```
  1: library STD;
  2: library IEEE;
  3: use IEEE.std_logic_1164.all;
  4:
  5: entity chip is
  6:     port(
  7:         cpu_add     :   in      std_logic_vector(7 downto 0);
  8:         cpu_data    :   inout   std_logic_vector(7 downto 0);
  9:         cpu_rd_wrn  :   in      std_logic;
 10:         start       :   in      std_logic;
 11:         clk         :   in      std_logic;
 12:         reset       :   in      std_logic;
 13:         mem_data    :   in      std_logic_vector(7 downto 0);
 14:         Vdd         :   in      std_logic;
 15:         Gnd         :   in      std_logic;
 16:         busy        :   out     std_logic;
 17:         mem_en      :   out     std_logic;
 18:         mem_add     :   out     std_logic_vector(7 downto 0)
 19:     );
 20: end chip;
 21:
 22: architecture structural of chip is
 23:
 24:     component Counter
 25:     port(
 26:         clk         :   in  std_logic;
 27:         hit_miss    :   in  std_logic;  --1 hit 0 miss
 28:         rd_wr       :   in  std_logic;  --1 read 0 write
 29:         start       :   in  std_logic;
 30:         Vdd         :   in  std_logic;
 31:         Gnd         :   in  std_logic;
 32:         reset       :   in  std_logic;
 33:         busy        :   out std_logic;
 34:         rd_wr_o     :   out std_logic;
 35:         cache_write :   out std_logic;
 36:         rm_wr_en    :   out std_logic;
 37:         wr_hit      :   out std_logic;
 38:         cpu_dout_en :   out std_logic;
 39:         mem_enable  :   out std_logic;
 40:         write_0     :   out std_logic;  --write word0 to cache
 41:         write_1     :   out std_logic;  --write word1 to cache
 42:         write_2     :   out std_logic;  --write word2 to cache
 43:         write_3     :   out std_logic  --write word3 to cache
 44:     );
 45:     end component;
 46:
 47:     component Cache_Block
 48:     port(
 49:         Data_In     :   in  std_logic_vector(7 downto 0);
 50:         Tag_In      :   in  std_logic_vector(2 downto 0);
 51:         Set_Valid   :   in  std_logic;
 52:         Rd_Wr       :   in  std_logic;
 53:         Cache_Write :   in  std_logic;
 54:         Col_En      :   in  std_logic_vector(3 downto 0);
 55:         Row_En      :   in  std_logic_vector(7 downto 0);
 56:         Tag_Wr_En   :   in  std_logic;
 57:         Gnd         :   in  std_logic;
 58:         reset       :   in  std_logic;
 59:         Data_Out    :   out std_logic_vector(7 downto 0);
 60:         Tag_Out     :   out std_logic_vector(2 downto 0);
 61:         Valid_Out   :   out std_logic
 62:     );
 63:     end component;
 64:
 65:     component Decoder
 66:     port(
 67:         Mem_Add :   in  std_logic_vector(4 downto 0);
 68:         Col_En  :   out std_logic_vector(3 downto 0);
 69:         Row_En  :   out std_logic_vector(7 downto 0)
 70:     );
 71:     end component;
 72:
 73:     component Hit_Miss
 74:     port(
 75:         tag1    :   in  std_logic_vector(2 downto 0);
 76:         tag2    :   in  std_logic_vector(2 downto 0);
 77:         Valid   :   in  std_logic;
 78:         HitMiss :   out std_logic
 79:     );
 80:     end component;
 81:
 82:     component Output_Enable
 83:     port(
 84:         in8     :   in  std_logic_vector(7 downto 0);
 85:         enable  :   in  std_logic;
 86:         out8    :   out std_logic_vector(7 downto 0)
 87:     );
 88:     end component;
 89:
 90:     component register8
 91:     port(
 92:         d       :   in  std_logic_vector(7 downto 0);
 93:         clk     :   in  std_logic;
 94:         reset   :   in  std_logic;
 95:         Gnd     :   in  std_logic;
 96:         q       :   out std_logic_vector(7 downto 0)
 97:     );
 98:     end component;
 99:
100:     component or2
101:     port(
102:         in1     :   in  std_logic;
103:         in2     :   in  std_logic;
104:         out1    :   out std_logic
105:     );
106:     end component;
107:
108:     component nor2
109:     port(
110:         in1     :   in  std_logic;
111:         in2     :   in  std_logic;
112:         out1    :   out std_logic
113:     );
114:     end component;
115:
116:     component and2
117:     port(
118:         in1     :   in  std_logic;
119:         in2     :   in  std_logic;
120:         out1    :   out std_logic
121:     );
122:     end component;
123:
124:     component nand2
```

```
125:     port(
126:         in1     :   in  std_logic;
127:         in2     :   in  std_logic;
128:         out1    :   out std_logic
129:     );
130:     end component;
131:
132:     component invX1
133:     port(
134:         in1     :   in  std_logic;
135:         out1    :   out std_logic
136:     );
137:     end component;
138:
139:     --register signals
140:     signal addr_reg_out     :   std_logic_vector(7 downto 0);
141:     signal data_reg_out     :   std_logic_vector(7 downto 0);
142:
143:     --cache enable signals
144:     signal col_dec_out      :   std_logic_vector(3 downto 0);
145:     signal col_en           :   std_logic_vector(3 downto 0);
146:     signal row_dec_out      :   std_logic_vector(7 downto 0);
147:
148:     --cache block signals
149:     signal data_bus         :   std_logic_vector(7 downto 0);
150:     signal cache_data_out   :   std_logic_vector(7 downto 0);
