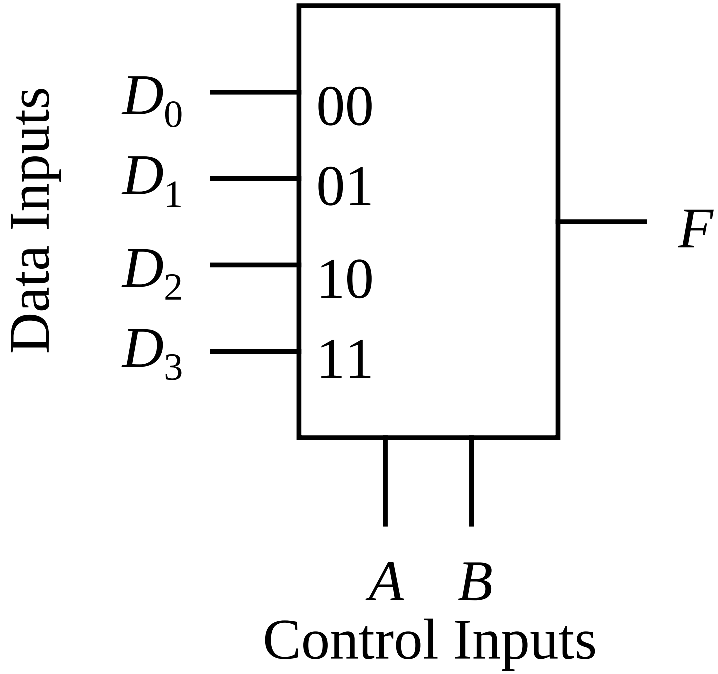
CMSC 313 Lecture 19

- Combinational Logic Components
- Programmable Logic Arrays
- Karnaugh Maps

Last Time & Before

- Returned midterm exam
- Half adders & full adders
- Ripple carry adders vs carry lookahead adders
- Propagation delay
- Multiplexers

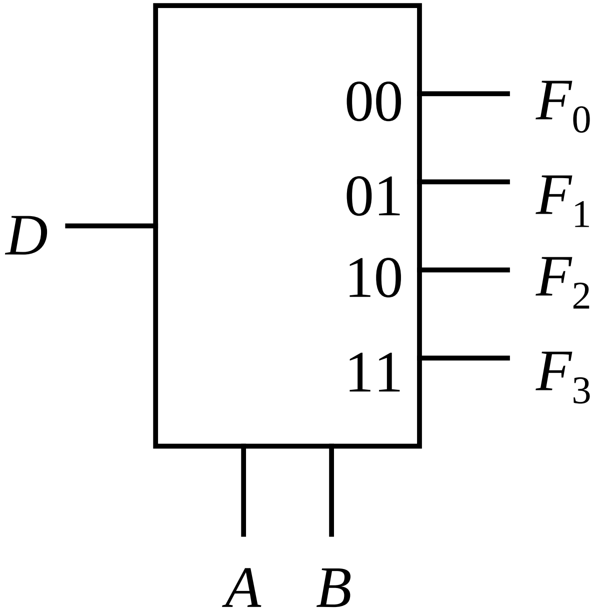
Multiplexer



A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \overline{A}\,\overline{B}\,D_0 + \overline{A}\,B\,D_1 + A\,\overline{B}\,D_2 + A\,B\,D_3$$

Demultiplexer

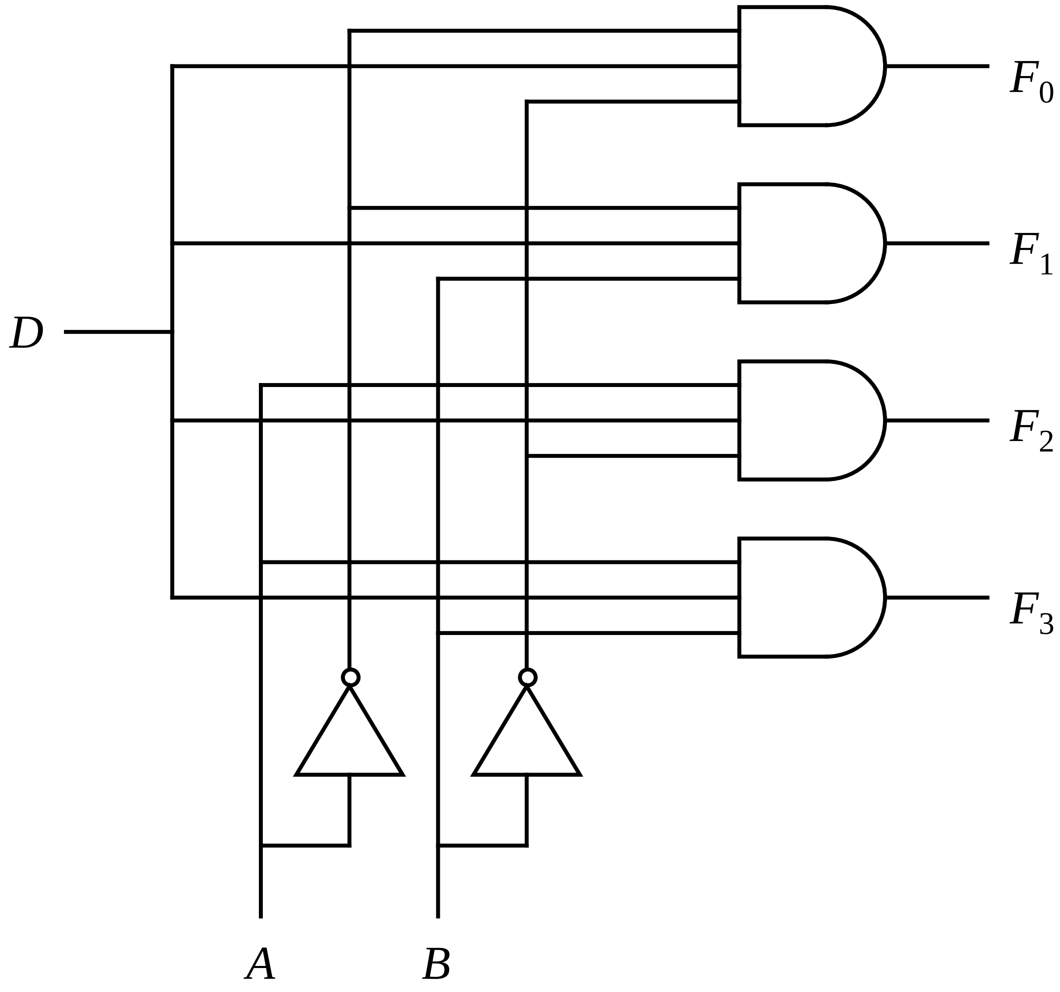


D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

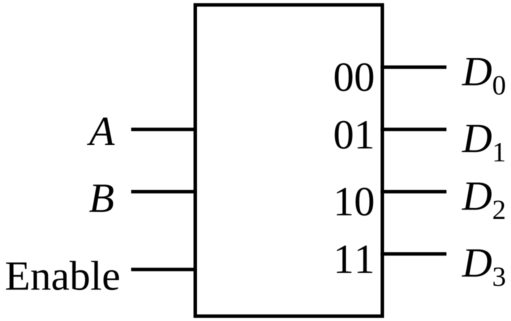
$$F_0 = D\overline{A}\,\overline{B} \qquad F_2 = DA\overline{B}$$

