BEE 271 Digital circuits and systems

Spring 2017

Lecture 2: Logic circuits and Karnaugh maps

Nicole Hamilton

https://faculty.washington.edu/kd1uj

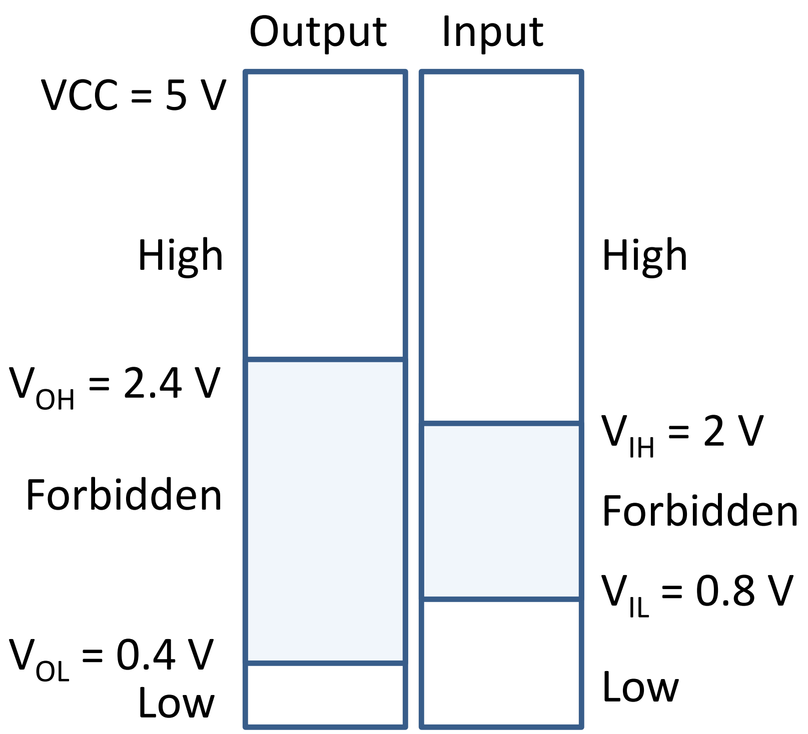
Topics

1. Review
   a. Binary numbers
   b. Boolean algebra
2. minterms and Maxterms
3. Sum of products
4. Product of sums
5. Karnaugh maps

We only have bits

We represent everything in bits, where each bit can be only a 0 or a 1.

Implemented as voltage levels in a digital circuit, e.g., shown here for TTL.



Numbers are positional

In decimal

1492

- 1's
- 10's
- 100's
- 1000's

In binary

1011

- 1's
- 2's
- 4's
- 8's

Numbers are positional

In decimal

1492

10^0

10^1

10^2

10^3

In binary

1011

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

Adding zeroes to the left doesn't change the value.

In decimal

001492 = 1492

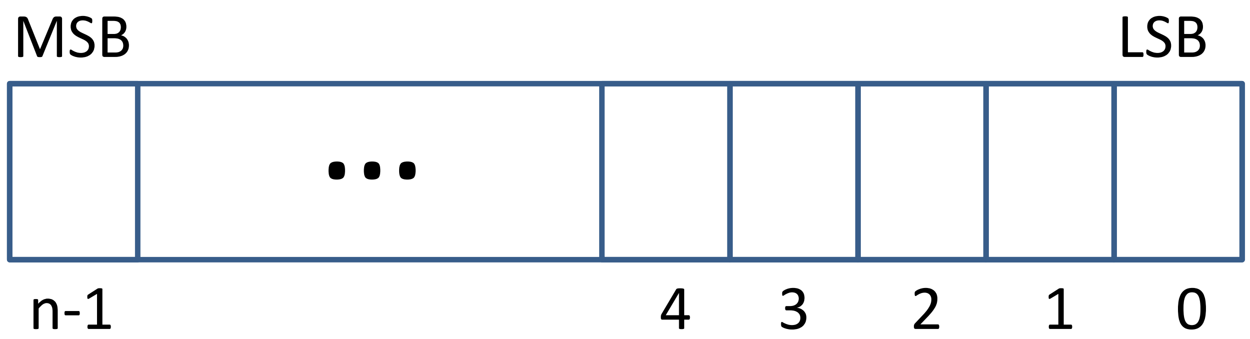
In binary

001011 = 1011

When we add numbers we get carries.

In decimal	In binary
110	*011*
1492	1011
+ 525	+ 011
2017	1110

Binary numbers

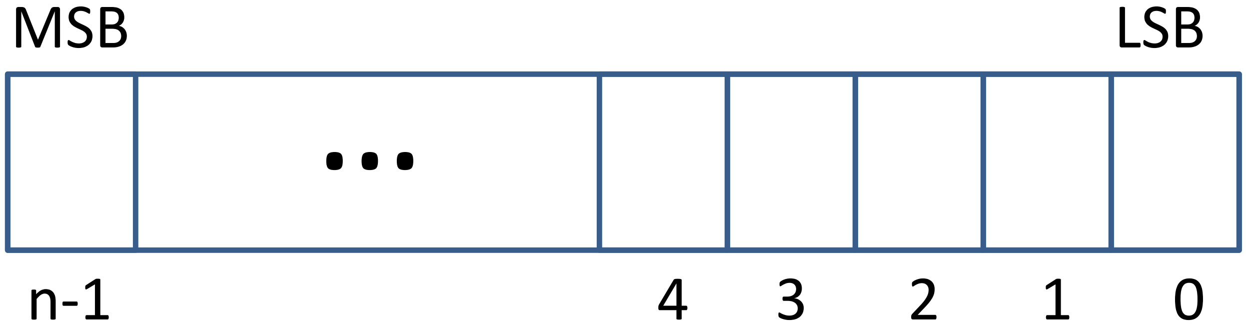


Numbering of the individual bits is from least significant bit (LSB) to most significant bit (MSB).

If b0 = 0, the number is even.
If b0 = 1, the number is odd.

Each bit represents a power of 2.

Value of a binary number



$$Value = \sum_{i=0}^{n-1} b_i 2^i$$

Hex

1. Hard to read long strings of nothing but 1’s and 0’s.
2. So we break it up into groups of 4 bits called *nibbles*, starting *at the LSB*.
3. Take each 4-bit group as a value from 0 to 15.
4. Values 10 to 15 written as A to F.

0111010010011111

0111 0100 1001 1111

7 4 9 F

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

In hex

A12D

$16^0 = 1$

$16^1 = 16$

$16^2 = 256$

$16^3 = 4096$

Exercise

What is binary 10100101 in decimal and hex?

Exercise

What is binary 10100101 in decimal and hex?