151:     signal cache_tag_out    :   std_logic_vector(2 downto 0);
152:     signal cache_valid_out  :   std_logic;
153:
154:     --hit detection
155:     signal hitmiss          :   std_logic;
156:
157:     --state machine signals
158:     signal w0               :   std_logic;
159:     signal w1               :   std_logic;
160:     signal w2               :   std_logic;
161:     signal w3               :   std_logic;
162:     signal dout_en          :   std_logic;
163:     signal aout_en          :   std_logic;
164:     signal rdwr             :   std_logic;
165:     signal nrdwr            :   std_logic;
166:
167:     --register write enable signals
168:     signal nstart           :   std_logic;
169:     signal reg_clk_en       :   std_logic;
170:
171:     --signals for turning off regular block select during write from mem
172:     signal col_dec_en       :   std_logic_vector(3 downto 0);
173:     signal rd_miss          :   std_logic;
174:     signal nrd_miss         :   std_logic;
175:     signal internal_busy    :   std_logic;
176:     signal nbusy            :   std_logic;
177:     signal e                :   std_logic;
178:
179:     signal delayed_rm       :   std_logic;
180:
181:     signal reset_nor_start  :   std_logic;
182:
183:     signal wr_hit           :   std_logic;
184:     signal wr_on_wr_hit     :   std_logic_vector(3 downto 0);
185:     signal wr_on_rd_miss    :   std_logic_vector(3 downto 0);
186:     signal cache_write      :   std_logic;
187:     signal rm_wr_en         :   std_logic;
188:
189:     for nreadwr, nrm, nb, not_start    :   invX1   use entity work.invX1(struc
tural);
190:     for data_reg, addr_reg  :   register8 use entity work.register8(structural
);
191:     for decode              :   Decoder use entity work.Decoder(structural);
192:     for cache               :   Cache_Block use entity work.Cache_Block(struct
ural);
193:     for hm                  :   Hit_Miss use entity work.Hit_Miss(structural);
194:     for state               :   Counter use entity work.Counter(structural);
195:     for outen0, outen1, outen2      :   Output_Enable use entity work.Output_E
nable(structural);
196:     for rnors               :   nor2 use entity work.nor2(structural);
197:     for reg_en, colen0, colen1, colen2, colen3 : or2 use entity work.or2(struc
tural);
198:     for rm0, rm1, rm2, rm3, colen_wr0, colen_wr1, colen_wr2, colen_wr3  :   an
d2 use entity work.and2(structural);
199:     for oen                 :   nand2 use entity work.nand2(structural);
200:
201:
202: begin
203:
204:     busy        <=  internal_busy;
205:     data_bus    <=  mem_data;
206:     nreadwr     :   invX1       port map(rdwr, nrdwr);
207:     data_reg    :   register8   port map(cpu_data, reg_clk_en, reset, Gnd, dat
a_reg_out);
208:     addr_reg    :   register8   port map(cpu_add, reg_clk_en, reset, Gnd, addr
_reg_out);
209:     decode      :   Decoder     port map(addr_reg_out(4 downto 0), col_dec_out
, row_dec_out);
210:     cache       :   Cache_Block port map(data_bus, addr_reg_out(7 downto 5), w
0, rdwr, cache_write, col_en, row_dec_out, w0, Gnd, reset, cache_data_out, cache_tag_o
ut, cache_valid_out);
211:     hm          :   Hit_Miss    port map(addr_reg_out(7 downto 5), cache_tag_o
ut, cache_valid_out, hitmiss);
212:     state       :   Counter     port map(clk, hitmiss, cpu_rd_wrn, start, Vdd,
 Gnd, reset, internal_busy, rdwr, cache_write, rm_wr_en, wr_hit, dout_en, aout_en, w0,
 w1, w2, w3);
213:     outen0      : Output_Enable port map(cache_data_out, dout_en, cpu_data);
214:     outen1      : Output_Enable port map(addr_reg_out, aout_en, mem_add);
215:
216:     --register clock enable
217:     not_start   :   invX1       port map(start, nstart);
218:     rnors       :   nor2         port map(reset, start, reset_nor_start);
219:     reg_en      :   or2         port map(reset_nor_start, clk, reg_clk_en);
220:
221:     mem_en      <=  aout_en;
222:
223:     --column enable signals for cache
224:     nrm         :   invX1       port map(rd_miss, nrd_miss);
225:     nb          :   invX1       port map(internal_busy, nbusy);
226:     oen         :   nand2       port map(rm_wr_en, internal_busy, e);
227:     rm0         :   and2        port map(e, col_dec_out(0), col_dec_en(0));
228:     rm1         :   and2        port map(e, col_dec_out(1), col_dec_en(1));
229:     rm2         :   and2        port map(e, col_dec_out(2), col_dec_en(2));
230:     rm3         :   and2        port map(e, col_dec_out(3), col_dec_en(3));
231:     colen0      :   or2         port map(w0, col_dec_en(0), col_en(0));
232:     colen1      :   or2         port map(w1, col_dec_en(1), col_en(1));
233:     colen2      :   or2         port map(w2, col_dec_en(2), col_en(2));
234:     colen3      :   or2         port map(w3, col_dec_en(3), col_en(3));
```

```
  235:     colen_wr0   :   and2        port map(wr_hit,  col_dec_out(0), wr_on_wr_hit
(0));
  236:     colen_wr1   :   and2        port map(wr_hit,  col_dec_out(1), wr_on_wr_hit
(1));
  237:     colen_wr2   :   and2        port map(wr_hit,  col_dec_out(2), wr_on_wr_hit
(2));
  238:     colen_wr3   :   and2        port map(wr_hit,  col_dec_out(3), wr_on_wr_hit
(3));
  239:
  240:     outen2      :   Output_Enable   port map(data_reg_out, nrdwr, data_bus);
  241:
  242: end structural;
```