Each person's choice can be organized into two simultaneous games, respecting player game. Develop a circuit that can tell if the 4 people whether they want to play Chess or Checkers, each a 2-

Amy, Bill, Carol, and Dennis each decide independently.  

Review Problem 20
module D_FF (q, d, reset, ctk);

input logic d, reset, ctk;
output logic q;

always @(posedge ctk) begin
    assign q = d; // on reset, set to 0
end

endmodule

// D Flip Flop w/Synchronous Reset
module

endmodule

end

$stop();

@ (posedge clk)
@ (negedge clk)

reset <= 0;
reset <= 1;
reset <= 0;
reset <= 1;

d <= 0;
d <= 1;
d <= 0;
d <= 1;

transition begin // Set up the reset signal

transition begin // Set up the clock

forever @ (clockdelay/2) clk <= ~clk;

do d' => 0;
d' => 1;
d' => 0;
d' => 1;

parameter clockdelay = 100;

module stimulus;

Verilog Testbench
Finite State Machines

Sequential Logic needs more complex design steps

History will be held in flip-flops

Traffic light controller, Sequence Lock...

Need to implement circuits that remember history

Implementatin of combinational logic as controller

State Table to specify functions (like Truth Table)

State Diagram to describe behavior

Readings: 6-6.4.7
Even: State = 0, Odd: State = 1

Input = P5 ⊕ I

Output = P5

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Example: Odd Parity Checker

Finite State Machine Example
Finite State Machine Example (cont.)

NS = Ps XOR Input, OUT = Ps

85