Review Problem 56

- Is the following good Verilog? If not, fix it. If so, draw the circuit it represents.

```verilog
logic a, b, c, d, e;

always_comb begin
    b = 0;
    c = 0;
    if (a)
        b = d;
    else
        c = d | e;
end
```
Review Problem 56

- Is the following good Verilog? If not, fix it. If so, draw the circuit it represents.

```verilog
logic a, b, c, d, e;

always_comb begin
    b = 0;
    c = 0;
    if (a)
        b = d;
    else
        c = d | e;
end
```

Slide 208: Complex “If” Synthesis
Advanced Verilog – "Case" Synthesis

input w;

parameter [1:0] A = 2'b00, B = 2'b01, C = 2'b10;
logic [1:0] ns, ps;

// Next State Logic
always_comb begin
  case (ps)
    A: if (w) ns = B;
        else ns = A;
    B: if (w) ns = C;
        else ns = A;
    C: if (w) ns = C;
        else ns = A;
  endcase
end
input w;

parameter [1:0] A = 2'b00, B = 2'b01, C = 2'b10;
logic [1:0] ns, ps;

// Next State Logic
always_comb begin
    case (ps)
        A: if (w) ns = B;
        else ns = A;
        B: if (w) ns = C;
        else ns = A;
        C: if (w) ns = C;
        else ns = A;
        default: ns = 2'bxx;
    endcase
end
Advanced Verilog – "Posedge" Statements

logic [1:0] ps, ns;
parameter [1:0] Start = 2'b01;

always_ff @ (posedge clk) begin
  if (reset)
    ps <= Start;
  else
    ps <= ns;
end
Advanced Verilog – “Posedge” Statements

logic [1:0] ps, ns;
parameter [1:0] Start = 2'b01;

always_ff @(posedge clk) begin
  if (reset)
    ps <= .Start;
  else
    ps <= ns;
end
```verilog
logic [3:0] out;

always @(posedge clk)
begin
    if (reset)
        out <= 4'b0000;
    else if (incr)
        out <= out + 4'b0001;
end
```
logic [3:0] out;

always_ff @(posedge clk) begin
    if (reset)
        out <= 4'b0000;
    else if (incr)
        out <= out + 4'b0001;
end

don't need
else out <= out.
Advanced Verilog – Histogram

logic [1:0] inVal;
logic [3:0] count00, count01, count10, count11;

always_ff @(posedge clk) begin
    case (inVal)
        2'b00: count00 <= count00 + 4'b0001;
        2'b01: count01 <= count01 + 4'b0001;
        2'b10: count10 <= count10 + 4'b0001;
        2'b11: count11 <= count11 + 4'b0001;
    endcase
end
Advanced Verilog – Histogram

logic [1:0] inVal;
logic [3:0] count00, count01, count10, count11;

always_ff @(posedge clk) begin
    case (inVal)
        2'b00: count00 <= count00 + 4'b0001;
        2'b01: count01 <= count01 + 4'b0001;
        2'b10: count10 <= count10 + 4'b0001;
        2'b11: count11 <= count11 + 4'b0001;
    endcase
end

Real system needs reset...
Advanced Verilog – Less Than

logic [3:0] a, b;

always_comb begin
    lessThan = (a<b);
end
Advanced Verilog – Less Than

```verilog
logic [3:0] a, b;

always_comb begin
    lessThan = (a < b);
end
```

Do subtraction.
Verilog values unsigned by default.
How subtract? **Add a sign bit.**
Overflow – can this overflow? YES. Add 1 more bit.
Or, realize that overflow just flips the sign.
Advanced Verilog – Sequential Statements

input logic [2:0] i0, i1;
output logic [2:0] o0, o1;

always_comb begin
  o0 = i0;
o1 = i1;
  if (o1 < o0) begin
    o1 = i0;
o0 = i1;
  end
end
Advanced Verilog – Sequential Statements

input logic [2:0] i0, i1;
output logic [2:0] o0, o1;

always_comb begin
  o0 = i0;
o1 = i1;
if (o1 < o0) begin
  o1 = i0;
o0 = i1;
end
end

What does it do? o0 is smaller of i0 til 1
o1 is larger
Advanced Verilog – For Loops

output logic [1:0] result;
input logic [1:0] vals [3:0];
integer i;

always_comb begin
    result = vals[0];
    for (i=1; i<4; i++)
        if (vals[i] != 2'b00)
            result = vals[i];
end
Advanced Verilog – For Loops

output logic [1:0] result;
input logic [1:0] vals [3:0];
integer i;

always_comb begin
    result = vals[0];
    for(i=1; i<4; i++)
        if (vals[i] != 2'bc0)
            result = vals[i];
end

Discuss delay of design – could convert to Parallel prefix.