

CSE369: Intro to Digital Design

L3: Karnaugh Maps, Simulation

Instructor: ~~Naomi Alterman~~

Isabel Froelich

Teaching Assistants:

Isabel Froelich
[Head TA]

Packard Stephenson

Mark Garcia

Ewan Kae

Violet Monserate

Avesta Rahimi

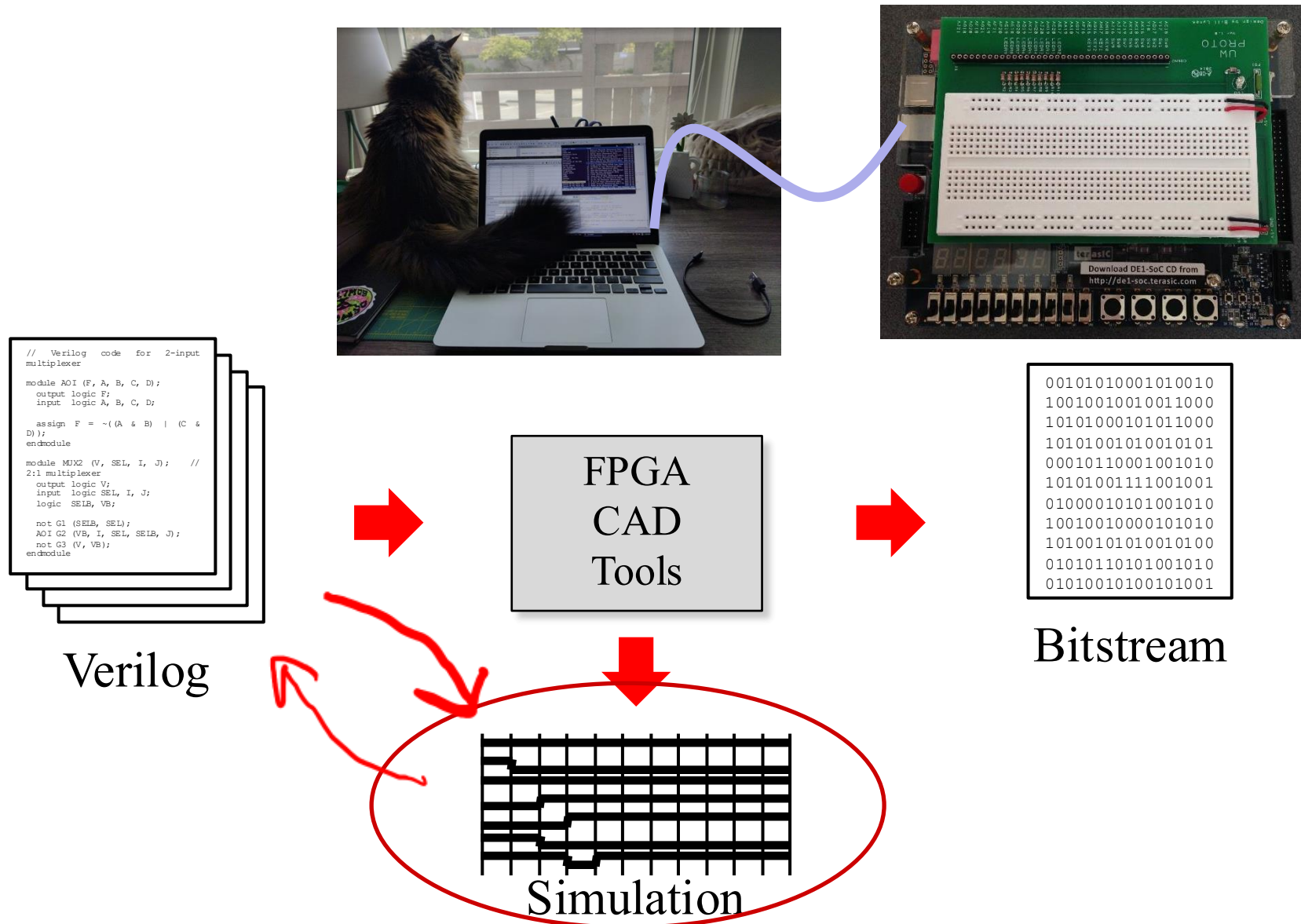
Administrivia

- Lab 3 out – Logic simplification with K-maps and Verilog
 - Due a week from tomorrow
 - Full credit for minimal logic

Today

- **Debugging, Simulations and Waveform Diagrams** → Hazard
- Karnaugh Maps (K-maps)
- Avoiding Hazards
- Design Examples

Using an FPGA



Testbenches

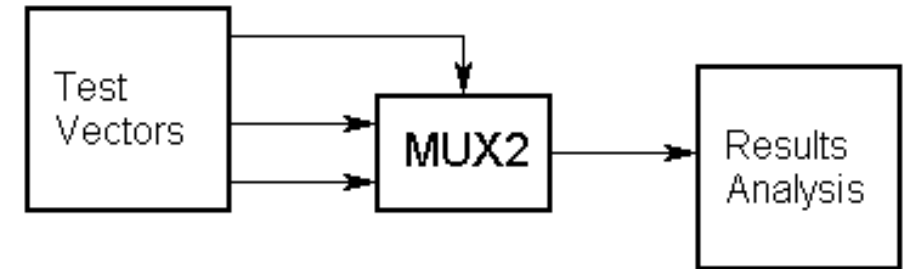
- **ModelSim** and **Icarus Verilog** are digital logic simulators
 - Run entirely on your computer and simulates the operation of your FPGA
 - Used for debugging the internals of buggy Verilog
 - No such thing as `printf()` in a physical circuit (at least not without microscopic tweezers)
- **Testbench** – a Verilog module that instantiates the circuit you're testing *and* provides a lil script to generate fake input signals
 - We need to mockup fake signals for *every* input to our module, *and* the timing of how they change.
 - Doesn't/can't get synthesized
 - **You need a testbench for every single module you write. Period.**