$$F_1 = D\overline{A}B \qquad F_3 = DAB$$

Gate-Level Implementation of DEMUX



Decoder

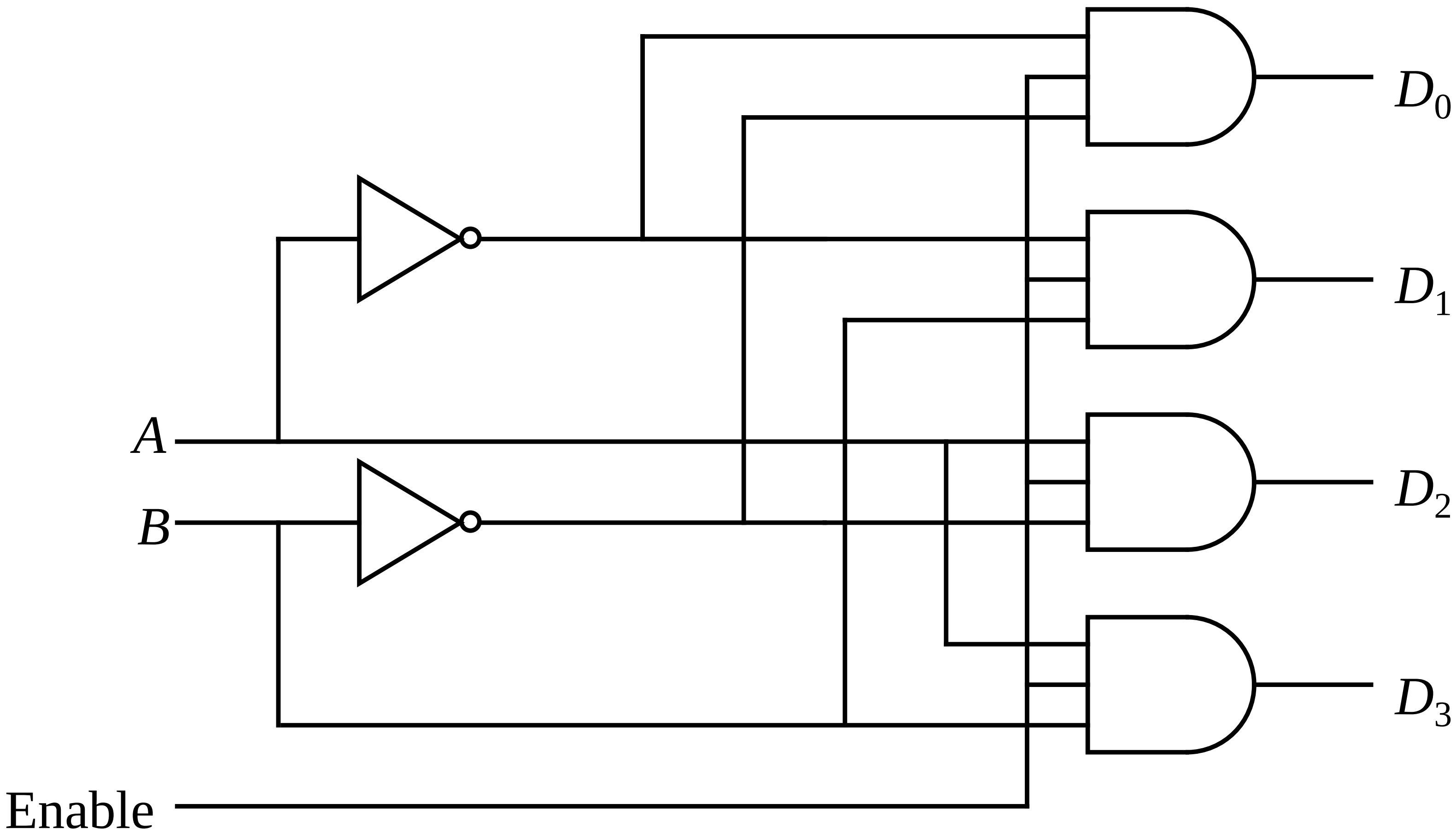


Enable = 1					
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Enable = 0					
A	B	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

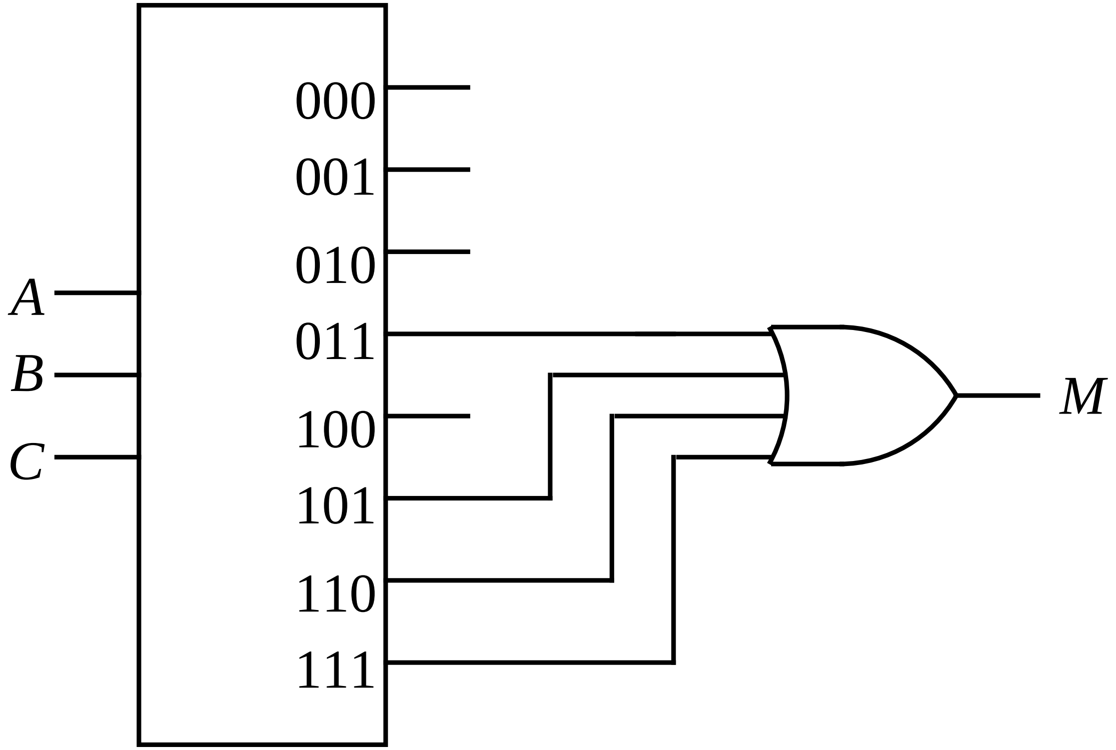
$D_0 = \overline{A}\,\overline{B}$ $D_1 = \overline{A}\,B$ $D_2 = A\,\overline{B}$ $D_3 = A\,B$

Gate-Level Implementation of Decoder



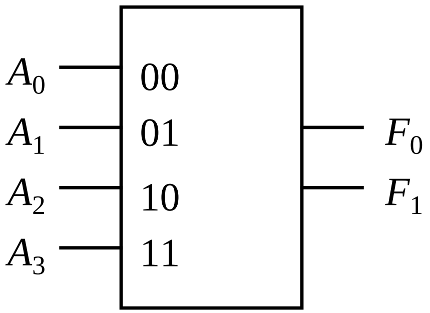
Decoder Implementation of Majority Function

- Note that the enable input is not always present. We use it when discussing decoders for memory.



