

CS 316: Gates and Logic

Kavita Bala

Fall 2007

Computer Science

Cornell University

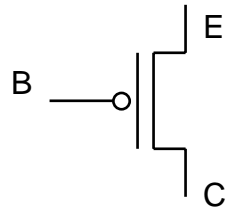
Announcements

- Class newsgroup created
 - Posted on web-page
- Use it for partner finding
- First assignment is to find partners

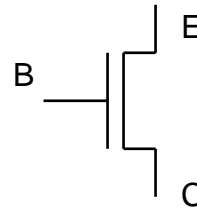
Kavita Bala, Computer Science, Cornell University

P and N Transistors

- PNP Transistor



- NPN Transistor

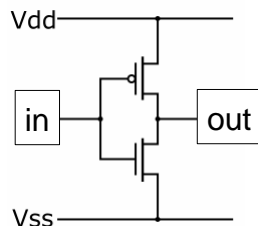


- Connect E to C when base = 0

- Connect E to C when base = 1

Kavita Bala, Computer Science, Cornell University

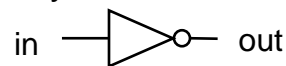
Inverter



In	Out
0	1
1	0

Truth table

- Function: NOT
- Called an inverter
- Symbol:

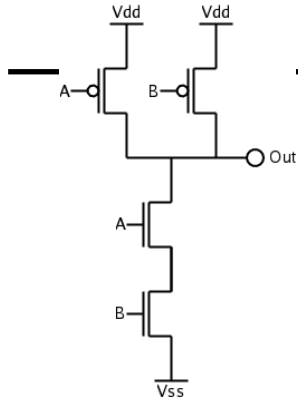


- Useful for taking the inverse of an input

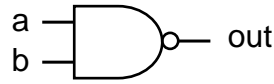
- CMOS: complementary-symmetry metal-oxide-semiconductor

Kavita Bala, Computer Science, Cornell University

NAND Gate



- Function: NAND
- Symbol:

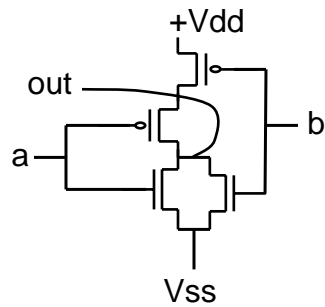


A	B	out
0	0	1
1	0	1
0	1	1
1	1	0

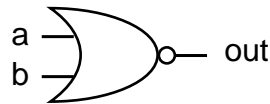


Kavita Bala, Computer Science, Cornell University

NOR Gate



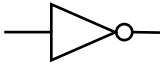

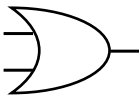
- Function: NOR
- Symbol:



A	B	out
0	0	1
1	0	0
0	1	0
1	1	0

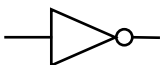


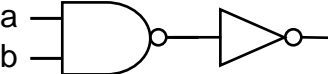
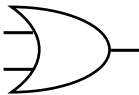
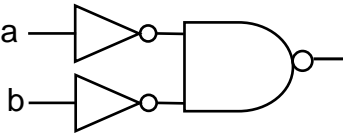
Kavita Bala, Computer Science, Cornell University

Building Functions

- NOT: 
- AND: 
- OR: 
- NAND and NOR are universal
 - Can implement any function with NAND or just NOR gates
 - useful for manufacturing

Kavita Bala, Computer Science, Cornell University

Building Functions

- NOT:  
- AND:  
a b
- OR:  
a b
- NAND and NOR are universal
 - Can implement any function with NAND or just NOR gates
 - useful for manufacturing

Kavita Bala, Computer Science, Cornell University

Logic Equations

- AND
 - out = $a b$ = $a \& b$ = $a \wedge b$
- OR
 - out = $a + b$ = $a | b$ = $a \vee b$
- NOT
 - out = \bar{a} = $!a$ = $\neg a$

Kavita Bala, Computer Science, Cornell University

Identities

- Identities useful for manipulating logic equations
 - For optimization & ease of implementation
 - $a + \bar{a} = 1$
 - $a + 0 = a$
 - $a + 1 = 1$
 - $a \bar{a} = 0$
 - $a 0 = 0$
 - $a 1 = a$
 - $a(b+c) = ab + ac$
 - $\overline{(a + b)} = \bar{a} \bar{b}$
 - $\overline{(a b)} = \bar{a} + \bar{b}$
 - $a + \bar{a} b = a + b$

Kavita Bala, Computer Science, Cornell University

Logic Manipulation

- Can specify functions by describing gates, truth tables or logic equations
- Can manipulate logic equations algebraically
- Can also use a truth table to prove equivalence
- Example: $(a+b)(a+c) = a + bc$

$$\begin{aligned}
 &(a+b)(a+c) \\
 &= aa + ab + ac + bc \\
 &= a + a(b+c) + bc \\
 &= a(1 + (b+c)) + bc \\
 &= a + bc
 \end{aligned}$$

a	b	c	a+b	a+c	LHS	bc	RHS
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Kavita Bala, Computer Science, Cornell University

Logic Minimization

- A common problem is how to implement a desired function most efficiently
- One can derive the equation from the truth table

a	b	c	minterm
0	0	0	\overline{abc}
0	0	1	$\overline{a}bc$
0	1	0	$a\overline{b}\overline{c}$
0	1	1	$a\overline{b}c$
1	0	0	$a\overline{b}\overline{c}$
1	0	1	$a\overline{b}c$
1	1	0	$ab\overline{c}$
1	1	1	abc

for all outputs that are 1, take the corresponding minterm, OR the minterms to obtain the result in "sum of products" form

- How does one find the most efficient equation?
 - Manipulate algebraically until satisfied
 - Use Karnaugh maps

Kavita Bala, Computer Science, Cornell University

Karnaugh maps

- Encoding of the truth table where adjacent cells differ in only one bit

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

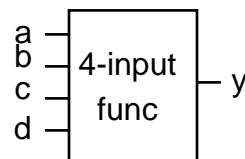
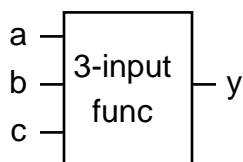
truth table
for AND

ab	00	01	11	10
	0	0	1	0

Corresponding
Karnaugh map

Kavita Bala, Computer Science, Cornell University

Bigger Karnaugh Maps



c \ ab	00	01	11	10
0				
1				

cd \ ab	00	01	11	10
00				
01				
11				
10				

Kavita Bala, Computer Science, Cornell University

Minimization with Karnaugh maps (1)

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

- Sum of minterms yields
– $\bar{a}\bar{b}c + \bar{a}bc + a\bar{b}c + abc$

Kavita Bala, Computer Science, Cornell University

Minimization with Karnaugh maps (2)

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

- Sum of minterms yields
– $\bar{a}bc + a\bar{b}c + abc + abc$
- Karnaugh maps identify which inputs are (ir)relevant to the output

		ab			
	c	00	01	11	10
0		0	0	0	1
1		1	1	0	1

Kavita Bala, Computer Science, Cornell University

Minimization with Karnaugh maps (2)

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

c \ ab	00	01	11	10
0	0	0	0	1
1	1	1	0	1

Kavita Bala, Computer Science, Cornell University

- Sum of minterms yields
 - $\bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}c$
- Karnaugh map minimization
 - Cover all 1's
 - Group adjacent blocks of 2^n 1's that yield a rectangular shape
 - Encode the common features of the rectangle
 - $out = a\bar{b} + \bar{a}c$

Karnaugh Minimization Tricks (1)

c \ ab	00	01	11	10
0	0	1	1	1
1	0	0	1	0

c \ ab	00	01	11	10
0	1	1	1	1
1	0	0	1	0

- Minterms can overlap
 - $out = b\bar{c} + a\bar{c} + ab$
- Minterms can span 2, 4, 8 or more cells
 - $out = \bar{c} + ab$

Kavita Bala, Computer Science, Cornell University

Karnaugh Minimization Tricks (2)

	ab			
cd	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	0	1
10	0	0	0	0

- The map wraps around
– out = \overline{bd}

	ab			
cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

- out = \overline{bd}

Kavita Bala, Computer Science, Cornell University

Karnaugh Minimization Tricks (3)

	ab			
cd	00	01	11	10
00	0	0	0	0
01	1	x	x	x
11	1	x	x	1
10	0	0	0	0

- “Don’t care” values can be interpreted individually in whatever way is convenient
– assume all x’s = 1
– out = d

	ab			
cd	00	01	11	10
00	1	0	0	x
01	0	x	x	0
11	0	x	x	0
10	1	0	0	1

- assume middle x’s = 0
– assume 4th column x = 1
– out = \overline{bd}

Kavita Bala, Computer Science, Cornell University