Optimization via K-Maps to 2-level forms

- Readings: 2.11-2.12.2, 2.14
- Sum of Products form: the OR of several AND gates, inversions over only inputs
  - $F = \overline{X}+Y\overline{Z}+XYZ$
- Circuit diagram & inversions:

On Sets and Off Sets

X	Y	Z	H
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

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

Two-Level Simplification

***Key Tool: The Uniting Theorem* — $A(\overline{B} + B) = A$**

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

$$F = A\overline{B} + AB = A(\overline{B} + B) = A$$

B's values change within the on-set rows

B is eliminated, A remains

A's values don't change within the on-set rows

A	B	G
0	0	1
0	1	0
1	0	1
1	1	0

$$G = \overline{A}\,\overline{B} + A\overline{B} = (\overline{A} + A)\overline{B} = \overline{B}$$

B's values stay the same within the on-set rows

A is eliminated, B remains

A's values change within the on-set rows

Essence of Simplification:
find two element subsets of the ON-set where only one variable changes its value. This single varying variable *can be eliminated!*

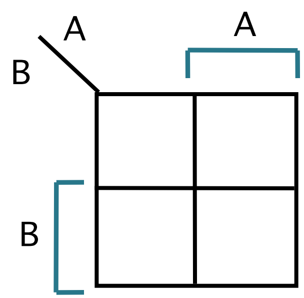
Karnaugh Maps

Karnaugh Map Method

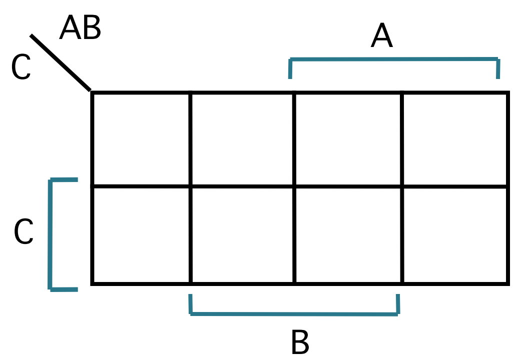
K-map is an alternative method of representing the truth table that helps visualize adjacencies in up to 4 dimensions

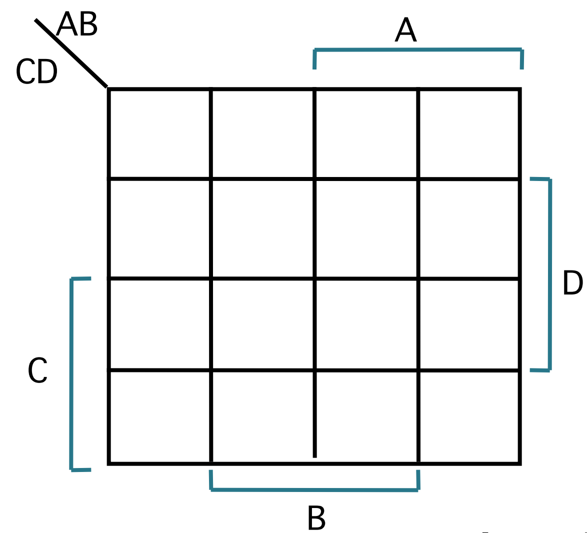
Beyond that, computer-based methods are needed