Value = 1*1 + 0*2 + 1*4 + 0*8 + 0*16
+ 1*32 + 0*64 + 1*128
= 1 + 4 + 32 + 128
= 165

10100101 = 1010 0101 = A5 hex
= A5 = 10*16 + 5 = 165

Exercise

What is binary 1111100 in decimal and hex?

Exercise

What is binary 1111100 in decimal and hex?

Value = 0*1 + 0*2 + 1*4 + 1*8 + 1*16 + 1*32 + 1*64
= 4 + 8 + 16 + 32 + 64
= 124

Only 7 bits given, extend with high-order zeros.

10100101 = 111 1100 = 0111 1100 = 7C hex

= 7*16 + 12 = 124

Converting to binary

1. Repeatedly integer divide by 2 until the result is 0.
2. At each step, the remainder is the next bit, starting with the LSB.

(We start at the LSB because the lowest bit is just odd or even.)

Convert 12 to binary

Value	Result	Remainder	
12	6	0	LSB
6	3	0	
3	1	1	
1	0	1	MSB

12 base 10 = 1100 binary = Hex C

Exercise: Convert 957 to binary

Value	Result	Remainder
957		

Exercise: Convert 957 to binary

Value	Result	Remainder	
957	478	1	LSB
478	239	0	
239	119	1	
119	59	1	
59	29	1	
29	14	1	
14	7	0	
7	3	1	
3	1	1	
1	0	1	MSB

957 decimal = 11 1011 1101 binary = 3BD hex

Exercise: Convert 1492 to hex

Value	Result	Remainder
1492		

Exercise: Convert 1492 to hex

Value	Result	Remainder	
1492	746	0	LSB
746	373	0	
373	186	1	
186	93	0	
93	46	1	
46	23	0	
23	11	1	
11	5	1	
5	2	1	
2	1	0	
1	0	1	MSB

1492 decimal = 101 1101 0100 binary = 5D4 hex

Chapter 2

Introduction to Logic Circuits

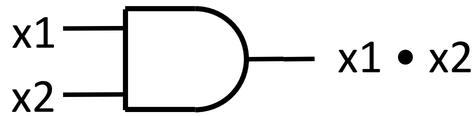
Boolean Algebra

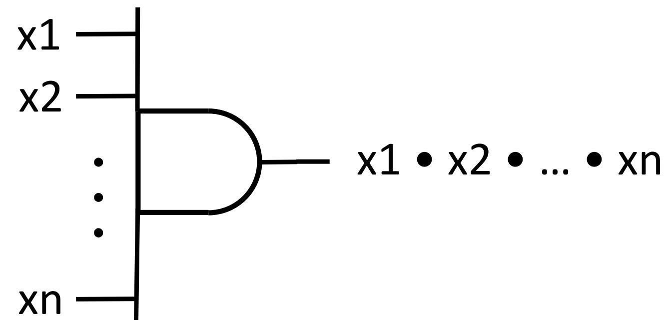
Values	0, 1
Variables	A, B, C, Sum, DoorOpen, ..
Operations	NOT, AND, OR

Operation	*Written as*
NOT X	as $\overline{X}$, X' or X*
X AND Y	X • Y, X Y or X & Y
X OR Y	X + Y

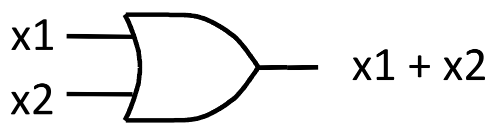
The basic gates.

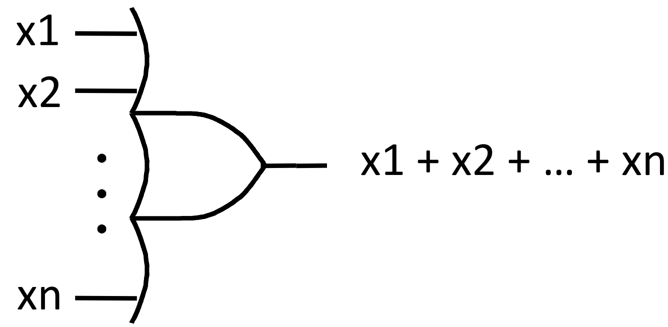
AND If all inputs are true, the output is true.



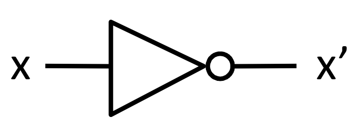


OR If any input is true, the output is true.

