

Example 8—A module of memory dimm

```

module dimm (adr, ba, rasx, casx, csx, wex, dqm, cke, data, clk,
              dev_id) ;

  parameter [31:0] MEM_WIDTH = 16, MEM_SIZE = 8; // in mbytes

  input [10:0] adr;
  input      ba;
  input      rasx, casx, csx, wex;
  input [ 7:0] dqm;
  input      cke;
  inout [63:0] data;
  input      clk;
  input [ 4:0] dev_id;

  genvar      i;

  generate
    case ({MEM_SIZE, MEM_WIDTH})
      {32'd8, 32'd16}: // 8Meg x 16 bits wide.
      begin
        // The generated instance names are word[3].p, word[2].p,
        // word[1].p, word[0].p, and the task read_mem
        for (i=0; i<4; i=i+1) begin:word
          sms_16b216t0 p(.clk(clk), .csb(csx), .cke(cke), .ba(ba[0]),
                      .addr(adr[10:0]), .rasb(rasx), .casb(casx),
                      .web(wex), .udqm(dqm[2*i+1]), .ldqm(dqm[2*i]),
                      .dqi(data[15+16*i:16*i]), .dev_id(dev_id[4:0]));
        end
        task read_mem;
          input [31:0] address;
          output [63:0] data;
          begin
            word[3].p.read_mem(address, data[63:48]);
            word[2].p.read_mem(address, data[47:32]);
            word[1].p.read_mem(address, data[31:16]);
            word[0].p.read_mem(address, data[15:0]);
          end
        endtask
      end
      {32'd16, 32'd8}: // 16Meg x 8 bits wide.
      begin
        // The generated instance names are byte[7].p, byte[6].p,
        // byte[5].p, byte[4].p, byte[3].p, byte[2].p, byte[1].p,
        // byte[0].p and the task read_mem
        for (i=0; i<8; i=i+1) begin:byte
          sms_16b208t0 p(.clk(clk), .csb(csx), .cke(cke), .ba(ba[0]),
                      .addr(adr[10:0]), .rasb(rasx), .casb(casx),
                      .web(wex), .dqm(dqm[i]),
                      .dqi(data[7+8*i:8*i]), .dev_id(dev_id[4:0]));
        end
        task read_mem;
          input [31:0] address;
          output [63:0] data;
          begin
            byte[7].p.read_mem(address, data[63:56]);
            byte[6].p.read_mem(address, data[55:48]);
            byte[5].p.read_mem(address, data[47:40]);
            byte[4].p.read_mem(address, data[39:32]);
            byte[3].p.read_mem(address, data[31:24]);
            byte[2].p.read_mem(address, data[23:16]);
            byte[1].p.read_mem(address, data[15: 8]);
            byte[0].p.read_mem(address, data[ 7: 0]);
          end
        endtask
      end
      // Other memory cases ...
    endcase
  endgenerate
endmodule

```