2-variable K-map



3-variable K-map

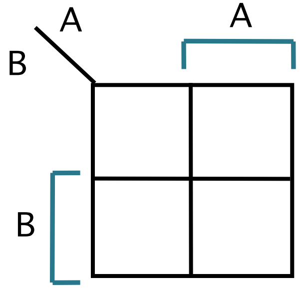




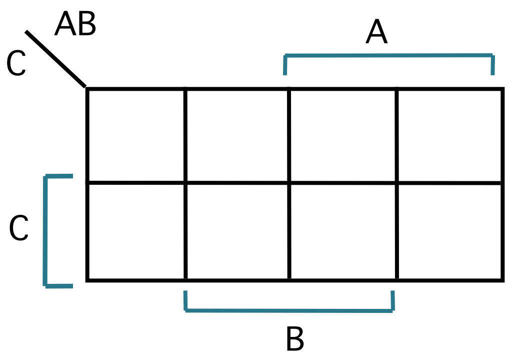
4-variable K-map

Truth Tables to K-Maps

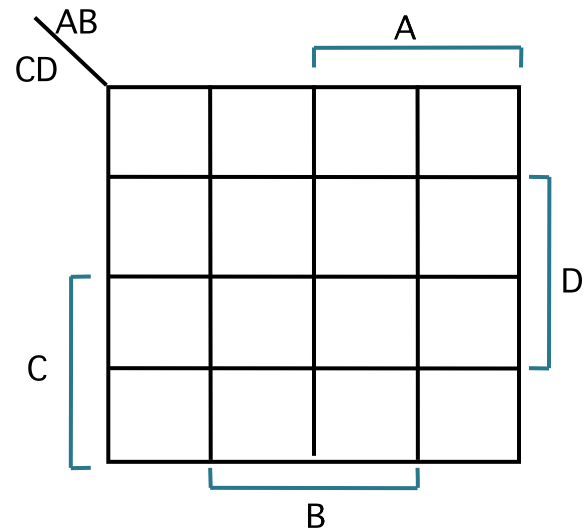
A	B	F
0	0	0
0	1	0
1	0	1
1	1	0



A	B	C	G
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0



A	B	C	D	H
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



K-Map Simplification

K-Map Method Examples

B \ A		A
	0	1
B	0	1

F =

B \ A		A
	1	1
B	0	0

G =

Cin \ A B			A	A
	0	0	1	0
Cin	0	1	1	1
		B	B	

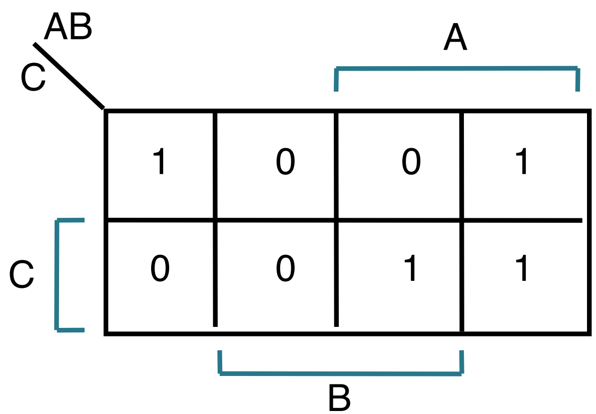
Cout =

C \ AB			A	A
	0	0	1	1
C	0	0	1	1
		B	B	

F(A,B,C) =

K-Map Simplification (cont.)

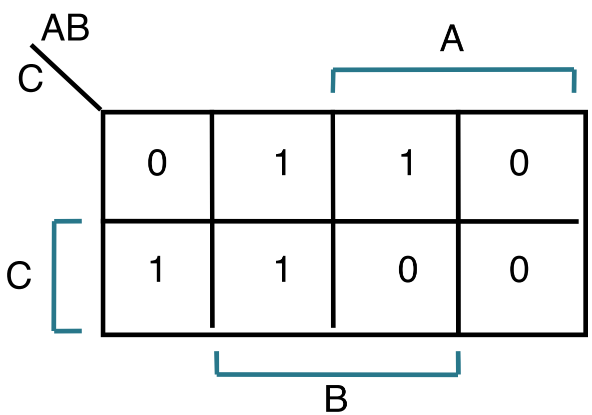
More K-Map Method Examples, 3 Variables



$F(A,B,C) = \bar{A}\,\bar{B}\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,\bar{B}\,C + A\,B\,C$