Priority Encoder

- **An encoder translates a set of inputs into a binary encoding.**
- **Can be thought of as the converse of a decoder.**
- **A priority encoder imposes an order on the inputs.**
- **A_i has a higher priority than A_{i+1}**

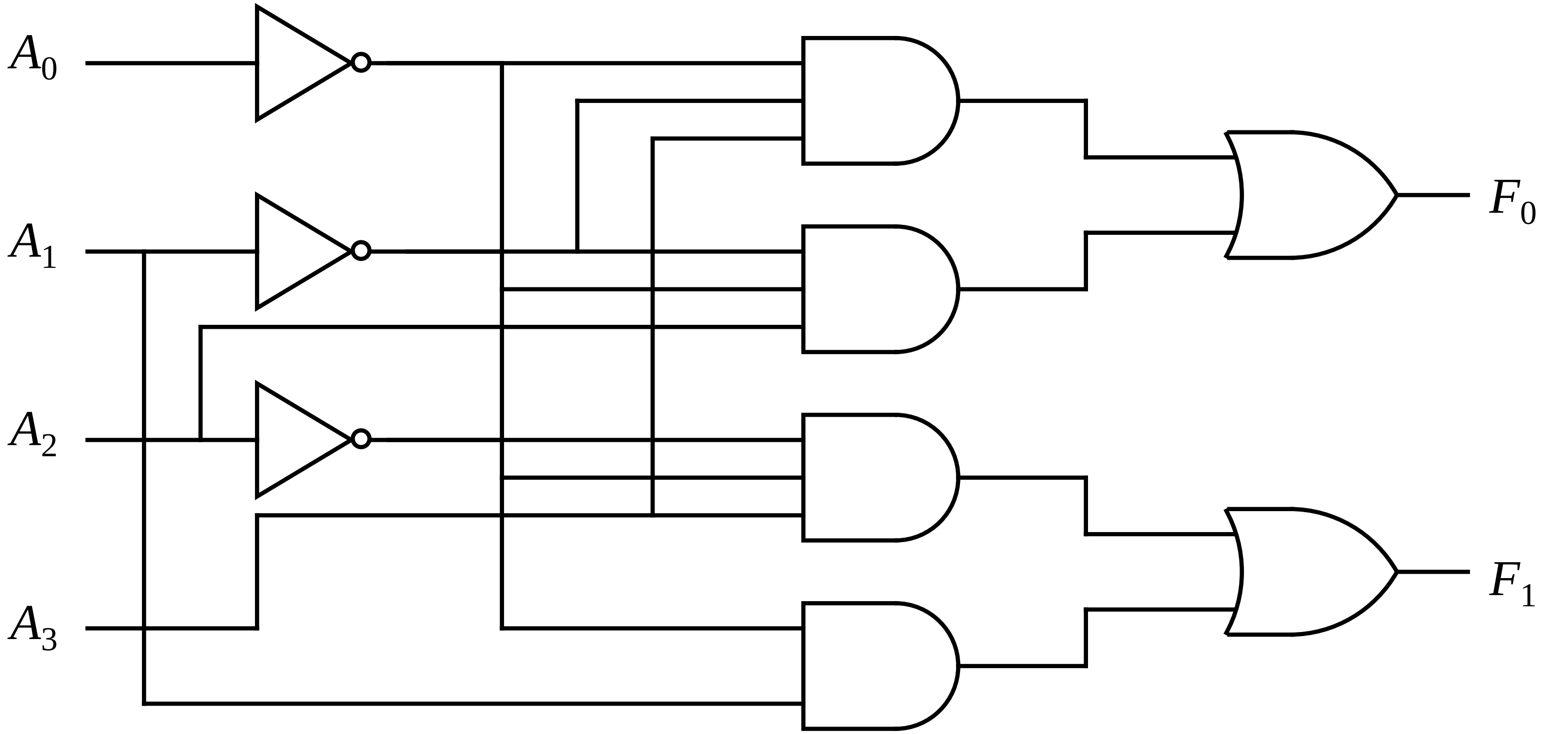


$$F_0 = \overline{A_0}\,\overline{A_1}\,A_3 + \overline{A_0}\,\overline{A_1}\,A_2$$

$$F_1 = \overline{A_0}\,\overline{A_2}\,A_3 + \overline{A_0}\,A_1$$

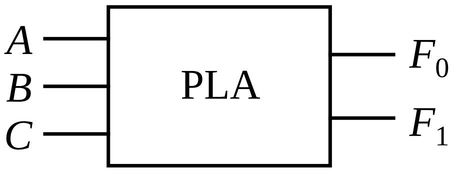
A_0	A_1	A_2	A_3	F_0	F_1
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

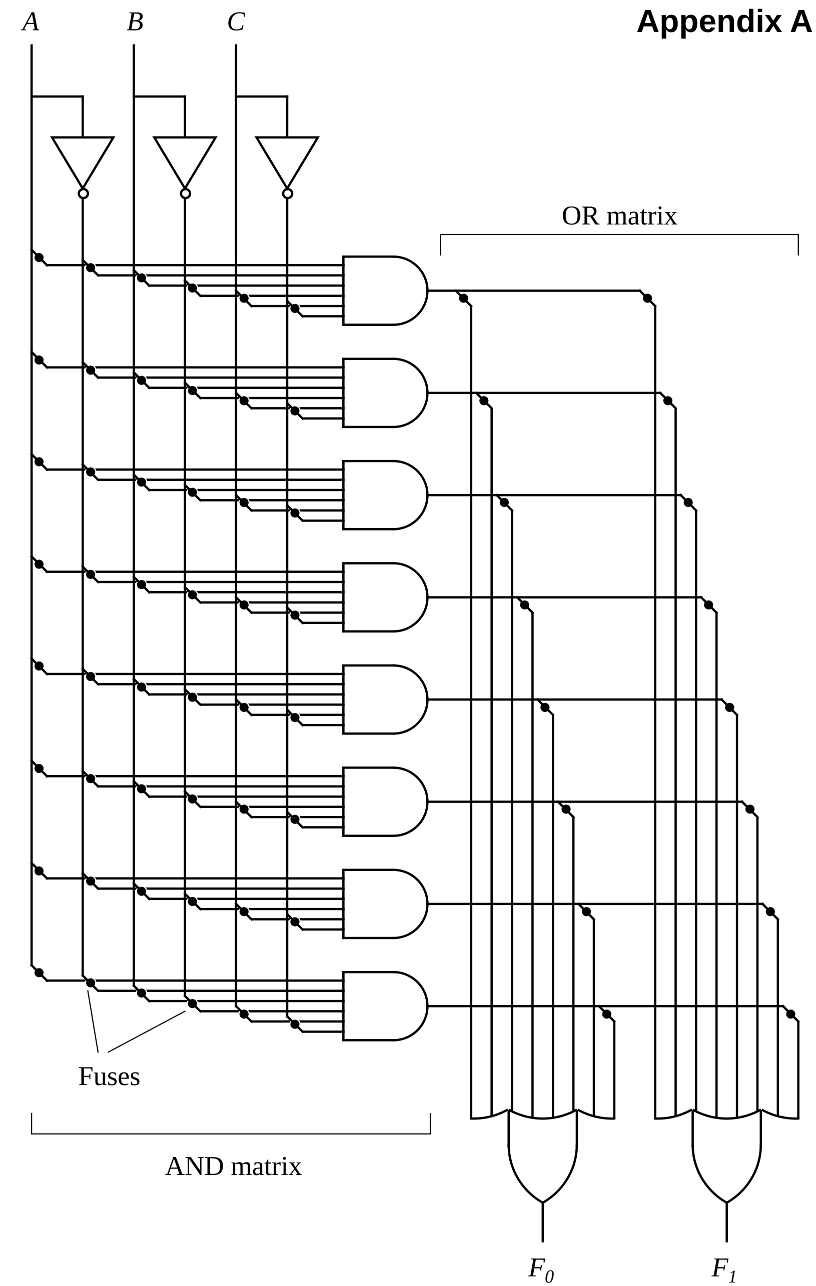
AND-OR Implementation of Priority Encoder



