notes w/ annotations

CpE111 Introduction to Computer Engineering

Dr. Minsu Choi

CH 3: Combinational Logic Design

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

What is Combinational Logic Design?

- Combinational logic deals with networks that use logic gates to combine the input variables as needed to produce logic functions -> the value of the output is determined by the current values of the inputs.
- Logic diagrams, truth tables (= function tables) and Boolean expressions are used to represent combinational logic designs.

Canonical Logic Forms

- Two types of structured forms are especially useful in logic design -> Sum-of-Products (SOP) & Product-of-Sums (POS).
- SOP form
- A SOP expression consists of AND terms that are ORed together.
- For a function to be in canonical SOP structure, every variable must appear in each term in either normal or complemented form. otherwise, the function is simply SOP form -> Canonical SOP ≠ SOP.

Example

ex) Suppose that we have variables A, B and C

$f = ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$

$g = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$

are in canonical SOP form.

$F = AB + \bar{A}C + B\bar{C}$ is in sop form, but not canonical SOP form!

$G(a,b,c) = a\bar{b}c + \bar{a}bc + ac$

is not in canonical sop form because of ac.

How to Convert SOP into Canonical SOP?

- Ex) $h(x,y,z) = xy + yz$ we use $(x+\bar{x}=1,\ z+\bar{z}=1)$

$= x \cdot y \cdot 1 + 1 \cdot y \cdot z = x \cdot y(z+\bar{z}) + (x+\bar{x}) \cdot y \cdot z$

$= xyz + xy\bar{z} + xyz + \bar{x}yz$

$= \underline{xyz} + \underline{xy\bar{z}} + \underline{\bar{x}yz}$

Each AND term has all variables.

⇒ Canonical SOP form!

Continued,

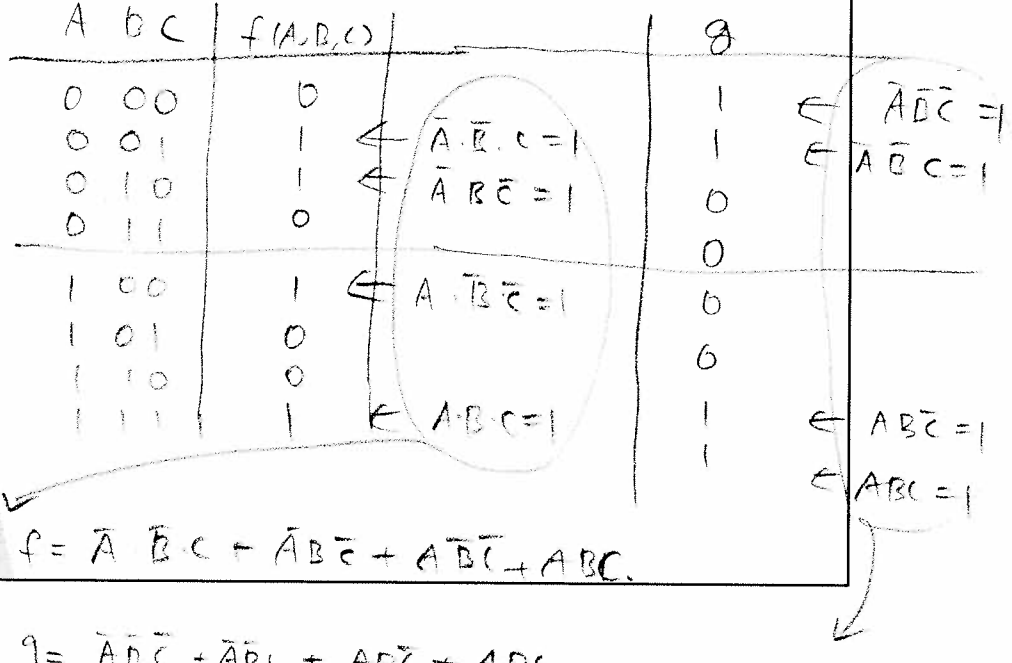
- Product-of-Sum form (POS): A POS express consists of OR terms that are ANDed together.
- Ex) $f(x,y) = (\bar{x}+y) \cdot (x+\bar{y})$ ⇒ canonical POS

