CS230: Digital Logic Design and Computer Architecture

Lecture 2: Logic gates and K-maps

https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html

https://www.cse.iitb.ac.in/~biswa/

Phones (smart/non-smart) on silence plz, Thanks

Logistics

- Join Piazza now
- Lab on Monday, January 9, 2 PM SL1 to SL3, attendance compulsory
- You can meet me and discuss if anything is not clear

- Problem set 1 by next week. Ungraded, for your practice only
- Detailed course content by end of this week

Range of Numbers

System	Range
Unsigned	$[0, 2^N - 1]$
Sign/Magnitude	$[-2^{N-1} + 1, 2^{N-1} - 1]$
Two's Complement	$[-2^{N-1}, 2^{N-1} - 1]$

Remember sign/magnitude has two zeros ☺

Sign Extension

To represent a signed number in 2’s complement form using large number of bits

Repeat the sign bit at the msbs as needed

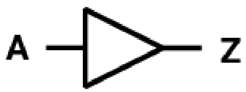
Overflow

1101 + 0101 ?


PAUSE

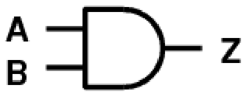
Common Logic Gates

Buffer



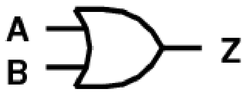
A	Z
0	0
1	1

AND



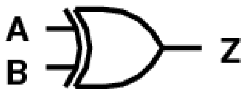
A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR



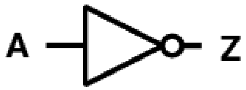
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

XOR



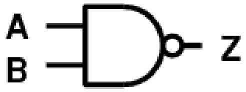
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Inverter



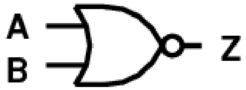
A	Z
0	1
1	0

NAND



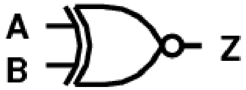
A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Universal Logic gates? Coffee points++

NAND and NOR

A bit of Boolean algebra

Dual

Operations with 0 and 1:

1. $X + 0 = X$ — 1D. $X \cdot 1 = X$
2. $X + 1 = 1$ — 2D. $X \cdot 0 = 0$

AND, OR with identities gives you back the original variable or the identity (dot: AND, plus: OR)

Idempotent Law:

3. $X + X = X$ — 3D. $X \cdot X = X$

AND, OR with self = self

Involution Law:

4. $\overline{(\overline{X})} = X$

double complement = no complement

Laws of Complementarity:

5. $X + \overline{X} = 1$ — 5D. $X \cdot \overline{X} = 0$

AND, OR with complement gives you an identity

Commutative Law:

6. $X + Y = Y + X$ — 6D. $X \cdot Y = Y \cdot X$

Just an axiom…

Contd.

Associative Laws:

7. $(X + Y) + Z = X + (Y + Z) = X + Y + Z$

7D. $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z) = X \bullet Y \bullet Z$

Parenthesis order does not matter

Distributive Laws:

8. $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$

8D. $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$

Axiom

Simplification Theorems:

9. $X \bullet Y + X \bullet \overline{Y} = X$

9D. $(X + Y) \bullet (X + \overline{Y}) = X$

10. $X + X \bullet Y = X$, how?

10D. $X \bullet (X + Y) = X$

11. $(X + \overline{Y}) \bullet Y = X \bullet Y$

11D. $(X \bullet \overline{Y}) + Y = X + Y$

Useful for simplifying expressions

Actually worth remembering — they show up a lot in real designs…

DeMorgan's Law (Can you prove it)?