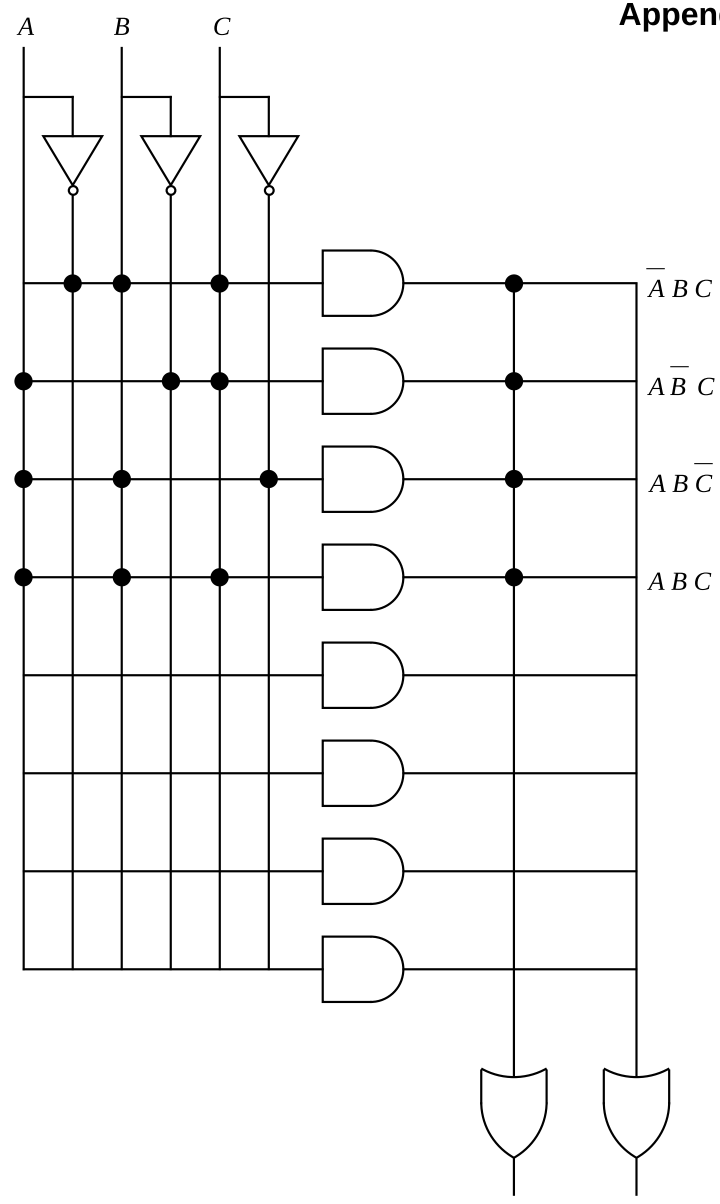
Programmable Logic Array

- A PLA is a customizable AND matrix followed by a customizable OR matrix.
- Black box view of PLA:



