ECE 2300
Digital Logic & Computer Organization
Spring 2025

Combinational Logic Minimization

Cornell University

Announcements

- Weekly calendar (including TA OH schedule) is posted on the course web

De Morgan's Theorem

- Very important, also known as De Morgan's *Law*

(T12) $(X1 \bullet X2 \bullet \ldots \bullet Xn)' = X1' + X2' + \ldots + Xn'$

(T12') $(X1 + X2 + \ldots + Xn)' = X1' \bullet X2' \bullet \ldots \bullet Xn'$

De Morgan Example

- By DeMorgan's Law

  $(X \bullet Y \bullet Z)' = X' + Y' + Z'$

- Proof by perfect induction

XYZ	(X•Y•Z)'	X'+Y'+Z'
000	1	1
001	1	1
010	1	1
011	1	1
100	1	1
101	1	1
110	1	1
111	0	0

Review: Minterms & Maxterms

ABC	**Minterm**	name	**Maxterm**	name
000	A'•B'•C'	m_0	A+B+C	M_0
001	A'•B'•C	m_1	A+B+C'	M_1
010	A'•B•C'	m_2	A+B'+C	M_2
011	A'•B•C	m_3	A+B'+C'	M_3
100	A•B'•C'	m_4	A'+B+C	M_4
101	A•B'•C	m_5	A'+B+C'	M_5
110	A•B•C'	m_6	A'+B'+C	M_6
111	A•B•C	m_7	A'+B'+C'	M_7

$(m_i)' = M_i$ according to De Morgan

$m_i = 1$ <=> input row *i* is selected

$Mi = 1$ <=> input row *i* is NOT selected

Review: Canonical Representations

- Problem: F = 1 if and only if A•B = (B+C')
- Step 1: Lay out the truth table
- Step 2: Derive canonical forms
  - Canonical sum (F=1 if input "hits" the on-set)

    F = A'•B'•C + A•B'•C + A•B•C' + A•B•C

    $= \sum_{A,B,C} (1,5,6,7)$

  - Canonical product (F=1 if input "avoids" the off-set)

    F = (A'•B'•C')'(A'•B•C')'(A'•B•C)'(A•B'•C')'

    = (A+B+C)(A+B'+C)(A+B'+C')(A'+B+C)

    $= \prod_{A,B,C} (0,2,3,4)$

ABC	F
000	0
001	1
010	0
011	0
100	0
101	1
110	1
111	1

- Step 3: Simplification (this lecture)

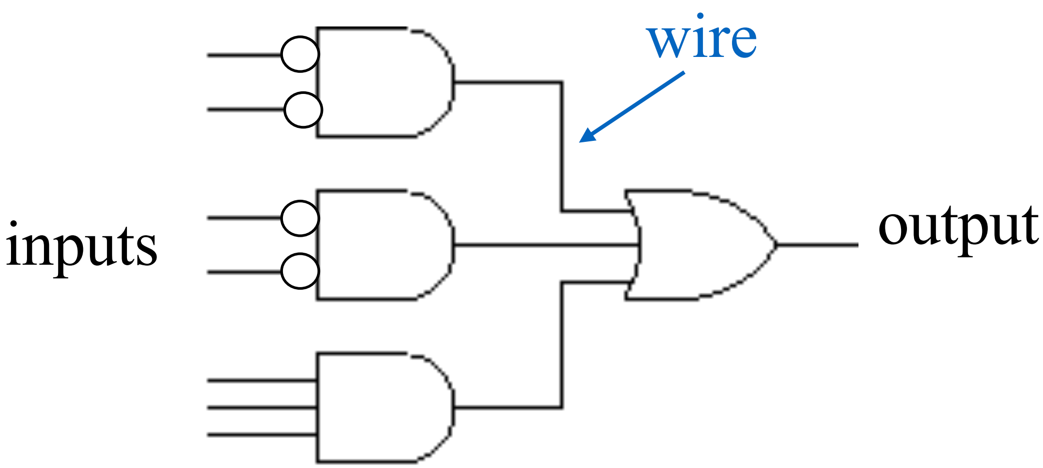
Combinational Logic

- Outputs depend *only* on current inputs
  - Example: Detect if the input is an odd number
  - Can be represented in two-level or multi-level forms
- In contrast, sequential logic has “memory” or “state”
  - Example: Detect if the last two inputs are odd
  - We’ll cover sequential logic later

