**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';

zero<= '1';

else

zero<= '1';

end if;

end process;

end ;

**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, "J:\Lab8\alu.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;**