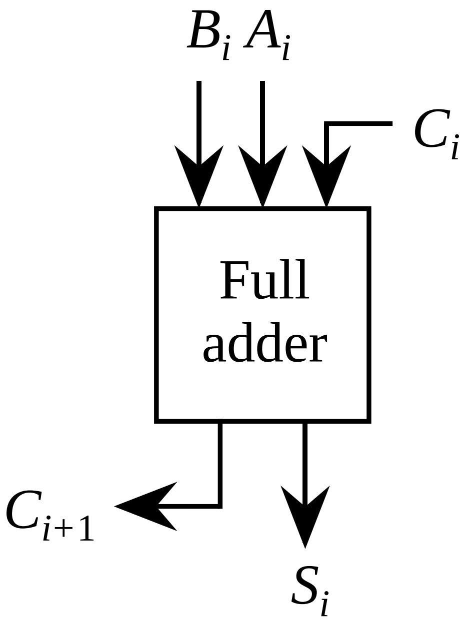


Simplified Representation of PLA Implementation of Majority Function

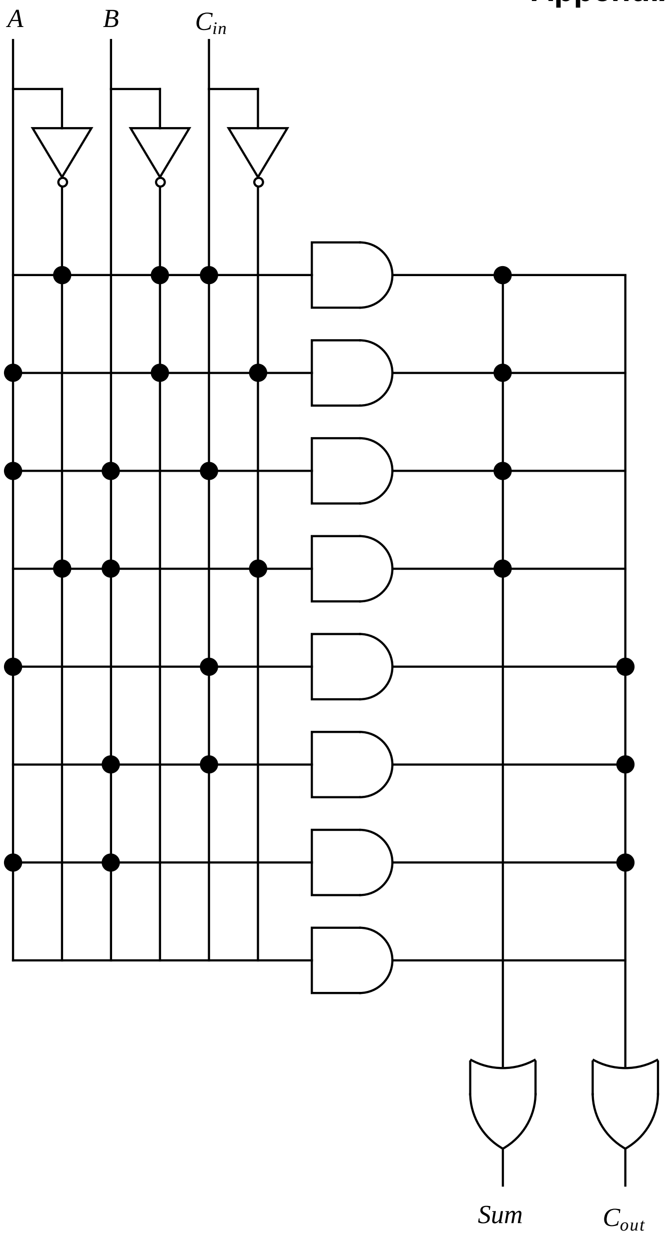


Full Adder

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



PLA Realization of Full Adder

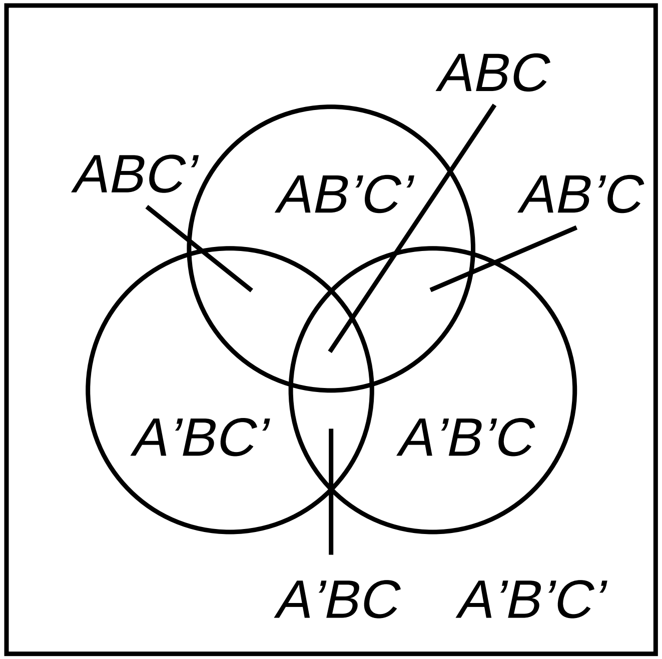


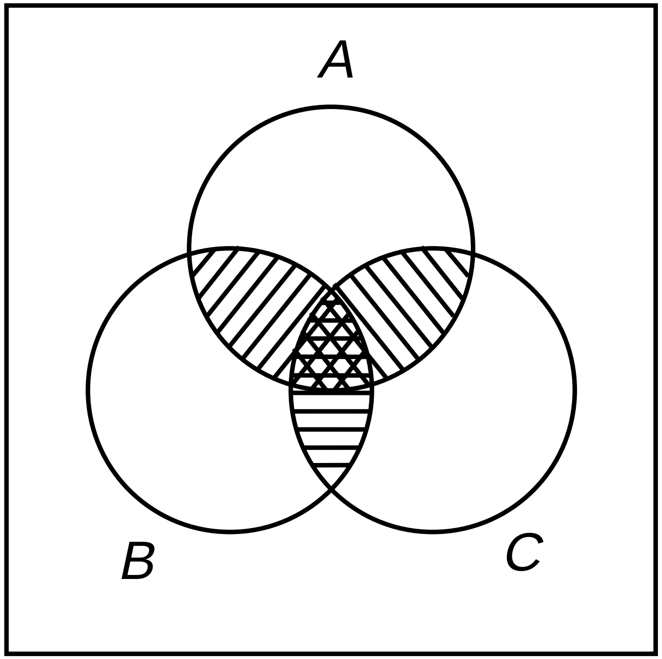
Reduction (Simplification) of Boolean Expressions

- It is usually possible to simplify the canonical SOP (or POS) forms.
- A smaller Boolean equation generally translates to a lower gate count in the target circuit.
- We cover three methods: algebraic reduction, Karnaugh map reduction, and tabular (Quine-McCluskey) reduction.

Karnaugh Maps: Venn Diagram Representation of Majority Function

- **Each distinct region in the "Universe" represents a minterm.**
- **This diagram can be transformed into a *Karnaugh Map*.**





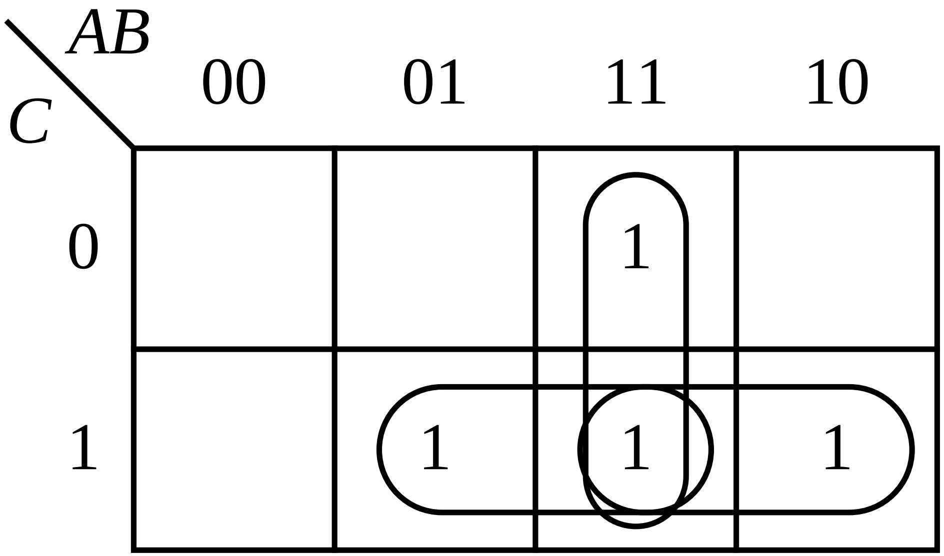
K-Map for Majority Function

- **Place a “1” in each cell that corresponds to that minterm.**
- **Cells on the outer edge of the map “wrap around”**

