Glitches

Unintended outputs (glitches or transients) can occur if you are monitoring the output of a counter with comparator. For example, if you want to generate a compare when the output of a binary counter reaches the value 3, it can actually occur before the count of 3 because the transition between the count 1 and 2 has a short time were the one of the count 1 (b01) is delayed just enough that when the count 2 occurs (b10) there is a short time when the output of the counter is 3 (b11). You can simulate this in Modelsim. Here is the Verilog code for the counter, comparator (if statement) and test bench.

```
module count(
input clk,
output reg equal3,
output reg [1:0] counter);

always@(posedge clk) begin
counter <= counter +1;
end

always@* begin
if (counter == 3)
        equal3 = 1;
else
        equal3 = 0;
end
endmodule
```

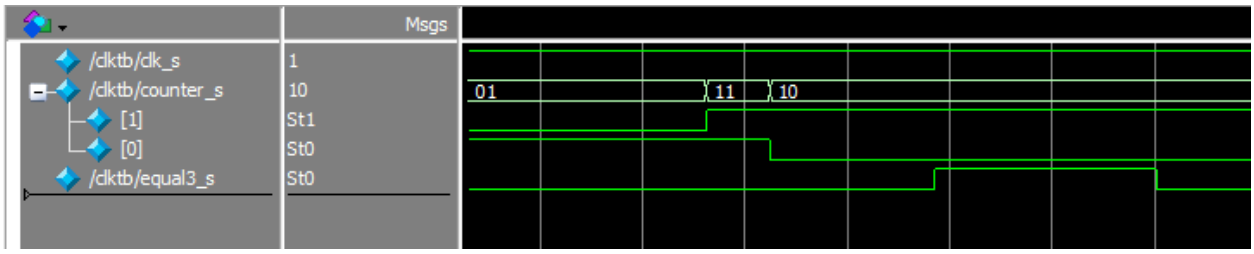
```
`timescale 1 ns/1 ns
module clktb();
reg clk_s;
wire [1:0]counter_s;
wire equal3_s;

count
t1(.clk(clk_s),.counter(counter_s),.eq
ual3(equal3_s));

always begin
clk_s <=0; #10;
clk_s <=1; #10;
end
endmodule
```

The Modesim output shows the equal 3 value going high before a count of 3.

When you zoom into the area around the transition from count 1 to 2 you see there is a short time when they are both high.



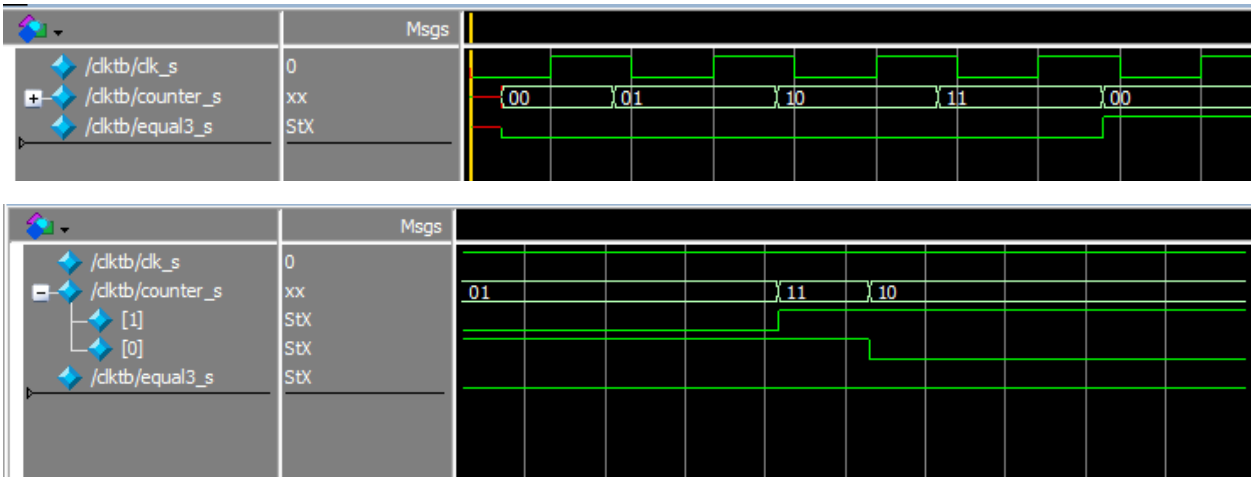
To fix this, you can hold the count value in a register just before the comparator so that it only changes on the rising edge of counter clock instead of continuously as we are doing now avoiding the counter transition from count 1 to count 2. The Verilog will look like this instead.

```
module count(
input clk,
output reg equal3,
output reg [1:0] counter);

always@(posedge clk) begin
counter <= counter +1;
end

always@(posedge clk) begin
if (counter == 3)
        equal3 = 1;
else
        equal3 = 0;
end
endmodule
```

If you run a simulation, you will now see that the comparator does not respond to the counter transist between the counts 1 and 2 even though the count transit still exists.



This can happen for other counts in a much larger counter. Generally speaking you should evaluate the comparator synchronously with your counter to avoid this problem.