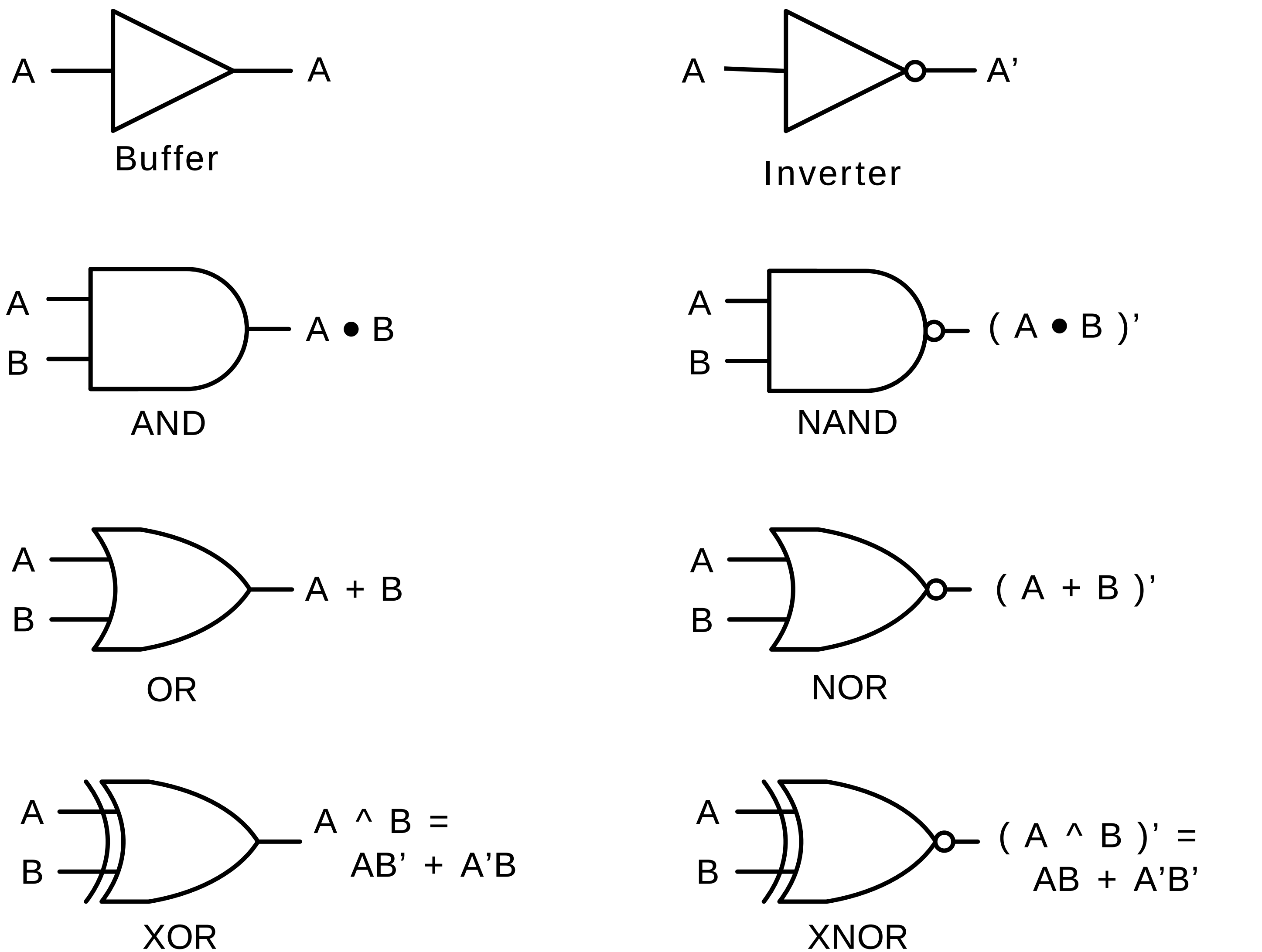




NOT The output is the inverse of the input.



A more complete set of gates



Truth tables

We describe Boolean functions with truth tables.

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

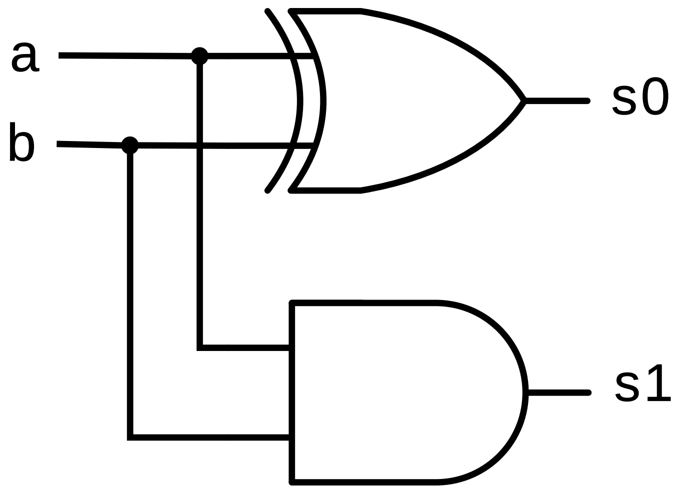
a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

a	NOT a
0	1
1	0

a	0	0	1	1
+b	+0	+1	+0	+1
s1 s0	0 0	0 1	0 1	1 0

a	b	s1	s0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Addition of one-bit binary numbers.

Truth tables

Deriving Boolean equations from truth tables:

a	b	s1	s0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

OR together *product* terms for each truth table row where the function is 1.

If input variable is 0, it appears in complemented form; if 1, it appears uncomplemented.

Truth tables

Deriving Boolean equations from truth tables:

a	b	s1	s0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

s0 = a ^ b

s1 = a b

Example: a full adder

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum =

Cout =

Example: a full adder

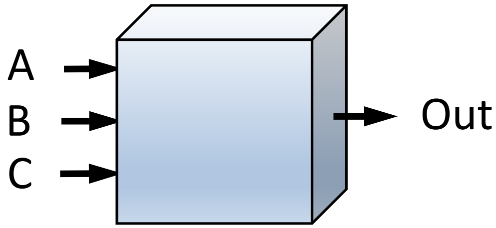
A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum = A’ B’ Cin + A’ B Cin’ + A B’ Cin’ + A B Cin

Cout = A’ B Cin + A B’ Cin + A B Cin’ + A B Cin

Example: A majority function

1. Output should = 1 if the majority of the inputs = 1.



Truth Table

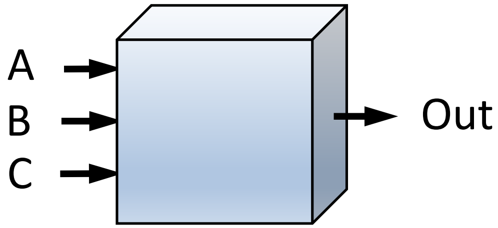
A	B	C	Out

2. Consider the unknown circuit a "black box".

3. Create a truth table.

Example: A majority function

4. Enumerate all the possible input combinations.

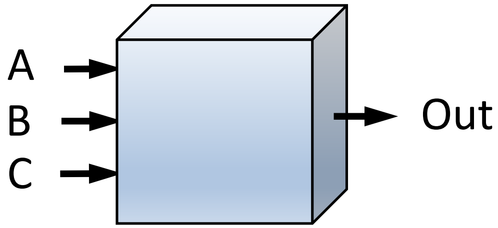


Truth Table

A	B	C	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Example: A majority function

5. Fill in the outputs.

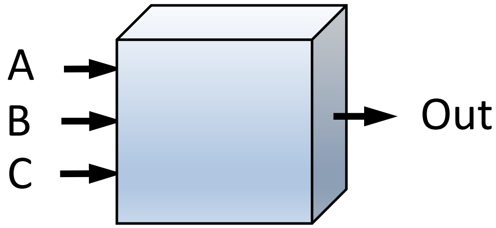


Truth Table

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Example: A majority function

5. Write the equation summing up all the 1's.



Truth Table

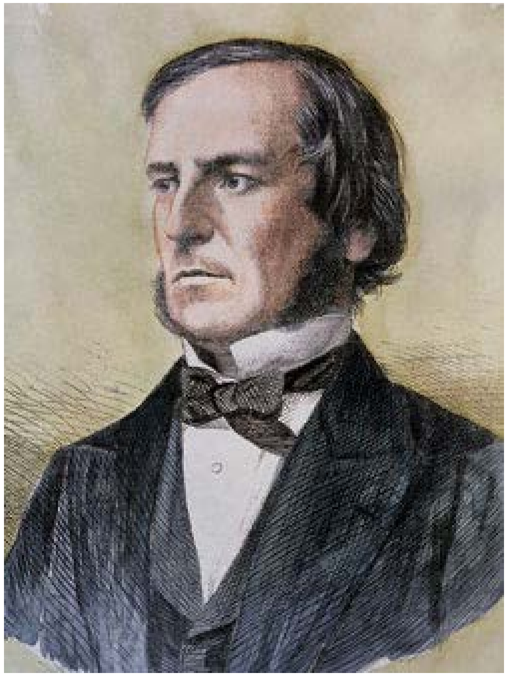
A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$Out = A' B C + A B' C + A B C' + A B C$

A deeper dive into Boolean algebra

Boolean algebra

Named after George Boole, who published an algebraic description of the processes involved in logical thought and reasoning in 1849.



https://en.wikipedia.org/wiki/George_Boole

Boolean algebra

In the 1930s, used by Claude Shannon to describe circuits built with switches, and thus with logic circuits.

https://en.wikipedia.org/wiki/Claude_Shannon

ax·i·om

/ˈaksēəm/

noun

a statement or proposition that is regarded as being established, accepted, or self-evidently true.

"the **axiom that** supply equals demand"

synonyms: accepted truth, general truth, dictum, truism, principle; More

- MATHEMATICS

  a statement or proposition on which an abstractly defined structure is based.

Axioms of Boolean Algebra

1a. $0 \bullet 0 = 0$

1b. $1 + 1 = 1$

2a. $1 \bullet 1 = 1$

2b. $0 + 0 = 0$

3a. $0 \bullet 1 = 1 \bullet 0 = 0$

3b. $1 + 0 = 0 + 1 = 1$

4a. If $x = 0$, then $x' = 1$

4b. If $x = 1$, then $x' = 0$

Notice the *duality*:

$\bullet \quad +$

$0 \quad 1$

Single-variable theorems

5a.	$x \bullet 0 = 0$	
5b.	$x + 1 = 1$	
6a.	$x \bullet 1 = x$	
6b.	$x + 0 = x$	
7a.	$x \bullet x = x$	Replication
7b.	$x + x = x$	
8a.	$x \bullet x' = 0$	
8b.	$x + x' = 1$	
9.	$(x')' = x$	

Easily proved by *perfect induction,* trying all the possibilities.

2 and 3-variable properties

10a.	$x \bullet y = y \bullet x$	Commutative
10b.	$x + y = y + x$	
11a.	$x \bullet (y \bullet z) = (x \bullet y) \bullet z$	Associative
11b.	$x + (y + z) = (x + y) + z$	
12a.	$x \bullet (y + z) = x \bullet y + x \bullet z$	Distributive
12b.	$x + y \bullet z = (x + y) \bullet (x + z)$	
13a.	$x + x \bullet y = x$	Absorption
13b.	$x \bullet (x + y) = x$	
14a.	$x \bullet y + x \bullet y' = x$	Combining
14b.	$(x + y) \bullet (x + y') = x$	

Easily proved by *perfect induction,* trying all the possibilities.

2 and 3-variable properties

15a. $(x \bullet y)' = x' + y'$ DeMorgan's theorem

15b. $(x + y)' = x' \bullet y'$

16a. $x + x' \bullet y = x + y$

16b. $x \bullet (x' + y) = x \bullet y$

17a. $x \bullet y + y \bullet z + x' \bullet z = x \bullet y + x' \bullet z$ Consensus

17b. $(x + y) \bullet (y + z) \bullet (x' + y) = (x + y) \bullet (x' + z)$

Easily proved by *perfect induction,* trying all the possibilities.

Can prove Boolean theorems by

1. Perfect induction
2. Algebraically
3. Venn diagrams

DeMorgan’s theorem by perfect induction

$$(x \bullet y)' = x' + y'$$

		LHS		RHS		
x	y	x • y	(x • y)'	x'	y'	x' + y'
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Identical

Proof of DeMorgan’s theorem by perfect induction, enumerating all the possibilities in a truth table.

Algebraic proof of the Combining theorem

Combining theorem:

14a. $x \bullet y + x \bullet y' = x$

14b. $(x + y) \bullet (x + y') = x$

$$
\begin{aligned}
x \bullet y + x \bullet y' &= x (y + y') \\
&= x
\end{aligned}
$$

$$
\begin{aligned}
(x + y) (x + y') &= x\,x + x\,y' + x\,y + y\,y' \\
&= x + x (y' + y) + 0 \\
&= x + x = x
\end{aligned}
$$

Algebraic proof of the Consensus theorem

Prove we can ignore this term.

Prove: $x\,y + x'\,z + \boxed{y\,z} = x\,y + x'\,z$

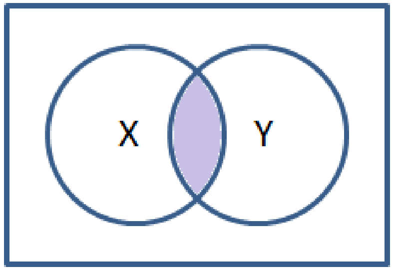
$(x + x') = 1$

$y\,z = (x + x')\,y\,z = x\,y\,z + x'\,y\,z$

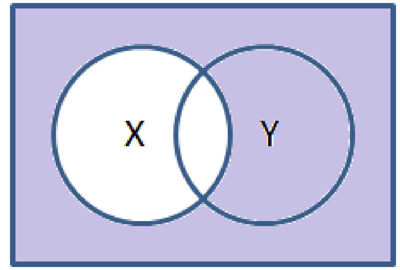
Substituting back into the original LHS:

$$\begin{aligned} x\,y + x'\,z + y\,z &= x\,y + x'\,z + (x\,y\,z + x'\,y\,z) \\ &= x\,y + x\,y\,z + x'\,z + x'\,y\,z \\ &= x\,y\,(1 + z) + x'\,z\,(1 + y) \\ &= x\,y + x'\,z \end{aligned}$$

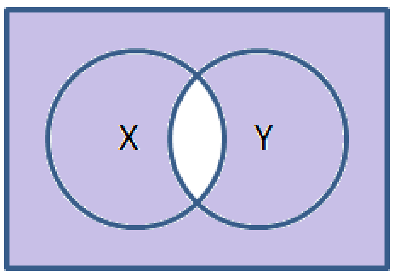
Proof of DeMorgan's Theorem



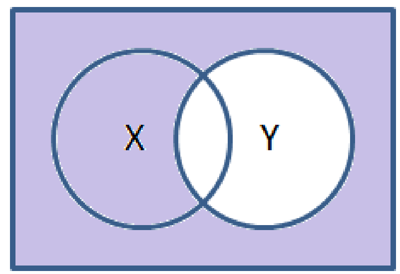
X • Y



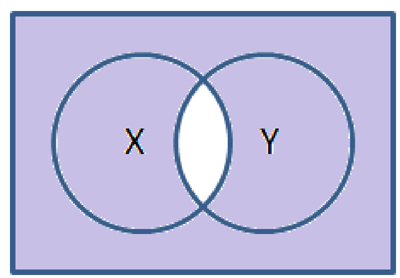
X'



(X • Y)'

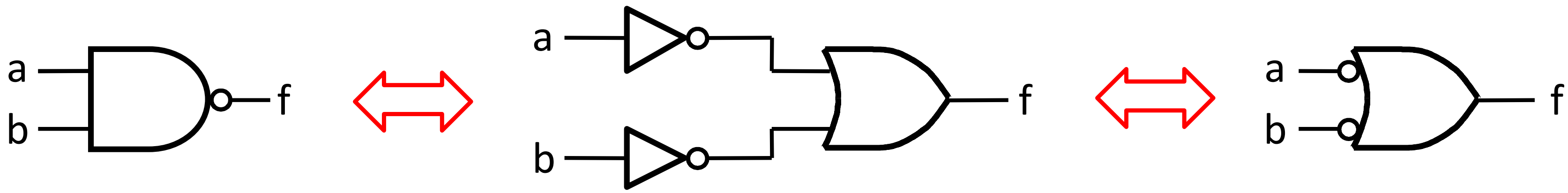


Y'

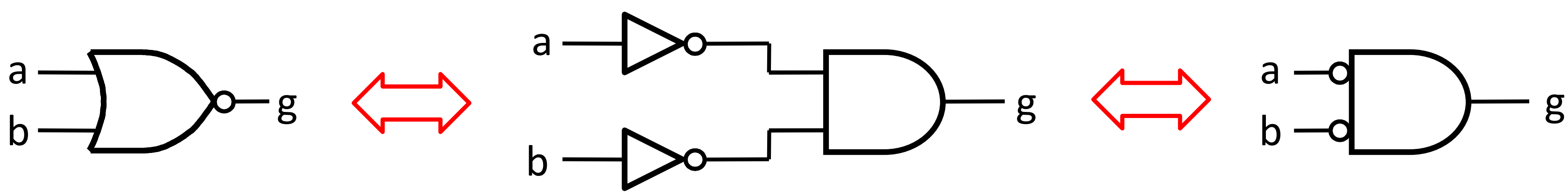


X' + Y'

Bubble pushing



$$f = (\, a\, b\,)' = a' + b'$$



$$g = (\, a + b\,)' = a'\, b'$$

DeMorgan’s theorem in terms of logic gates.

Operator precedence

Highest	NOT	x'
	AND	$\bullet$
Lowest	OR	$+$

Example: $x + y \bullet z' = x + (y \bullet (z'))$

Parentheses can be used to specify a different order of evaluation, for example:

$$((x + y) \bullet z)'$$

We tend to omit the $\bullet$ when the meaning is clear.

Minimization

Often relies on these Boolean theorems:

1. $a + a\,b = a\,(1 + b) = a$
2. $a\,b + a\,b' = a\,(b + b') = a$
3. $(a + b)\,(a + b') = a$
4. $a + a = a$

Synthesis is the process of beginning with a description of the *desired* functional behavior and then generating a circuit that *realizes* that behavior.

Exercise: Synthesize this function

x1	x2	f(x1, x2)
0	0	1
0	1	1
1	0	0
1	1	1

Exercise: Synthesize this function

x1	x2	f(x1, x2)
0	0	1
0	1	1
1	0	0
1	1	1

f(x1, x2) = x1’ x2’ + x1’ x2 + x1 x2

We like to simplify both

x1’ x2’ + x1’ x2 = x1’ (x2’ + x2) = x1’

x1’ x2 + x1 x2 = (x1’ + x1) x2 = x2

To do that, we add a copy of the middle term. We can do that because x + x = x.

Exercise: Synthesize this function

x1	x2	f(x1, x2)
0	0	1
0	1	1
1	0	0
1	1	1

$$f(x1, x2) = x1'\ x2' + x1'\ x2 + x1\ x2$$

Since x + x = x, we can replicate the middle term:

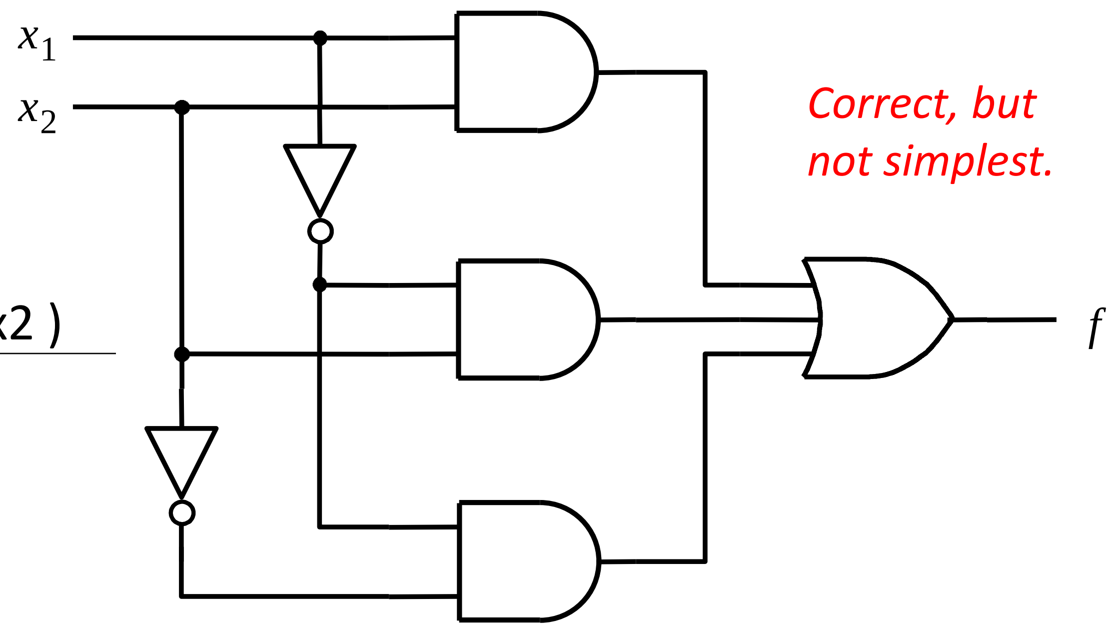
$$f(x1, x2) = x1'\ x2' + x1'\ x2 + x1'\ x2 + x1\ x2$$

Using the distributive property:

$$f(x1, x2) = x1'\ (x2' + x2) + (x1' + x1)\ x2$$

$$= x1' + x2$$

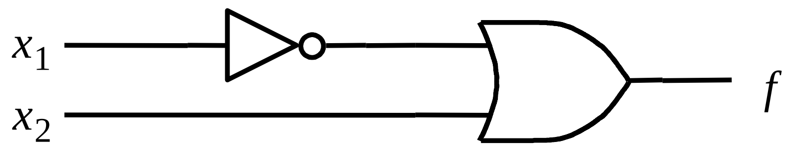
x1	x2	f(x1, x2)
0	0	1
0	1	1
1	0	0
1	1	1



Correct, but not simplest.

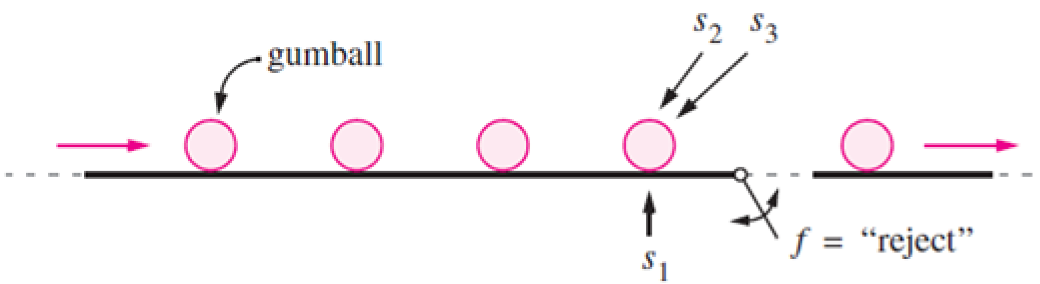
(a) Canonical sum-of-products

f(x1, x2) = x1’ + x2



(b) Minimal-cost realization

Figure 2.20. Two implementations of the function in Figure 2.19.



(a) Conveyor and sensors

s1 = 1 → Too light
s2 = 1 → Too small
s3 = 1 → Too big
f = 1 → Reject the gumball

Reject if the ball is too large or both too small and too light.

s_1	s_2	s_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) Truth table

Figure 2.21. A bubble gumball factory.

s_1	s_2	s_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

We could OR together one term per row where f = 1.

f = s1’ s2’ s3 + s1’ s2 s3 + s1 s2’ s3 + s1 s2 s3’ + s1 s2 s3
Duplicating last term and collecting terms:
= s1’ s3 (s2’ + s2) + s1 s3 (s2’ + s2) + s1 s2 (s3’ + s3)
= s1’ s3 + s1 s3 + s1 s2
= (s1’ + s1) s3 + s1 s2
= s3 + s1 s2

Two ways to synthesize a function

Sum of products: *Include* all rows where f = 1 using minterms.

Product of sums: *Exclude* all rows where f = 0 using Maxterms.

minterms and Maxterms

A *minterm* is 1 for only one row.

A *Maxterm* is 0 for only one row.

Minterms and maxterms for all possible combinations of 3 variables

Row	x1	x2	x3	Minterm	Maxterm
0	0	0	0	m0 = x1' x2' x3'	M0 = x1 + x2 + x3
1	0	0	1	m1 = x1' x2' x3	M1 = x1 + x2 + x3'
2	0	1	0	m2 = x1' x2 x3'	M2 = x1 + x2' + x3
3	0	1	1	m3 = x1' x2 x3	M3 = x1 + x2' + x3'
4	1	0	0	m4 = x1 x2' x3'	M4 = x1' + x2 + x3
5	1	0	1	m5 = x1 x2' x3	M5 = x1' + x2 + x3'
6	1	1	0	m6 = x1 x2 x3'	M6 = x1' + x2' + x3
7	1	1	1	m7 = x1 x2 x3	M7 = x1' + x2' + x3'

Minterms are small m

Maxterms are big M

Minterms (small m)

Row	x1	x2	x3	Minterm
0	0	0	0	m0 = x1' x2' x3'
1	0	0	1	m1 = x1' x2' x3
2	0	1	0	m2 = x1' x2 x3'
3	0	1	1	m3 = x1' x2 x3
4	1	0	0	m4 = x1 x2' x3'
5	1	0	1	m5 = x1 x2' x3
6	1	1	0	m6 = x1 x2 x3'
7	1	1	1	m7 = x1 x2 x3

A *minterm* is 1 for only one row.

It's an AND expression in which each of the input variables appears once.

Each variable can be in complemented, or uncomplemented, e.g., x' or x.

To match a row in a truth table, use the *uncomplemented* form to match a 1 and the *complemented* form to match a 0.

For example, x1 x2' x3 matches the row where
(x1, x2, x3) = (1, 0, 1)

Maxterms (big M)

Row	x1	x2	x3	Maxterm
0	0	0	0	M0 = x1 + x2 + x3
1	0	0	1	M1 = x1 + x2 + x3'
2	0	1	0	M2 = x1 + x2' + x3
3	0	1	1	M3 = x1 + x2' + x3'
4	1	0	0	M4 = x1' + x2 + x3
5	1	0	1	M5 = x1' + x2 + x3'
6	1	1	0	M6 = x1' + x2' + x3
7	1	1	1	M7 = x1' + x2' + x3'

A *maxterm* is a 0 for only one matching row.

It's an OR expression in which each of the input variables appears once.

Each variable can be in complemented, or uncomplemented, e.g., x' or x.

To match a row in a truth table, use the *complemented* form to match a 1 and the *uncomplemented* form to match a 0.

For example, x1' + x2 + x3' matches the row where (x1, x2, x3) = (1, 0, 1).

Sum of products

Include all rows where f = 1 using minterms.

$$f = \sum \left(m_i \bullet f_i \right)$$

Where f_i is the desired result for row i.
If f_i is 0, we can eliminate that term.

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \sum \left(m_i \bullet f_i \right)$$

Using minterms for the rows where we want ones:

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \sum \left(m_i \bullet f_i \right)$$

Using minterms for the rows where we want ones:

f = Σm(1, 4, 5, 6) = m1 + m4 + m5 + m6

= (m1 + m5) + (m4 + m6)

= (x1’ x2’ x3 + x1 x2’ x3) + (x1 x2’ x3’ + x1 x2 x3’)

= (x1’ + x1) x2’ x3 + x1 (x2’ + x2) x3’

= x2’ x3 + x1 x3’

Product of sums

Exclude all rows where f = 0 using Maxterms.

$$f = \prod \left(M_i + f_i \right)$$

Where f_i is the desired result for row i.
If f_i is 1, we can eliminate that term.