$g(x,y) = x \cdot (x+\bar{y})$ ⇒ POS

Extracting Canonical Forms

- Truth table (= function table) -> Boolean expression in canonical SOP.
  1. Select rows with output = 1.
  2. Look up the input bits & construct AND terms.
  3. Then OR them to get the canonical SOP form.

Examples



Minterms & Maxterms

- Easy ways to express C SOPs & C POSs, respectively.
- Ex) Three variable case A, B, C
  1. If complemented -> 0, if not complicated -> 1.
  2. Then find binary #.
  3. Convert it into decimal.

$\bar{A}\bar{B}\bar{C} = m_0$ ($000_2 = 0$)
$\bar{A}\bar{B}C = m_1$
$\bar{A}B\bar{C} = m_2$
$\bar{A}BC = m_3$
$A\bar{B}\bar{C} = m_4$
$A\bar{B}C = m_5$
$AB\bar{C} = m_6$
$ABC = m_7$

Examples

$f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$

$= m_0 + m_3 + m_4 + m_7$

Summation of minterms $= \sum m(0,3,4,7)$

Summation sign sigma

This shorthand notation provides a compact way to express SOP functions.

ex) Let's expand the function $g(a,b,c) = \sum m(1,2,5)$

$= m_1 + m_2 + m_5$

$= \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}c$ #

Continued,

- Maxterm: $M_i = \overline{m_i}$
- Ex) 3-variable case ← DeMorgan theorem

$M_0 = \overline{m_0} = \overline{\bar{A}\cdot\bar{B}\cdot\bar{C}} = A+B+C$

$M_1 = \overline{m_1} = A+B+\bar{C}$ → (If complimented = 1, If not complimented = ϕ) in this case.

$M_7 = \overline{m_7} = \bar{A}+\bar{B}+\bar{C}$

- Ex)

$g = (A+B+\bar{C})\cdot(A+\bar{B}+\bar{C})\cdot(\bar{A}+\bar{B}+\bar{C})$

$= M_1 \cdot M_3 \cdot M_7 = \prod M(1,3,7)$

product sign p_i

SOP <-> POS <-> Truth Table

- Ex) 3-var case

$F = m_1 + m_2 + m_5 + m_6$

$= \Sigma m(1,2,5,6)$ (SOP) easy →

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

(T T)

(POS) we should look for entries where the minterms are ϕ.

$M_0 = \overline{m_0}$ $M_3 = \overline{m_3}$ $M_4 = \overline{m_4}$ $M_7 = \overline{m_7}$

So, $F = \prod(0,3,4,7)$

$M_i = \overline{m_i}$

① Select rows with output = ϕ where maxterm = 1 ⇐ minterm = ϕ

② Find maxterm expression.

Exclusive-OR & Equivalence Operation

- When only a single input is 1 exclusively the output is 1.
- XOR and XNOR

XOR2 → XOR (read "O-plus")

A B	A⊕B
0 0	0
0 1	1
1 0	1
1 1	0

XOR symbol

⇒ SOP

$A \oplus B = A\bar{B} + \bar{A}B$

POS?

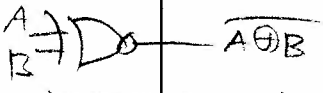
$A \oplus B = (A+B)\cdot(\bar{A}+\bar{B})$

$XNOR = \overline{A \oplus B} = AB + \bar{A}\bar{B}$ in SOP

⇒ $\overline{A \oplus B} = 1$ iff $A = B$

⇒ XNOR is referred to as the equivalence function.

