Figure 2.19: Published version left out D[0] and D[1] in instantiations

```
module skip(CLK, RST, D, Q);
  input CLK, RST;
  input [1:0] D;
  output wire [1:0] Q;

  DFF R0(CLK, RST, D[0],, Q[0]); //QN is unconnected
  DFF R1(CLK, RST, D[1],, Q[1]);
endmodule

//Behavioral code modeling a D flipflop with
//complimentary outputs
module DFF(CLK, RST, D, QN, Q);
  input CLK, RST, D;
  output reg QN, Q;
  always_ff @(posedge CLK, negedge RST)
    if (!RST) begin
      QN <= 1'b1;
      Q <= 1'b0;
    end
    else begin
      QN <= ~D;
      Q <= D;
    end
endmodule
```

Table 3.1: Verilog net types

Net Type	Function
wire	Normal interconnects between instances
tri	Exactly the same as wire
wand	Wired AND. Models open drain/open collector devices
triand	Exactly the same as wand
wor	Wired OR. Used in now-obsolete emitter-coupled logic
trior	Exactly the same as wor
trireg	Models nets with capacitive storage. Holds last driven value
tri0	Nets that pull down when not driven
tri1	Nets that pull up when not driven

Table 3.12: Bitwise OR Resolution

|	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

Table 3.13: Bitwise XOR Resolution

^	0	1	X	Z
0	0	1	X	X
1	1	0	X	X
X	X	X	X	X
Z	X	X	X	X

Page 61, 2nd to last paragraph, although should be through

Page 83, references to figure 3.22 should be references to figure 3.29

Page 96, second paragraph: left and right are switched. Corrected text is shown below.

The difference between arithmetic shift right and simple shift right is that arithmetic shift extends the sign bit when used with signed operands. Arithmetic shift left works exactly the same as a simple shift left under all conditions.

page 143, second paragraph, a little more than half way down, $clog(WIDTH) is used where $clog2(WIDTH) is intended. This appears twice in the latter half of that paragraph.

Figure 9.13: uses const where it should use parameter