12. $\overline{(\boldsymbol{X} + \boldsymbol{Y} + \boldsymbol{Z} + \cdots)} = \overline{\boldsymbol{X}}.\overline{\boldsymbol{Y}}.\overline{\boldsymbol{Z}}. \ldots$

12D. $\overline{(\boldsymbol{X}.\boldsymbol{Y}.\boldsymbol{Z}. \ldots)} = \overline{\boldsymbol{X}} + \overline{\boldsymbol{Y}} + \overline{\boldsymbol{Z}} + \ldots$

- **Think of this as a transformation**
  - Let's say we have:

$$F = A + B + C$$

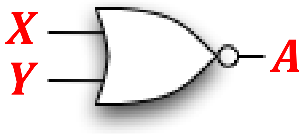
  - Applying DeMorgan's Law (12), gives us

$$\boldsymbol{F} = \overline{\overline{(\boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C})}} = \overline{(\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}.\overline{\boldsymbol{C}})}$$

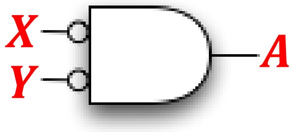
At least one of A, B, C is TRUE --> It is **not** the case that A, B, C are **all** false

Contd. with a Truth Table

$$A = \overline{(X + Y)} = \overline{X}\,\overline{Y}$$

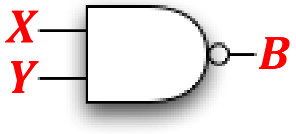


NOR is equivalent to AND with inputs complemented

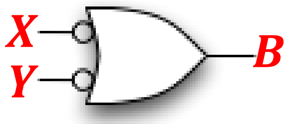


X	Y	$\overline{X+Y}$	$\overline{X}$	$\overline{Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

$$B = \overline{(XY)} = \overline{X} + \overline{Y}$$



NAND is equivalent to OR with inputs complemented



X	Y	$\overline{XY}$	$\overline{X}$	$\overline{Y}$	$\overline{X} + \overline{Y}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Remember: It is not

$$\overline{(\boldsymbol{X}.\boldsymbol{Y})} = \bar{\boldsymbol{X}}.\bar{\boldsymbol{Y}}$$

$$\overline{(\boldsymbol{X}+\boldsymbol{Y})} = \bar{\boldsymbol{X}}+\bar{\boldsymbol{Y}}$$

Definitions of interest

- A ***normal term*** is a product or sum term in which no variable appears more than once.
  - *Examples:* $a, \bar{a}, a+c, \bar{a}cd$ are normal terms; $\bar{a}+a, \bar{a}a$ are not normal terms.

- A ***minterm*** of *n* variables is a normal product term with *n* literals. There are 2^n such product terms.
  - *Examples of 3-variable minterms:* $\bar{a}bc, abc$
  - *Example:* $\bar{a}b$ is not a 3-variable minterm.
- A ***maxterm*** of *n* variables is a normal sum term with *n* literals. There are 2^n such sum terms.
  - *Examples of 3-variable maxterms:* $\bar{a}+b+c, a+b+c$

Definitions of interest

- A ***sum of products (SOP)*** expressions is a set of product (AND) terms connected with logical sum (OR) operators.
  - *Examples:* $a, \bar{a}, ab+c, \bar{a}c+bde, a+b$ are SOP expressions.

- A ***product of sum (POS)*** expressions is a set of sum (OR) terms connected with logical product (OR) operators.
  - *Examples:* $a, \bar{a}, a+b+c, (\bar{a}+c)(b+d)$ are POS expressions.

Definitions of interest

- The ***canonical sum of products (CSOP)*** form of an expression refers to rewriting the expression as a sum of minterms.
  - *Examples for 3-variables:* $\bar{a}bc + abc$ is a CSOP expression; $\bar{a}b + c$ is not.
- The ***canonical product of sums (CPOS)*** form of an expression refers to rewriting the expression as a product of maxterms.
  - *Examples for 3-variables:* $(\bar{a}+b+c)(a+b+c)$ is a CPOS expression; $(\bar{a}+b)c$ is not.

- There is a close correspondence between the truth table and minterms and maxterms.

SOP: Sum of Products

Also known as **disjunctive normal form** or **minterm expansion**

Find all the input combinations (minterms) for which the output of the function is TRUE.

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

0 1 1   1 0 0   1 0 1   1 1 0

- Each row in a truth table has a minterm
- A minterm is a product (AND) of literals
- Each minterm is TRUE for that row (and only that row)

All Boolean equations can be written in SOP form

Contd.

