Review Problem 19

- Extend this Verilog code to also show the letter "A" on input pattern 1010 (ten) and "F" on pattern 1111 (fifteen).

```verilog
module seg7 (bcd, leds);
    input logic [3:0] bcd;
    output logic [6:0] leds;

    always_comb begin
        case (bcd)
            4'b0000: leds = 7'b0111111;
            4'b0001: leds = 7'b0000110;
            4'b0010: leds = 7'b1011011;
            4'b0011: leds = 7'b1001111;
            4'b0100: leds = 7'b1100110;
            4'b0101: leds = 7'b1101111;
            4'b0110: leds = 7'b1111101;
            4'b0111: leds = 7'b0000111;
            4'b1000: leds = 7'b1111111;
            4'b1001: leds = 7'b1101111;
            default: leds = 7'bX;
        endcase
    end
endmodule
```
Debugging Complex Circuits

- Complex circuits require careful debugging
  - Rip up and retry?
- Ex. Debug a 9-input odd parity circuit
  - True if an odd number of inputs are true
Debugging Complex Circuits (cont.)

A
B 3-Parity Out
C

A
B
C
Out

Wrong
Wrong
Debugging Approach

- Test all behaviors.
  - All combinations of inputs for small circuits, subcircuits.

- Identify any incorrect behaviors.

- Examine inputs and outputs to find earliest place where value is wrong.
  - Typically, trace backwards from bad outputs, forward from inputs.
  - Look at values at intermediate points in circuit.

- **DO NOT RIP UP, DEBUG!**
Combinational vs. Sequential Logic

Network implemented from logic gates. The presence of feedback distinguishes between sequential and combinational networks.

Sequential = combinational + feedback

Combinational logic
no feedback among inputs and outputs
outputs are a pure function of the inputs
e.g., seat belt light:
  (Dbelt, Pbelt, Passenger) mapped into (Light)

Dbelt → Logic Circuit → Seat Belt Light
Pbelt
Passenger
Hazards/Glitches

- Circuit can temporarily go to incorrect states

![Circuit Diagram]

- Must filter out temporary states
Safe Sequential Circuits

- Clocked elements on feedback, perhaps outputs
  - Clock signal synchronizes operation
  - Clocked elements hide glitches/hazards
Basic D Flip Flop

// Basic D flip-flop

module basic_D_FF (q, d, clk);
    output logic q;
    input logic d, clk;

    always_ff @(posedge clk) begin
        q <= d; // ALWAYS use <= to assign to clocked elements
    end
endmodule