

## chip\_test.vhd

```

1: -- Entity: chip_test
2: -- Architecture : test
3: -- Author: cpatel2
4: -- Created On: 11/01/05
5: --
6: library IEEE;
7: use IEEE.std_logic_1164.all;
8: use IEEE.std_logic_textio.all;
9: use IEEE.std_logic_arith.all;
10: use STD.textio.all;
11:
12: entity chip_test is
13:
14: end chip_test;
15:
16: architecture test of chip_test is
17:
18:     component chip
19:     port (
20:         cpu_add      : in  std_logic_vector(7 downto 0);
21:         cpu_data      : inout std_logic_vector(7 downto 0);
22:         cpu_rd_wrn    : in  std_logic;
23:         start         : in  std_logic;
24:         clk           : in  std_logic;
25:         reset         : in  std_logic;
26:         mem_data      : in  std_logic_vector(7 downto 0);
27:         Vdd           : in  std_logic;
28:         Gnd           : in  std_logic;
29:         busy          : out std_logic;
30:         mem_en        : out std_logic;
31:         mem_add       : out std_logic_vector(7 downto 0));
32:     end component;
33:
34:
35:
36:     for c1 : chip use entity work.chip(structural);
37:
38:     signal Vdd, Gnd: std_logic;
39:     signal cpu_add, cpu_data, mem_data, mem_add: std_logic_vector(7 downto 0);
40:     signal cpu_rd_wrn, reset, clk, start, clock, busy, mem_en: std_logic;
41:
42:     signal clk_count: integer:=0;
43:     shared variable done      : boolean :=false;
44:     procedure print_output is
45:         variable out_line: line;
46:
47:     begin
48:         write (out_line, string' (" Clock: "));
49:         write (out_line, clk_count);
50:         write (out_line, string' (" Start: "));
51:         write (out_line, start);
52:         write (out_line, string' (" Cpu Read/Write: "));
53:         write (out_line, cpu_rd_wrn);
54:         write (out_line, string' (" Reset: "));
55:         write (out_line, reset);
56:         writeline(output, out_line);
57:
58:         write (out_line, string' (" CPU address: "));
59:         write (out_line, cpu_add);
60:         write (out_line, string' (" CPU data: "));
61:         write (out_line, cpu_data);
62:         writeline(output, out_line);
63:
64:         write (out_line, string' (" Memory data: "));
65:         write (out_line, mem_data);
66:         writeline(output, out_line);
67:         writeline(output, out_line);
68:
69:         write (out_line, string' (" Busy: "));
70:         write (out_line, busy);
71:         write (out_line, string' (" Memory Enable: "));
72:         write (out_line, mem_en);
73:         writeline(output, out_line);
74:
75:         write (out_line, string' (" Memory Address: "));
76:         write (out_line, mem_add);
77:         writeline(output, out_line);
78:
79:         write (out_line, string' (" -----"))
80:     );
81:     writeline(output, out_line);
82:
83: end print_output;
84:
85:
86:
87: begin
88:
89:     Vdd <= '1';
90:     Gnd <= '0';
91:     clk <= clock;
92:
93:     c1 : chip port map (cpu_add, cpu_data, cpu_rd_wrn, start, clk, reset, mem_data, Vdd, Gnd, busy, mem_en, mem_add);
94:
95:     clking : process
96:     begin
97:         if done then
98:             wait;
99:         else
100:             clock<= '1', '0' after 5 ns;
101:             wait for 10 ns;
102:         end if;
103:     end process clking;
104:
105:     io_process: process
106:
107:         file infile : text is "./chip_in.txt";
108:         variable out_line: line;
109:         variable buf: line;
110:         variable value: std_logic_vector(7 downto 0);
111:         variable value1: std_logic;
112:
113:     begin
114:
115:         while not (endfile(infile)) loop
116:
117:             wait until rising_edge(clock);
118:             print_output;
119:
120:             readline(infile, buf);
121:             read(buf, value);
122:             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:     wait until falling_edge(clock);
141:
142:     readline(infile, buf);
143:     read(buf, value);
144:     mem_data <= value;
145:
146:     clk_count <= clk_count+1;
147:
148:     print_output;
149:
150:     end loop;
151:     done := true;
152:     wait;
153:
154: end process io_process;
155:
156:
157: end test;
```