CpE2210 Introduction to Digital Logic

Dr. Minsu Choi

CH 4: Combinational Logic Design

MISSOURI S&T

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

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.

MISSOURI S&T

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

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.

$F = AB + \bar{A}\bar{c} + B\bar{c}$ 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 ↙.

How to Convert SOP into Canonical SOP?

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

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

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

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

Each AND term has all variables.

$\Rightarrow$ Canonical SOP form!

Continued,

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

$g(x,y) = x \cdot (x+\bar{y}) \Rightarrow$ 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

A	B	C	f(A,B,C)		g	
0	0	0	0		1	$\bar{A}\bar{B}\bar{C}=1$
0	0	1	1	$\bar{A}\cdot\bar{B}\cdot C=1$	1	$\bar{A}\bar{B}C=1$
0	1	0	1	$\bar{A}B\bar{C}=1$	0	
0	1	1	0		0	
1	0	0	1	$A\cdot\bar{B}\bar{C}=1$	0	
1	0	1	0		0	
1	1	0	0		1	$AB\bar{C}=1$
1	1	1	1	$A\cdot B\cdot C=1$	1	$ABC=1$

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

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

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 $= \Sigma m(0,3,4,7)$

summation sign sigma

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

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

$= m_1 + m_2 + m_5$

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

Continued,

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

$\vdots$

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

- 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

pi

SOP <-> POS <-> Truth Table

- Ex) 3-var case easy. →

$F = m_1 + m_2 + m_5 + m_6$

$= \Sigma m(1,2,5,6)$

SOP

T.T.

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

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.

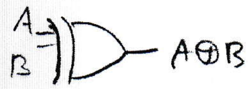
② Find maxterm expression.

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	$A \oplus B$
0 0	0
0 1	1
1 0	1
1 1	0



XOR symbol

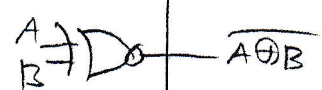
⇒ SOP
$A \oplus B = A\bar{B} + \bar{A}B$
POS?
$A \oplus B = (A+B)\cdot(\bar{A}+\bar{B})$

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

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

⇒ XNOR is referred to as the equivalence function.

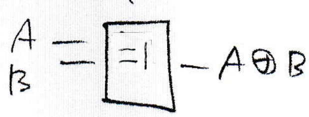
A B	$\overline{A \oplus B}$
0 0	1
0 1	0
1 0	0
1 1	1



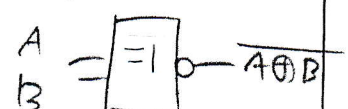
XNOR symbol.

"equals to 1"

IEEE Symbols



XOR



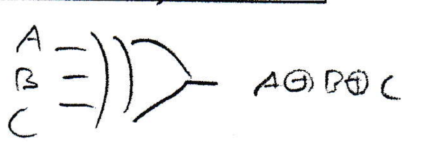
- 3 input and higher circuits -> cascading.

ex) $A \oplus B \oplus C$ $\begin{cases} = 0 & \text{if there are an even \# of 1's at the input} \\ = 1 & \text{" odd "} \end{cases}$

→ So called $\boxed{\text{odd-function}}$

ABC	$A \oplus B \oplus C$
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

=



its even function?...

$\overline{A \oplus B \oplus C}$ $\begin{cases} = 1 & \text{if there are even \# of 1s.} \\ = 0 & \text{" odd \# "} \end{cases}$

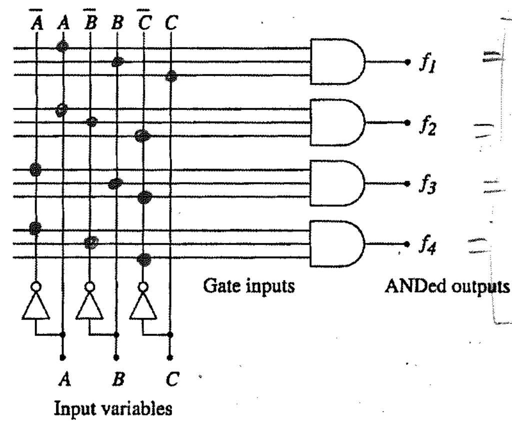
Logic Arrays

- 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)

ex)