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \prod \left(M_i + f_i \right)$$

Using Maxterms for the rows where we want zeros:

f =

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \prod \left(M_i + f_i \right)$$

Using Maxterms for the rows where we want zeros:

f = ΠM(0, 2, 3, 7) = M0 • M2 • M3 • M7

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \prod \left(M_i + f_i \right)$$

Using Maxterms for the rows where we want zeros:

f = ΠM(0, 2, 3, 7) = M0 • M2 • M3 • M7

= (x1 + x2 + x3) (x1 + x2’ + x3) (x1 + x2’ + x3’) (x1’ + x2’ + x3’)

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \prod \left(M_i + f_i \right)$$

Using Maxterms for the rows where we want zeros:

f = ΠM(0, 2, 3, 7) = M0 • M2 • M3 • M7

= (x1 + x2 + x3) (x1 + x2’ + x3) (x1 + x2’ + x3’) (x1’ + x2’ + x3’)

= ((x1 + x3) + x2) ((x1 + x3) + x2’) (x1 + (x2’ + x3’)) (x1’ + (x2’ + x3’))

Using Maxterms for the rows where we want zeros:

$f = M0 \bullet M2 \bullet M3 \bullet M7$

$= (x1 + x2 + x3)(x1 + x2' + x3)(x1 + x2' + x3')(x1' + x2' + x3')$

$= ((x1 + x3) + x2) ((x1 + x3) + x2') (x1 + (x2' + x3'))(x1' + (x2' + x3'))$

Combining theorem:

14a. $x \bullet y + x \bullet y' = x$

14b. $(x + y) \bullet (x + y') = x$

$f = ((x1 + x3) + x2) ((x1 + x3) + x2') (x1 + (x2' + x3')) (x1' + (x2' + x3'))$

$= (x1 + x3) (x2' + x3')$

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \prod \left(M_i + f_i \right)$$

Using Maxterms for the rows where we want zeros:

f = ΠM(0, 2, 3, 7) = M0 • M2 • M3 • M7

= (x1 + x2 + x3) (x1 + x2’ + x3) (x1 + x2’ + x3’) (x1’ + x2’ + x3’)

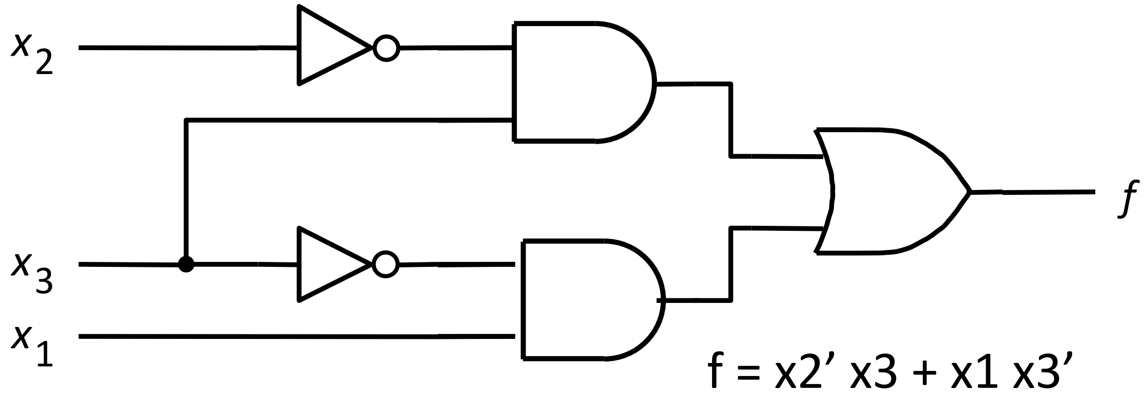
= ((x1 + x3) + x2) ((x1 + x3) + x2’) (x1 + (x2’ + x3’)) (x1’ + (x2’ + x3’))

= (x1 + x3)(x2 + x2’)(x2’ + x3’)(x1 + x1’)

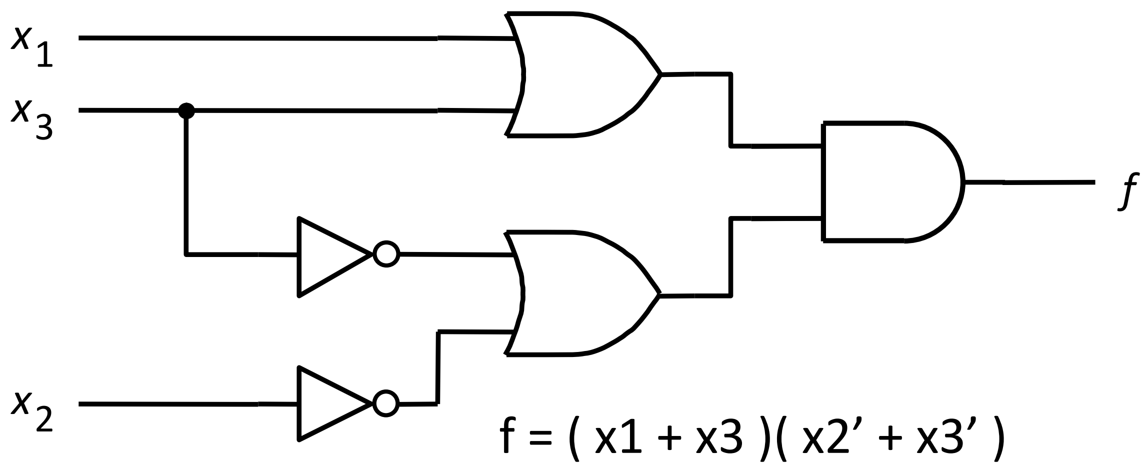
= (x1 + x3)(x2’ + x3’)

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



(a) A minimal sum-of-products realization



(b) A minimal product-of-sums realization

Figure 2.24. Two realizations of the function.

Exercise: A 3-variable function we'd like to synthesize

Row	x1	x2	x3	f(x1, x2, x3)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Are POS and SOP solutions always equivalent cost?

1. Does it matter how many rows are 1s and how many are 0s? Why or why not?
2. Does it matter which rows are 1s or 0s in relation to each other?

Karnaugh maps

Want simplest forms but the algebra is difficult.

Karnaugh maps

Invented by Maurice Karnaugh in 1954 as a graphical method for simplifying Boolean equations.

http://www.ithistory.org/sites/default/files/honor-roll/Maurice%20Karnaugh.jpg

Karnaugh maps

Truth table

row	a b	f
0	0 0	
1	0 1	
2	1 0	
3	1 1	

⇨

		b	
		0	1
a	0	*0*	*1*
	1	*2*	*3*

		a	
		0	1
b	0	*0*	*2*
	1	*1*	*3*

Map rows in a truth table to cells in a matrix.

May choose either assignment of columns and rows.