Two-Level Logic: Sum-of-Products

- Sum of product terms (SOP)
  - e.g., A’•B’ + A’•C’ + A•B•C

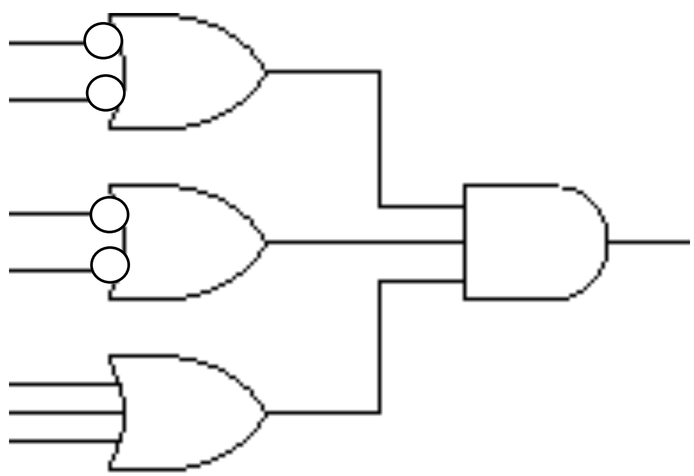
- Circuits look something like this
  - A *bubble* indicates the signal is inverted



AND-OR

Two-Level Logic: Product-of-Sums

- Product of sum terms (POS)
  - e.g., (A'+C') • (B'+C') • (A+B+C)
- Circuits look something like this



OR-AND

Combinational logic can be expressed as SOP or POS

Algebraic Simplification

- Apply theorems to canonical sum (or product) to reduce (1) the number of terms, and (2) the number of literals in each term

- Results in a more compact expression and lower cost digital logic implementation

- We focus on *minimizing two-level logic* in this lecture

Algebraic Simplification Example

- **Binary adder**
  - inputs: A, B, Carry-in (Cin)
  - outputs: Sum, Carry-out (Cout)

- **Truth Table → Canonical sum**

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S = A'•B'•Cin + A'•B•Cin' + A•B'•Cin' + A•B•Cin

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin

Algebraic Simplification Example

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin

= A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)

= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin (combining)

= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)

= B•Cin + A•Cin + A•B•Cin' + A•B•Cin (combining)

= B•Cin + A•Cin + A•B (combining)

We can apply these theorems in a more intuitive fashion using a *Karnaugh Map*

Reduction in Hardware Cost

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin

= B•Cin + A•Cin + A•B

3 inverters 4 three-input ANDs 1 four-input OR	→	3 two-input ANDs 1 three-input OR

Karnaugh Map (K-Map)

- **Idea: Use combining and idempotency theorems *visually* to simplify canonical forms into two-level SOP or POS**

- **Multidimensional representation of a truth table**

- **<u>Adjacent cells</u> represent minterms (or maxterms) that <u>differ by exactly one literal</u>**
  - **Cyclic encoding along each dimension**

- **At most two variables per dimension**

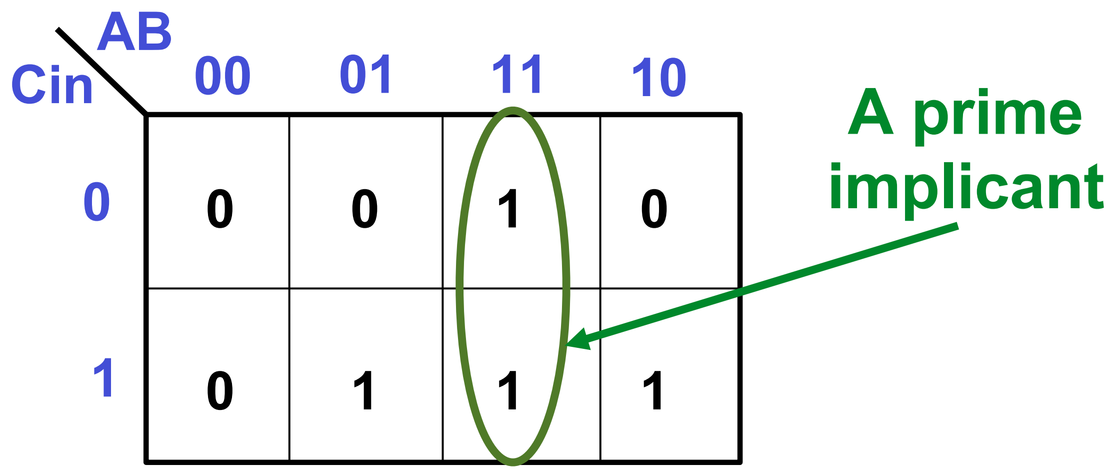
K-Map for Cout

A	B	Cin	S	Cout
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

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

Some K-Map Definitions

- **1-cell**: Minterm of a canonical sum
- **Implicant**: Set of adjacent 1-cells
  - A rectangle in K-Map
  - Number of cells contained must be a power of 2
- **Prime implicant**: Implicant that cannot be contained in a larger implicant



SOP Minimization Using K-Map

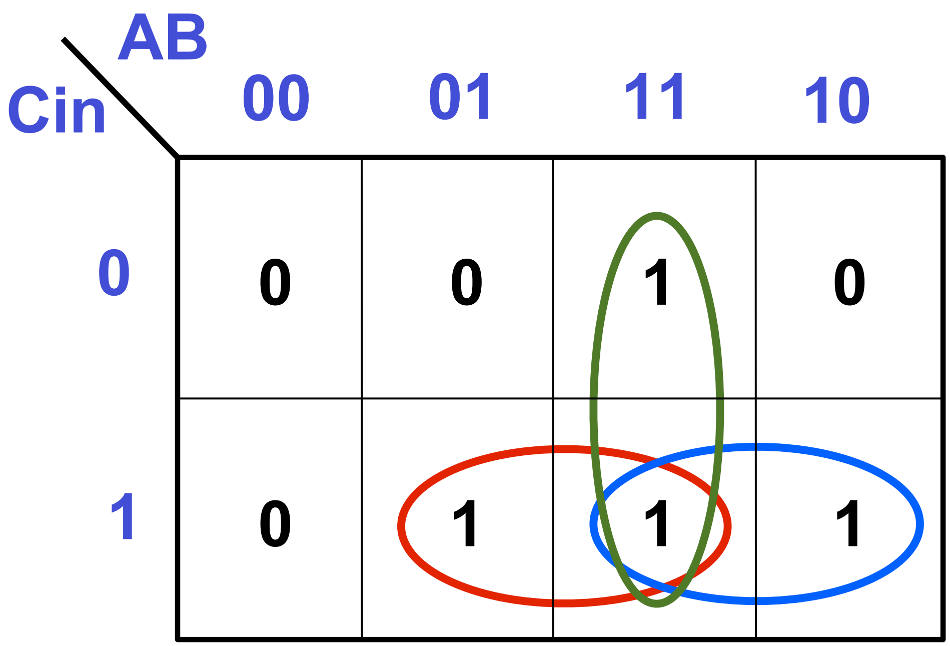
- Goal: Cover all 1-cells with the minimum number of prime implicants

- Procedure
  - Plot 1's corresponding to minterms of function
  - Circle largest possible rectangular sets of 1's
    - Must be power of 2
    - Including "wrap-around" sets
  - Repeat until all minterms are covered
  - OR product terms derived from each circle

- Minimizes the gate count and inputs in the SOP form
  - Solution may not be unique

Simplifying Cout Using a K-Map

$$\begin{aligned}
Cout &= A' \cdot B \cdot Cin + A \cdot B' \cdot Cin + A \cdot B \cdot Cin' + A \cdot B \cdot Cin \\
&= A' \cdot B \cdot Cin + A \cdot B' \cdot Cin + A \cdot B \cdot Cin' + A \cdot B \cdot Cin + A \cdot B \cdot Cin && \text{(idempotency)} \\
&= B \cdot Cin + A \cdot B' \cdot Cin + A \cdot B \cdot Cin' + A \cdot B \cdot Cin && \text{(combining)} \\
&= B \cdot Cin + A \cdot B' \cdot Cin + A \cdot B \cdot Cin' + A \cdot B \cdot Cin + A \cdot B \cdot Cin && \text{(idempotency)} \\
&= B \cdot Cin + A \cdot Cin + A \cdot B \cdot Cin' + A \cdot B \cdot Cin && \text{(combining)} \\
&= B \cdot Cin + A \cdot Cin + A \cdot B && \text{(combining)}
\end{aligned}$$

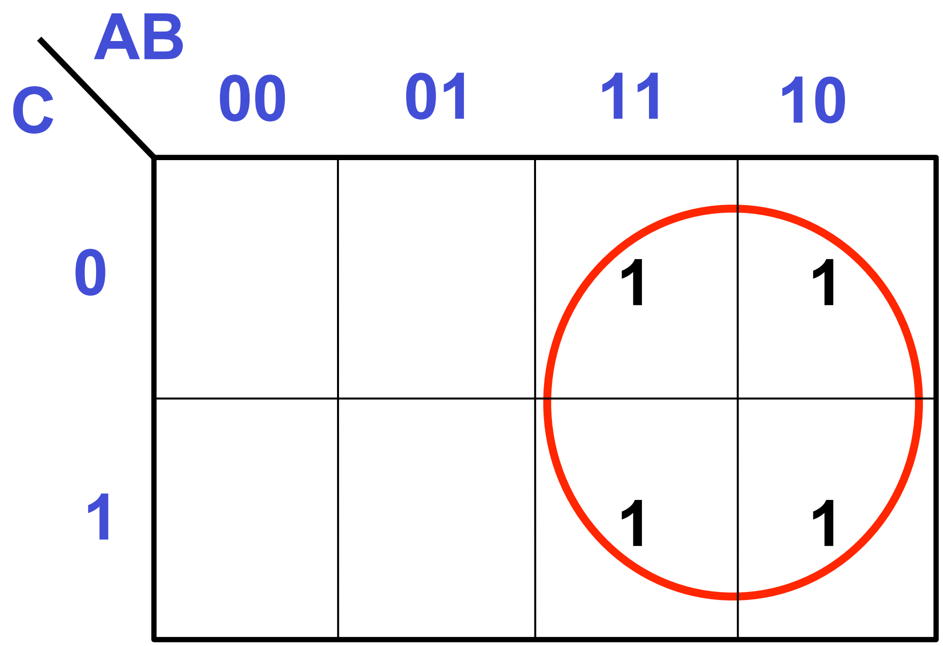


Circles (ovals) indicate applying combining theorem

Idempotency theorem allows circles to overlap

3 Variable K-Map Example

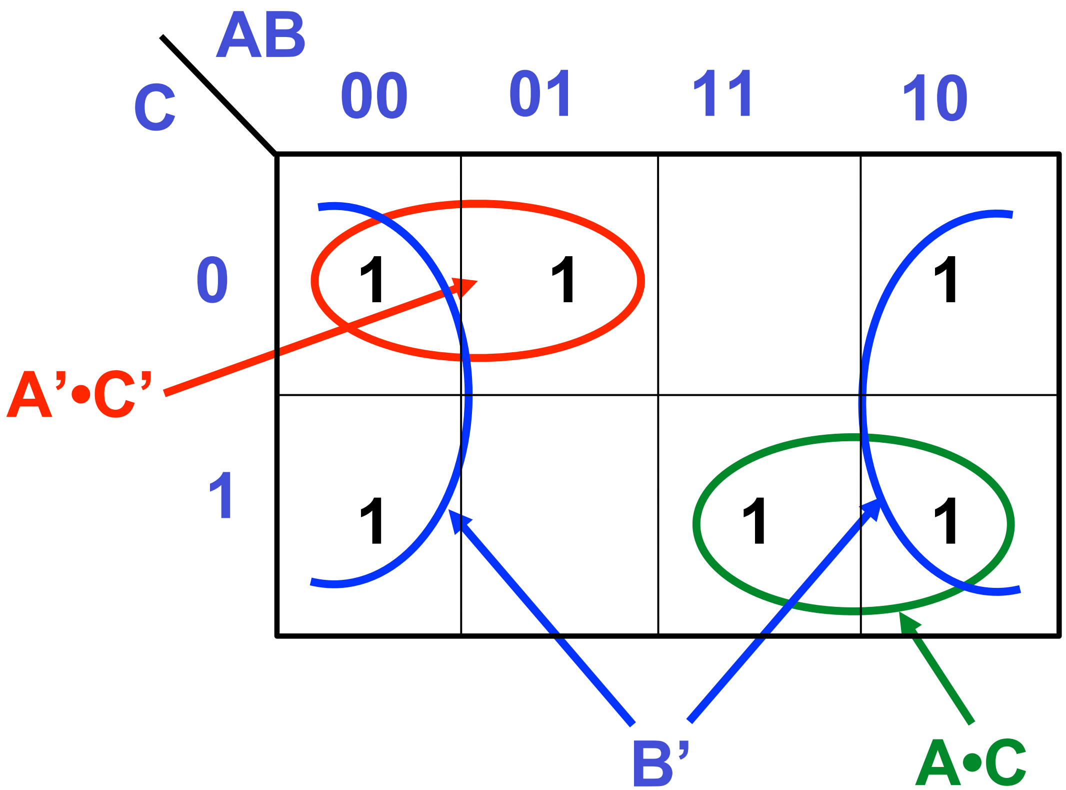
$$F=\Sigma_{A,B,C}(4,5,6,7)$$



F = A

Another Example

$F=\Sigma_{A,B,C}(0,1,2,4,5,7)$



F = A'•C' + B' + A•C

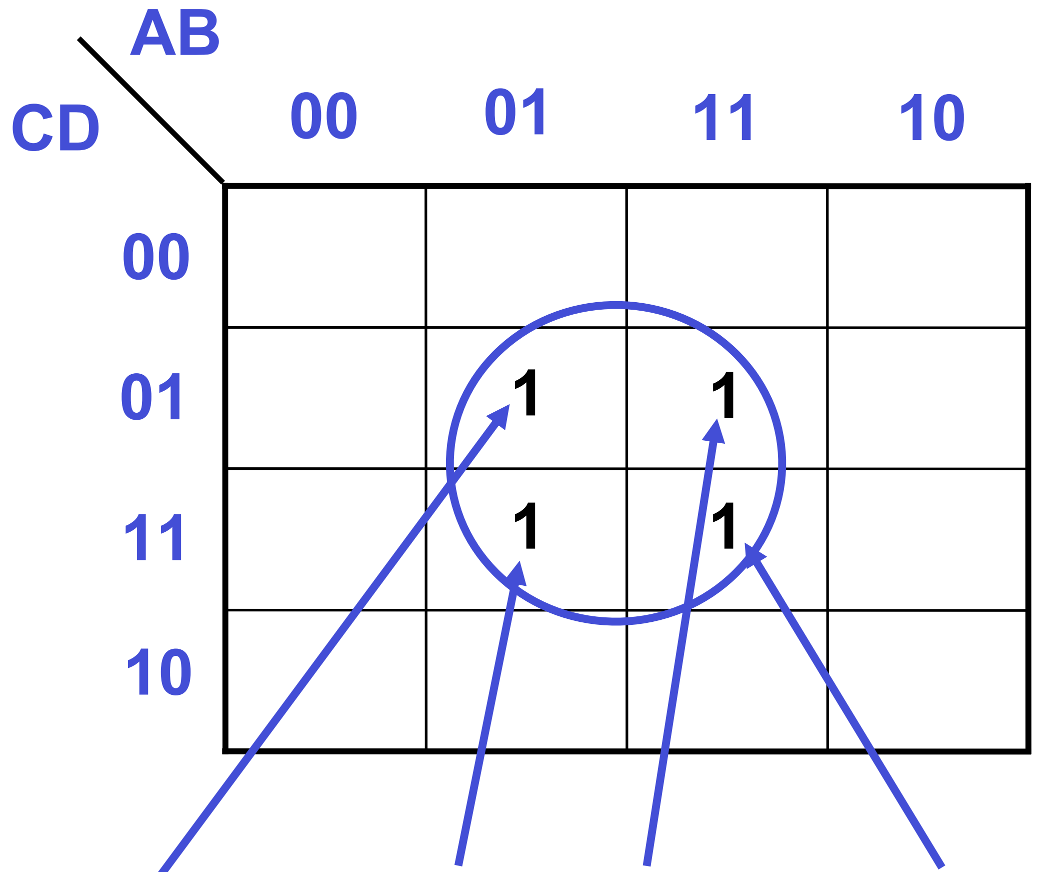
Intuition Behind K-Map

- Spatial encoding!