+ → no connection

✚ ← 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,

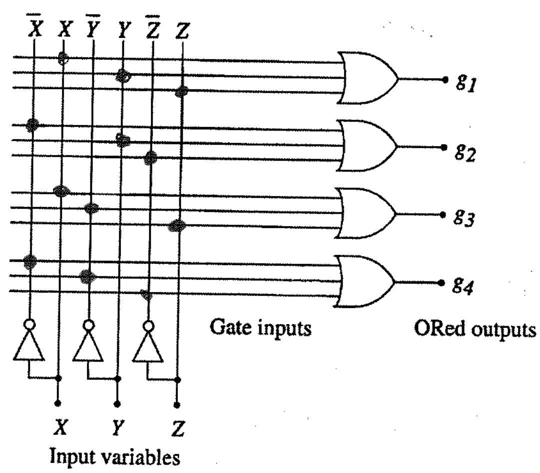
- OR array (produces maxterms)
- Ex)

$g_1 = X + Y + Z = M_0$

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

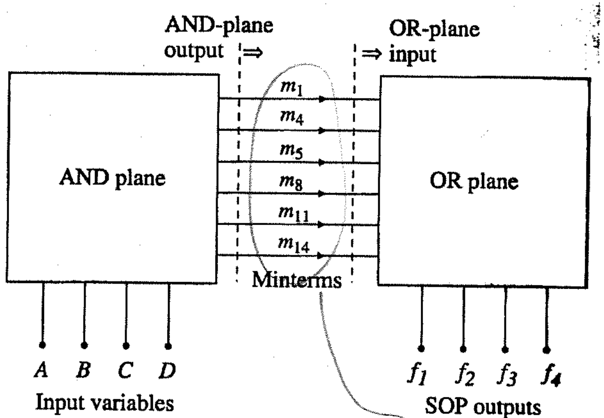
$g_3 = X + \bar{Y} + Z = M_2$

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



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

AND-OR PLA (Programmable Logic Array) for SOP



⇒ SOP $f_n = \sum_i m_i$

OR-AND PLA for POS

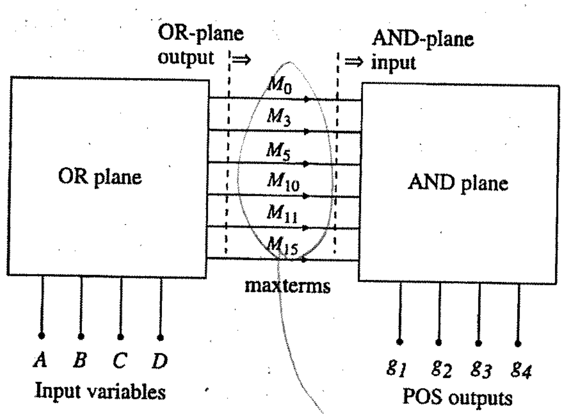
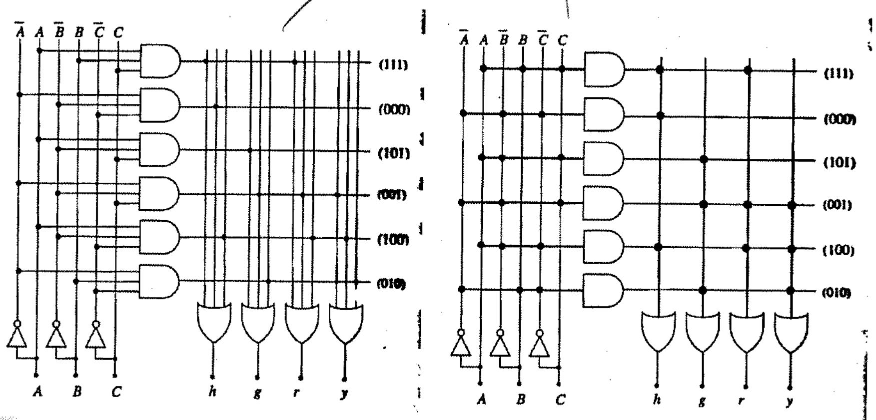


