```
// This is a Verilog description for an 4 x 16 register file

`timescale 1ns / 1ns

module regfile4x16a
  (input clk,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output [15:0] rdDataA,
   input [2:0] rdAddrB,
   output [15:0] rdDataB);

   reg [15:0]    reg0, reg1, reg2, reg3;

   assign rdDataA = rdAddrA == 0 ? reg0 :
                    rdAddrA == 1 ? reg1 :
                    rdAddrA == 2 ? reg2 :
                    rdAddrA == 3 ? reg3 : 0;
   assign rdDataB = rdAddrB == 0 ? reg0 :
                    rdAddrB == 1 ? reg1 :
                    rdAddrB == 2 ? reg2 :
                    rdAddrB == 3 ? reg3 : 0;

   always @(posedge clk) begin
      if (write)
        case (wrAddr)
          0: begin
             reg0 <= wrData;
          end
          1: begin
             reg1 <= wrData;
          end
          2: begin
             reg2 <= wrData;
          end
          3: begin
             reg3 <= wrData;
          end
        endcase // case (wrAddr)
   end // always @ (posedge clk)
endmodule
=========================================


// This is a Verilog description for an 4 x 16 register file

`timescale 1ns / 1ns

module regfile4x16b
  (input clk,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output reg [15:0] rdDataA,
   input [2:0] rdAddrB,
   output reg [15:0] rdDataB);

   reg [15:0]    reg0, reg1, reg2, reg3;

   always @(*) begin
      case (rdAddrA)
```

```
        0: rdDataA = reg0;
        1: rdDataA = reg1;
        2: rdDataA = reg2;
        3: rdDataA = reg3;
        default: rdDataA = 16'hXXXX;
      endcase
   end
   always @(*) begin
      case (rdAddrB)
        0: rdDataB = reg0;
        1: rdDataB = reg1;
        2: rdDataB = reg2;
        3: rdDataB = reg3;
        default: rdDataB = 16'hXXXX;
      endcase
   end
   always @(posedge clk) begin
      if (write)
        case (wrAddr)
          0: reg0 <= wrData;
          1: reg1 <= wrData;
          2: reg2 <= wrData;
          3: reg3 <= wrData;
        endcase // case (wrAddr)
   end // always @ (posedge clk)
endmodule
=============================================


// This is a Verilog description for an 4 x 16 register file

`timescale 1ns / 1ns

module regfile4x16c
  (input clk,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output reg [15:0] rdDataA,
   input [2:0] rdAddrB,
   output reg [15:0] rdDataB);

   reg [15:0]    reg0, reg1, reg2, reg3;

   always @(*) begin
      case (rdAddrA)
        0: rdDataA = reg0;
        1: rdDataA = reg1;
        2: rdDataA = reg2;
        3: rdDataA = reg3;
        default: rdDataA = 16'hXXXX;
      endcase
      case (rdAddrB)
        0: rdDataB = reg0;
        1: rdDataB = reg1;
        2: rdDataB = reg2;
        3: rdDataB = reg3;
        default: rdDataB = 16'hXXXX;
      endcase
   end
   always @(posedge clk) begin
      if (write)
        case (wrAddr)
          0: reg0 <= wrData;
          1: reg1 <= wrData;
          2: reg2 <= wrData;
          3: reg3 <= wrData;
        endcase // case (wrAddr)
   end // always @ (posedge clk)
endmodule
=====================================


// This is a Verilog description for an 8 x 16 register file

`timescale 1ns / 1ns

module regfile8x16a
  (input clk,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output [15:0] rdDataA,
   input [2:0] rdAddrB,
   output [15:0] rdDataB);

   reg [15:0]    regfile [0:7];

   assign rdDataA = regfile[rdAddrA];
   assign rdDataB = regfile[rdAddrB];

   always @(posedge clk) begin
      if (write) regfile[wrAddr] <= wrData;
   end
endmodule
=========================================


// This is a Verilog description for an 8 x 16 register file

`timescale 1ns / 1ns

module regfile8x16b
  (input clk,
   input reset,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output [15:0] rdDataA,
   input [2:0] rdAddrB,
   output [15:0] rdDataB);

   reg [15:0]    regfile [0:7];

   assign rdDataA = regfile[rdAddrA];
   assign rdDataB = regfile[rdAddrB];

   always @(posedge clk) begin
      if (reset) begin
         regfile[0] <= 0;
         regfile[1] <= 0;
         regfile[2] <= 0;
         regfile[3] <= 0;
         regfile[4] <= 0;
         regfile[5] <= 0;
         regfile[6] <= 0;
         regfile[7] <= 0;
      end else begin
         if (write) regfile[wrAddr] <= wrData;
      end // else: !if(reset)
   end
endmodule
=====================================


// This is a Verilog description for an 8 x 16 register file

`timescale 1ns / 1ns

module regfile8x16c
  (input clk,
   input reset,
   input write,
   input [2:0] wrAddr,
   input [15:0] wrData,
   input [2:0] rdAddrA,
   output [15:0] rdDataA,
   input [2:0] rdAddrB,
   output [15:0] rdDataB);

   reg [15:0]    regfile [0:7];

   assign rdDataA = regfile[rdAddrA];
   assign rdDataB = regfile[rdAddrB];

   integer       i;
   always @(posedge clk) begin
      if (reset) begin
         for (i = 0; i < 8; i = i + 1) begin
            regfile[i] <= 0;
         end
      end else begin
         if (write) regfile[wrAddr] <= wrData;
      end // else: !if(reset)
   end
endmodule
```