Minterm Index	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

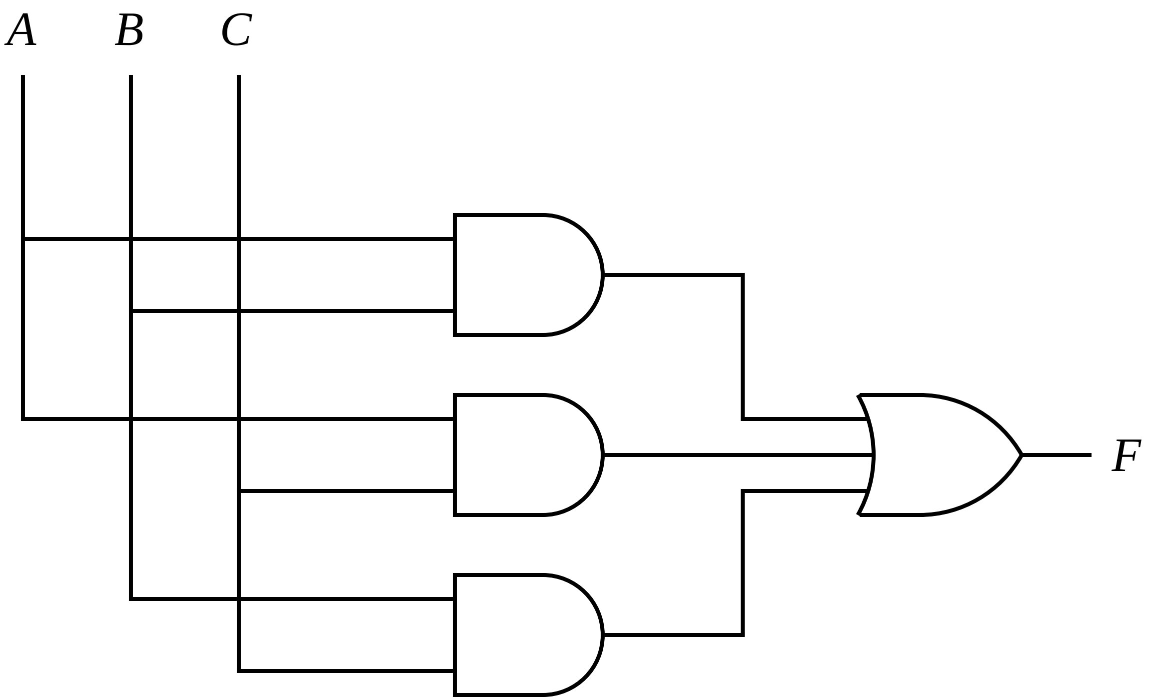
C \ AB	00	01	11	10
0			1	
1		1	1	1

Adjacency Groupings for Majority Function



- F = BC + AC + AB

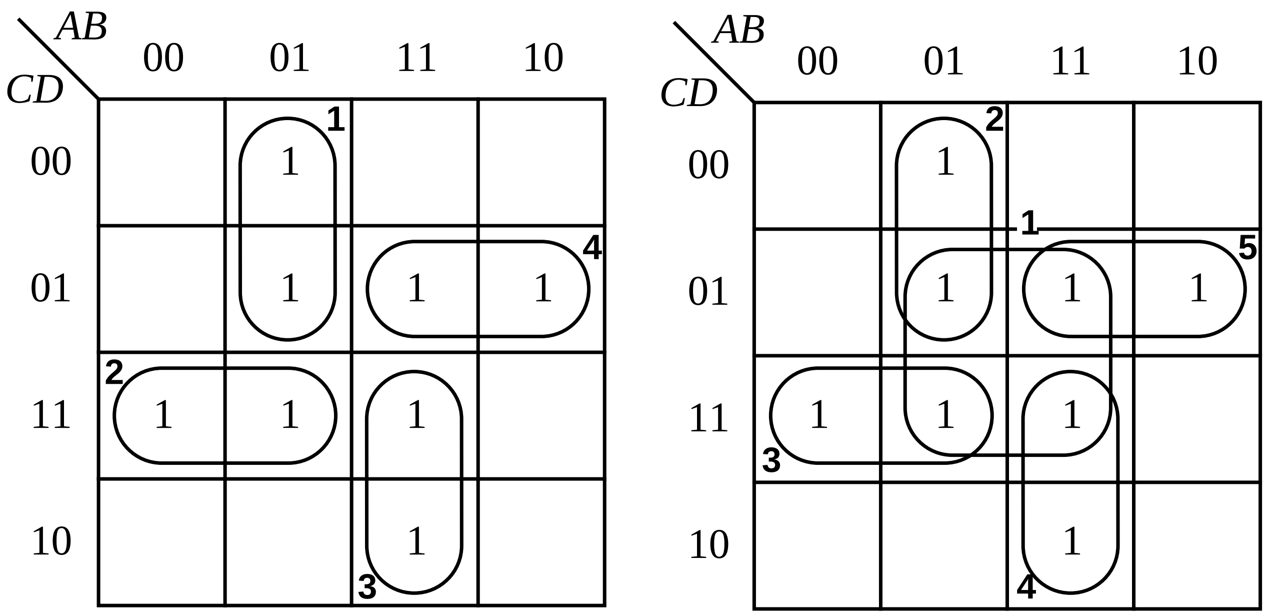
Minimized AND-OR Majority Circuit



- F = BC + AC + AB
- The K-map approach yields the same minimal two-level form as the algebraic approach.

K-Map Groupings

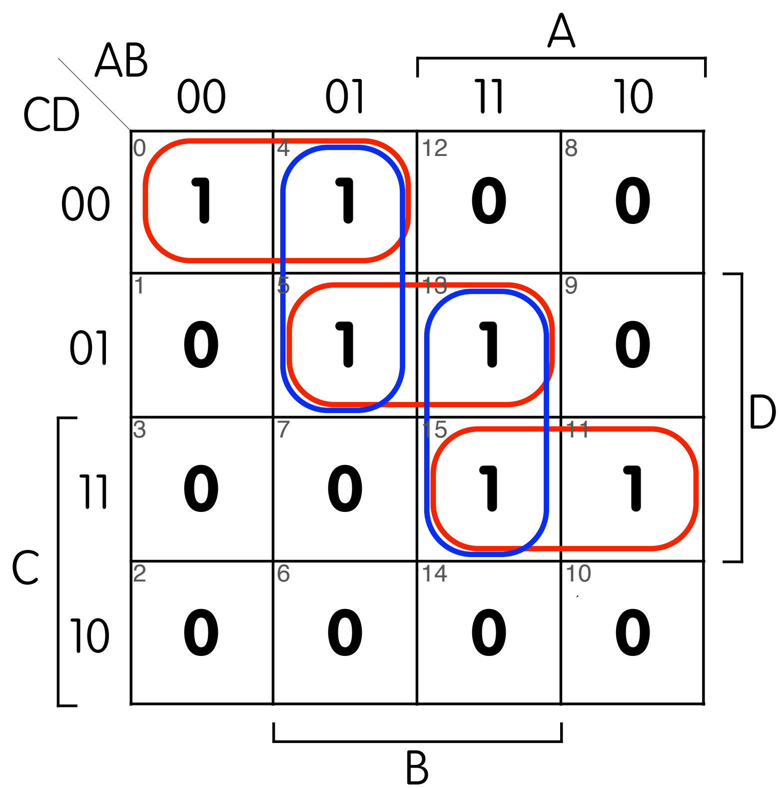
- **Minimal grouping is on the left, non-minimal (but logically equivalent) grouping is on the right.**
- **To obtain minimal grouping, create *smallest* groups first.**