Figure 3.15 An OR-AND PLA

⇒ POS $g_m = \prod_j M_j$

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

Example



=> Final Boolean functions from the given programmed PLA.

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

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

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

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

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.

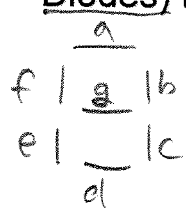
ABCD	Decimal	ABCD	Decimal
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

⇒ 4 bit binword to express single Dec digit.

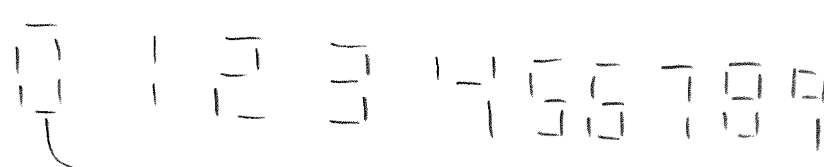
- Binary combinations 1010 through 1111 are not used.

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.

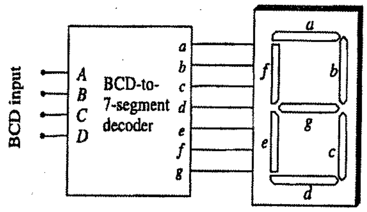


Formulation of base-10 digits using a 7-seg display



ex) abcdef 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

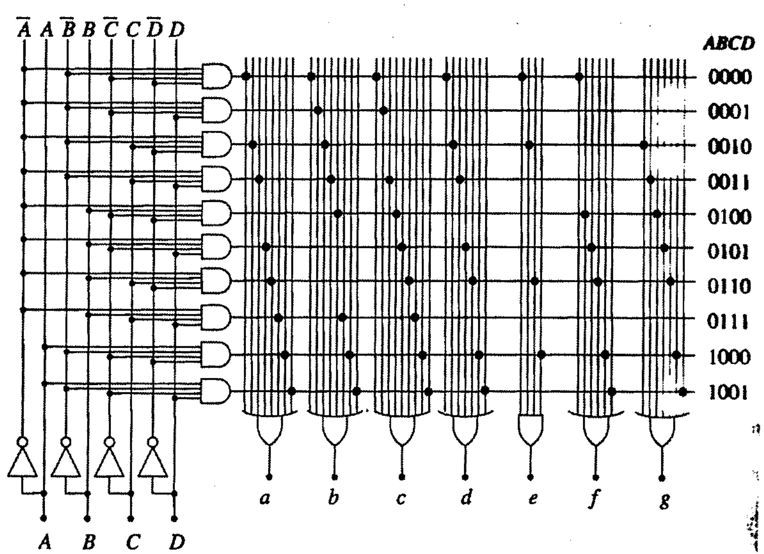
- -> decoder has 7 outputs -> 7 functions
  1. a = a(A,B,C,D)
  2. b = b(A,B,C,D) ....
- SOP expressions?

⊗ Find minterms from the column for a then OR them up.

$$a = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D$$

PLA Implementation

minterms



m_0
m_1
⋮
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: $A+\bar{A}=1$, $1\cdot X=X$ ex) $(A+\bar{A})\cdot X = 1\cdot X = X$
- where X is any group of logic variables.

Continued,

- Ex)

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

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

$$= AB + AC = \underbrace{A(B+C)}_{\text{has only two gates}}$$

- -> 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.

