```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4   use STD.textio.all;
5   use ieee.std_logic_textio.all;
6   use work.insbuffer_type.all;
7
8   entity file_io_tb is
9   end file_io_tb;
10
11
12  architecture behave of file_io_tb is
13
14    ------------------------------------------------------------------------
    --
15    -- Declare the Component Under Test
16    ------------------------------------------------------------------------
    --
17
18    signal tb_clk: std_logic;
19    signal tb_alu_out: std_logic_vector(127 downto 0);
20    signal rom_data: insBuffer;
21    signal count:integer :=0;
22    signal alu_out: std_logic_vector(127 downto 0);
23
24    constant period : time := 20 ns;
25
26
27    ------------------------------------------------------------------------
    --
28    -- Instantiate and Map UUT
29    ------------------------------------------------------------------------
    --
30    file file input : text;
31    file file_output: text;
32
33
34
35    begin
36        UUT: entity pipiline_unit port map(
37            ins_in => rom_data,
38            clk => tb_clk,
39            ALU_o => tb_alu_out
40            );
41
42
43
44
    ------------------------------------------------------------------------
45    -- This procedure reads the file input_vectors.txt which is located in
    the
46    -- simulation project area.
47    -- It will read the data in and send it to the ripple-adder component
48    -- to perform the operations.  The result is written to the
49    -- output_results.txt file, located in the same directory.
50
    ------------------------------------------------------------------------
51    process
52      variable ILINE : line;
53      variable INS_LINE  : std_logic_vector(24 downto 0);
54      variable count: integer:=0;
```

```
55      variable v_out : line;
56
57    begin
58      file_open(file_input, "opcode.txt",  read_mode);
59      file_open(file_output, "alu_out.txt", write_mode);
60
61
62      while not endfile(file_input) loop
63        readline(file input, ILINE);
64        read(ILINE, INS_LINE);
65        rom_data(count)<= INS_LINE;
66        count := count+1;
67      end loop;
68      file_close(file_input);
69
70
71      for i in 0 to count +4 loop
72          wait for period;
73          write(v_out, tb_alu_out);
74          writeline(file_output, v_out);
75      end loop;
76      file_close(file_output);
77
78    end process;
79
80
81
82    clock: process
83    begin
84        tb_clk <='0';
85        wait for 10ns;
86        tb_clk<='1';
87        wait for 10ns;
88    end process;
89  end behave;
```