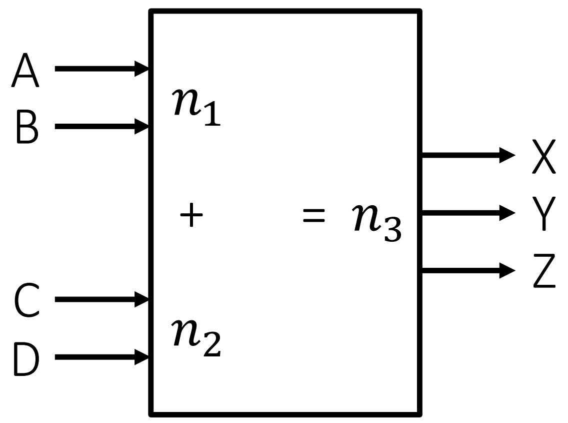
Lecture Outline

- Karnaugh Maps (K-maps)
- Design Examples

Design Example: 2-bit Adder

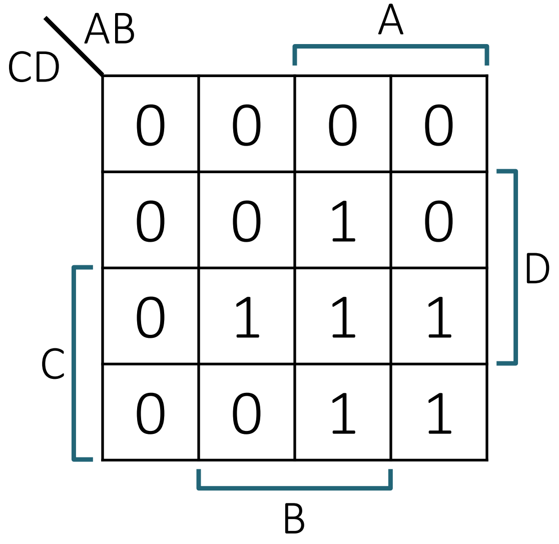
- Block Diagram and Truth Table:



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

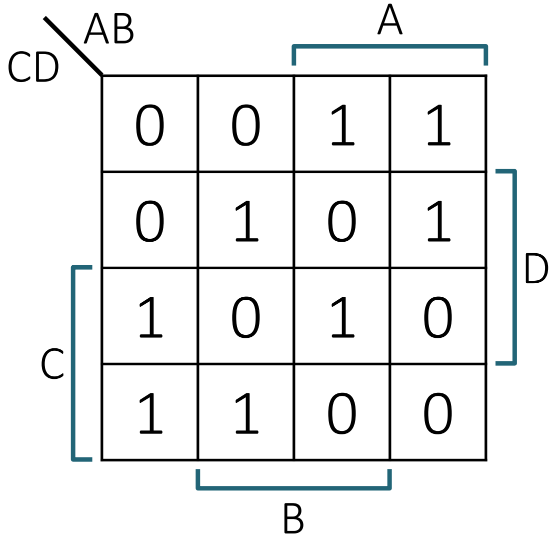
Design Example: 2-bit Adder

K-map for X



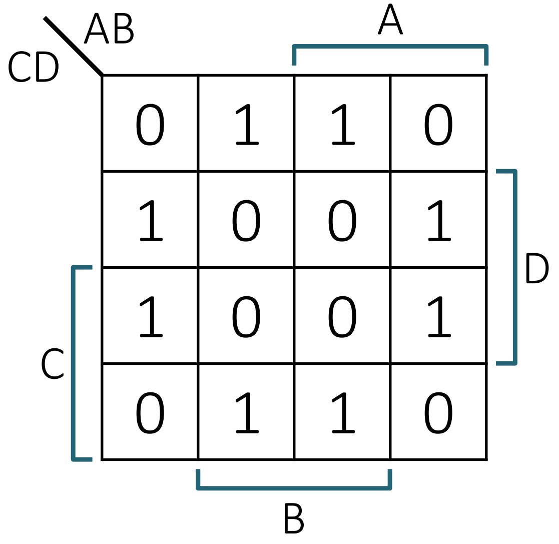
0	0	0	0
0	0	1	0
0	1	1	1
0	0	1	1

K-map for Y



0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0

K-map for Z



0	1	1	0
1	0	0	1
1	0	0	1
0	1	1	0

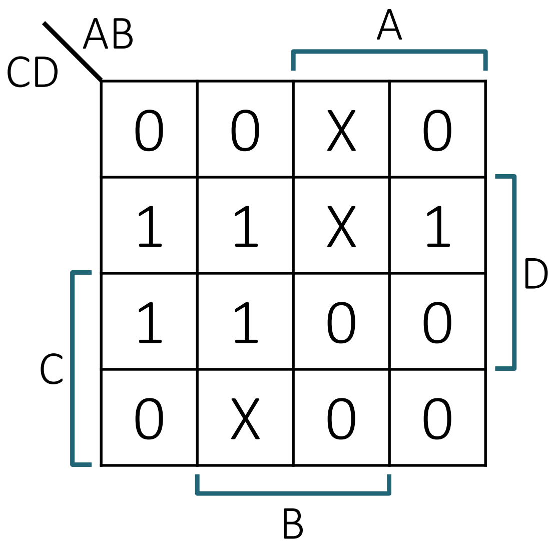
$X =$

$Y =$

$Z =$

Don’t Cares

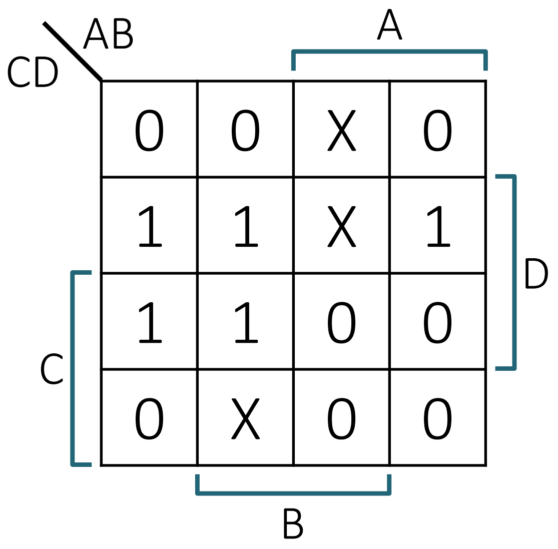
- Use symbol ‘X’ to mean it can be either a 0 or 1
  - Make choice to simplify final expression



0	0	X	0
1	1	X	1
1	1	0	0
0	X	0	0

Let all $\mathrm{X} = 0$:

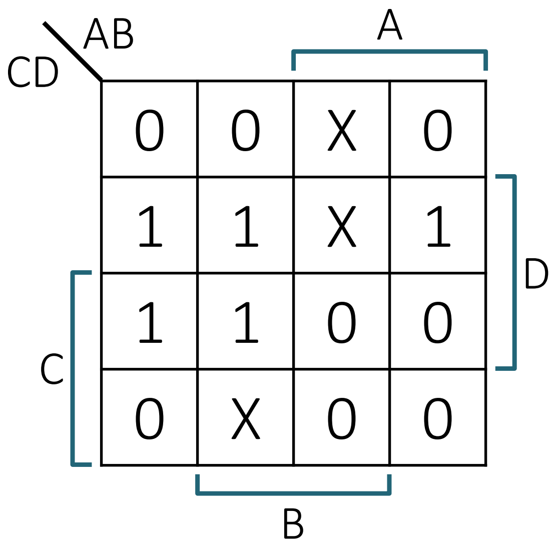
$\mathrm{F} =$



0	0	X	0
1	1	X	1
1	1	0	0
0	X	0	0

Let all $\mathrm{X} = 1$:

$\mathrm{F} =$



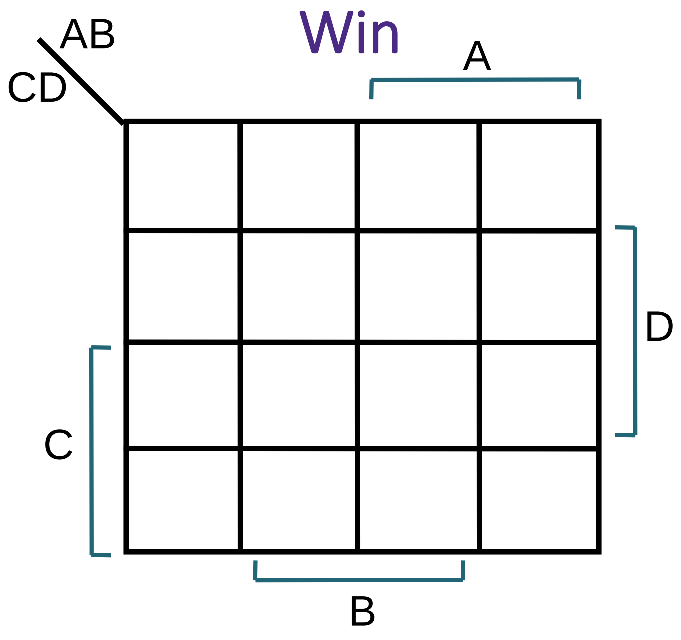
0	0	X	0
1	1	X	1
1	1	0	0
0	X	0	0

Choose wisely:

$\mathrm{F} =$

Design Example: Rock-Paper-Scissors

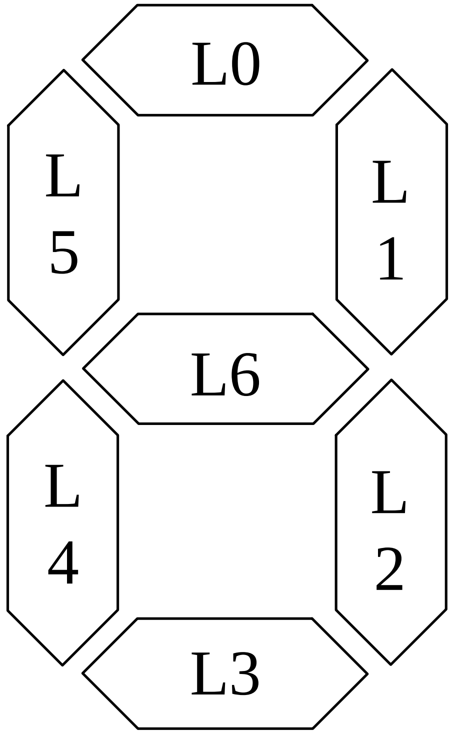
- Rock (00), Paper (01), Scissors (10) for two players P0 and P1
- **Output:** Win = Winner's ID (0/1)
  Tie = 1 if Tie, 0 else



P1		P0			
A	B	C	D	Win	Tie
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		

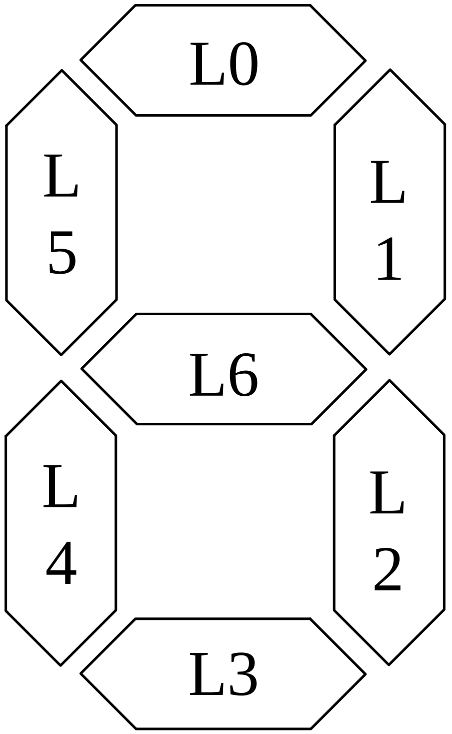
Case Study: Seven-Segment Display

- Chip to drive digital display

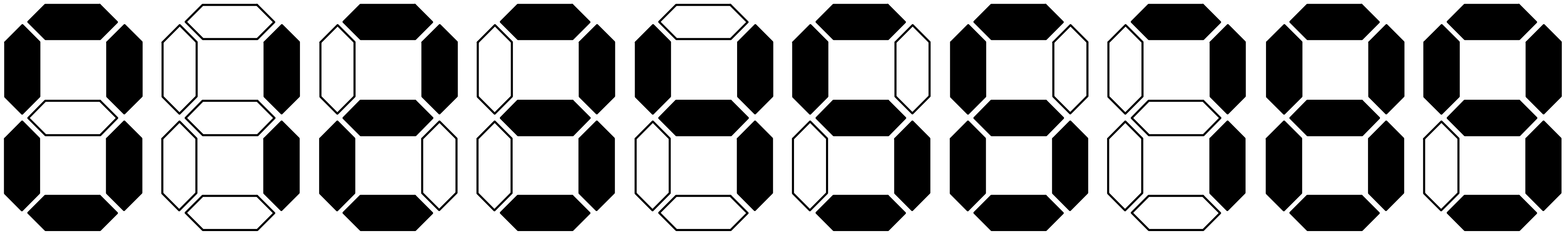


B3	B2	B1	B0	Val
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Case Study: Seven-Segment Display