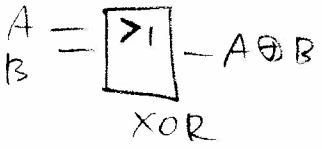
A B	A⊙B
0 0	1
0 1	0
1 0	0
1 1	1



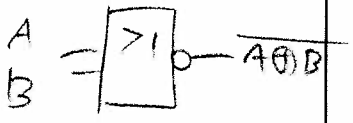
XNOR symbol.

IEEE Symbols

regular OR function "≥"



XOR

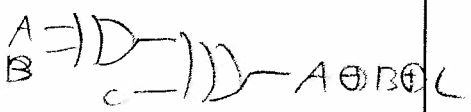


- 3 input and higher circuits -> cascading.

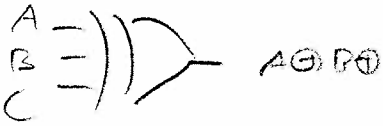
ex) $A \oplus B \oplus C$ 1 = 0 if there are an even # of 1s at the input
= 1 " odd "

↳ So called odd-function

A B C	A⊕B⊕C
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1



=

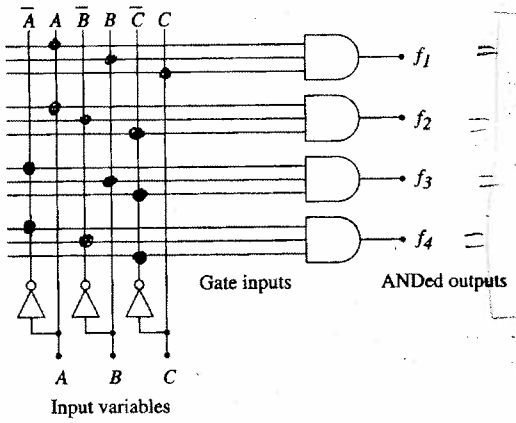


its even function?...

$\overline{A \oplus B \oplus C}$ 1 = 1 if there are even # of 1s
= 0 " odd # "

Logic Arrays

- Structured networks that can be configured to produce specific forms of logic expressions.
- AND array (minterms are used to program it)
- Uncommitted AND logic array -> programming (= make connections accordingly)



connection.
minterms we want to program

$m_7 = A \cdot B \cdot C$

$m_4 = A \cdot \bar{B} \cdot \bar{C}$

$m_2 = \bar{A} \cdot B \cdot \bar{C}$

$m_0 = \bar{A} \cdot \bar{B} \cdot \bar{C}$

Continued,

- OR array (produces maxterms)
- Ex)