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

$$\begin{matrix} & 0\ 1\ 1 & & 1\ 0\ 0 & & 1\ 0\ 1 & & 1\ 1\ 0 & & 1\ 1\ 1 \\ F = & \overline{A}BC & + & A\overline{B}\overline{C} & + & A\overline{B}C & + & AB\overline{C} & + & ABC \end{matrix}$$

Activates this term

- Only the shaded product term — $A\overline{B}C$ $= 1 \cdot \overline{0} \cdot 1$ — will be 1

Contd.

- Standard “shorthand” notation
  - If we agree on the order of the variables in the rows of truth table…
    - then we can enumerate each row with the decimal number that corresponds to the binary number created by the input pattern

A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	1	**100 = decimal 4 so this is minterm #4, or m4**
1	0	1	1	
1	1	0	1	
1	1	1	1	**111 = decimal 7 so this is minterm #7, or m7**

f =

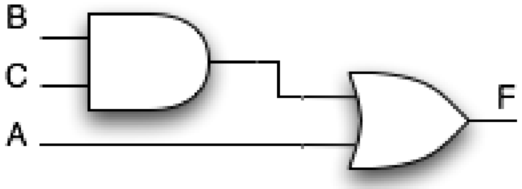
We can write this as a sum of products

Or, we can use a summation notation

Contd.

A	B	C	minterms
0	0	0	$\bar{A}\bar{B}\bar{C}$ = m0
0	0	1	$\bar{A}\bar{B}C$ = m1
0	1	0	$\bar{A}B\bar{C}$ = m2
0	1	1	$\bar{A}BC$ = m3
1	0	0	$A\bar{B}\bar{C}$ = m4
1	0	1	$A\bar{B}C$ = m5
1	1	0	$AB\bar{C}$ = m6
1	1	1	ABC = m7

Shorthand Notation for Minterms of 3 Variables



2-Level AND/OR Realization

***F* in canonical form:**

$F(A,B,C) = \sum m(3,4,5,6,7)$

$= m3 + m4 + m5 + m6 + m7$

$F =$

canonical form ≠ minimal form

F

We are on the same page?

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2 x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1 x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1 x_2 x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2 x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1 x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

POS: Product of Sum

Find all the input combinations (maxterms) for which the output of the function is FALSE.

Product of Sums (POS)

product

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

sums

Each sum term represents one of the "**zeros**" of the function

This input

0 0 0 | 0 0 1 | 0 1 0

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

Activates this term

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

For the given input, only the shaded sum term will equal 0

$$A + \overline{B} + C = 0 + \overline{1} + 0$$

Anything ANDed with 0 is 0; Output F will be 0

The function evaluates to FALSE (i.e., output is 0) if **any** of the Sums (maxterms) causes the output to be 0

Contd.

1. **Minterm to Maxterm conversion:**
   rewrite minterm shorthand using maxterm shorthand
   replace minterm indices with the indices not already used
   E.g., $F(A,B,C) = \sum m(3,4,5,6,7) = \prod M(0,1,2)$

2. **Maxterm to Minterm conversion:**
   rewrite maxterm shorthand using minterm shorthand
   replace maxterm indices with the indices not already used
   E.g., $F(A,B,C) = \prod M(0,1,2) = \sum m(3,4,5,6,7)$

3. **Expansion of F to expansion of $\bar{F}$:**

$$\text{E.g., } F(A,B,C) = \sum m(3,4,5,6,7) \longrightarrow \bar{F}(A,B,C) = \sum m(0,1,2)$$

$$= \prod M(0,1,2) \longrightarrow = \prod M(3,4,5,6,7)$$

4. **Minterm expansion of F to Maxterm expansion of $\bar{F}$:**
   rewrite in Maxterm form, using the same indices as F

$$\text{E.g., } F(A,B,C) = \sum m(3,4,5,6,7) \longrightarrow \bar{F}(A,B,C) = \prod M(3,4,5,6,7)$$

$$= \prod M(0,1,2) \longrightarrow = \sum m(0,1,2)$$

K-Maps

- Karnaugh Map (K-map) method
  - K-map is an alternative method of representing the truth table that helps visualize adjacencies in up to 6 dimensions
  - Physical adjacency ↔ Logical adjacency

