Introduction to Computer Engineering – EECS 203
http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/

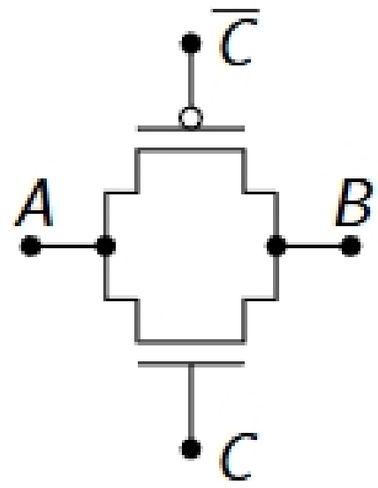
Instructor: Robert Dick
Office: L477 Tech
Email: dickrp@northwestern.edu
Phone: 847-467-2298

TA: Neal Oza
Office: Tech. Inst. L375
Phone: 847-467-0033
Email: nealoza@u.northwestern.edu

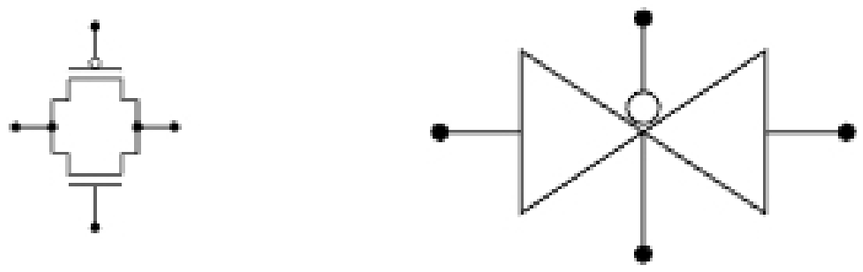
TT: David Bild
Office: Tech. Inst. L470
Phone: 847-491-2083
Email: d-bild@northwestern.edu

NORTHWESTERN
UNIVERSITY

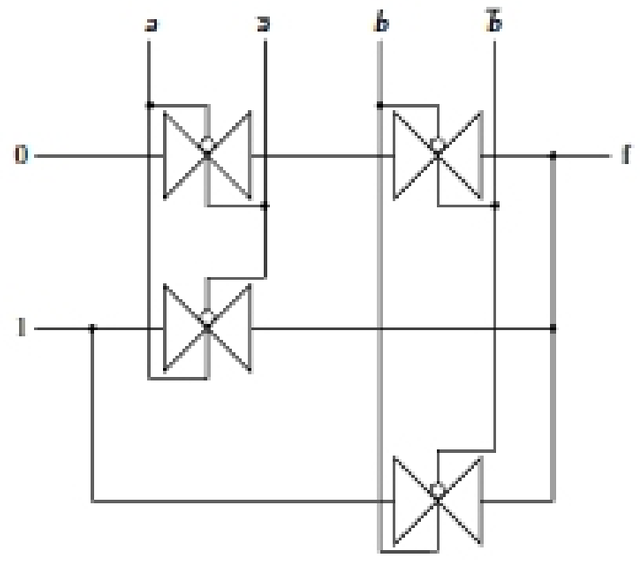
CMOS transmission gate (TG)