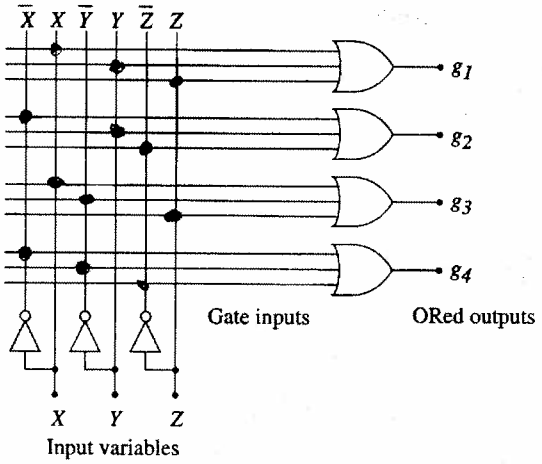
$f_1 = X + Y + Z = M_0$

$g_2 = \bar{X} + Y + \bar{Z} = M_5$

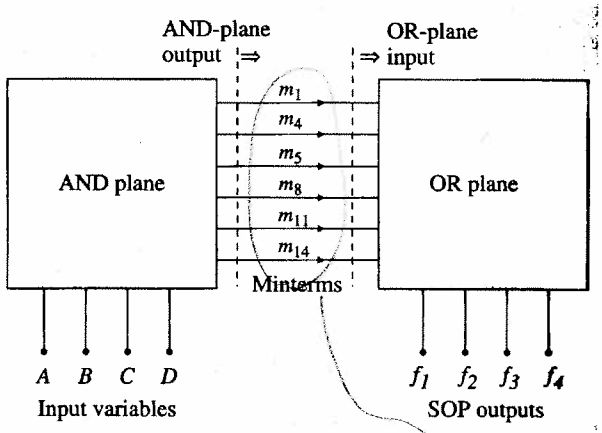
$g_3 = X + \bar{Y} + Z = M_2$

$g_4 = \bar{X} + \bar{Y} + \bar{Z} = M_7$

- Combine AND & OR array -> SOP & POS arrays.



AND-OR PLA (Programmable Logic Array) for SOP



⇒ SOP $f_n = \sum m_i$

OR-AND PLA for POS

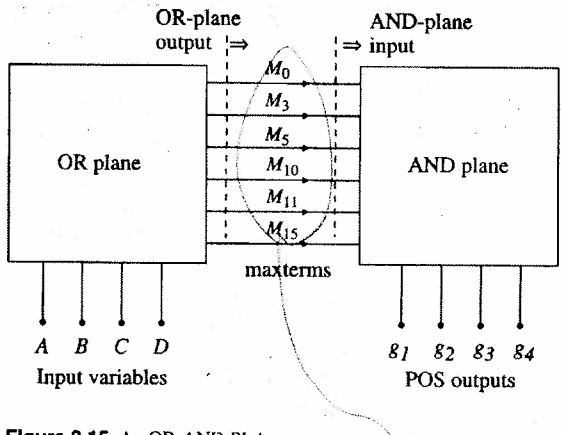
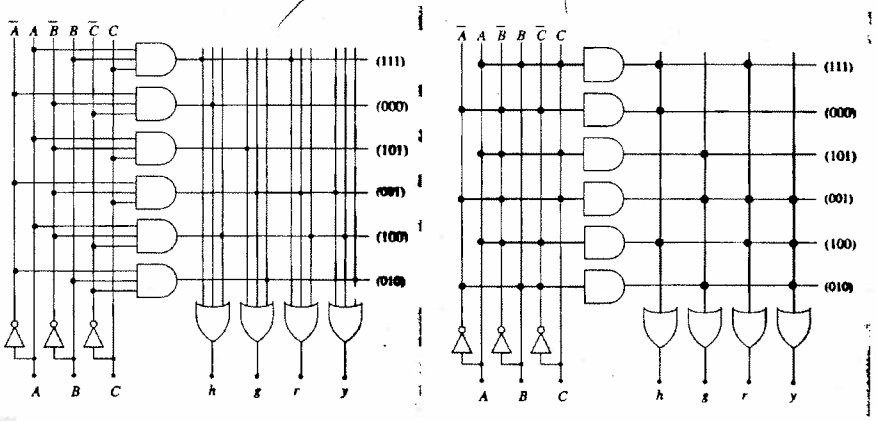


Figure 3.15 An OR-AND PLA

⇒ POS $g_m = \prod M_j$

these are equivalent!
(the second one is simplified version)

Example



$h = ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

$g = A\bar{B}C + \bar{A}BC + A\bar{B}\bar{C}$

$r = ABC + \bar{A}BC + A\bar{B}\bar{C}$

$y = \bar{A}BC + A\bar{B}C + \bar{A}\bar{B}\bar{C}$

=> final Boolean functions from the given programmed PLA

Pros and Cons of Logic Arrays

- Pros: Rapid implementation & prototyping of complex digital networks.
- Cons: Resulting circuit will probably not be the most efficient use of gates and the design itself will not be the fastest implementation that can be achieved.
- Complex PLA-based programmable devices are called PLDs (Programmable Logic Devices).
- Good example of PLD: FPGAs (Field-Programmable Gate Arrays) -> very powerful logic circuits that can be used to implement highly complex logic networks.
- CAD tools are used to implement and program custom logic networks on FPGAs.

good example of combinational logic design!