2-variable K-map

A \ B	0	1
0	00	01
1	10	11

3-variable K-map

A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110

4-variable K-map

AB \ CD	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

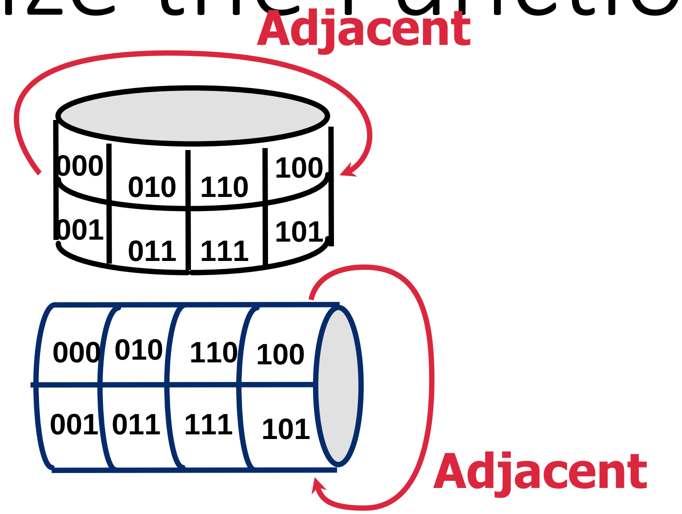
Coffee points

Why 11 before 10 ?

***Numbering Scheme:* 00, 01, 11, 10 is called a**
“Gray Code” — only a *single bit (variable) changes*
from one code word and the next code word

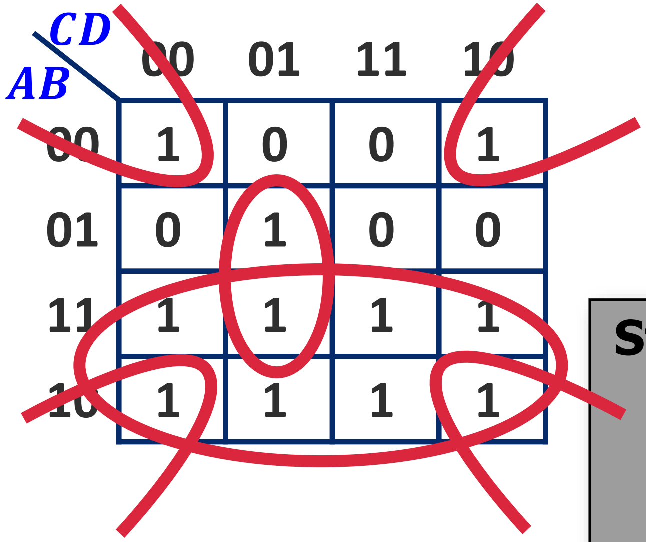
How? To minimize the Function

A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110



K-map adjacencies go "around the edges"
Wrap around from first to last column
Wrap around from top row to bottom row

How?



