```
/*
The arbiter desgin shown below uses the dataflow level
of abstraction. Of course you could also have done this
at the behavioral level using if-then-else or case statements.

To illustrate the behavior, the eight inputs and eight outputs
are declared individually to illustrate the behavior on the
timing diagram. You did NOT have to do this; you could have
declared the inputs and outputs as bus variables.

The assumption is the request inputs (r0 thru r7) and the
grant outputs (g0 thru g7) are active high signals.

*/

module arbiter(r0, r1, r2, r3, r4, r5, r6, r7, g0, g1, g2, g3,
        g4, g5, g6, g7);

input r0,r1,r2,r3,r4,r5,r6,r7;  // request signals
output g0,g1,g2,g3,g4,g5,g6,g7; // grant signals

// define the grant outputs (active high)

assign g0 = r0;         // highest priority
assign g1 = ~r0 & r1;
assign g2 = ~r0 & ~r1 & r2;
assign g3 = ~r0 & ~r1 & ~r2 & r3;
assign g4 = ~r0 & ~r1 & ~r2 & ~r3 & r4;
assign g5 = ~r0 & ~r1 & ~r2 & ~r3 & ~r4 & r5;
assign g6 = ~r0 & ~r1 & ~r2 & ~r3 & ~r4 & ~r5 & r6;
assign g7 = ~r0 & ~r1 & ~r2 & ~r3 & ~r4 & ~r5 & ~r6 & r7; // lowest priority

endmodule


module top;
reg R0,R1,R2,R3,R4,R5,R6,R7;

wire G0, G1, G2, G3, G4, G5, G6, G7;

// instantiate arbiter

arbiter g2(R0,R1,R2,R3,R4,R5,R6,R7,G0,G1,G2,G3,G4,G5,G6,G7);

// conduct exhaustive test

initial
begin
  R0=0; R1=0; R2=0; R3=0; R4=0; R5=0; R6=0; R7=0; // no request case
  #5 R0=1; R1=1'bx; R2=1'bx; R3=1'bx; R4=1'bx; R5=1'bx;
     R6=1'bx; R7=1'bx;
  #5 R0 = 0; R1=1;
  #5 R1 = 0; R2=1;
  #5 R2 = 0; R3=1;
  #5 R3 = 0; R4=1;
  #5 R4 = 0; R5=1;
  #5 R5 = 0; R6=1;
  #5 R6 = 0; R7=1;
  #5 R7 = 0;        // no request case
  #5 $finish;
  end

endmodule
```