BCD & 7-Segment Display

- Binary-Coded Decimal (BCD) is a binary counting system for the base-10 digits 0 through 9 -> 4 bits required & A, B, C, D denote individual bits.

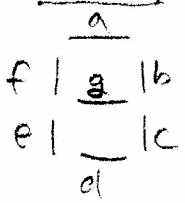
ABCD	Decimal	ABCD	Decimal
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

⇒ 4 bit binword to express single dec digit.

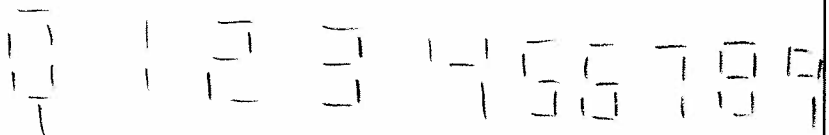
- Binary combinations 1010 through 1111 are not used.

Ex) An application of BCD: BCD to 7-segment decoder.

- 7-segment display: a common type of numerical display that usually uses 7 LEDs (Light-Emitting Diodes) to represent decimal digits.

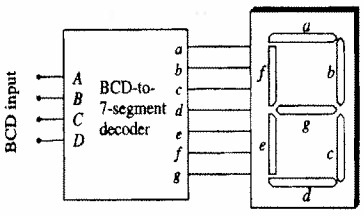


Formulation of base-10 digits using a 7-seg display



(ex) abcdefg are ON & g is off.

BCD-to-7-Segment Decoder Function



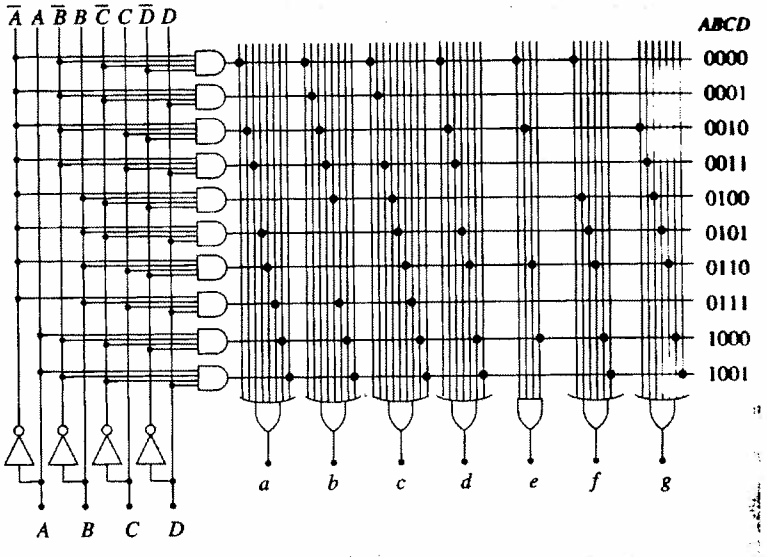
ABCD	Digit	a	b	c	d	e	f	g
0000	0	1	1	1	1	1	1	0
0001	1	0	1	1	0	0	0	0
0010	2	1	1	0	1	1	0	1
0011	3	1	1	1	1	0	0	1
0100	4	0	1	1	0	0	1	1
0101	5	1	0	1	1	0	1	1
0110	6	1	0	1	1	1	1	1
0111	7	1	1	1	0	0	0	0
1000	8	1	1	1	1	1	1	1
1001	9	1	1	1	0	0	1	1

- -> decoder has 7 outputs -> 7 functions
  1. a = a(A,B,C,D)
  2. b = b(A,B,C,D) ....
- SOP expressions?

Find minterms from the column a then OR them up.