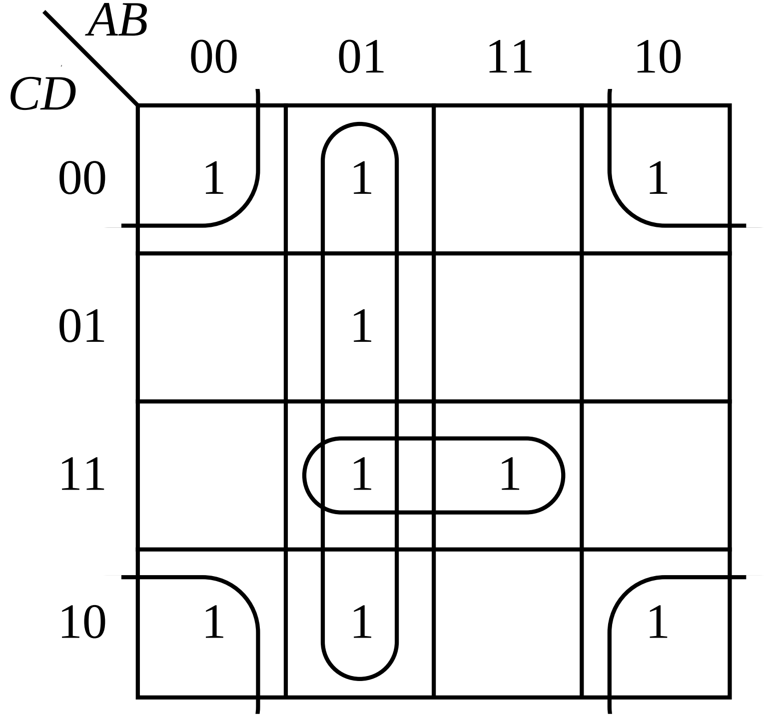
$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

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

Example Requiring More Rules



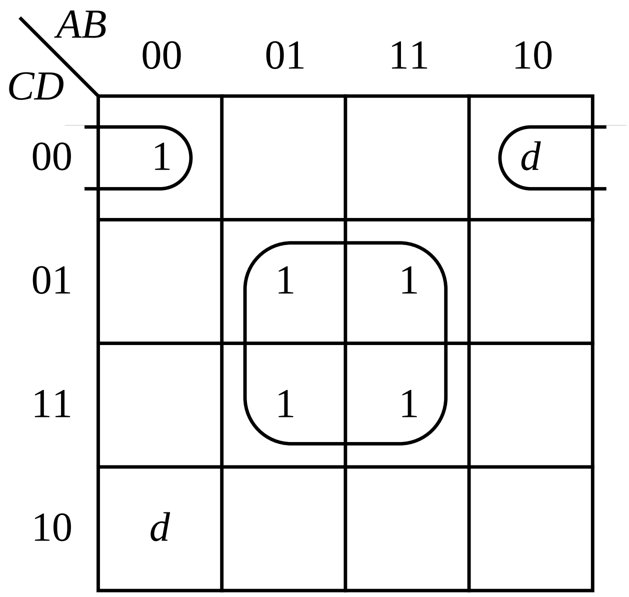
K-Map Corners are Logically Adjacent



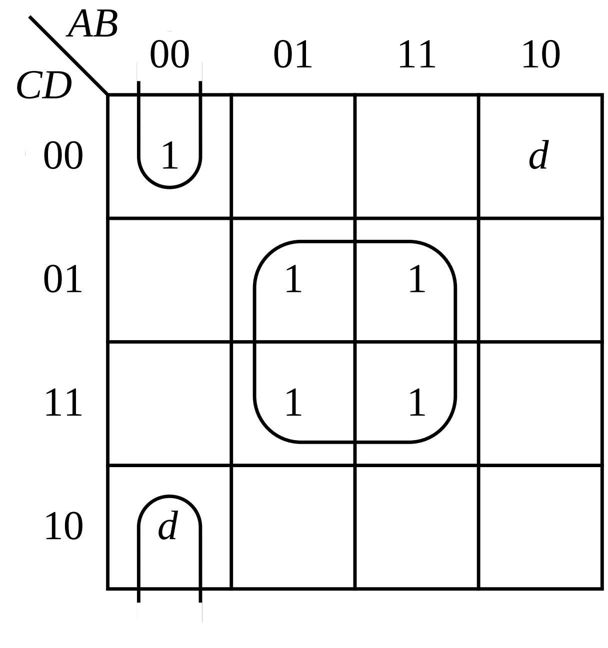
$$F = B\,C\,D + \overline{B}\,\overline{D} + \overline{A}\,B$$

K-Maps and Don't Cares

- There can be more than one minimal grouping, as a result of don't cares.



$$F = \overline{B}\,\overline{C}\,\overline{D} + B\,D$$

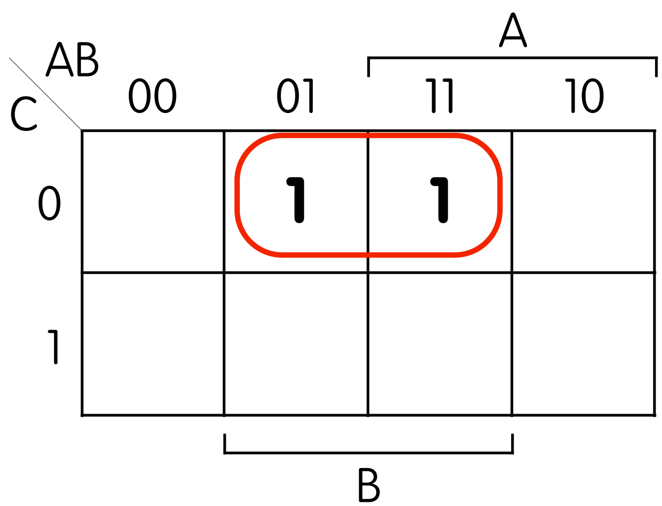


$$F = \overline{A}\,\overline{B}\,\overline{D} + B\,D$$

Gray Code

- Two bits: 00, 01, 11, 10
- Three bits: 000, 001, 011, 010, 110, 111, 101, 100
- Successive bit patterns only differ at 1 position
- For Karnaugh maps, adjacent 1's represent minterms that can be simplified using the rule:

