```vhdl
-- Entity: chip_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_test is

end chip_test;

architecture test of chip_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;



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

  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;
    shared variable done    :   boolean :=false;
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;



begin

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

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

  clking : process
  begin
    if done 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_in.txt";
    variable out_line: line;
    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);
      print_output;

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

      readline(infile, buf);
      read(buf, value);
      cpu_data <= value;

      readline(infile, buf);
      read(buf, value1);
      cpu_rd_wrn <= value1;

      readline(infile, buf);
      read(buf, value1);
      start <= value1;

      readline(infile, buf);
      read(buf, value1);
      reset <= value1;

      wait until falling_edge(clock);

      readline(infile, buf);
      read(buf, value);
      mem_data <= value;

      clk_count <= clk_count+1;

      print_output;

    end loop;
    done := true;
    wait;

  end process io_process;


end test;
```