$a = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D$

PLA Implementation



minterms

m_0

m_9

Karnaugh Maps (= K-maps)

- Canonical SOP & POS forms can be simplified, but the types of gates and their placement in the logic network will be "random" in that they cannot be predicted -> This design technique is called random logic -> More systematic way? K-maps.
- Karnaugh maps allow us to simply Boolean functions using a visual mapping technique that helps us recognize Boolean reductions by their locations on a grid.
- The technique of K-maps relies on the following two identities:

$$\begin{cases} A + \bar{A} = 1 \\ 1 \cdot X = X \end{cases}$$

ex) $(A + \bar{A}) \cdot X = 1 \cdot X = X$

- where X is any group of logic variables.

Continued,

- Ex)

$$g(A,B,C) = \underbrace{ABC + AB\bar{C}}_{} + \underbrace{A\bar{B}C + A\bar{B}\bar{C}}_{}$$

$$= AB(C + \bar{C}) + AC(\bar{B} + B)$$

$$= AB + AC = A(B + C)$$

↳ has only two gates

- -> K-maps can do reductions in a systematic way.

1. Start with a function table.
2. Map the input-output combinations to a rectangular grid array.
3. Locate the terms where the identity $(x + \bar{x}) = 1$ can be used to simply the function.

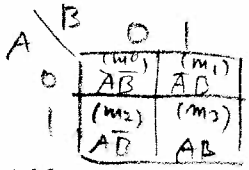
Continued,

- K-maps can be applied to functions with arbitrary # of variables.
- Only 2, 3, and 4-variable cases will be discussed.
- For more # of variables, computer programs can be used.

(Since they are computationally complex).

