("Error: b=" & std_logic'image(b));
        --report("Error: f=" & std_logic'image(f));
        errors := errors + 1;
      end if;
      vectornum := vectornum + 1;
      end loop;

if errors=0 then

  report "NO ERRORS" & integer'image(vectornum)  & "tests completed"
severity failure;

  else


report integer'image(errors) & " ERRORS in " &


    integer'image(vectornum) & "tests" severity failure;

  end if;


      end process;
end;
```

Submitted by: Piyusha Jahagirdar

Waveform with the results: