Karnaugh Maps & Combinational Logic Design

ECE 152A – Winter 2012

Reading Assignment

- Brown and Vranesic
  - 4 Optimized Implementation of Logic Functions
    - 4.1 Karnaugh Map
    - 4.2 Strategy for Minimization
      - 4.2.1 Terminology
      - 4.2.2 Minimization Procedure
    - 4.3 Minimization of Product-of-Sums Forms
    - 4.4 Incompletely Specified Functions
    - 4.8 Cubical Representation
      - 4.8.1 Cubes and Hypercubes

Reading Assignment

- Roth
  - 1 Introduction Number Systems and Conversion
    - 1.4 Representation of Negative Numbers
    - 1.5 Binary Codes
  - 4 Applications of Boolean Algebra Minterm and Maxterm Expansions
    - 4.5 Incompletely Specified Functions

Reading Assignment

- <u>Roth</u> (cont)
  - 5 Karnaugh Maps
    - 5.1 Minimum Forms of Switching Functions
    - 5.2 Two- and Three-Variable Karnaugh Maps
    - 5.3 Four-Variable Karnaugh Maps
    - 5.4 Determination of Minimum Expressions Using Essential Prime Implicants
    - 5.5 Five-Variable Karnaugh Maps

Canonical Forms

- The canonical Sum-of-Products (SOP) and Product-of-Sums (POS) forms can be derived directly from the truth table but are (by definition) not simplified
  - Canonical SOP and POS forms are “highest cost”, two-level realization of the logic function
  - The goal of simplification and minimization is to derive a lower cost but equivalent logic function

Simplification

- Reduce cost of implementation by reducing the number of literals and product (or sum) terms
  - Literals correspond to gate inputs and hence both wires and the size (fan-in) of the first level gates in a two-level implementation
  - Product (Sum) terms correspond to the number of gates in the first level of a two-level implementation and the size (fan-in) of the second level gate

Simplification

- Algebraic Simplification
  - Using theorems and properties of Boolean Algebra
    - Difficult with large number of variables and complex Boolean expressions
    - Most often incorporated into CAD Tools
- Karnaugh Maps
  - Graphical representation of logic function suitable for manual simplification and minimization

Two-Variable Karnaugh Map

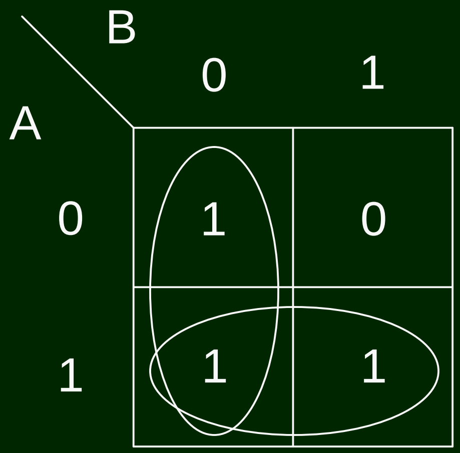
- Location of minterms and maxterms on a two-variable map
  - Index is the same, expansion is complementary

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

A \ B	0	1
0	M_0	M_1
1	M_2	M_3

Two-Variable Karnaugh Map

- Simplification using $xy + xy' = x$ and $x + x'y = x + y$
  - F= Σ *m* (0,2,3)



F = A’B’ + AB’ + AB
F = B’ (A’ + A) + AB
F = B’ + AB
F = (B’ + A) (B’ + B)
F = B’ + A

Three-Variable Karnaugh Map

- Location of three-variable minterms

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

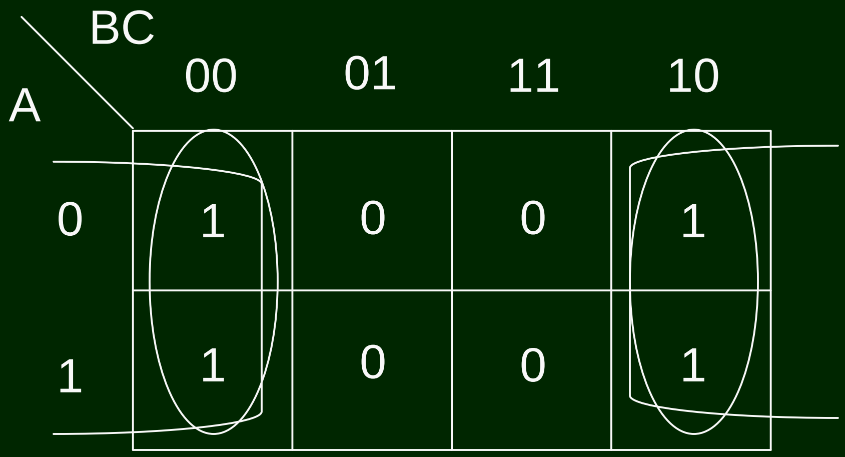
Three-Variable Karnaugh Map

- Adjacent cells differ in the value of only one variable
  - Known as *Gray* coding
  - Topological adjacency equates to algebraic adjacency

$$\begin{tikzcd}
000 \arrow[r] & 001 \arrow[r] & 011 \arrow[r] & 010 \arrow[d] \\
100 \arrow[u] & 101 \arrow[l] & 111 \arrow[l] & 110 \arrow[l]
\end{tikzcd}$$

Three-Variable Karnaugh Map

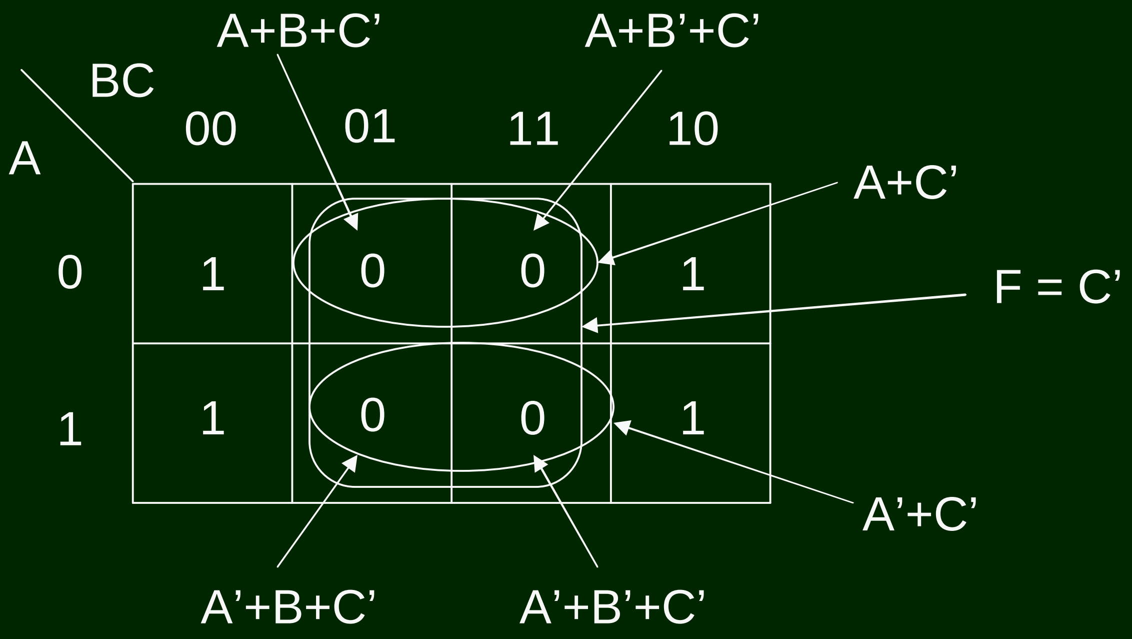
- Three Variable Sum-of-Products Simplification
  - Groupings of 4 (2^2)



F = A’B’C’ + AB’C’ + A’BC’ + ABC’
F = (A’ + A) B’C’ + (A’ + A) BC’
F = B’C’ + BC’
F = (B’ + B) C’
F = C’

Three-Variable Karnaugh Map

- Three Variable Product-of-Sums Simplification
  - Groupings of 4 (2^2)



Four-Variable Karnaugh Map

- Location of four-variable minterms

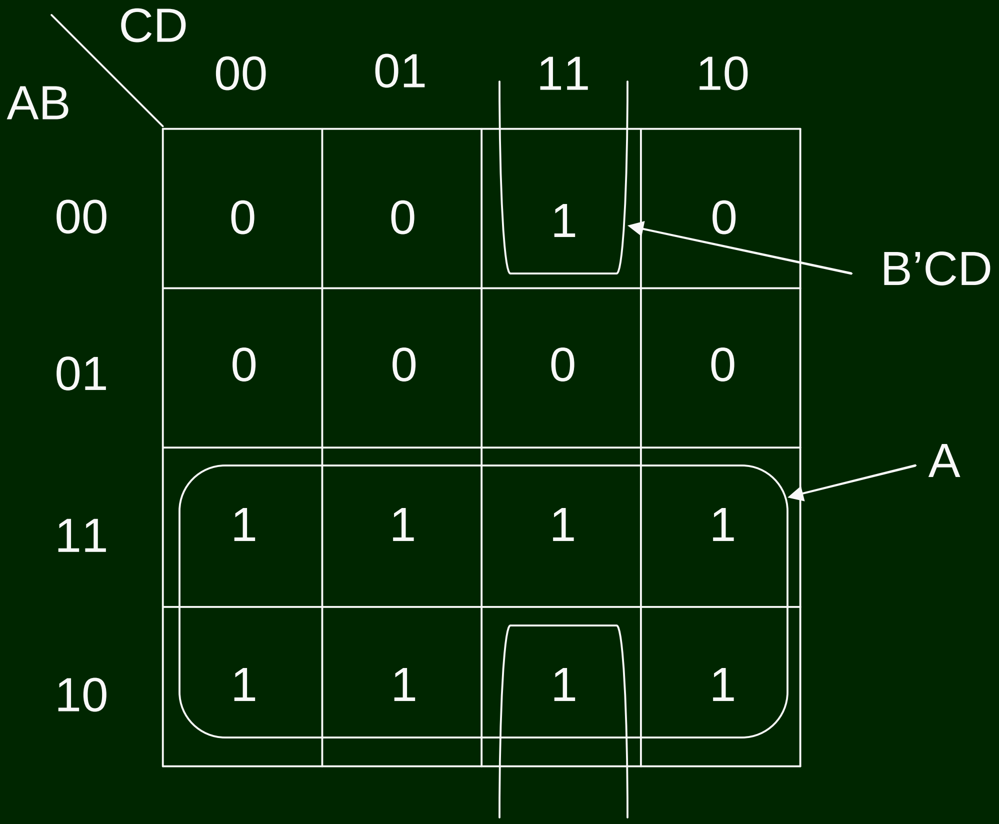
AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Four-Variable Karnaugh Map

- Four-bit Gray code

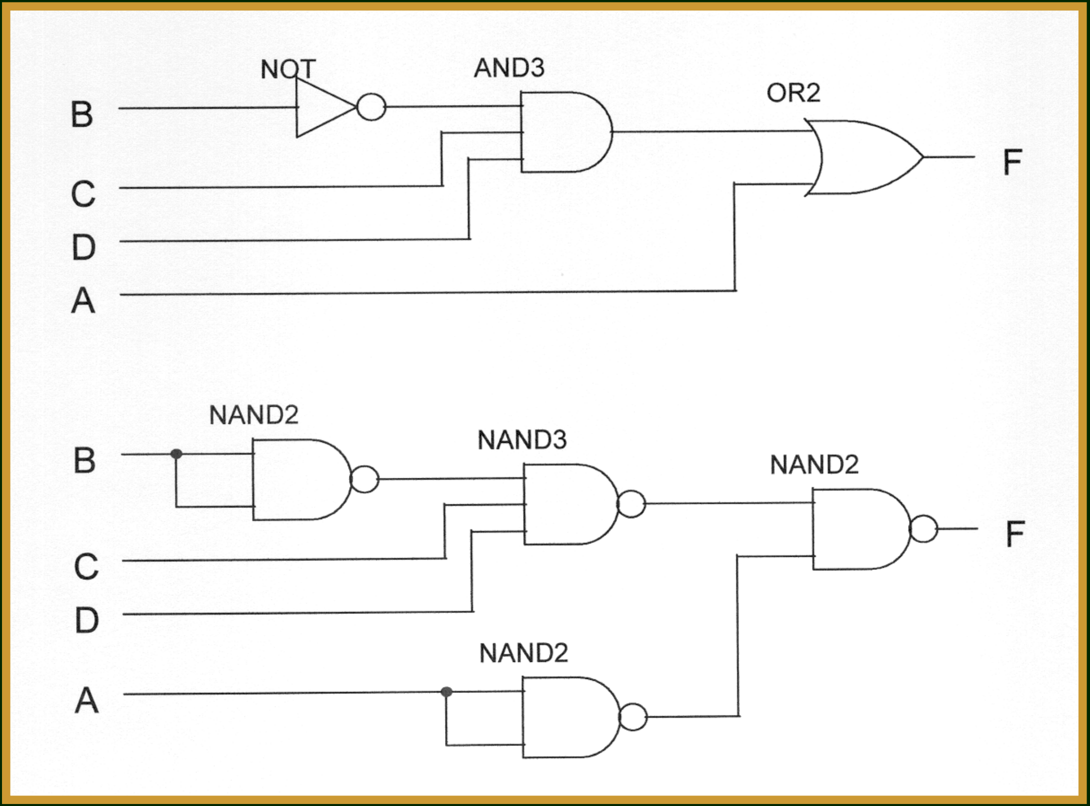
$$\begin{tikzcd}
0000 \arrow[r] & 0001 \arrow[r] & 0011 \arrow[r] & 0010 \arrow[d] \\
0100 \arrow[d] & 0101 \arrow[l] & 0111 \arrow[l] & 0110 \arrow[l] \\
1100 \arrow[r] & 1101 \arrow[r] & 1111 \arrow[r] & 1110 \arrow[d] \\
1000 & 1001 \arrow[l] & 1011 \arrow[l] & 1010 \arrow[l]
\end{tikzcd}$$

Four-Variable Sum-of-Products Map

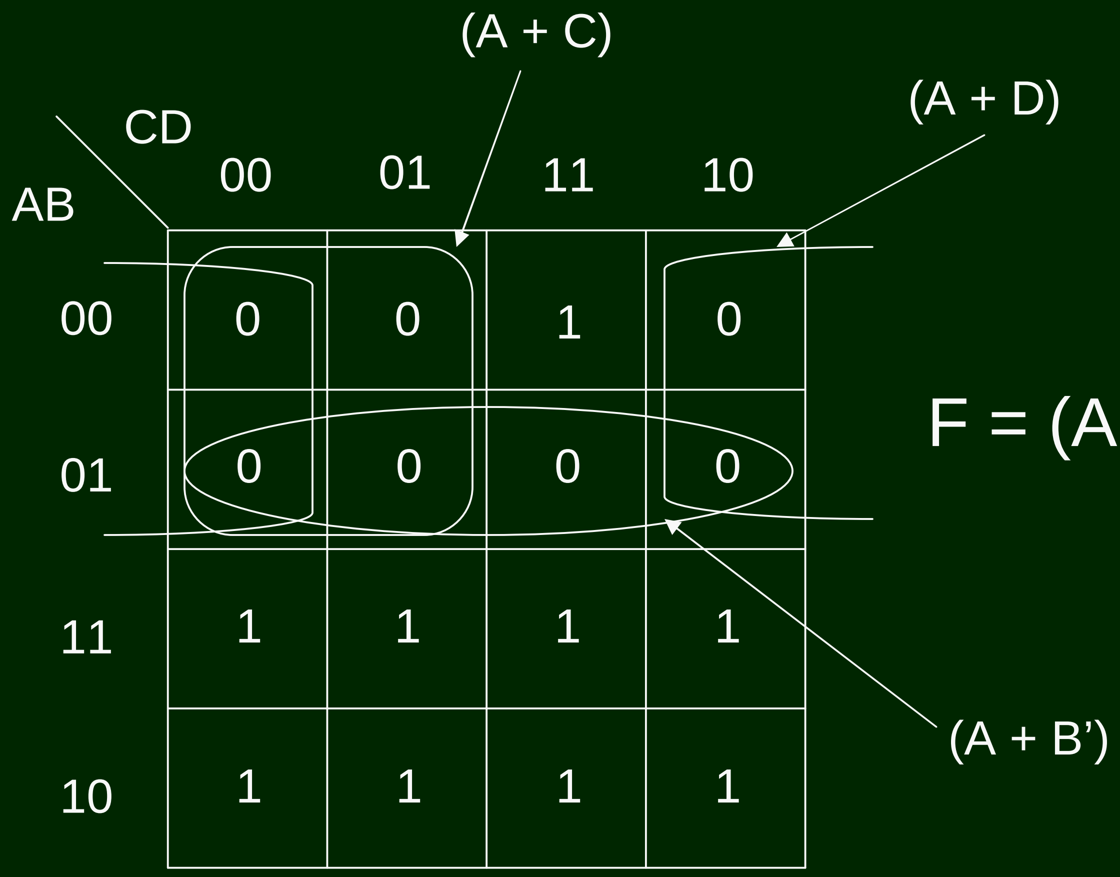


$F = A + B'CD$

Implementation with AND/OR/NOT & NAND gates



Four-Variable Product-of-Sums Map



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

Algebraic conversion between SOP and POS forms

- Multiplying out
  - POS → SOP

F = (A + B’)(A + C)(A + D)

$$\begin{array}{l} A + B' \\ \underline{A + C} \\ AA + AB' \\ \underline{\qquad AC + B'C} \\ A + AB' + AC + B'C \\ A + B'C \\ \underline{A + D} \\ AA + AB'C + AD + B'CD \end{array}$$

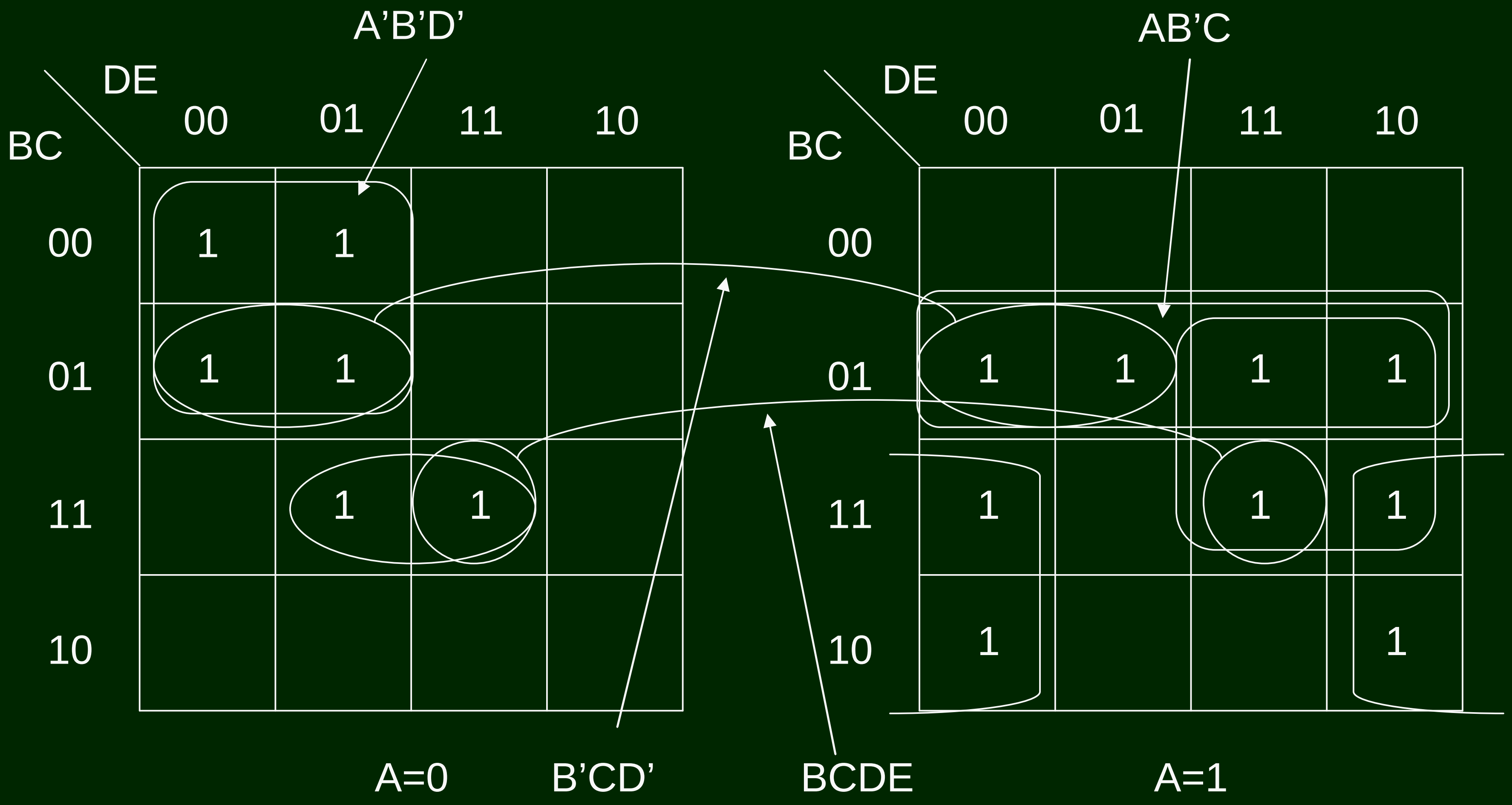
F = A + B’CD

- Factoring
  - SOP → POS

F = A + B’CD
F = (A + B’)(A + CD)

F = (A + B’)(A + C)(A + D)

Five-Variable Karnaugh Maps



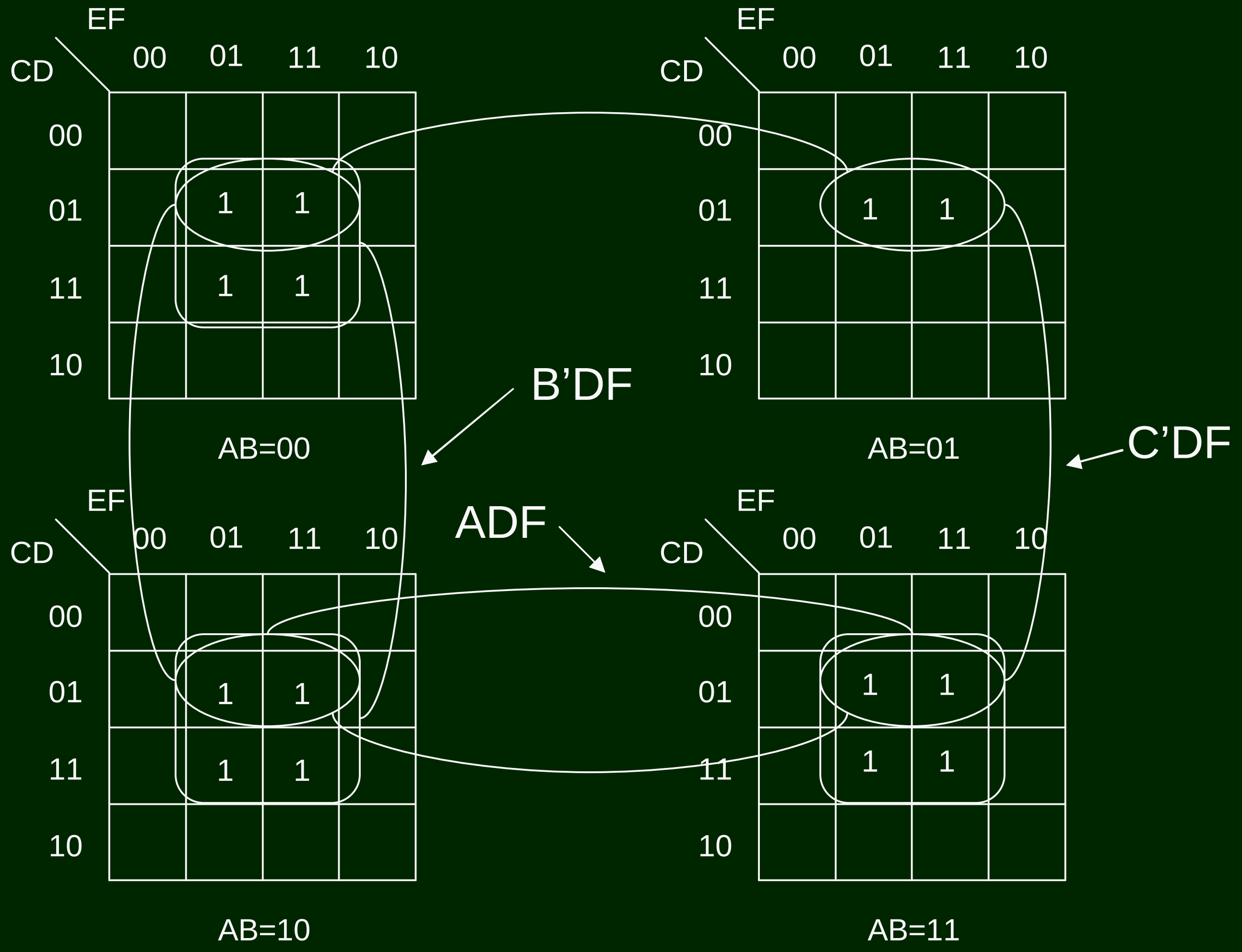
A=0

BC \ DE	00	01	11	10
00	1	1		
01	1	1		
11		1	1	
10				

A=1

BC \ DE	00	01	11	10
00				
01	1	1	1	1
11	1		1	1
10	1			1

Six-Variable Karnaugh Map



Terminology

- Literal
  - *An appearance of a variable or its complement*
- Implicant
  - *Any minterm and/or product term for which the value of the function equals 1 (in SOP form) or any maxterm and/or sum term for which the value of the function equals 0 (in POS form)*

Terminology

- Prime Implicant
  - *An implicant that cannot be combined into another implicant that has fewer literals*
- Essential Prime Implicant
  - *A prime implicant that includes at least one minterm not covered by any other prime implicant*

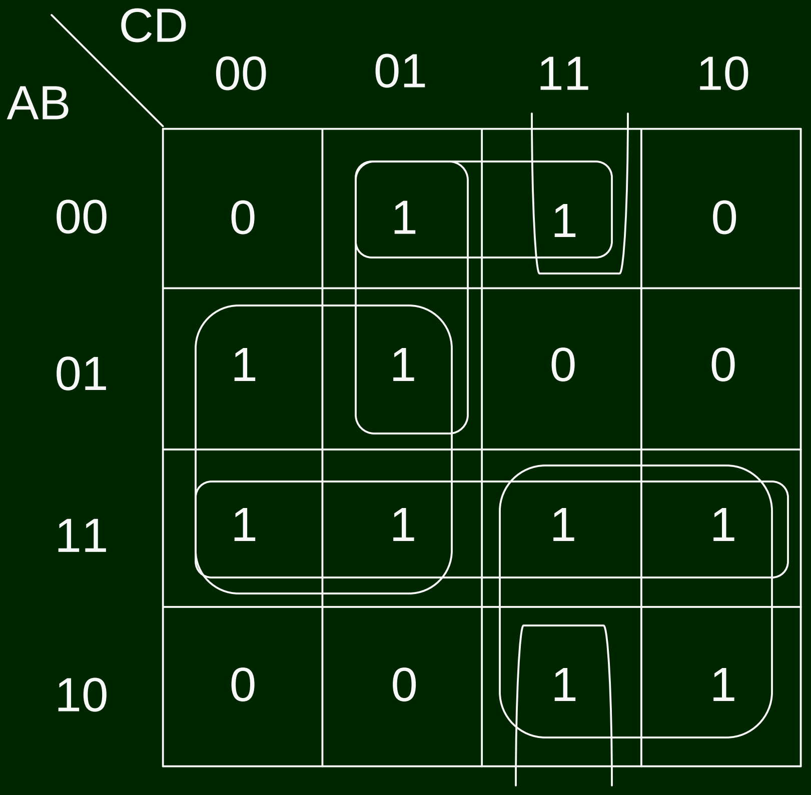
Terminology

- Cover
  - *A collection of implicants that accounts for (covers) all minterms (or maxterms) for which a given function equals 1 in SOP form (or 0 in POS form)*
- Cost
  - *An heuristic figure of merit determined generally from the number of product (sum) terms and the number of literals in a given cover*

Minimization Procedure

- Generate all prime implicants for the given function
- Find the set of all essential prime implicants
- If the set of essential prime implicants covers the function, this set is the desired cover
  - Otherwise, determine the nonessential prime implicants that should be added to form a complete, minimal cost cover

Minimization Example



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

10 Implicants (minterms)
6 Prime Implicants
- BC’, AB, AC,
- B’CD, A’B’D, A’C’D

2 Essential Prime Implicants
- BC’, AC

Final Cover with A’B’D
- F= A’B’D + BC’ + AC

Combinational Logic Circuit Design

- Specify combinational function using
  - Truth Table,
  - Karnaugh Map, or
  - Canonical sum of minterms (product of maxterms)
- This is the creative part of digital design
  - Design specification may lend itself to any of the above forms

Combinational Logic Circuit Design

- Find minimal POS or SOP form of the logic function
  - Technology can determine whether POS or SOP is appropriate solution
  - Nature of function and cost of implementation can determine whether POS or SOP is better solution

Combinational Logic Circuit Design

- Implement design using AND/OR (or NAND) gates or OR/AND (or NOR) gates
  - In most technologies NAND and NOR implementations are superior
    - In terms of both size and speed
- Simulate design and verify functionality and performance
  - Design should always be verified before committing to fabrication

Combinational Design Example 1

- Design Specification
  - Design a logic network that takes as its input a 4-bit, one's complement number and generates a 1 if that number is odd (0 is not odd)
    - Label the inputs A, B, C and D, where A is the most significant bit
  - Implement your design in standard sum-of-products representation using only NAND gates

Combinational Design Example 1

- Recall representation of fixed-point, signed and unsigned numbers from ECE 15A (lecture #14)

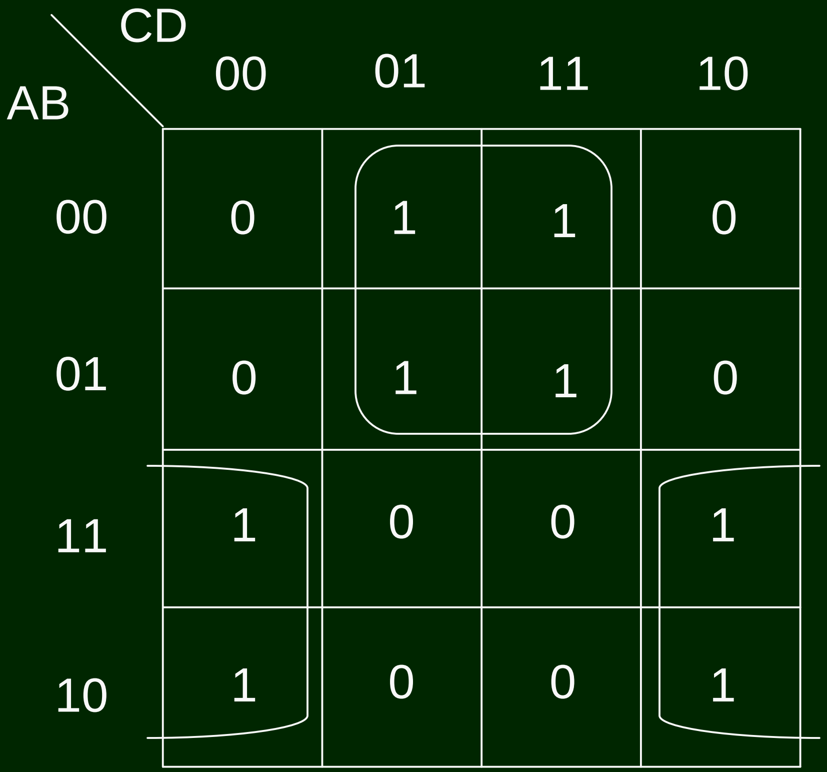
Binary	Unsigned	Sign-Magnitude	One's Complement	Two's Complement
000	0	+0	+0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Design Example 1 – Truth Table

- Odd, One's complement numbers

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

Design Example 1 – Karnaugh Map



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

F = A’D + AD’

Incompletely Specified Functions

- Some logic functions have input combinations that can never occur
  - Examples:
    - Sensors indicating a mutually exclusive event has occurred
    - Processor flags indicating a result was both positive and negative
    - Interlocked switches that can never be closed at the same time

Incompletely Specified Functions

- Conditions called “don’t cares”
  - For minterms/maxterms associated with “don’t care” input combinations, assign output value of 0 or 1 to generate the minimum cost cover
  - On Karnaugh Map, represent “don’t cares” with X and group with minterms (maxterms) to create prime implicants
    - Any X’s not covered can be ignored and will default to 0 (in SOP form) or 1 (in POS form)

Design Example 2

- Design Specification
  - Design a combinational circuit that takes as its input a Binary Coded Decimal (BCD) digit (four bits) and outputs a 1 if the input is an even number (not zero)
  - Recall Binary Coded Decimal representation from ECE 15A
    - Not most economical representation
      - 10 valid combinations per 4 bits
      - 100 valid combinations per byte

Design Example 2

- BCD Example

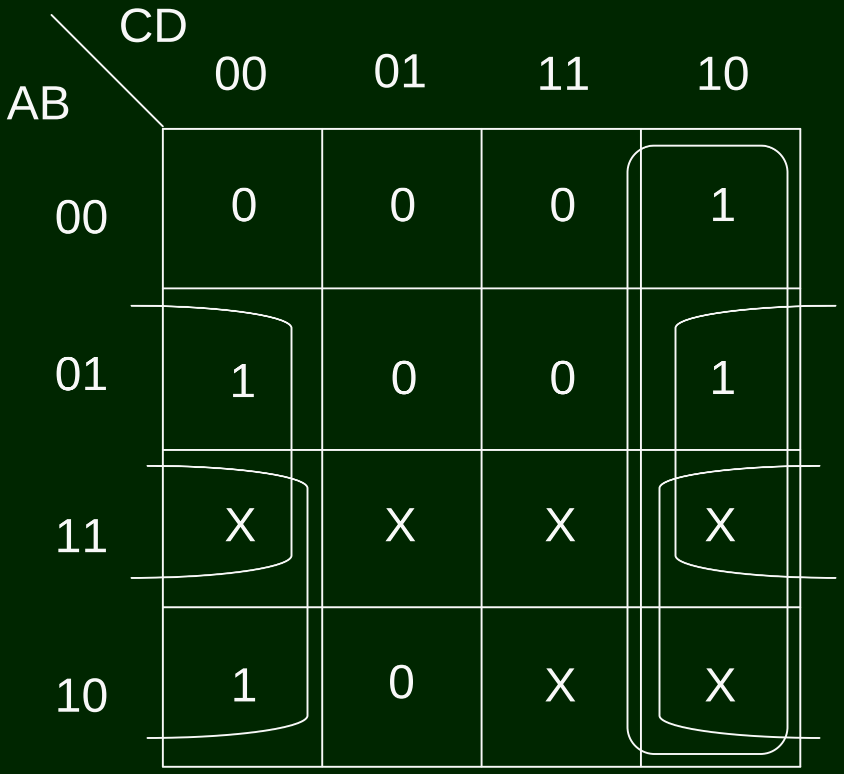
BCD	Value	BCD	Value
0000	0	1000	8
0001	1	1001	9
0010	2	1010	X
0011	3	1011	X
0100	4	1100	X
0101	5	1101	X
0110	6	1110	X
0111	7	1111	X

Design Example 2

- Canonical Forms for Incompletely Specified Functions
  - For design example, function determined directly from design specification
    - Even numbers, not 0

$$\boxed{\begin{array}{l}\sum m(2,4,6,8) + d(10,11,12,13,14,15) \\ \prod M(0,1,3,5,7,9) \bullet d(10,11,12,13,14,15)\end{array}}$$

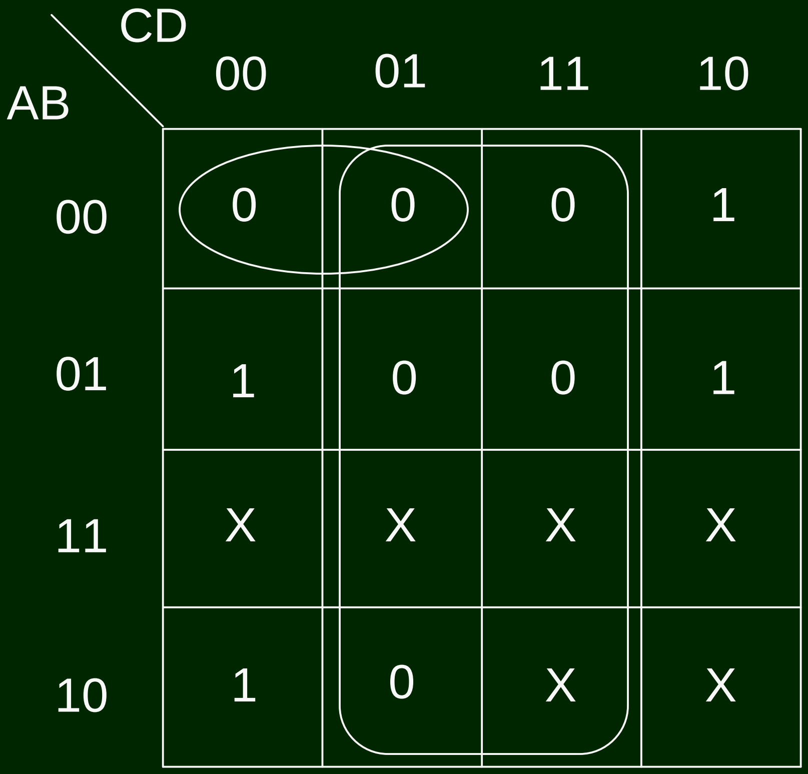
Design Example 2 – SOP Karnaugh Map



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

F = BD’ + AD’ + CD’

Design Example 2 – POS Karnaugh Map



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

F = D’ (A + B + C)

Design Example 3

- Design Specification
  - In this problem, you are to design the combinational circuit that controls the ceiling lights in my downstairs hallway
  - There are three wall switches: one at the front door (A), one at the back door (B) and one in the family room (C)
  - When I walk in the front door, the ceiling lights are off, the A switch is ON and both the B and C switches are OFF
  - From these initial conditions, changing the position of any switch should turn the lights on; changing the position of any switch (again) should turn the lights off, etc.

Design Example 3 – Karnaugh Map

A \ BC	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Initial Conditions:
A=1, B=0, C=0 and F=0

Changing the position of any switch causes the light to come on

Changing the position of any switch again causes the light to go off

And finally…

$$F = A'B'C' + AB'C + A'BC + ABC' = \text{XNOR}(A,B,C)$$

Design Example Review

- From Design Specification to Implementation:
  - Example 1
    - Generate truth table from specification
  - Example 2
    - Generate sum of minterms (product of maxterms) from specification
  - Example 3
    - Generate Karnaugh map from specification