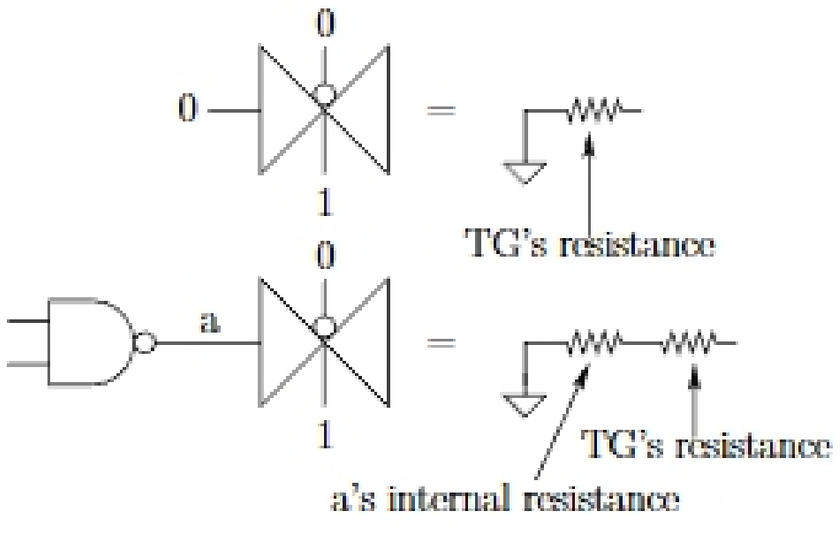
Other TG diagram



TG example



Impact of control on input



Practice w/ Boolean Minimization

Simplify this expression so that it has the minimal possible literal count:

$$(a + \overline{b})\,\overline{(\overline{a}b + cd)}$$

Two-Level Logic and Canonical Forms

- The previous example illustrated one standard representation (product of sums).
- These standard forms are known collectively as two-level logic:
  - Product of Sums (POS) e.g.

$$f = (a + \overline{b})\,(\overline{a} + b)$$

  - Sum of Products (SOP) e.g.

$$f = a\overline{b} + ab$$

  - Can you see why these two are equivalent?
  - Why is this known as two-level logic?

Canonical forms - SOP

For Sum of Products (SOP) the canonical form is constructed out of *minterms*.

- Product term in which all variables appear in complemented or uncomplemented forms once
- For an n-input function corresponds to one of of 2^n possible input combinations
- Use binary representation to enumerate minterms

Canonical forms – SOP

x	y	z	term	symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{x}\,\overline{y}\,\overline{z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{x}\,\overline{y}\,z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{x}\,y\overline{z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{x}\,yz$	m_3	0	0	0	1	0	0	0	0
1	0	0	$x\overline{y}\,\overline{z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$x\overline{y}\,z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$xy\overline{z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	xyz	m_7	0	0	0	0	0	0	0	1

Using Canonical Sum of Products Representation

Convenient representation uses $\sum$ operator and minterms:

$$f(x_1, \ldots, x_n) = \sum m_I$$

For given function f, list all minterms for which the function is true:

$$\begin{aligned} f(a,b,c) &= ab + \overline{a}\,\overline{c} \\ &= (m_6 + m_7) + (m_0 + m_2) \\ &= \sum m\,(0,2,6,7) \end{aligned}$$

Canonical forms – POS

For Products of Sums (POS) the canonical form is constructed out of *maxterms*.

- Sum term in which all variables appear in complemented or uncomplemented forms once
- Use binary representation to enumerate maxterms (note: function is not true for maxterms)

Canonical forms – POS

x	y	z	term	symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$x+y+z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$x+y+\overline{z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$x+\overline{y}+z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$x+\overline{y}+\overline{z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{x}+y+z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{x}+y+\overline{z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{x}+\overline{y}+z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{x}+\overline{y}+\overline{z}$	M_7	1	1	1	1	1	1	1	0

Using Canonical Products of Sums Representation

Convenient representation uses $\prod$ operator and minterms:

$$f(x_1, \ldots, x_n) = \prod M_I$$

For given function f, list all maxterms for which the function is false:

$$\begin{aligned} f(a,b,c) &= (a+\overline{c}\,)\,(\overline{a}+b) \\ &= M_1 \cdot M_3 \cdot M_4 \cdot M_5 \\ &= \prod M\,(1,3,4,5) \end{aligned}$$

More examples of two-level logic

$$f(a,b,c,d) = \sum m(1,4,8,10,13,15)$$

$$f(w,x,y,z) = \prod M(5,13,14)$$

$$f(u,v) = u + \overline{u}\,\overline{v}$$

Logic minimization motivation

- Want to reduce area, power consumption, delay of circuits
- Hard to exactly predict circuit area from equations
- Can approximate area with SOP cubes
- Minimize number of cubes and literals in each cube
- Algebraic simplification difficult
  - Hard to guarantee optimality

