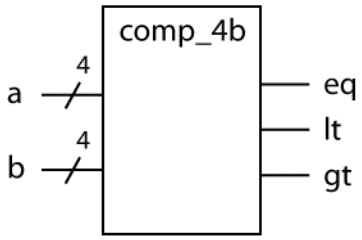
Verilog Testbench CheatSheet


```
module comp_4b(a, b, eq, lt, gt);
    input  [3:0] a,b;
    output reg eq, lt, gt;

    always @ (a,b)
    begin
       if (a == b)
       begin
          eq = 1; lt = 0;  gt = 0;
       end
       else if (a > b)
       begin
          eq = 0; lt = 0;  gt = 1;
       end
       else if (a < b)
       begin
          eq = 0; lt = 1;  gt = 0;
       end
       else
       begin
          eq = 0; lt = 0;  gt = 0;
       end
    end
endmodule
```




```
module tb_comp_4b(   );
   reg [3:0] a, b;       //- stimulus outputs
   wire eq, lt, gt;      //- DUT outputs

   //- DUT instantiation
   comp_4b MY_COMP (
      .a  (a),
      .b  (b),
      .eq (eq),
      .lt (lt),
      .gt (gt)   );

   initial
      begin
         //- initial values of a & b
         a = 'hA;
         b = 'hB;

         //- a & b values 20 time units later
         #20 a = 'hB;
             b = 'hB;

         //- a & b values 20 time units later
         #20 a = 4'b1011;
             b = 4'b0001;

         //- a & b values 20 time units later
         #20 a = 4'b0001;
             b = 4'b0001;
      end

endmodule
```

```
module
cntr_udclr_nb(clk,clr,up,ld,D,count,rco);
    input  clk, clr, up, ld;
    input  [n-1:0] D;
    output   reg [n-1:0] count;
    output   reg rco;

    //- default data-width
    parameter n = 8;

    always @(posedge clr, posedge clk)
    begin
        if (clr == 1)       //- asynch reset
           count <= 0;
        else if (ld == 1)   //- load value
           count <= D;
        else if (up == 1)   //- increment
           count <= count + 1;
        else if (up == 0)   //- decrement
           count <= count - 1;
    end

    //- handles the direction dependent RCO
    always @(count, up)
    begin
       if ( up == 1 && &count == 1'b1)
          rco = 1'b1;
       else if (up == 0 && |count == 1'b0)
          rco <= 1'b1;
       else
          rco <= 1'b0;
    end

endmodule
```

```
module tb_comp_nb(   );
   reg [7:0] D;
   reg clk, clr, up, ld;
   wire [7:0] count;
   wire rco;

   cntr_udclr_nb MY_CNTR (
       .clk   (clk),
       .clr   (clr),
       .up    (up),
       .ld    (ld),
       .D     (D),
       .count (count),
       .rco   (rco)   );

   //- Generate periodic clock signal
   initial
   begin
      clk = 0;   //- init signal
      forever  #10 clk = ~clk;
   end;

   initial
      begin
         clk = 0;
         up = 1;
         ld = 0;
         D =  'hFB;
         clr = 0;

         //- send out LD pulse
         #10 ld = 1;
         #30 ld = 0;

         //- change count direction
         #200 up = 0;

      end

endmodule
```