Verilog Testbenches

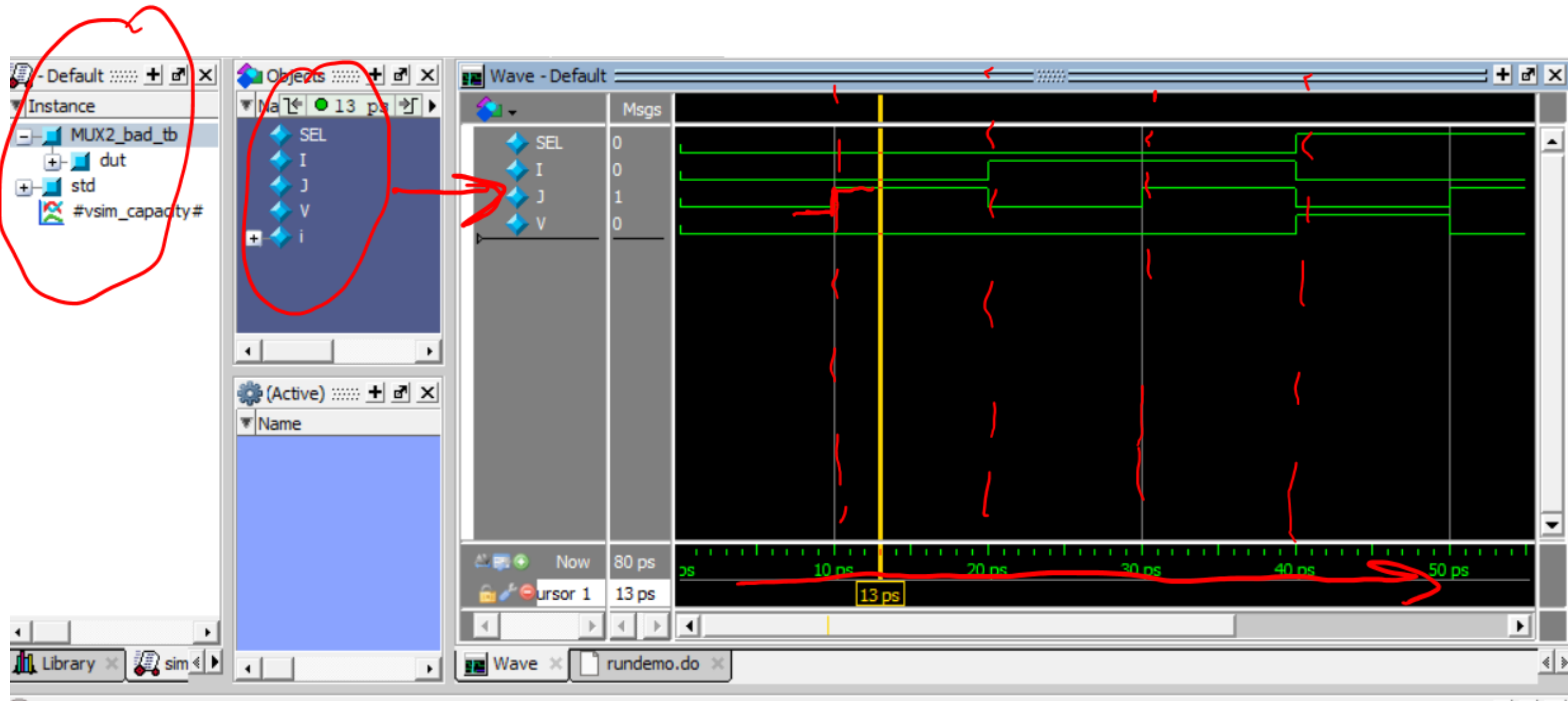
No ports (arrow pointing to the empty parentheses in the module declaration)

```
module MUX2_tb ();  
    logic SEL, I, J; // mocked inputs  
    logic V;         // observed output  
  
    initial begin  
        SEL = 1; I = 0; J = 0; #10; // t=0: S=1, I=0, J=0 -> V=0  
        I = 1; #10; // t=10: S=1, I=1, J=0 -> V=1  
        SEL = 0; #10; // t=20: S=0, I=1, J=0 -> V=0  
        J = 1; #10; // t=30: S=0, I=1, J=1 -> V=1  
        $finish; // end simulation  
    end  
  
    MUX2 dut (.V, .SEL, .I, .J);  
  
endmodule // MUX2_tb
```

(A red arrow points from the 'initial begin' block to the 'end' keyword.)



← **"Device Under Test"** (arrow pointing to the `MUX2 dut` instantiation)

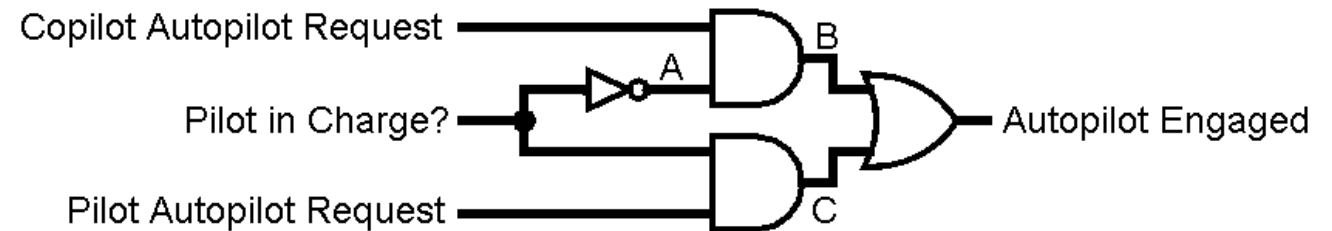


Circuit Timing Behavior



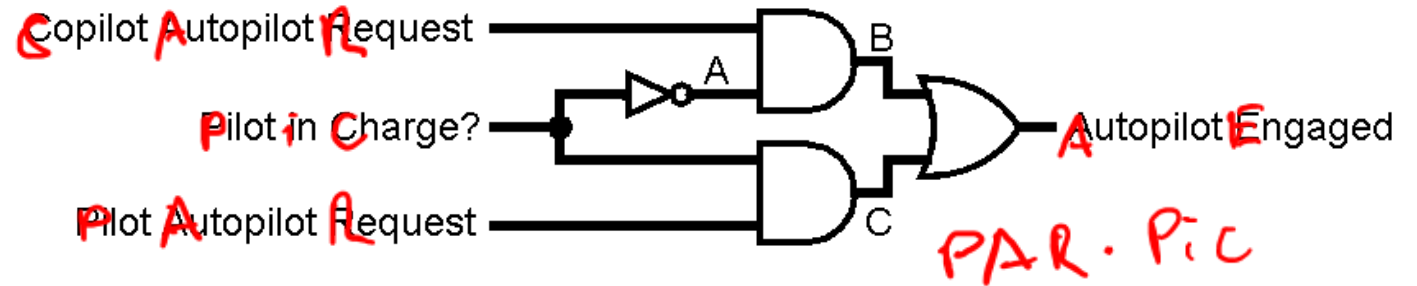
Who is in charge?

- Pilot?
- Copilot?
- Autopilot?