$$ABC' + A'BC' = (A + A')BC' = 1\ BC' = BC'$$



Karnaugh Maps

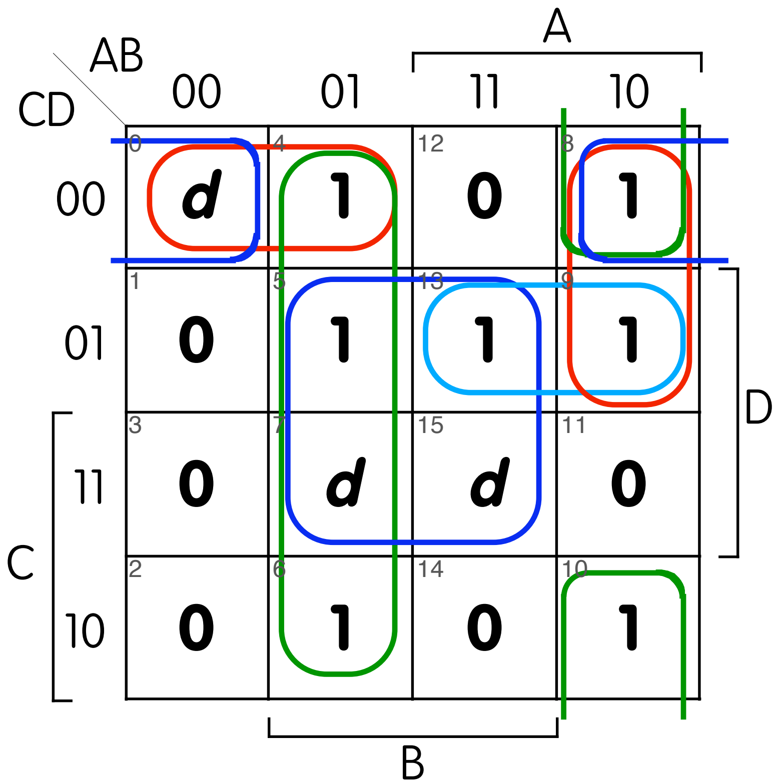
- **Implicant:** rectangle with 1, 2, 4, 8, 16 ... 1's
- **Prime Implicant:** an implicant that cannot be extended into a larger implicant
- **Essential Prime Implicant:** the only prime implicant that covers some 1
- **K-map Algorithm (not from M&H):**

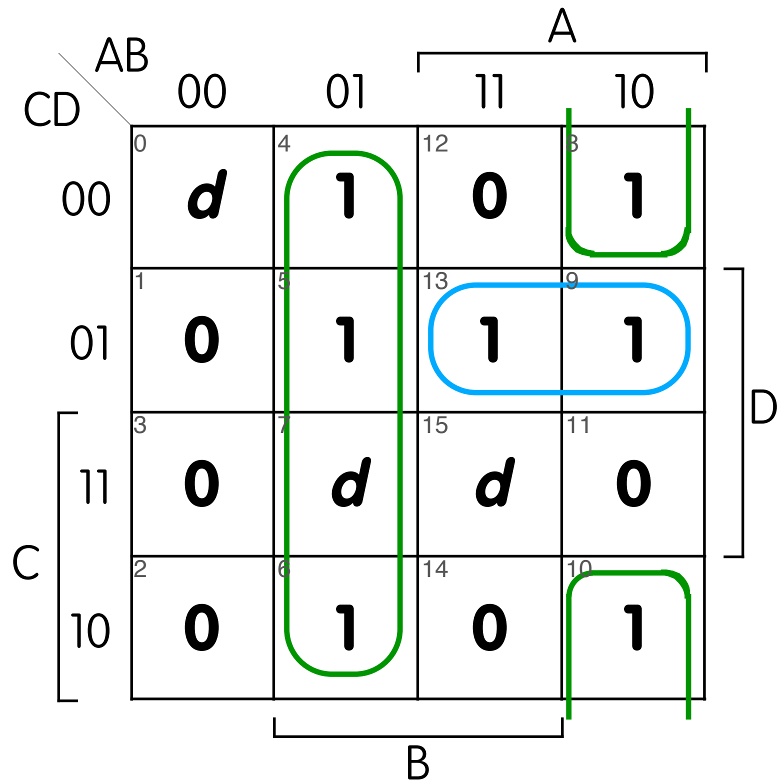
  1. Find ALL the prime implicants. Be sure to check every 1 and to use don't cares.

  2. Include all essential prime implicants.

  3. Try all possibilities to find the minimum cover for the remaining 1's.

K-map Example





A'B + AC'D + AB'D'

Notes on K-maps

- Also works for POS
- Takes 2^n time for formulas with n variables
- Only optimizes two-level logic
  - Reduces number of terms, then number of literals in each term
- Assumes inverters are free
- Does not consider minimizations across functions
- Circuit minimization is generally a hard problem
- Quine-McCluskey can be used with more variables
- CAD tools are available if you are serious

Circuit Minimization is Hard

- Unix systems store passwords in encrypted form.
  - User types in x, system computes f(x) and looks for f(x) in a file.
- Suppose we us 64-bit passwords and I want to find the password x, such that f(x) = y. Let

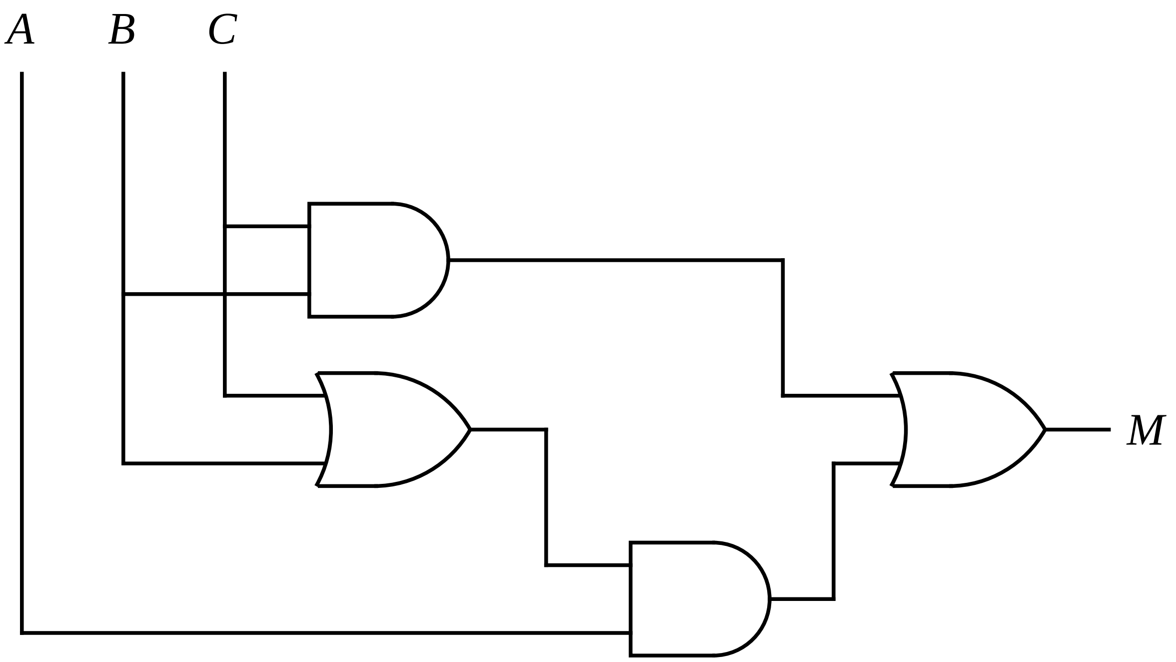
  $g_i(x)$ = 0 if f(x) = y and the ith bit of x is 0

  1 otherwise.

- If the ith bit of x is 1, then $g_i(x)$ outputs 1 for every x and has a very, very simple circuit.
- If you can simplify every circuit quickly, then you can crack passwords quickly.

3-Level Majority Circuit

- **K-Map Reduction results in a reduced two-level circuit (that is, AND followed by OR. Inverters are not included in the two-level count). Algebraic reduction can result in multi-level circuits with even fewer logic gates and fewer inputs to the logic gates.**



CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10