Example

Truth table

row	a b	f
0	0 0	*1*
1	0 1	*1*
2	1 0	*0*
3	1 1	*1*

⇨

		b	
		0	1
a	0	*1*	*1*
	1	*0*	*1*

		a	
		0	1
b	0	*1*	*0*
	1	*1*	*1*

Fill in the desired output values.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	a’	
		0	1	
a	0	1	1	b
	1	0	1	

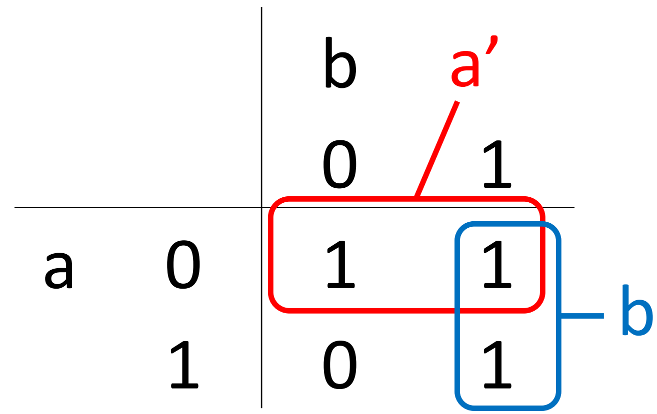
Use the Combining property to group neighboring cells where the output should be the same.

14a. $x \bullet y + x \bullet y' = x$

Truth table

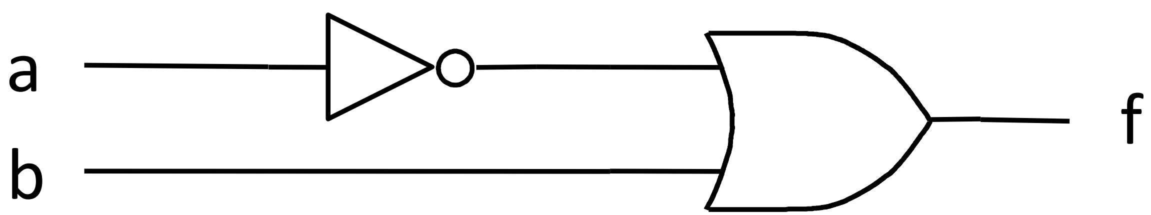
row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map



f = a’ + b

Form the minimal SOP solution.



A function of 3 variables

Truth table

row	a b c	f
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
4	1 0 0	
5	1 0 1	
6	1 1 0	
7	1 1 1	

Karnaugh map

		bc			
		00	01	11	10
a	0	*0*	*1*	*3*	*2*
	1	*4*	*5*	*7*	*6*

Map the rows to a 2 x 4 matrix.
Columns are arranged so each differs by only 1 bit from the next.

Example

Truth table

row	a b c	f
0	0 0 0	*0*
1	0 0 1	*0*
2	0 1 0	*0*
3	0 1 1	*1*
4	1 0 0	*0*
5	1 0 1	*1*
6	1 1 0	*1*
7	1 1 1	*1*

Karnaugh map

		bc			
		00	01	11	10
a	0	*0*	*0*	*1*	*0*
	1	*0*	*1*	*1*	*1*

Example

Truth table

row	a b c	f
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1

Karnaugh map

		bc			
		00	01	11	10
a	0	0	0	1	0
	1	0	1	1	1

$$f = a\,b + a\,c + b\,c$$

A function of four variables

Truth table

row	a b c d	f
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	
3	0 0 1 1	
4	0 1 0 0	
5	0 1 0 1	
6	0 1 1 0	
7	0 1 1 1	
8	1 0 0 0	
9	1 0 0 1	
10	1 0 1 0	
11	1 0 1 1	
12	1 1 0 0	
13	1 1 0 1	
14	1 1 1 0	
15	1 1 1 1	

Karnaugh map

		cd 00	01	11	10
ab	00	*0*	*1*	*3*	*2*
	01	*4*	*5*	*7*	*6*
	11	*12*	*13*	*15*	*14*
	10	*8*	*9*	*11*	*10*

		ab 00	01	11	10
cd	00	*0*	*4*	*12*	*8*
	01	*1*	*5*	*13*	*9*
	11	*3*	*7*	*15*	*11*
	10	*2*	*6*	*14*	*10*

Map the rows to a 4 x 4 matrix either way. (I use the top one.)

Example

Truth table

row	a b c d	f
0	0 0 0 0	*0*
1	0 0 0 1	*0*
2	0 0 1 0	*0*
3	0 0 1 1	*1*
4	0 1 0 0	*1*
5	0 1 0 1	*0*
6	0 1 1 0	*1*
7	0 1 1 1	*0*
8	1 0 0 0	*0*
9	1 0 0 1	*0*
10	1 0 1 0	*0*
11	1 0 1 1	*1*
12	1 1 0 0	*1*
13	1 1 0 1	*0*
14	1 1 1 0	*1*
15	1 1 1 1	*0*

Karnaugh map

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

Copy the outputs to the Karnaugh map.

Example

Truth table

row	a b c d	f
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	1
5	0 1 0 1	0
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	0
9	1 0 0 1	0
10	1 0 1 0	0
11	1 0 1 1	1
12	1 1 0 0	1
13	1 1 0 1	0
14	1 1 1 0	1
15	1 1 1 1	0

Karnaugh map

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

$f = b\,d' + b'\,c\,d$

Identify the prime implicants.

Notice that the Karnaugh map "wraps" vertically and horizontally.

Example

Truth table

row	a b c d	f
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	1
5	0 1 0 1	0
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	0
9	1 0 0 1	0
10	1 0 1 0	0
11	1 0 1 1	1
12	1 1 0 0	1
13	1 1 0 1	0
14	1 1 1 0	1
15	1 1 1 1	0

Karnaugh map

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

f = b d’ + b’ c d

If we’re collecting 1’s for an SOP solution, we can leave out the 0’s.

SOP terminology

Literal	A variable or its complement, e.g., *x* or *x'*.
Product term	A product, e.g., *x y' z*, of some number of literals.
Implicant	A product terms for which the output is 1. That product term *implies* the output is true.
Prime implicant	An implicant that cannot be combined with another with fewer literals.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	
		0	1
a	0	1	1
	1	0	1

$$f = a' + b$$

Each row or cell where f = 1 is an *implicant*.
The ***prime*** *implicants* are a’ and b.

Cover A collection of implicants that account for all cases for which the output = 1.

Essential prime implicant

A *prime implicant* that must be included in any *cover*.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	a’	
		0	1	
a	0	1	1	b
	1	0	1	

f = a’ + b

a’ and b form a *cover* for f.

Both are *essential prime implicants*.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	a’	
		0	1	
a	0	1	1	b
	1	0	1	

f = a’ + b

For a function of n variables, there will be 2^n rows in the truth table and 2^n cells in the Karnaugh map.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	a’	
		0	1	
a	0	1	1	b
	1	0	1	

f = a’ + b

The number of cells in an implicant must be a power of 2.

Truth table

row	a b	f
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

Karnaugh map

		b	$\color{red}{a'}$
		0	1
a	0	1	1
	1	0	1

$f = a' + b$

For a function of *n* variables, if an implicant has *k* literals, it must cover 2^{n-k} cells.