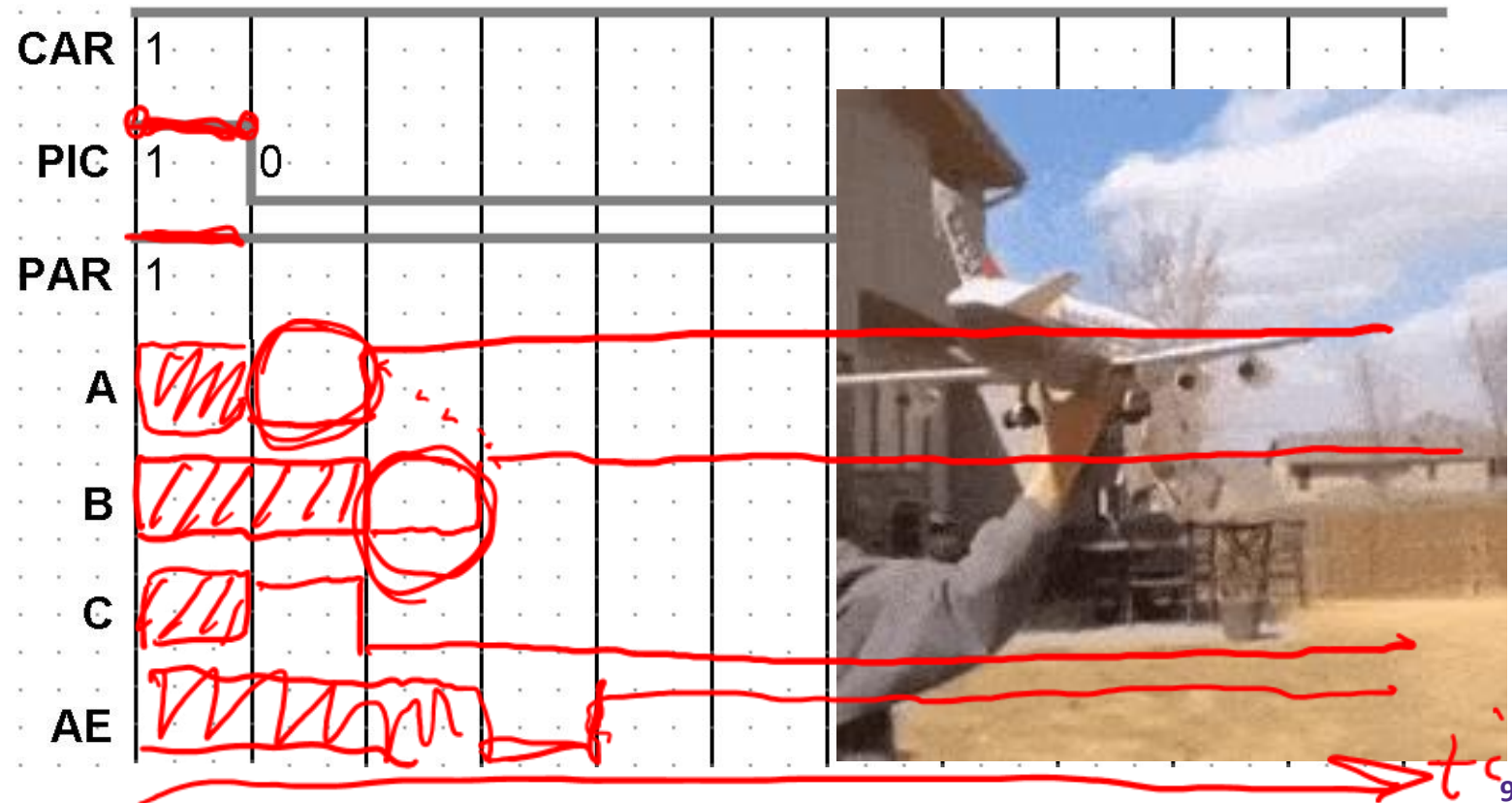


Circuit Timing Behavior

- Assume 1 ns (3 tick) delay for all gates



Hazards/Glitches:
Circuits can temporarily go to incorrect states!



Lecture Outline

- Debugging, Simulations and Waveform Diagrams
- **Karnaugh Maps (K-maps)**
- Avoiding Hazards
- Design Examples

On and Off Sets

- *On Set* is the set of input patterns where the function is TRUE

□ Here on set = $\{\bar{A}\bar{B}C, \bar{A}BC, A\bar{B}\bar{C}, A\bar{B}C\}$

- *Off Set* is the set of input patterns where the function is FALSE

□ Here off set = $\{\bar{A}\bar{B}\bar{C}, \bar{A}B\bar{C}, A\bar{B}C, ABC\}$

- **Recall:** Use the On Set for *Sum of Products* (SoP) and the Off Set for *Product of Sums* (PoS)

□ Considered **two-level** Boolean expressions



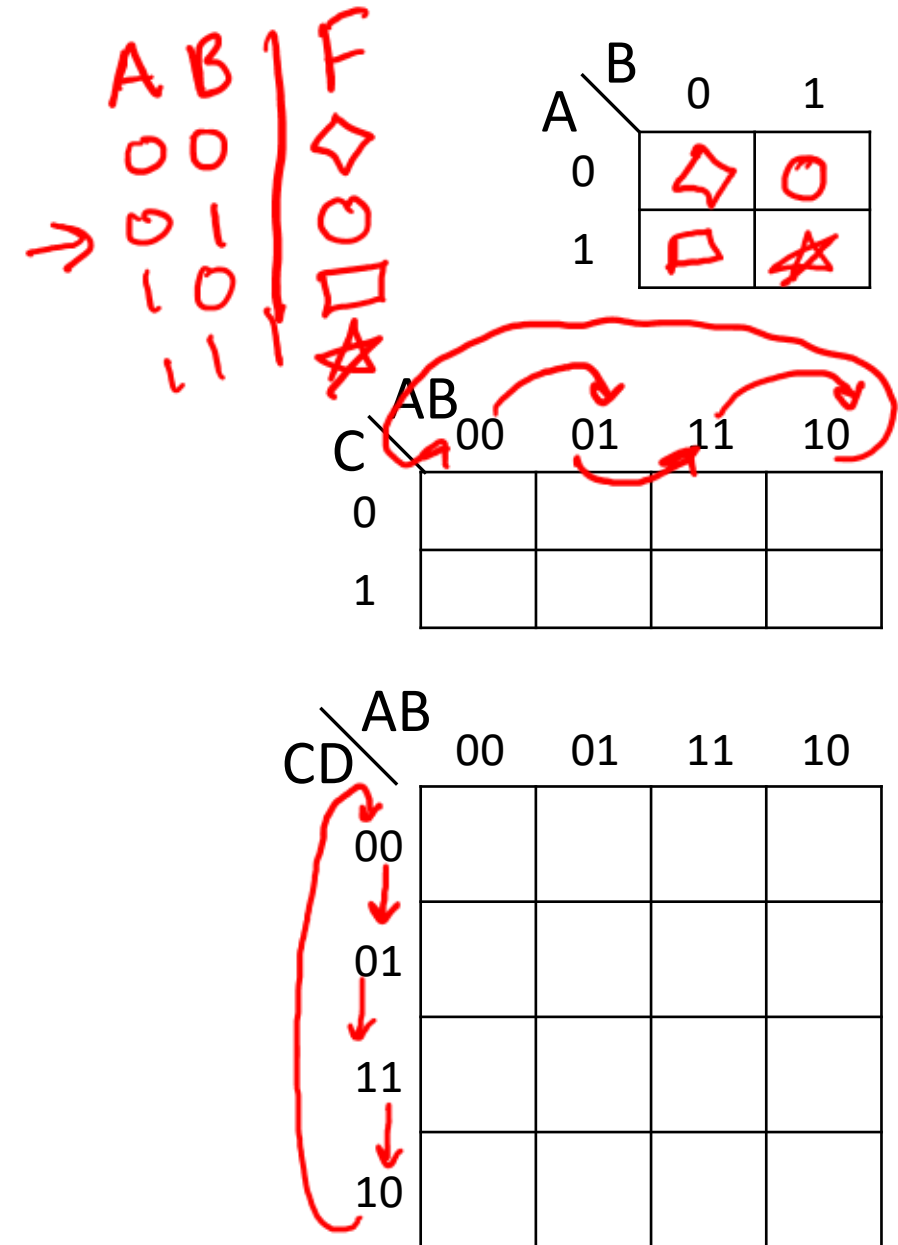
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Two-Level Simplification

- Using Sum of Products, “neighboring” input combinations simplify
 - “Neighboring”: inputs that differ by a single signal
 - *The Uniting Theorem*: $A(\bar{B} + B) = A$ ←
 - e.g., $\underline{AB} + \underline{\bar{A}B} = \underline{B}$ $\bar{A}\underline{BC} + \bar{A}\underline{B}\bar{C} = \bar{A}\underline{B}$
- **Goal:** Find neighboring subsets of the On Set to eliminate variables and simplify the expression
- **Technique:** A **Karnaugh map** (“K-map”) is a method of rearranging a truth table to *visualize* the terms that can be simplified with the uniting theorem

Karnaugh Maps

- Rearrange truth table into a 2D grid with inputs on the edges and outputs in the grid cells
- If more than 2 inputs, group them along the edges and write out all combinations
neighboring cells change only by 1 input ("**Gray code**")

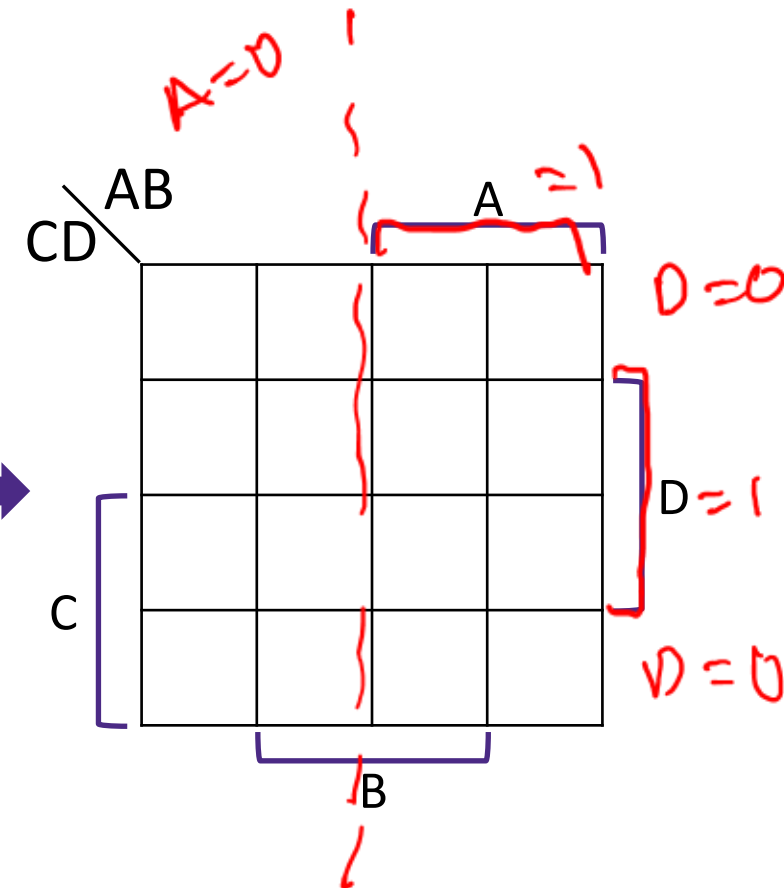


Karnaugh Maps

- Can also draw it using brackets instead of bits. A bracket means the bit is “asserted”/1 in those columns or rows

4 Inputs:

CD \ AB	00	01	11	10
00				
01				
11				
10				



K-map Example: Majority Circuit

- Filling in a Karnaugh map:

binary (handwritten arrow pointing down)

	A	B	C	F
<i>row</i> (handwritten)	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	1
	1	0	0	0
	1	0	1	1
	1	1	0	1
	1	1	1	1

grey (handwritten arrow pointing right)

AB	00	01	11	10
C				
0	0	0	1	0
1	0	1	1	1

- Each row of truth table corresponds to ONE cell of Karnaugh map
- Note the jump when you go from input 011 to 100
(*most mistakes made here*)

K-map Example: Majority Circuit

- Filling in alternate Karnaugh map:

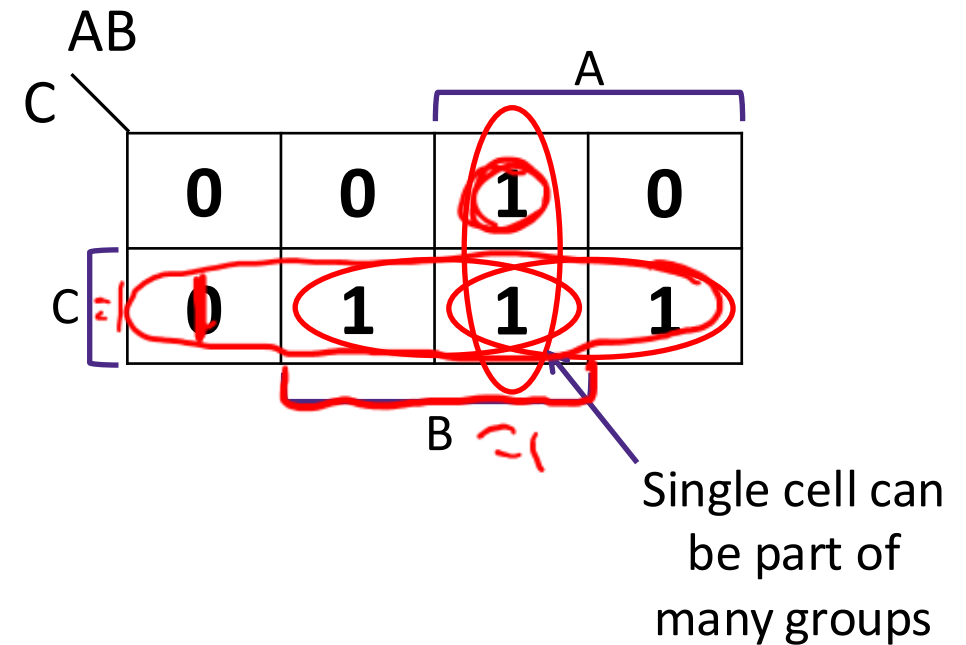
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

		BC			
A		00	01	11	10
0		0			0
1		0			

- Each row of truth table corresponds to ONE cell of Karnaugh map
- Note the jump when you go from input 001 to 010 and 101 to 110 (*most mistakes made here*)

K-map Simplification

- Group neighboring 1's with circles
 - Each circle becomes an equation term
 - Make sure **all** 1's are circled
- $F = BC + AB + AC$
- Larger groups become smaller terms
 - The single 1 in top row $\rightarrow AB\bar{C}$
 - Vertical group of two 1's $\rightarrow AB$
 - If entire lower row was 1's $\rightarrow C$



General K-map Rules

- Only group in powers of 2
 - Grouping should be of size $2^i \times 2^j$
 - Applies for both directions
- Wraps around in all directions
 - “Corners” case is extreme example
- Always choose largest groupings possible
 - Avoid single cells whenever possible
- $F = BD + \overline{B}\overline{D} + ACD$

$$\overline{A}BCD + A\overline{B}CD + A\overline{B}CD + \overline{B}CD + \overline{A}BCD$$

		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	0	1	1	0
	11	0	1	1	1
	10	1	0	0	1

- 1) NOT a valid group
- 2) IS a valid group
- 3) IS a valid group
- 4) “Corners” case
- 5) 1 of 2 good choices here

Don't Cares

- Use symbol 'X' to mean it can be either a 0 or 1
- Make choice to simplify final expression

CD \ AB	A			
	0	0	X	0
C	1	1	X	1
	1	1	0	0
	0	X	0	0
B				D

Let all X = 0:

$$F = \bar{A}D + \bar{B}\bar{C}D$$

CD \ AB	A			
	0	0	X	0
C	1	1	X	1
	1	1	0	0
	0	X	0	0
B				D

Let all X = 1:

$$F = \bar{A}D + \bar{C}D + AB\bar{C} + \bar{A}BC$$

CD \ AB	A			
	0	0	X	0
C	1	1	X	1
	1	1	0	0
	0	X	0	0
B				D

Choose wisely:

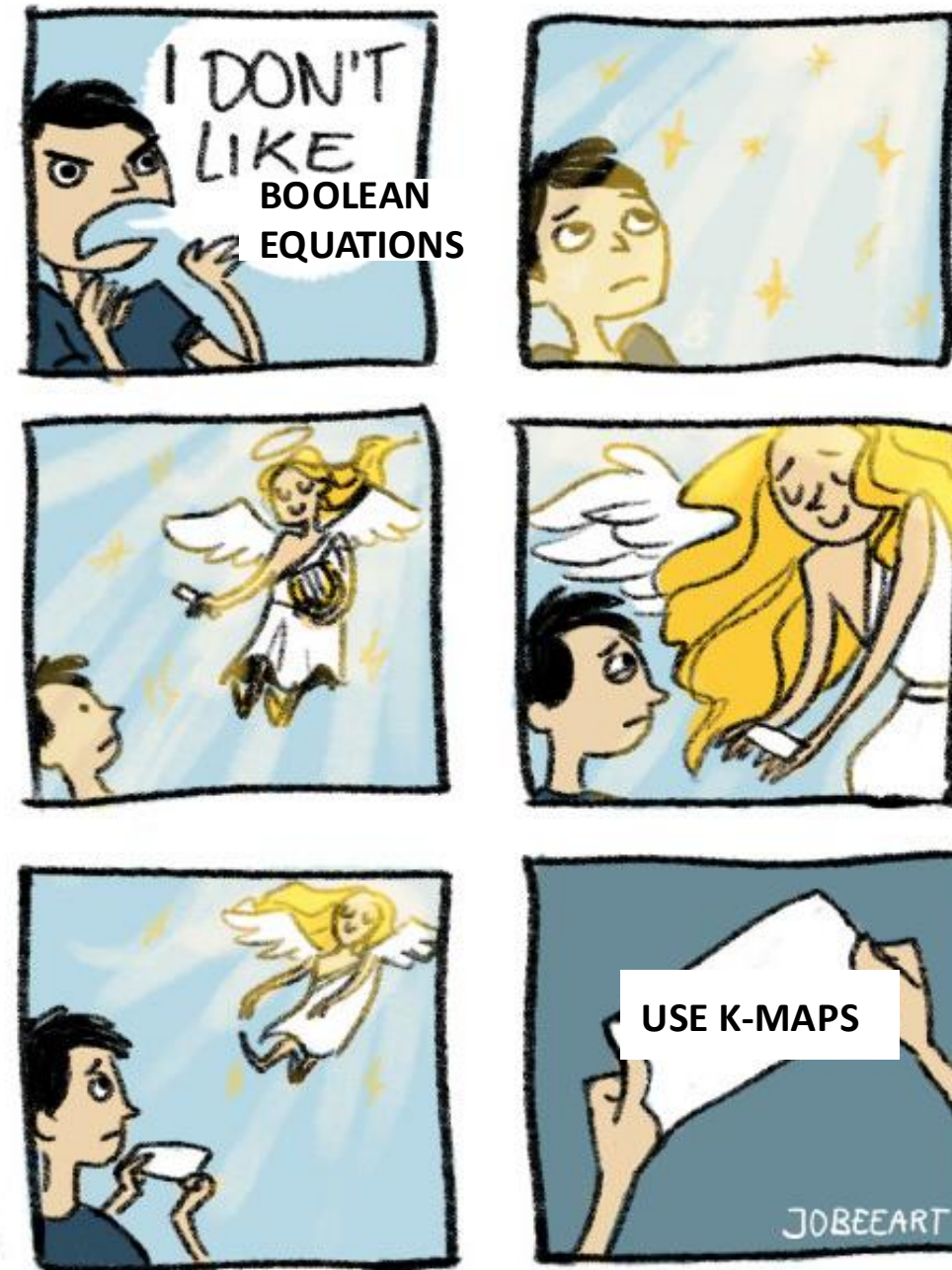
$$F = \bar{A}D + \bar{C}D$$

Two-Level Simplification (at SCALE!)

- As an aside: K-Maps are for ≤ 4 dimensions
- For more, we use a computer implementation of the same technique called the “Quine-McCluskey algorithm”
 - This is part of what Quartus is doing when you hit the “synthesize” button

Ed





Miso Moment



12:22

Lecture Outline

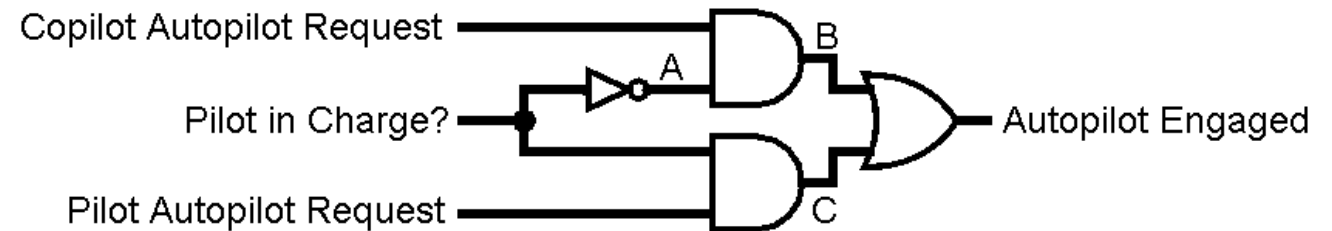
- Debugging, Simulations and Waveform Diagrams
- Karnaugh Maps (K-maps)
- **Avoiding Hazards**
- Design Examples

Timing Diagrams Matter



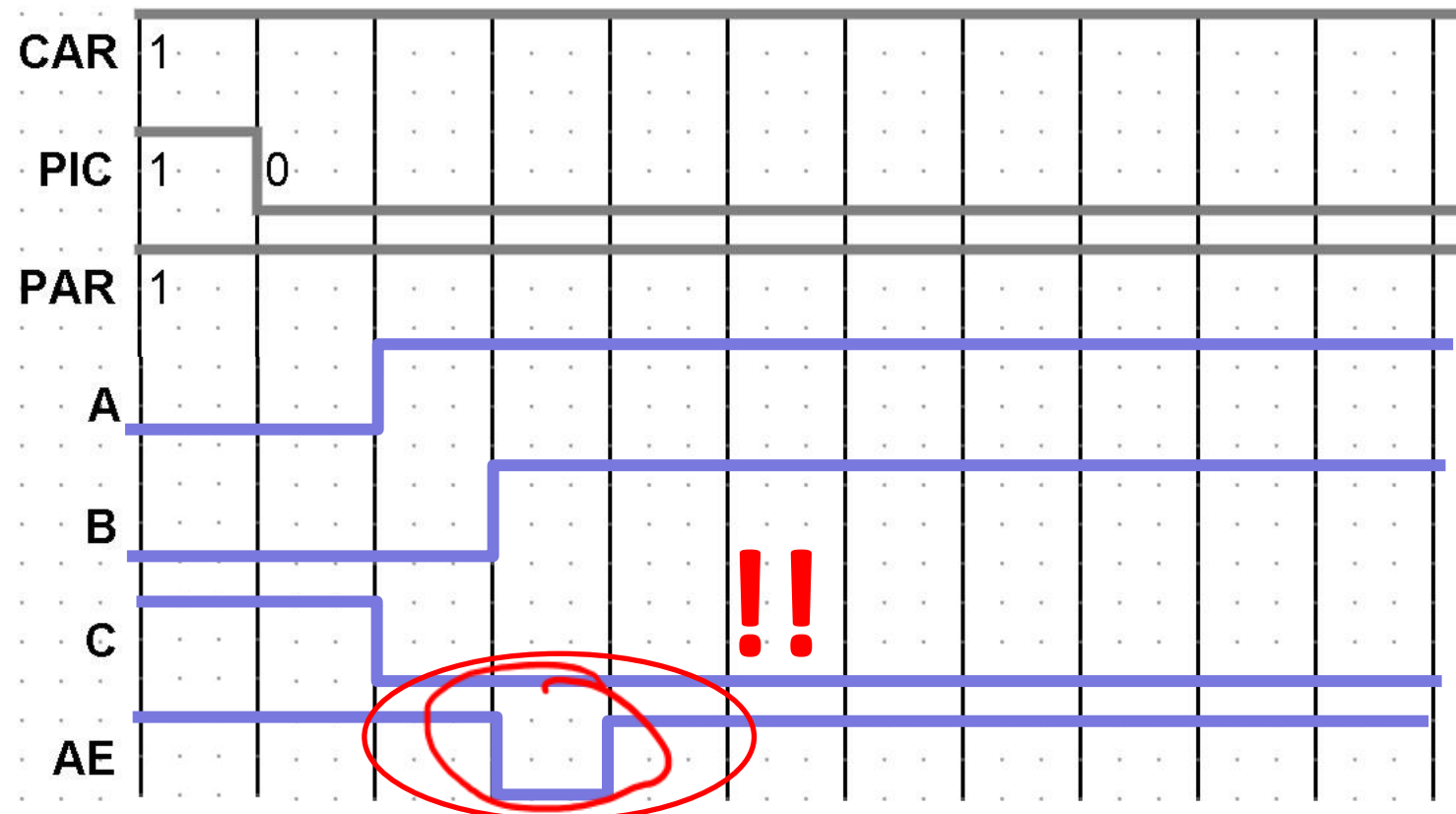
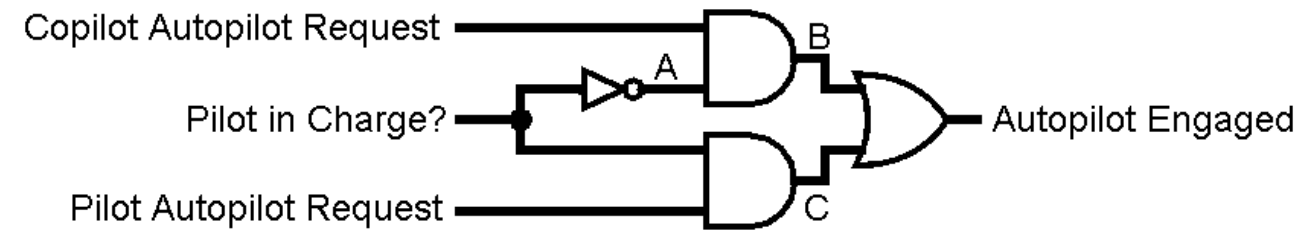
Who is in charge?

- Pilot?
- Copilot?
- Autopilot?



Circuit Timing: Hazards/Glitches

- Circuits can temporarily go to incorrect states!
- Assuming gate delay of 1 ns / 3 ticks



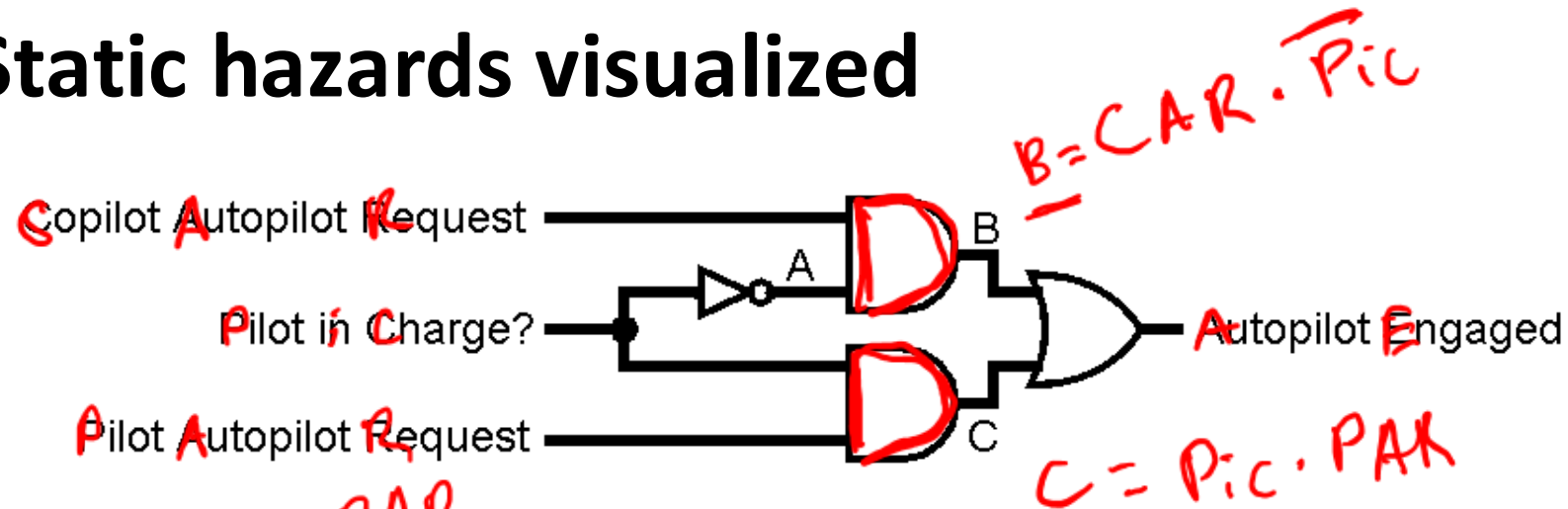
Circuit Hazards

- Momentary wiggles in output of combinational logic before it “settles” on the right answer
 - Mostly doesn’t matter (we’ll see why next week)
 - But when it does matter, it can make the plane crash 🐼
 - Can be avoided with **redundant** gates (eg, not fully simplified equations)
- Classification:
 - “Static-0” (output spikes to 1 before settling to 0)
 - Only a problem for PoS equations
 - “Static-1” (output dips to 0 before settling to 1)
 - Only a problem for SoP equations
 - “Dynamic” (lots of wiggles)

369 / 371



Static hazards visualized



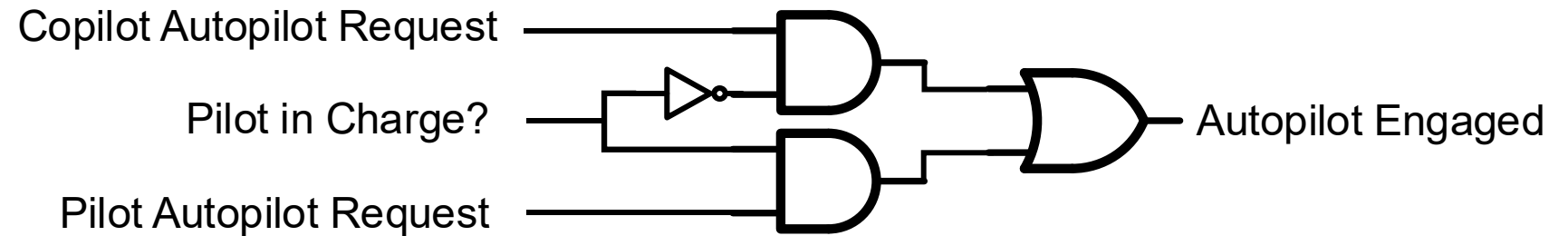
Handwritten red annotations: CAR, PAR and P_{ic}

	00	01	11	10
0	0	0	1	1
1	0	1	1	0

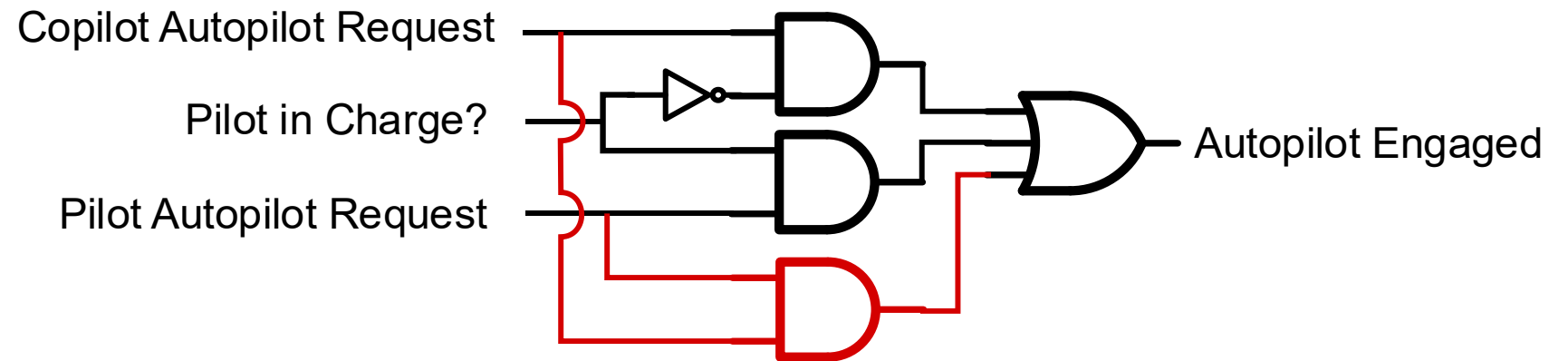
Handwritten red annotations: C and B

Design Tradeoffs

**Minimal
logic**



**No static
hazards**



Lecture Outline

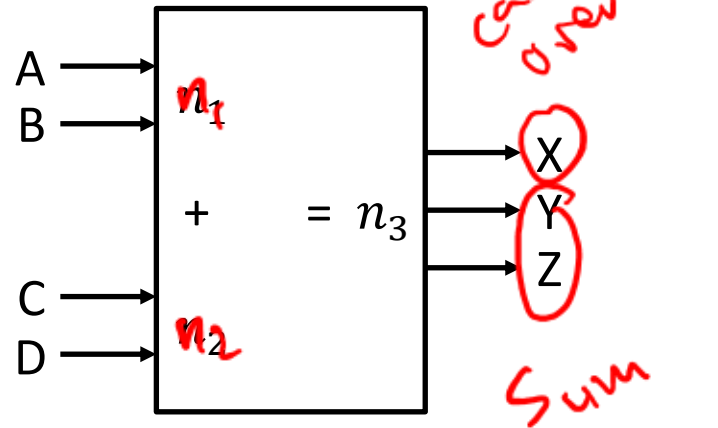
- Debugging, Simulations and Waveform Diagrams
- Karnaugh Maps (K-maps)
- Avoiding Hazards
- **Design Examples**

Design Example: 2-bit Adder

- Block Diagram and Truth Table:

Handwritten binary addition:

$$\begin{array}{r} 11 \\ + 11 \\ \hline 110 \end{array}$$



We'll need a new k-map for **each** output

Handwritten note: input

A	B	C	D	X	Y	Z
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Design Example: 2-bit Adder

Left
→

K-map for X

	AB	A	
CD			
	0	0	0
	0	0	1
C	0	1	1
	0	0	1
		B	
			D

K-map for Y

	AB	A	
CD			
	0	0	1
	0	1	0
C	1	0	1
	1	1	0
		B	
			D

K-map for Z

	AB	A	
CD			
	0	1	1
	1	0	0
C	1	0	0
	0	1	1
		B	
			D

right
↙

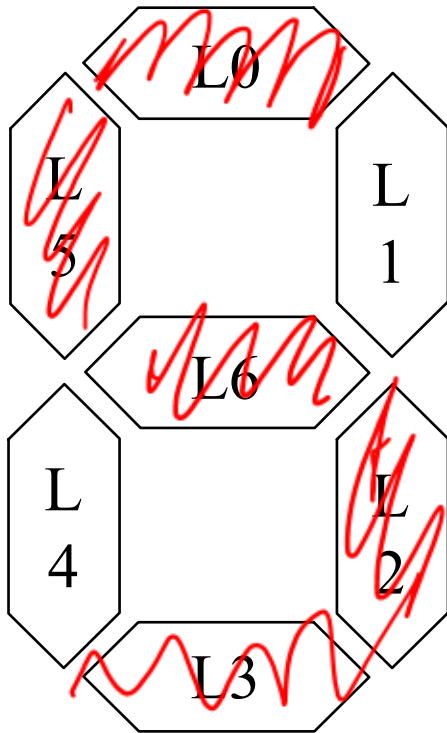
$$X = AC + ABD + BCD$$

$$Y =$$

$$Z = \bar{B}D + B\bar{D} = D \text{ XOR } B$$

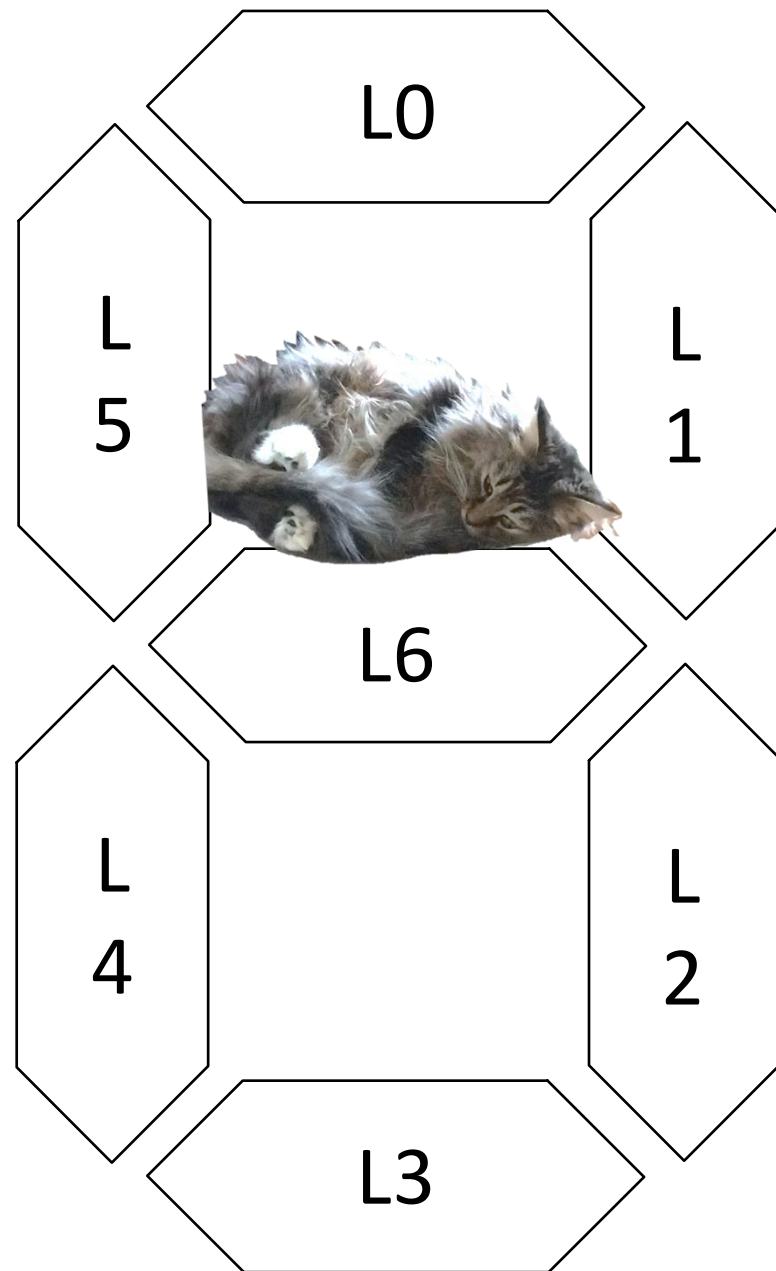
Case Study: Seven-Segment Display

- Chip to drive digital display

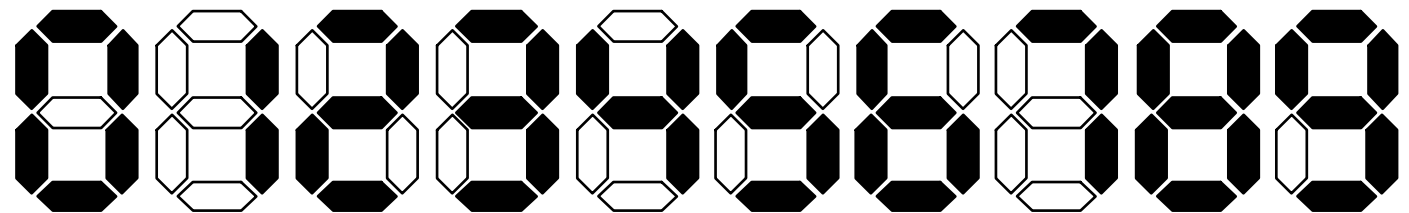


0

B3	B2	B1	B0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1



B3	B2	B1	B0	Val	L0	L1	L2	L3	L4	L5	L6
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1

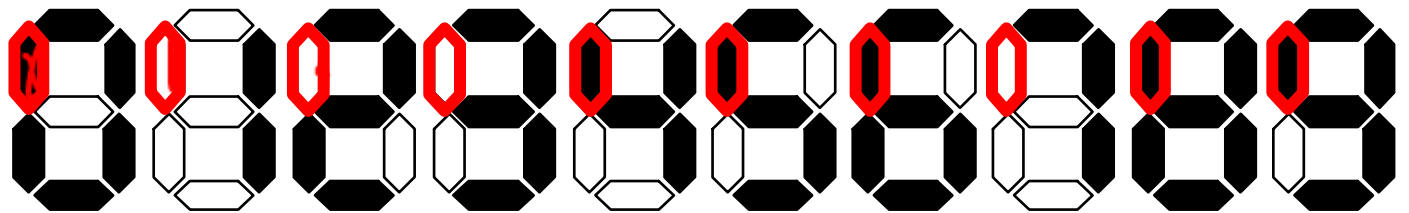
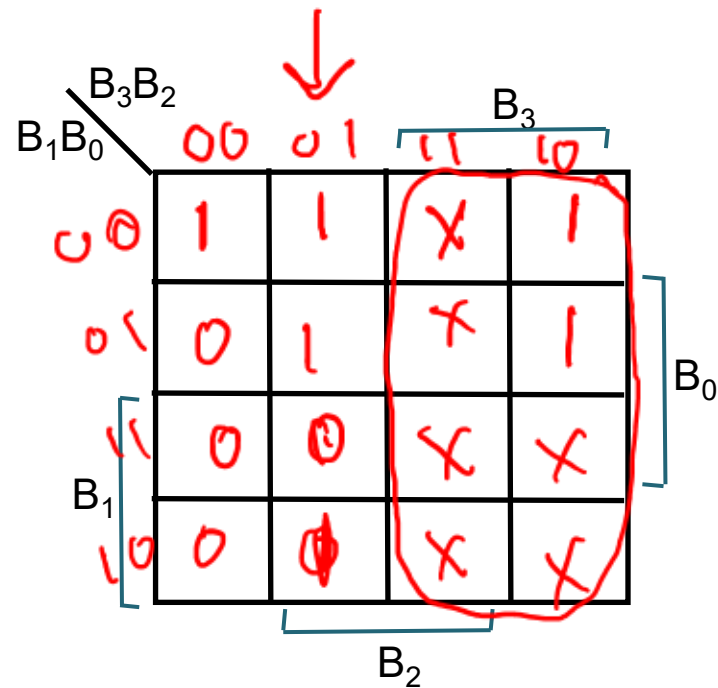


Case Study: Seven-Segment Display

□ Implement **L5**:

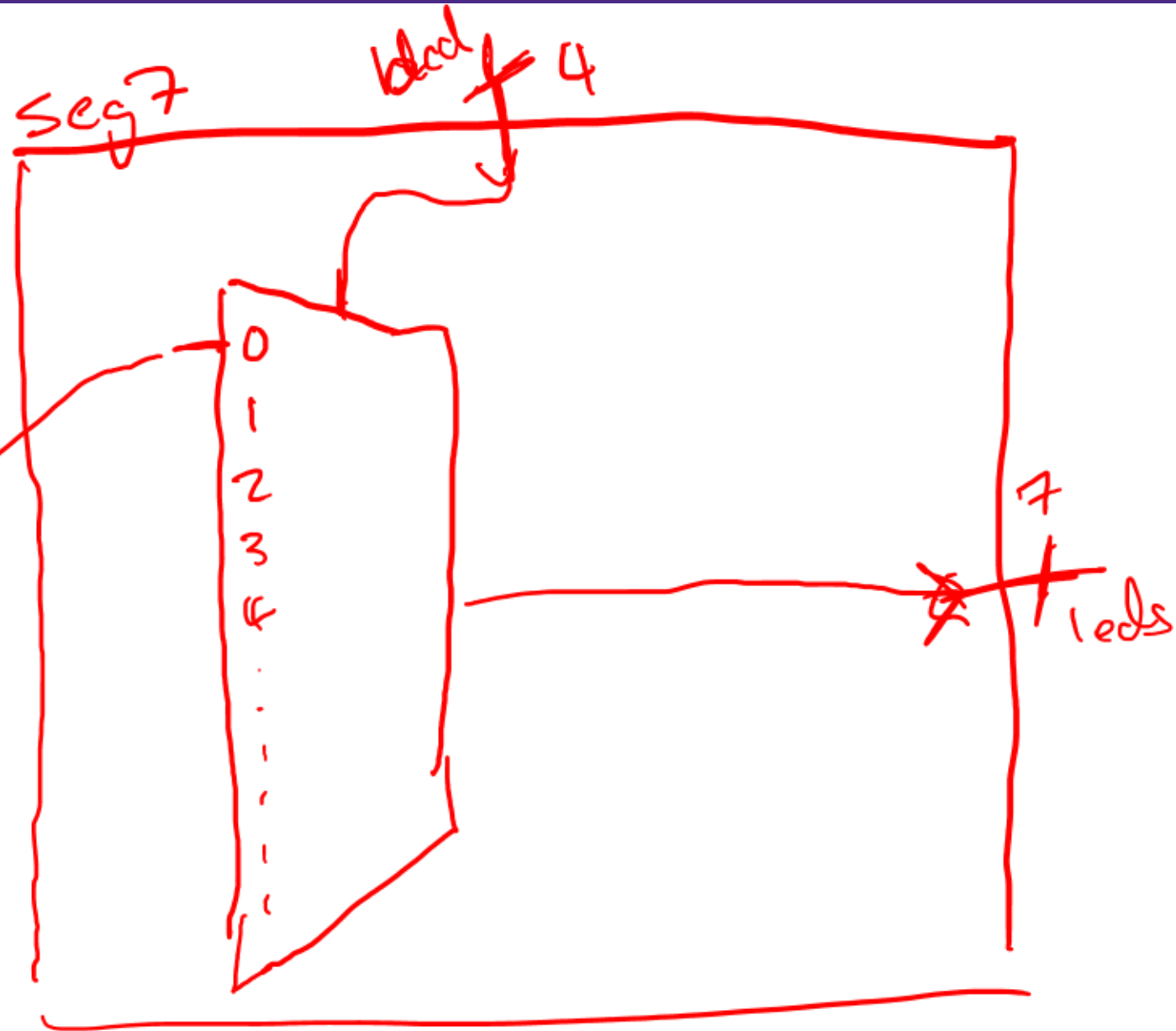
columns, row

B3	B2	B1	B0	L5
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1



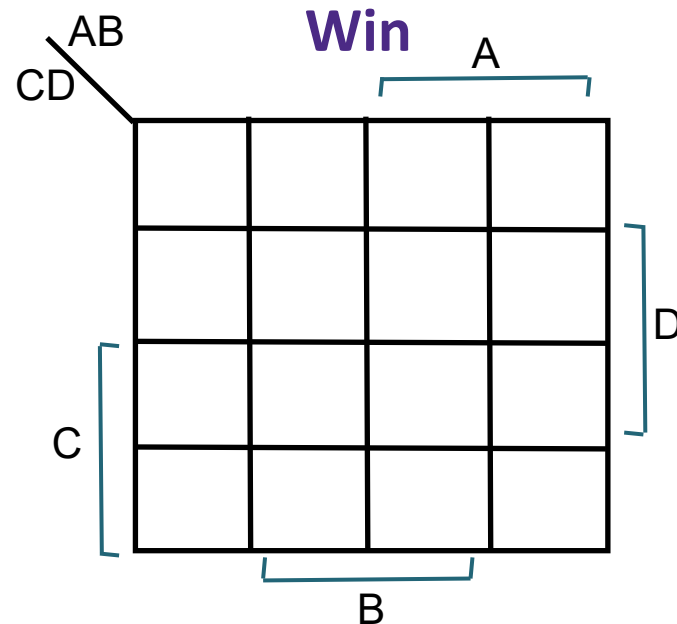
7-Seg Display in Verilog

```
module seg7 (bcd, leds);  
    input logic [3:0] bcd;  
    output logic [6:0] leds;  
  
    always_comb  
        case (bcd)  
            // 3210  
            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
```



Practice Problem: Rock-Paper-Scissors

- Rock (00), Paper (01), Scissors (10)
for two players P0 and P1
- Output:** Win = Winner's ID (0/1)
Tie = 1 if Tie, 0 else



P1		P0		Win	Tie
A	B	C	D		
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		