A \ B	0	1
0	(m_0) $\bar{A}\bar{B}$	(m_1) $\bar{A}B$
1	(m_2) $A\bar{B}$	(m_3) AB

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

Examples

ex) AND

A B	AND
0 0	0
0 1	0
1 0	0
1 1	1

A \ B	0	1
0	0	0
1	0	1

Since we have only one, no reduction can be done

$m_3 = AB$ AND

ex)

A B	f(A,B)
0 0	1
0 1	1
1 0	1
1 1	0

A \ B	0	1
0	1	1
1	1	0

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

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

so, $f = \bar{A} + \bar{B} = \overline{A \cdot B}$ NAND

De Morgan's theorem.

ex)

A B	f	g
0 0	0	1
0 1	1	0
1 0	1	0
1 1	1	0

A \ B	0	1
0	0	1
1	1	1

→ B

A

$\Rightarrow f = A + B$ OR

A \ B	0	1
0	1	0
1	0	0

$g = \bar{A}\,\bar{B} = \overline{A+B}$ NOR

DeMorgan's theorem.

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.

ex)

A \ BC	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$
1	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$AB\bar{C}$

=

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

- 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

ex)

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

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

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

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

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

further algebraic reduction, if possible.

ex)

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

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

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

$g = A + \bar{C}$

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 ($\simeq$ 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)

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

considered as 1s.

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

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

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

better

If considered as 0s

1	0	0	X
1	0	1	X

$\bar{B}\bar{C}$ ABC

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

Alternative 3-Variable Layout

Group A & B rather than B&C

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

Top & bottom edges are adjacent in this case.

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

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

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

Don't care minterm notation : $\Sigma X m(2,6)$

" Maxterm " : $\Pi X M(2,6)$

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.

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}

Examples

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

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

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

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

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

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

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

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

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

$= ACD$

$g = ACD + \bar{B}$

Minterm notation & NAND-NAND logic

- Any Boolean expression in Minterm notation can be directly realized in NAND-NAND logic.
- Simple (NAND gates only) and fast (only 2-3 ~~two~~ gate delay).
- Minimal NAND-NAND expression can be found by K-map.

or dc(3,7)

Example

$F = \Sigma m(0,1,5) + \Sigma x m(3,7)$ ← Don't care conditions.

Canonical SOP: $F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + A\,\overline{B}\,C$

minimal ~~NOR~~ NAND-NAND expression?

① K-map

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

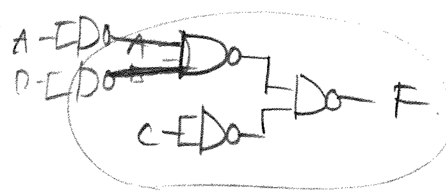
② minimal SOP: $F = \overline{A}\,\overline{B} + C$

③ minimal NAND-NAND: using involution theorem & DeMorgan's

$F = \overline{\overline{\overline{A}\,\overline{B} + C}}$ $\overset{\text{DeMorgan}}{=}$ $\overline{(\overline{\overline{A}\cdot\overline{B}})\cdot(\overline{C})}$

~~DeMorgan~~

④ Logic diagram:



Maxterm notation & NOR-NOR logic

- Any Boolean express in Maxterm notation can be directly realized in NOR-NOR logic.
- Simple and fast.
- Minimal NOR-NOR expression can be found by K-map.

$\approx dc(3,7)$

Example

$F = \Pi M(2,4,5) + \Pi XM(3,7)$

canonical POS: $F = (A+\bar{B}+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+\bar{B}+C)$

minimal NOR-NOR:

① K-map

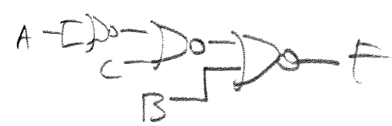
A \ BC	00	01	11	10
0	1	1	X	0
1	0	1	X	0

② minimal pos: $(\bar{A}+C)\cdot(\bar{B})$

③ Minimal NOR-NOR:

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

④ Diagram



Program Completed

University of Missouri-Rolla

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