Review Problem 20

Amy, Bill, Carol, and Dennis each decide independently whether they want to play Chess or Checkers, each a 2-player game. Develop a circuit that can tell if the 4 people can be organized into two simultaneous games, respecting each person’s choice.
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
```

```plaintext
A
B
C
```

Out
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

- Readings: 5-5.4.4

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

**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

Copilot Autopilot Request

Pilot in Charge?

Autopilot Engaged

Pilot Autopilot Request

<table>
<thead>
<tr>
<th>CAR</th>
<th>PIC</th>
<th>PAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>AE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Must filter out temporary states
Safe Sequential Circuits

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

![Diagram of sequential circuits with clocks and logic network]

Clock

Clock
Data
Compute Valid Compute Valid Compute

Longest combination path
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