```
-- Entity: chip_full_test
-- Architecture : test
-- Author: cpatel2
-- Created On: 11/01/05
--
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;
use IEEE.std_logic_arith.all;
use STD.textio.all;

entity chip_full_test is

end chip_full_test;

architecture test of chip_full_test is

  component chip
    port (
      cpu_add    : in  std_logic_vector(7 downto 0);
      cpu_data   : inout  std_logic_vector(7 downto 0);
      cpu_rd_wrn : in  std_logic;
      start      : in  std_logic;
      clk        : in  std_logic;
      reset      : in  std_logic;
      mem_data   : in  std_logic_vector(7 downto 0);
      Vdd        : in  std_logic;
      Gnd        : in  std_logic;
      busy       : out std_logic;
      mem_en     : out std_logic;
      mem_add    : out std_logic_vector(7 downto 0));
  end component;



    shared variable done    :   boolean := false;

  signal Vdd, Gnd: std_logic;
  signal cpu_add, cpu_data, mem_data, mem_add: std_logic_vector(7 downto 0);
  signal cpu_rd_wrn, reset, clk, start, clock, busy, mem_en: std_logic;

  signal clk_count: integer:=0;

procedure print_output is
   variable out_line: line;

   begin
   write (out_line, string' (" Clock: "));
   write (out_line, clk_count);
   write (out_line, string'(" Start: "));
   write (out_line, start);
   write (out_line, string'(" Cpu Read/Write: "));
   write (out_line, cpu_rd_wrn);
   write (out_line, string'(" Reset: "));
   write (out_line, reset);
   writeline(output, out_line);

   write (out_line, string' (" CPU address: "));
   write (out_line, cpu_add);
   write (out_line, string'(" CPU data: "));
   write (out_line, cpu_data);
   writeline(output, out_line);

   write (out_line, string'(" Memory data: "));
   write (out_line, mem_data);
   writeline(output, out_line);
   writeline(output, out_line);

   write (out_line, string'(" Busy: "));
   write (out_line, busy);
   write (out_line, string'(" Memory  Enable: "));
   write (out_line, mem_en);
   writeline(output, out_line);

   write (out_line, string'(" Memory  Address: "));
   write (out_line, mem_add);
   writeline(output, out_line);

   write (out_line, string'(" ------------------------------------------")
);
   writeline(output, out_line);


end print_output;

for c1 : chip use entity work.chip(structural);

begin

  Vdd <= '1';
  Gnd <= '0';
  clk <= clock;

  c1 : chip port map (cpu_add, cpu_data, cpu_rd_wrn, start, clk, reset, mem_da
ta, Vdd, Gnd, busy, mem_en, mem_add);

  clking : process
  begin
    if  done = true then
        wait;
    else
        clock<= '1', '0' after 5 ns;
        wait for 10 ns;
    end if;
  end process clking;

  io_process: process

    file infile  : text is "./chip_full_in.txt";
    variable buf: line;
    variable value: std_logic_vector(7 downto 0);
    variable value1: std_logic;

  begin

    while not (endfile(infile)) loop

      wait until rising_edge(clock);

      readline(infile, buf);

      readline(infile, buf);
      read(buf, value);
      cpu_add <= value;
```

```
  123:
  124:       readline(infile, buf);
  125:       read(buf, value);
  126:       cpu_data <= value;
  127:
  128:       readline(infile, buf);
  129:       read(buf, value1);
  130:       cpu_rd_wrn <= value1;
  131:
  132:       readline(infile, buf);
  133:       read(buf, value1);
  134:       start <= value1;
  135:
  136:       readline(infile, buf);
  137:       read(buf, value1);
  138:       reset <= value1;
  139:
  140:       clk_count <= clk_count+1;
  141:
  142:       wait until falling_edge(clock);
  143:
  144:       readline(infile, buf);
  145:       read(buf, value);
  146:       mem_data <= value;
  147:
  148:     end loop;
  149:     done := true;
  150:     wait;
  151:
  152:   end process io_process;
  153:
  154: print_process: process
  155:
  156:   variable out_line: line;
  157:
  158: begin
  159:
  160:   wait until ((falling_edge(clock) and start ='1') or busy'EVENT or mem_en'EVE
NT);
  161:   wait for 1 ns;
  162:   print_output;
  163:
  164: end process print_process;
  165:
  166: end test;
```