F =

In the K-map, adjacency wraps from left to right and from top to bottom



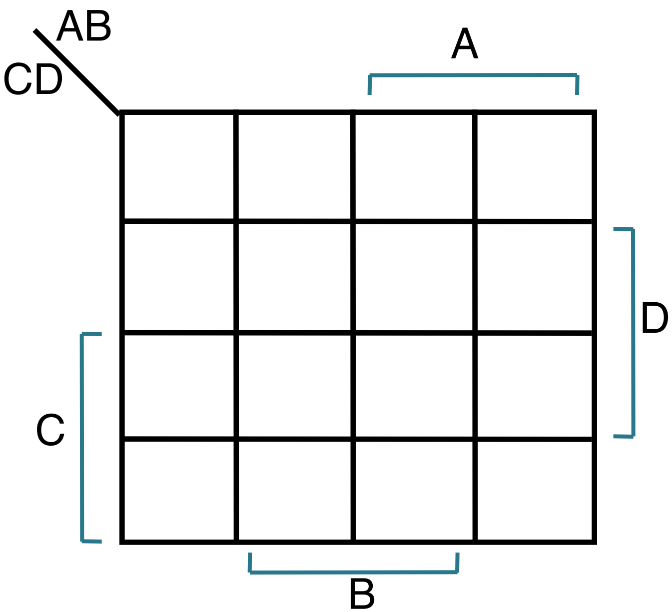
$\bar{F}$ simply replace 1's with 0's and vice versa