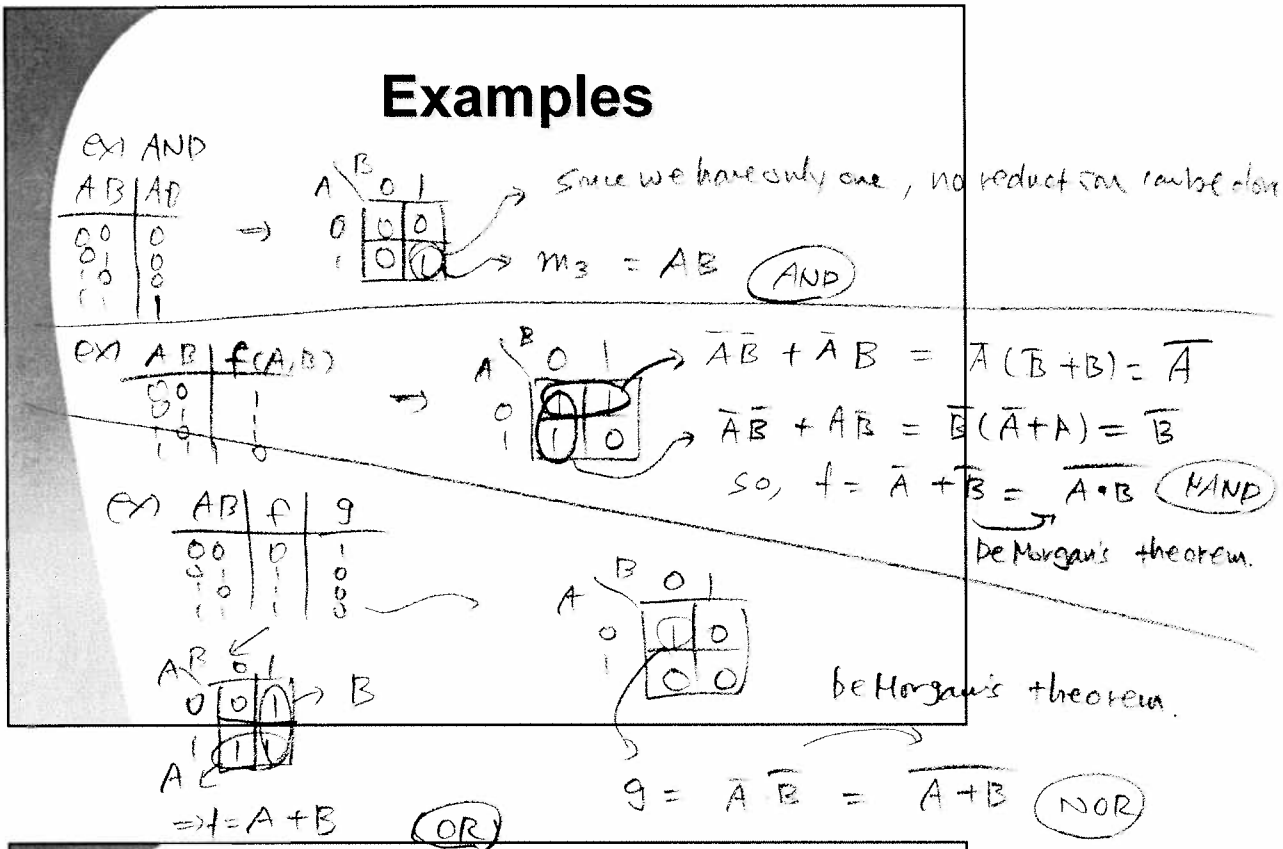
2-Variable K-maps

- Express the function in grid-like table.
- List all possible minterms & the resulting output value of the function -> $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and AB
- Construct a basic map.



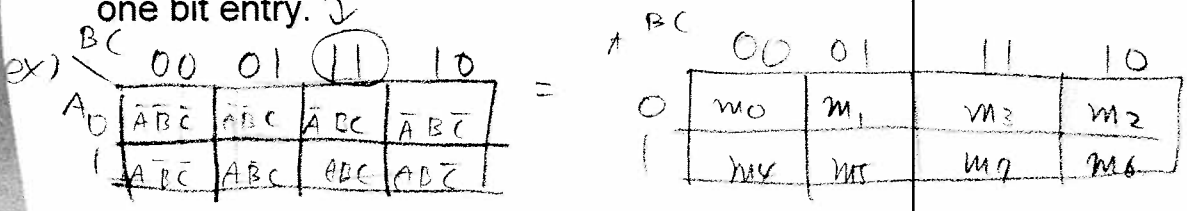
- Lookup truth table & construct K-map.
- Locate groups of 1, 2 or 4 adjacent 1s and simplify; each group called "one cell".

Examples



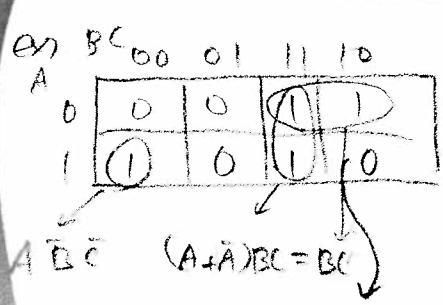
3-Variable K-Maps

- Create a map using the variable A with B,C (or A,B with C) so that adjacent boxes differ by only one bit entry.



- 1, 2, 4 or 8 adjacent 1s can be grouped and reduced.
- Left and right edges are also adjacent; allowing us to "wrap" the map into a cylinder.

Examples



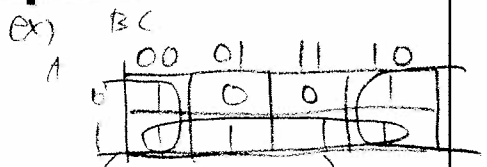
$A\bar{B}\bar{C}$ $(A+\bar{A})BC = BC$

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

$f = A\bar{B}\bar{C} + BC + \bar{A}B$

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

further algebraic reduction, if possible.



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

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

$g = A + \bar{C}$

Summary of Simplification Rules

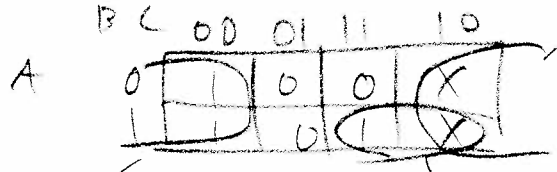
- A group of one minterm gives a term with all three factors A, B and C.
- A group of two minterms reduces to a term with two factors.
- A group of four minterms reduces to a term with one factor.
- A group of all eight minterms is equivalent to a logic 1.

1s should be grouped in order that reduction rule $(A+\bar{A})\cdot C = C$ can be used.

# of groups should be minimized (≙ 1s in each group should be maximized)

"Don't Care" Conditions

- The output produced by a particular set of inputs can be either 0 or 1 without affecting the behavior of the function -> called "don't care" condition & denoted by X.
- Ex)



considered as 1s.

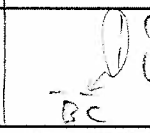
$(A+\bar{A})\cdot(B+\bar{B})\,\bar{C} = \bar{C}$

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

$f = A\cdot B + \bar{C}$

better

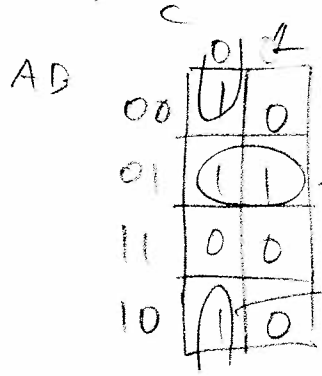
If considered as 0's



$f = AB\bar{C} + \bar{B}\cdot\bar{C}$

Alternative 3-Variable Layout

Group A & B rather than B & C



← Top & bottom edges are adjacent in this case.

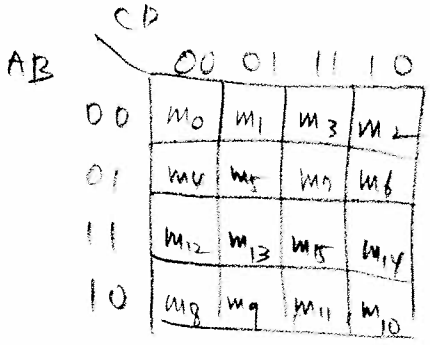
$\bar{A}B(C+\bar{C}) = \bar{A}B$

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

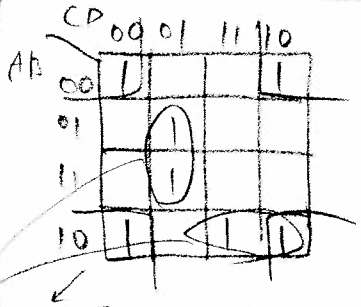
$f = \bar{A}B + \bar{B}\bar{C}$

4-Variable K-Maps

- Group A,B and C,D to draw a K-map.
- Group 1, 2, 4, 8 or 16 adjacent entries of 1s.
- Top-down and left-right edges are adjacent.



Examples



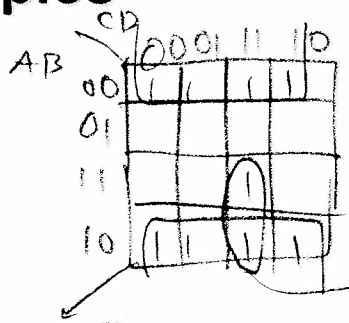
$(A+\bar{A})\bar{B}(C+\bar{C})\bar{D} = \bar{B}\bar{D}$

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

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

$f = B\bar{C}D + A\bar{B}C + \bar{B}\bar{D}$

$= B\bar{C}D + \bar{B}(AC + \bar{D})$



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

$A(B+\bar{B})CD$

$= ACD$

$g = ACD + \bar{B}$


Program Completed
University of Missouri-Rolla
© 2003 Curators of University of Missouri
UMR
UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.