4 Variable K-Map

CD \ AB	00	01	11	10
00				
01				
11				
10				

Combining Theorem in Action

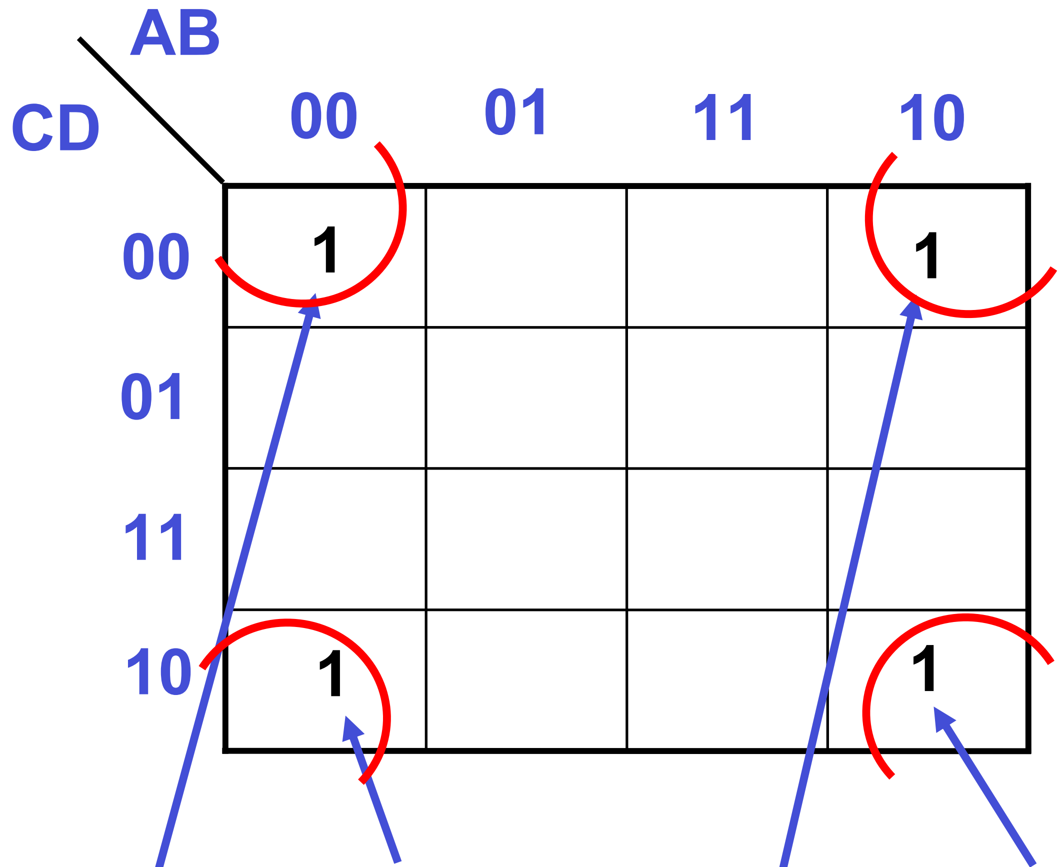


A'BC'D + A'BCD + ABC'D + ABCD

A'BD + ABD

BD

Combining Theorem in Action



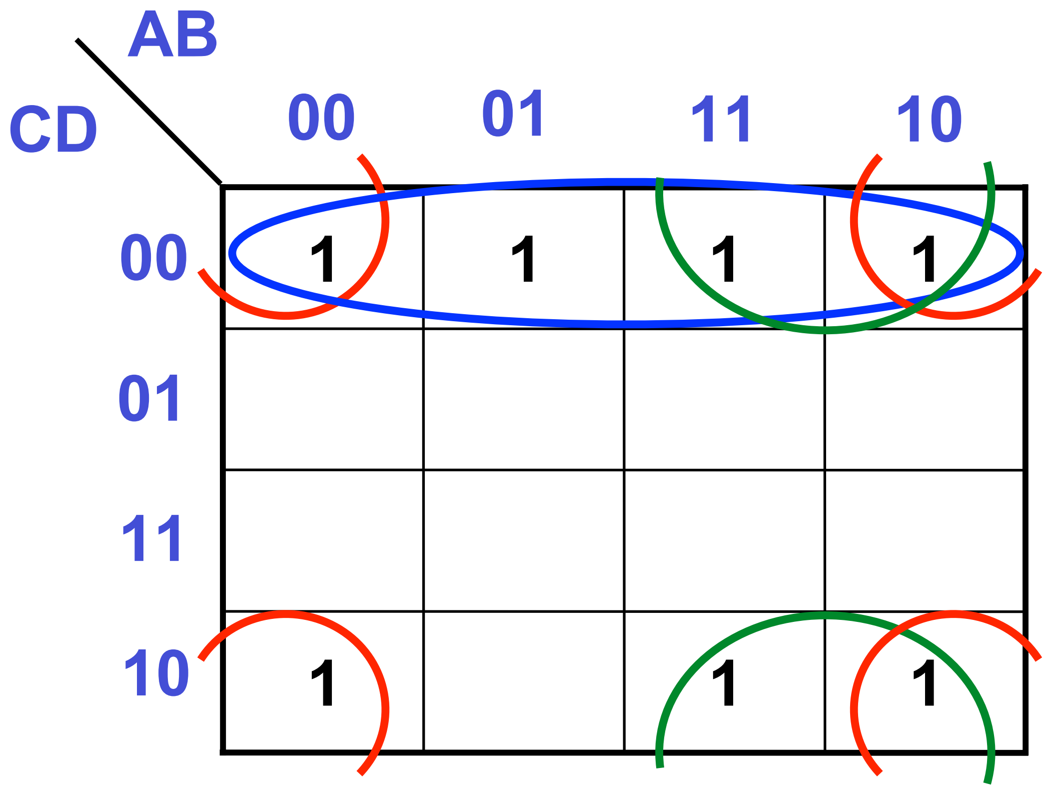
A'B'C'D' + A'B'CD' + AB'C'D' + AB'CD'

A'B'D' + AB'D'

B'D'

4 Variable Karnaugh Map Example

- Detect all even digits from 0 to 15 except 6
  - $F=\Sigma_{A,B,C,D}(0,2,4,8,10,12,14)$



