

```

1: library STD;
2: library IEEE;
3: use IEEE.std_logic_1164.all;
4: use IEEE.std_logic_textio.all;
5: use STD.textio.all;
6:
7: entity Counter_Test is
8:
9: end Counter_Test;
10:
11: architecture test of Counter_Test is
12:
13:     component Counter
14:     port(
15:         clk          : in std_logic;
16:         hit_miss     : in std_logic; --1 hit 0 miss
17:         rd_wr        : in std_logic; --1 read 0 write
18:         start        : in std_logic;
19:         Vdd           : in std_logic;
20:         Gnd           : in std_logic;
21:         reset        : in std_logic;
22:         busy         : out std_logic;
23:         rd_wr_o       : out std_logic;
24:         cache_write   : out std_logic;
25:         rm_wr_en      : out std_logic;
26:         wr_hit        : out std_logic;
27:         cpu_dout_en   : out std_logic;
28:         mem_enable    : out std_logic;
29:         write_0       : out std_logic; --write word0 to cache
30:         write_1       : out std_logic; --write word1 to cache
31:         write_2       : out std_logic; --write word2 to cache
32:         write_3       : out std_logic; --write word3 to cache
33:     );
34: end component;
35:
36: signal s_clk      : std_logic;
37: signal s_hit_miss : std_logic := '0';
38: signal s_rd_wr    : std_logic := 'Z';
39: signal s_start    : std_logic := '0';
40: signal s_reset    : std_logic;
41: signal s_busy     : std_logic;
42: signal s_rd_wr_o  : std_logic;
43: signal s_cache_write : std_logic;
44: signal s_rm_wr_en : std_logic;
45: signal s_wr_hit   : std_logic;
46: signal s_cpu_dout_en : std_logic;
47: signal s_mem_enable : std_logic;
48: signal s_write_0   : std_logic;
49: signal s_write_1   : std_logic;
50: signal s_write_2   : std_logic;
51: signal s_write_3   : std_logic;
52:
53: shared variable done : boolean := false;
54:
55: for asdt : Counter use entity work.Counter(structural);
56:
57: begin
58:
59:     asdt : Counter port map(
60:         clk      => s_clk,
61:         hit_miss => s_hit_miss,
62:         rd_wr    => s_rd_wr,

```

```

63:         start    => s_start,
64:         Vdd      => '1',
65:         Gnd      => '0',
66:         reset    => s_reset,
67:         busy     => s_busy,
68:         rd_wr_o  => s_rd_wr_o,
69:         cache_write => s_cache_write,
70:         rm_wr_en => s_rm_wr_en,
71:         wr_hit   => s_wr_hit,
72:         cpu_dout_en => s_cpu_dout_en,
73:         mem_enable => s_mem_enable,
74:         write_0  => s_write_0,
75:         write_1  => s_write_1,
76:         write_2  => s_write_2,
77:         write_3  => s_write_3
78:     );
79:
80:     c : process
81:     begin
82:         if done then
83:             wait;
84:         else
85:             s_clk <= '1', '0' after 5 ns;
86:             wait for 10 ns;
87:         end if;
88:     end process;
89:
90:     main : process
91:
92:         variable v_clk      : std_logic;
93:         variable v_hit_miss : std_logic;
94:         variable v_rd_wr    : std_logic;
95:         variable v_busy     : std_logic;
96:         variable v_start    : std_logic;
97:         variable v_write_0  : std_logic;
98:         variable v_write_1  : std_logic;
99:         variable v_write_2  : std_logic;
100:        variable v_write_3  : std_logic;
101:
102:     begin
103:         s_reset    <= '1';
104:         wait for 20 ns;
105:         s_reset    <= '0';
106:         wait for 10 ns;
107:         s_rd_wr    <= '1';
108:         s_start    <= '1';
109:         wait for 10 ns;
110:         s_rd_wr    <= 'Z';
111:         s_start    <= '0';
112:         s_hit_miss <= '0';
113:         wait for 190 ns;
114:         s_start    <= '1';
115:         s_rd_wr    <= '1';
116:         wait for 10 ns;
117:         s_rd_wr    <= 'Z';
118:         s_start    <= '0';
119:         s_hit_miss <= '1';
120:         wait for 40 ns;
121:         s_rd_wr    <= '0';
122:         s_start    <= '1';
123:         s_hit_miss <= '0';
124:         wait for 10 ns;

```

```
125:      s_rd_wr      <= 'Z';
126:      s_start      <= '0';
127:      wait for 50 ns;
128:      s_rd_wr      <= '0';
129:      s_start      <= '1';
130:      s_hit_miss    <= '1';
131:      wait for 10 ns;
132:      s_rd_wr      <= 'Z';
133:      s_start      <= '0';
134:      wait for 50 ns;
135:      done := true;
136:      wait;
137:    end process;
138: end test;
```