$\bar{F}(A,B,C) = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C}$

$\bar{F}$ =

4-Variable K-Map

K-map Method Examples: 4 variables

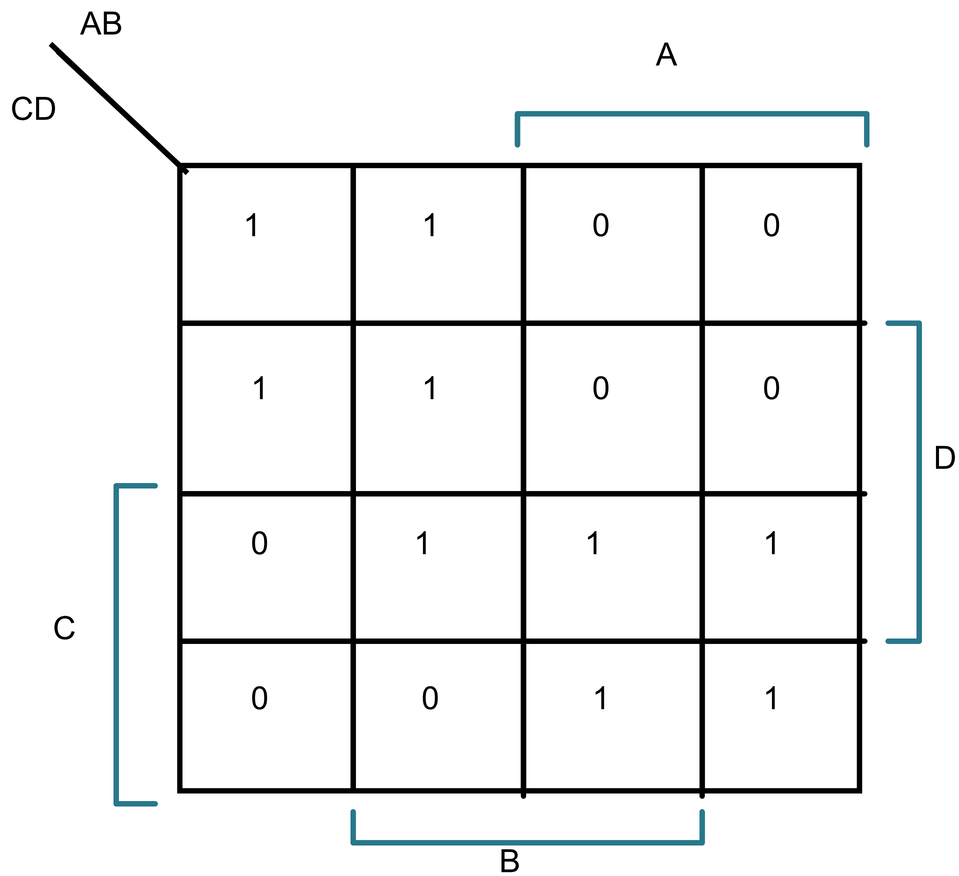


$$F = \overline{A}D + BD + \overline{B}C + A\overline{B}D$$

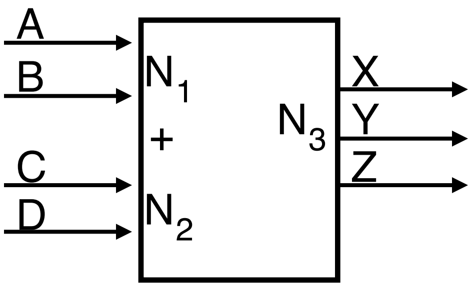
K-Map Example

$$F = (A \text{ xor } C) * D + A\overline{C}\,\overline{D} + \overline{A}BC\overline{D}$$

K-Map Example with Multiple Solutions



Design Example: 2-bit Adder



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

Block Diagram and Truth Table

Design Example (cont.)

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

K-map for X

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

K-map for Y

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

K-map for Z

X =

Z =

Y =

Don't Cares

Don't Cares can be treated as 1's or 0's if it is advantageous to do so

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

If all X=0, then

F =

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

If all X=1, then

F =

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

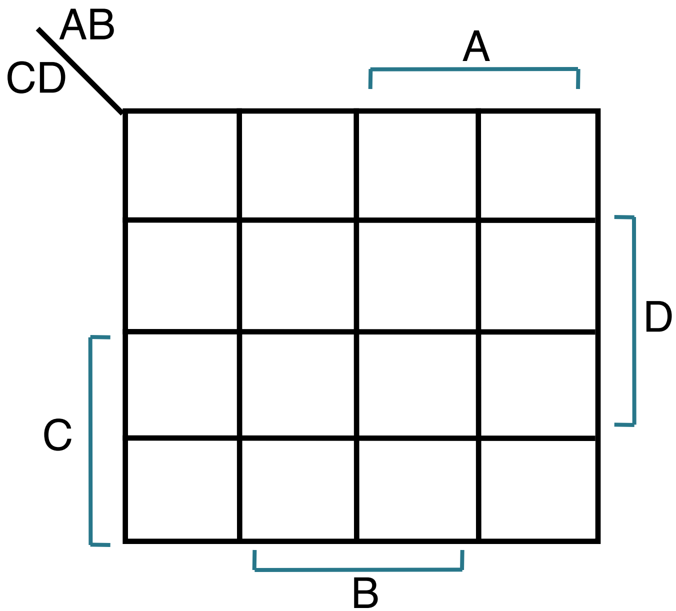
Using Don't Cares, then

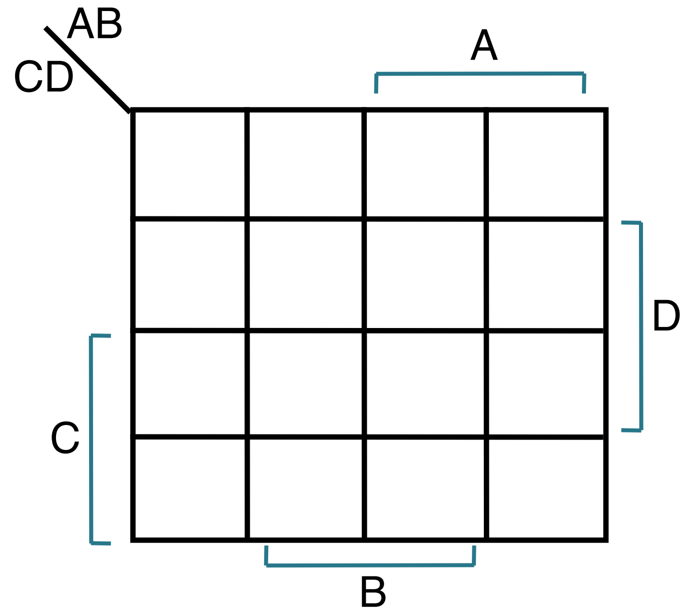
F =

Design Example: Rock-Paper-Scissors

- Rock (00), Paper (01), Scissors (10) for two players.
- Output: Winner = Winner's ID (0/1)
  Tie = 1 if Tie, 0 if not