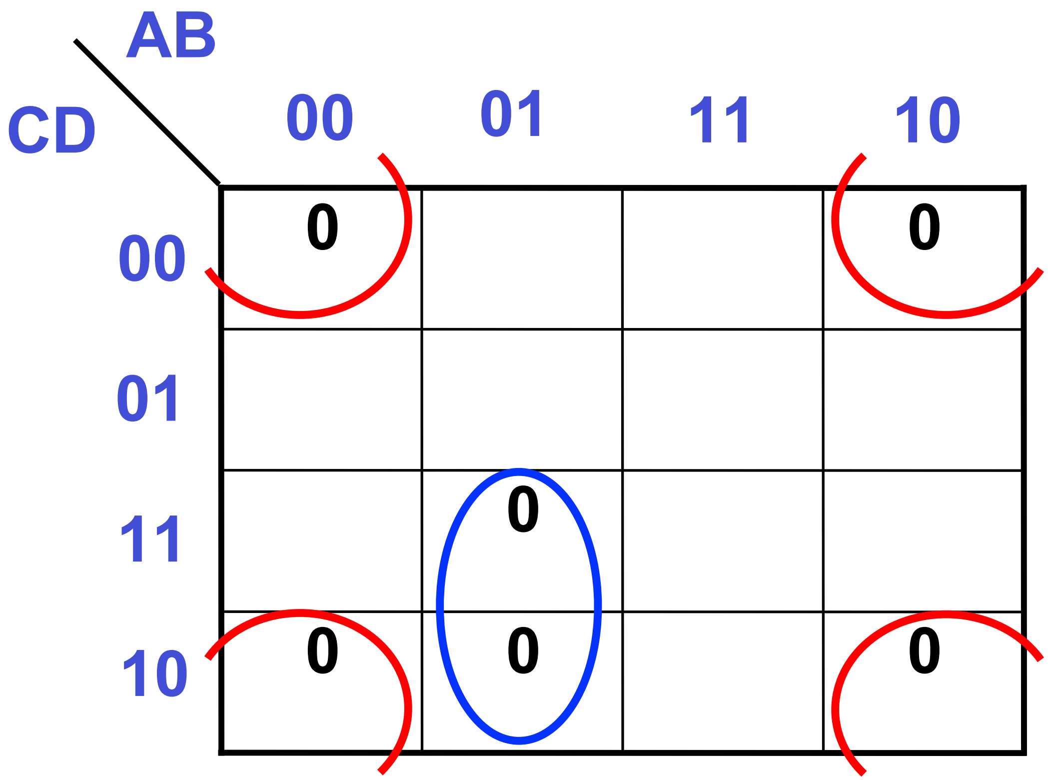
F = B'D' + AD' + C'D'

Minimizing Product-of-Sums

- Procedure
  - Plot 0’s corresponding to maxterms of function
  - Circle largest possible rectangular sets of 0’s
    - Must be power of 2
    - Including “wrap-around” sets
  - Repeat until all maxterms are covered
  - AND sum terms derived from each circle

Product-of-Sums Example

$F=\Pi_{A,B,C,D}(0, 2, 6, 7, 8, 10)$



Corners:
B+D

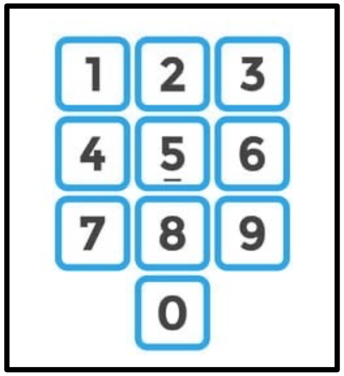
Other:
A+B'+C'

$F=(B+D)\cdot(A+B'+C')$

Don’t Cares Combinations

- Sometimes, for a given input combination, the output is unspecified or irrelevant
  - Such as an input combination that will never occur based on the problem definition
  - We call this input combination a Don’t Care
  - Represent as a ‘d’ (or ‘x’) in the truth table and K-map

- Example: Detect when an odd decimal digit (except 3) is pressed on a keypad
  - Four input bits are still required, but inputs 0-9 would appear
  - 10-15 are “don’t care” values



Don’t Cares in Karnaugh Map

- Don’t cares can be used as 1-cells as needed for minimizing SOP or 0-cells for POS
- Only circle if doing so creates a larger prime implicant (and thus a more minimal expression)
- Example: Detect when an odd decimal digit (except 3) is pressed on a keypad
  - Inputs 10-15 will never occur

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

F = BD + C’D

Next Class

Combinational Building Blocks
(H&H 2.8)