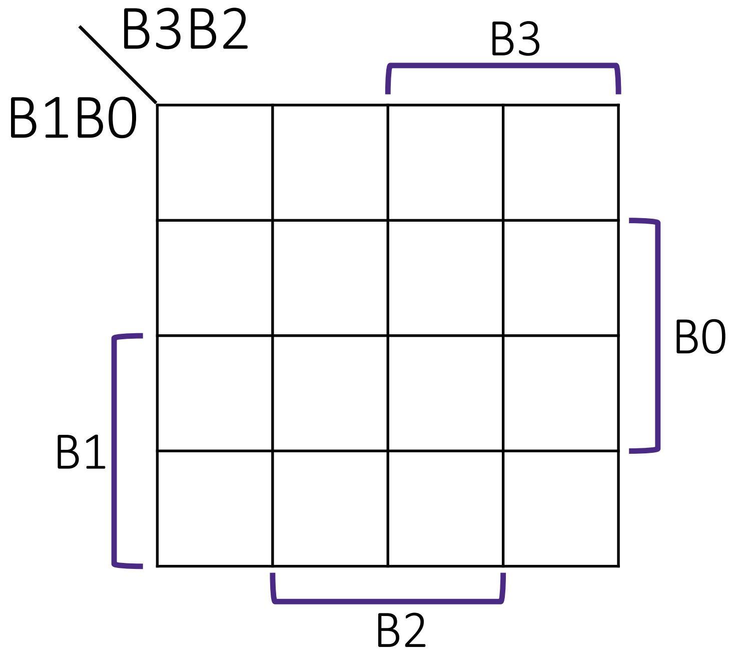
B3	B2	B1	B0	Val	L0	L1	L2	L3	L4	L5	L6
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1



Case Study: Seven-Segment Display

- Implement L5:

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              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
```

Procedural Blocks

- `assign`: continuous assignment
  - *e.g.* `assign F = ~((A & B) | (C & D));`

- `initial`: executes once at time zero
  - Set initial values (simulation only!!!)
  - Define testbench waveforms (and `monitor`)
  - *e.g.*

```
initial begin
     for(i=0; i<8; i=i+1)
        {SEL, I, J} = i;   #10;
end
```

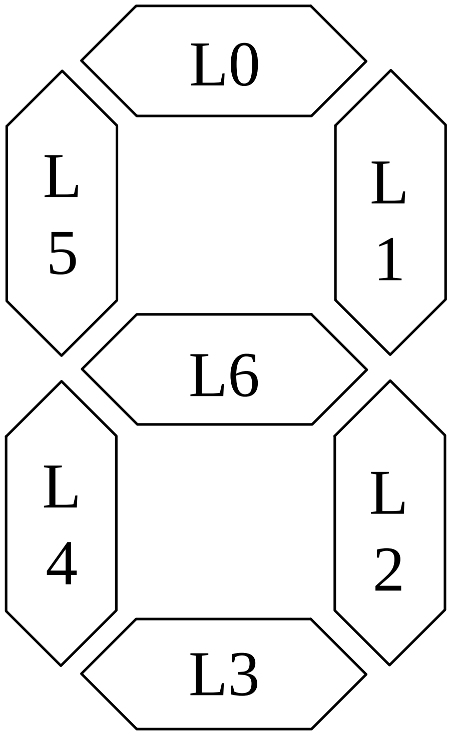
Procedural Blocks

- `always`: loop to execute over and over again
  - Block gets triggered by a *sensitivity list*
  - Any object that is assigned a value in an `always` statement must be declared as a variable (`logic`).
  - Examples:
    - `always @ (a or b or c)` ↔ `always @ (a, b, c)`
    - `always @ (*)` implicitly contains all read signals within the block

- `always_comb`: special SystemVerilog for CL
  - Similar to `always @(*)`, but generally more robust
  - *Only for use with combinational logic!!!*

Verilog: Extend 7-Seg to Hex

- Show “A” on 0b1010 (ten) to “F” on 0b1111 (fifteen)



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

  always_comb
    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
```

Circuit Implementation Techniques

- **Truth Tables** – “Black box” circuit description
- **Boolean Algebra** – Math form for optimization
  - ***K-Maps*** – Alternate simplification technique
- **Circuit Diagrams**
- **Verilog** – Simulation & mapping to FPGAs