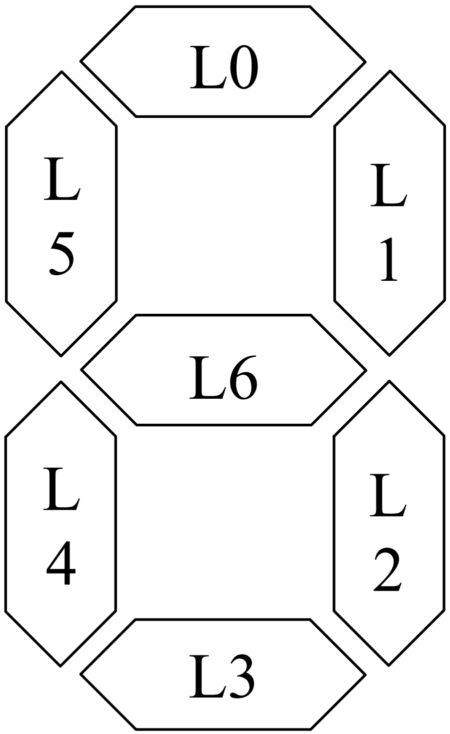
Rock, Paper, Scissors (cont.)





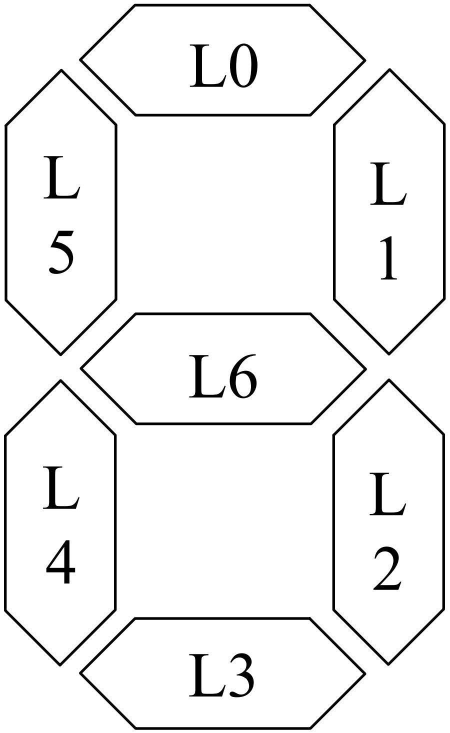
Case Study: Seven Segment Display

- Chip to drive digital display

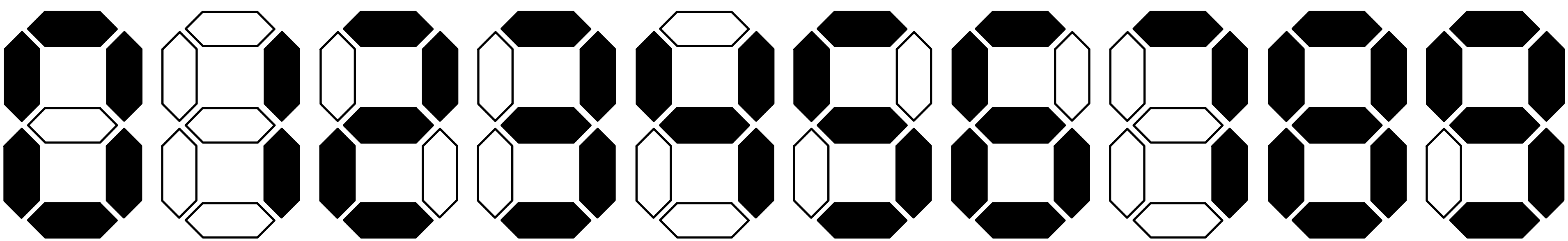


B3	B2	B1	B0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0

Case Study (cont.)



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 (cont.)

- 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

- 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: Circuit Implementation Techniques

- Truth Tables - Case-by-case circuit description
- Boolean Algebra - Math form for optimization
- K-Maps - Simplification technique
- Circuit Diagrams - TTL Implementations
- Verilog – Simulation & Mapping to FPGAs