Logic minimization motivation

- K-maps work well for small problems
  - Too error-prone for large problems
  - Don't ensure optimal prime implicant selection
- Quine-McCluskey optimal and can be run by a computer
  - Too slow on large problems
- Some advanced heuristics usually get good results fast on large problems
- Want to learn how these work and how to use them?
- Take Advanced Digital Logic Design

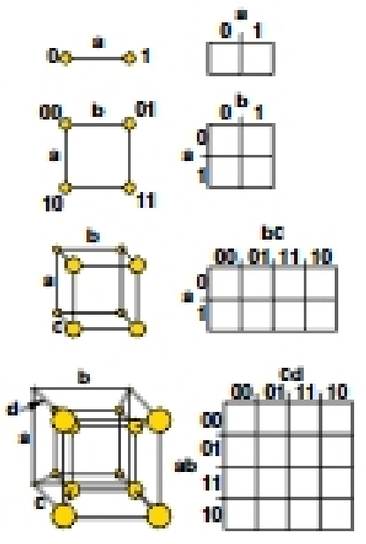
Boolean function minimization

- Algebraic simplification
  - Not systematic
  - How do you know when optimal solution has been reached?
- Optimal algorithm, e.g., Quine-McCluskey
  - Only fast enough for small problems
  - Understanding these is foundation for understanding more advanced methods
- Not necessarily optimal heuristics
  - Fast enough to handle large problems

Karnaugh maps (K-maps)

- Fundamental attribute is adjacency
- Useful for logic synthesis
- Helps logic function visualization
- General Idea: Circle groups of output values (typically 1's)
- Result: Circled terms correspond to minimized product terms

Karnaugh maps



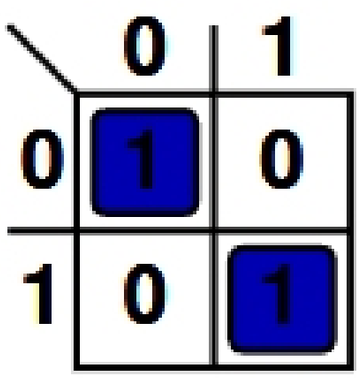
Sum of products (SOP) - Truth table

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

$$f = (\bar{a}\,\bar{b}) + (ab)$$

Sum of products (SOP) - KMap

Equivalent way of expressing the same function:



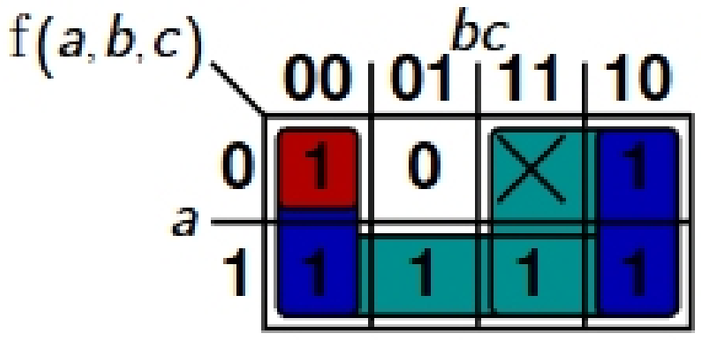
$(a\bar{b}) + (ab)$

Some definitions

- *implicant* - a product term (or sum term) which covers/includes one or more minterms (or maxterms)
- *prime implicant* - implicant that cannot be covered by a more general implicant (i.e. one with fewer literals)
- *essential prime implicants* - cover an output of the function that no other prime implicant (or sum thereof) is able to cover

Implicants

For now, treat × as a "wildcard"



Prime implicants are not covered by other implicants Essential prime implicants uniquely cover minterms

K-map example

- Minimize $f(a, b, c, d) = \sum(1, 3, 8, 9, 10, 11, 13)$
- f(a, b, c, d) $= a\bar{b} + \bar{b}\,d + a\bar{c}\,d$