$$F(A, B, C, D) = \sum m(0, 2, 5, 8, 9, 10, 11, 12, 13, 14, 15)$$

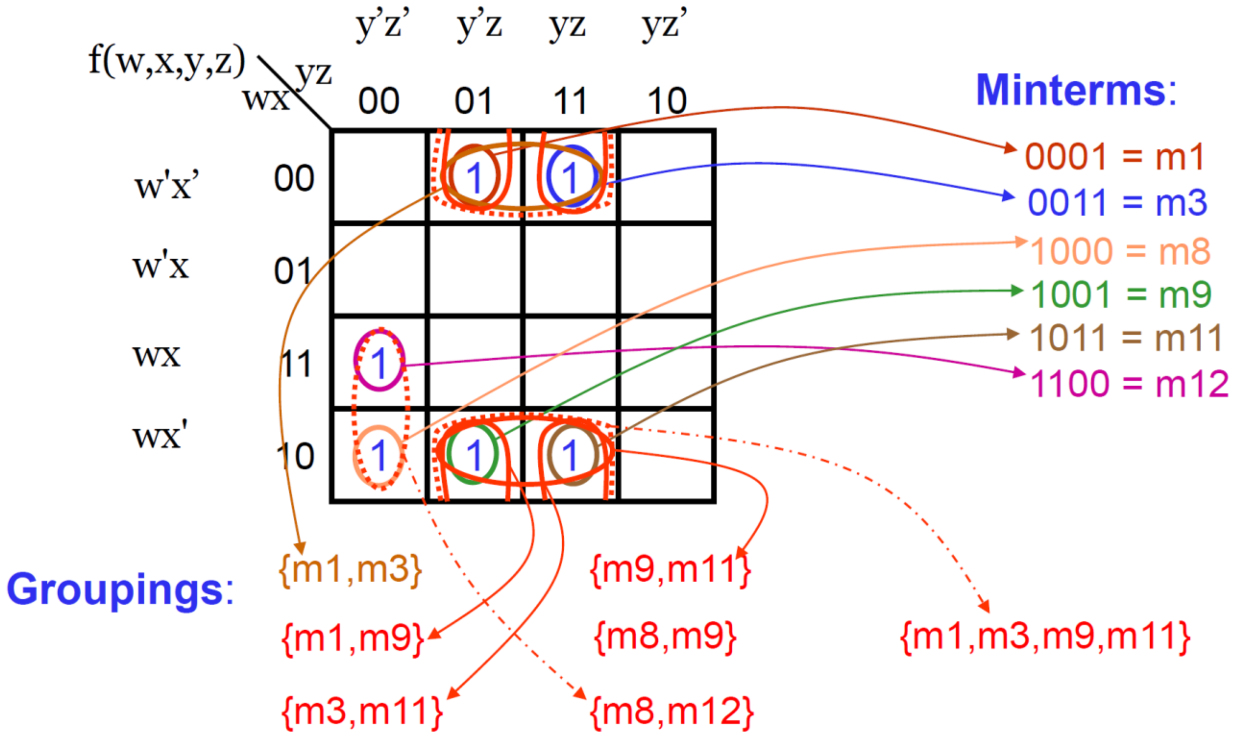
$$F = A + \bar{B}\bar{D} + B\bar{C}D$$

Strategy for "circling" rectangles on Kmap:

Biggest "oops!" that people forget:

Some more examples

Example: Grouping Minterms



Simplified Expression: f(w,x,y,z) = {m1,m3,m9,m11} + {m8,m12}

$$= \overline{x}z + w\overline{y}\,\overline{z}$$

Some more examples

Example: Grouping Maxterms

f(w,x,y,z)

		y+z	y+z'	y'+z'	y'+z
	wx \ yz	00	01	11	10
w+x	00	0	1	1	0
w+x'	01	0	0	0	0
w'+x'	11	1	0	0	0
w'+x	10	1	1	1	0

Maxterms:

0000 = M0
0010 = M2
0100 = M4
0101 = M5
0110 = M6
0111 = M7
1010 = M10
1101 = M13
1110 = M14
1111 = M15

Groupings (note: all groupings are not listed):

{M0,M2,M4,M6}, {M4,M5,M6,M7}, {M5,M7,M13,M15},
{M2,M6,M10,M14}, {M6,M7,M14,M15}

Simplified Expression: f(w,x,y,z) = {M0,M2,M4,M6}{M5,M7,M13,M15}{M2,M6,M10,M14}

$$= (w+z)(\overline{x}+\overline{z})(\overline{y}+z)$$

Why minimize?

Efficient resource usage

Resource scarcity

Summary

- Very simple guideline:
  - Circle all the rectangular blocks of 1's in the map, using the fewest possible number of circles
    - Each circle should be as large as possible
  - Read off the implicants that were circled
  - Some of them may be "don't care" (X) Try it yourself

- More formally:
  - A Boolean equation is minimized when it is written as a sum of the fewest number of prime implicants
  - Each circle on the K-map represents an implicant
  - The largest possible circles are prime implicants

PAUSE

Combinational Circuits

- Combinational logic is often grouped into larger building blocks to build more complex systems
- Hides the unnecessary gate-level details to emphasize the function of the building block
- Output is only dependent on the input
- We now examine:
  - Decoder
  - Multiplexer
  - Full adder

Textbook reading

Chapter 2.1 to 2.7 of H&H

Coffee points:

- Dhananjay 210050044
- Guramrit 210050061

तुमचा दिवस चांगला जावो