Review Problem: Basic Verilog

- Write the Verilog for a 2-input gate that is TRUE when an odd number of inputs are true.

```verilog
module xor1(x, A, B)
    output x;
    input A, B;
    assign x = (A & ~B) | (~A & B);
endmodule
```

// VERILOG FOR A $\oplus$ B = \overline{AB} + A\overline{B}
Review Problem: flip-flops

- The following two flip-flops are subtly different, but both useful. The difference in code is shown in bold. What is the difference in their behavior?

```
module D_FF1 (q, d, reset, clk);
    output q;
    input d, reset, clk;
    reg q;

    always @(posedge clk)
    if (reset)
        q <= 0;
    else
        q <= d;
endmodule

module D_FF2 (q, d, reset, clk);
    output q;
    input d, reset, clk;
    reg q;

    always @(posedge clk or posedge reset)
    if (reset)
        q <= 0;
    else
        q <= d;
endmodule
```

SYNCHRONOUS RESET
ONLY RESETS ON CLOCK EDGE

ASYNCHRONOUS RESET
WILL RESET AT ANY TIME
Case Study: Seven Segment Display

- Chip to drive digital display

<table>
<thead>
<tr>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Case Study (cont.)

<table>
<thead>
<tr>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>Val</th>
<th>L0</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>L5</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Case Study (cont.)

■ Implement L5:

<table>
<thead>
<tr>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>L5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

L5 = B3 + B1B0 + B2B0 + B2B1

(E.G. FROM K-MAP)
Verilog RTL: just describe what you want

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

    always @(*)
        case (bcd)
            // 3210            6543210
            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'b1101101;
            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
    endmodule
Review Problem

- 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 [3:0] bcd;
  output reg [6:0] leds;

  always @(*)
  case (bcd)
    // BCD[]          LEDS[]
    // 3210           6543210
    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'b1101101;
    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
endmodule
```
NEVER put a logic gate between the clock and DFF’s CLK input.
Flipflop Realities 2: Clock Period, Applying Stimulus

- Clock Period?
- Apply Inputs when?
\( T_{\text{setup}}, T_{\text{hold}}, \text{Clk} \rightarrow \text{Q} \)

- Flipflops require their inputs be stable for time period around clock edge

\[ \text{D} \quad \begin{array}{c|c|c} \text{Can Change} & \text{Stable} & \text{Can Change} \\ \hline \end{array} \]

\[ \text{Clock} \quad \begin{array}{c} T_{\text{setup}} \end{array} \quad \begin{array}{c} T_{\text{hold}} \end{array} \quad \text{Clk} \rightarrow \text{Q} \]

\[ \text{Q} \quad \begin{array}{c} \text{Old Value} \end{array} \quad \begin{array}{c} \text{New Value} \end{array} \]
Timing Definitions

- \( T_{\text{setup}} \): Time \( D \) must be stable BEFORE clock edge
  - Adds to critical path delay

- \( \text{Clk-} \rightarrow \text{Q} \): Time from clock edge to \( Q \) changing
  - Adds to critical path delay

- \( T_{\text{hold}} \): Time \( D \) must be stable AFTER clock edge
  - Sets minimum path from \( Q \) of one DFF to \( D \) of another
Flipflop Realities 3: External Inputs

- External inputs aren’t synchronized to the clock

D \_\_\_

Clk \_\_\_

Q \_\_\_

LOGIC 0

LOGIC 1

META-INSTABLE
Dealing with Metastability

■ Single DFF

External → D flipflop

D                   Q

Clk

METASTABLE
RANDOM VALUE

■ 2 DFFs in series

External → D flipflop → D flipflop

D                   Q

D                   Q

Clk                 Clk

PURE VALUE
0 OR 1

■ 2 DFFs in parallel

External → D flipflop

D                   Q

Clk

D                   Q

Clk

NOT GOOD!
OUTPUT COULD TAKE
DIFFERENT VALUES