



CSE140: Components and Design Techniques for Digital Systems

Introduction

Instructor: Pietro Mercati

Slides from Prof. Tajana Simunic Rosing

Welcome to CSE 140!

<http://cseweb.ucsd.edu/classes/fa15/cse140/>

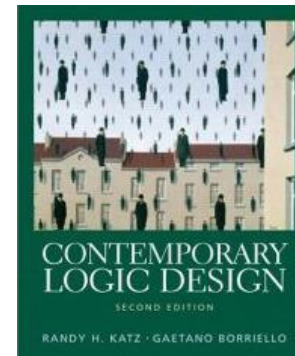
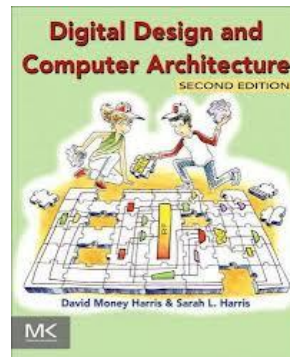
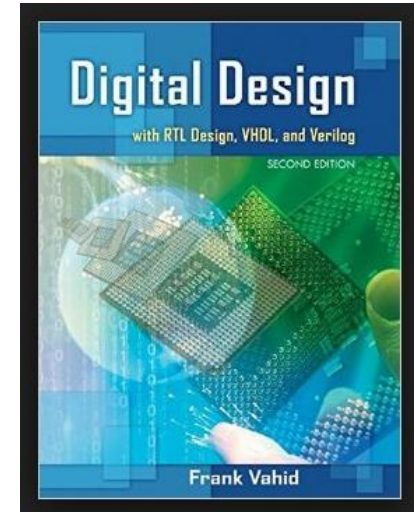
- **Instructor: Pietro Mercati**
 - Email: pimercat@eng.ucsd.edu;
 - please put “CSE140” in the subject line
 - Office Hours:
 - Tue 3.30-5.30pm CSE 2109
 - Tue 8/9 4-6pm CSE 2109
- **TAs and Tutors**
 - Office hours listed on the class website/Piazza
- **Discussion sessions:** Mon 5-6pm
- **Grades/HWs:** <http://ted.ucsd.edu>; submit one file in pdf format
- **Announcements and online discussion:** <https://piazza.com>
 - “CSE140_S216_MERCATI” → SIGN UP SOON !!!!

Class website

- https://cseweb.ucsd.edu/classes/su16_2/cse140-a/index.html

Textbooks and Recommended Readings

- Recommended textbook:
 - digital design by F. Vahid
- Other recommended textbooks:
 - Digital Design by F. Vahid
 - Digital Design & Computer Arch.
 - by David & Sarah Harris
 - Contemporary Logic Design
 - by R. Katz & G. Borriello
- Lecture slides are derived from the slides designed for all three books



Grading



- 4 Homeworks
 - 5% each
- Midterm
 - 30%
- Final
 - 50%
- Homeworks are out on Tuesdays and are due on the following Tuesday before the lecture starts (i.e. before 11am)
- Homework 1 will be out right after this class
- Submission via TED

Some Class Policies

- Academic Honesty

- Studying together in groups is encouraged
- Turned-in work must be ***completely your own***
- Both “giver” and “receiver” are equally culpable
- Cheating on HW/ exams: **F in the course.**
- Any instance of cheating will be referred to Academic Integrity Office

- Late:

- Late submissions will not be accepted.

CSE140 & other CSE classes

Layers of abstraction

CSE 30

CSE 141

CSE 140

Application
Software

Operating
Systems

Architecture

Micro-
architecture

Logic

Digital
Circuits

Analog
Circuits

Devices

Physics

programs

device drivers

instructions
registers

datapaths
controllers

adders
memories

AND gates
NOT gates

amplifiers
filters

transistors
diodes

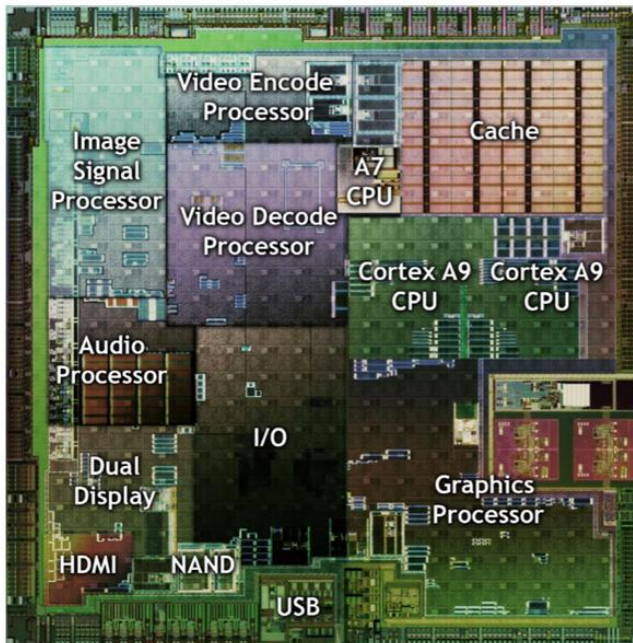
electrons

Abstraction:

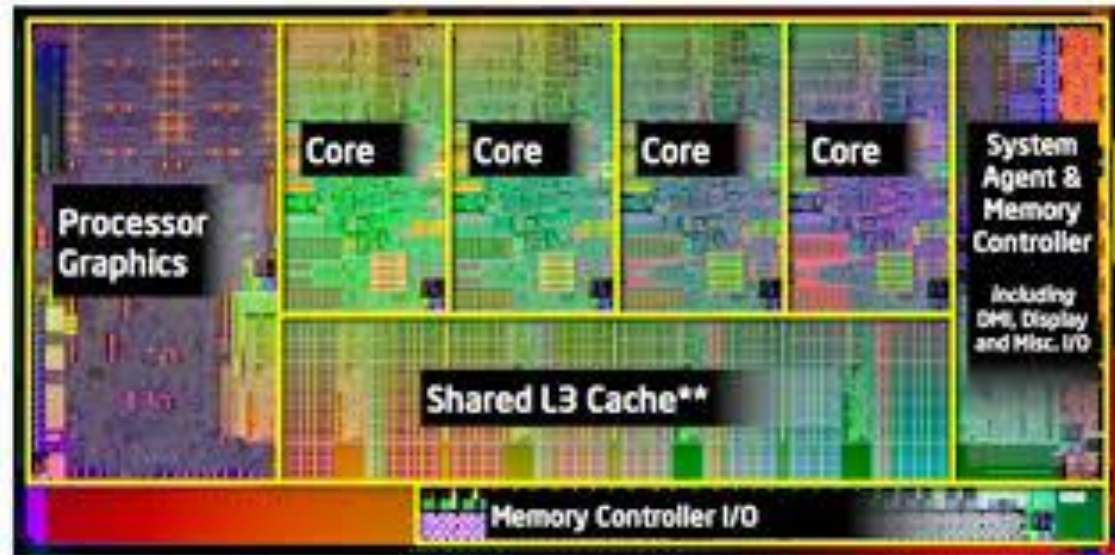
A way to simplify by hiding details
from other layers

Why Study Digital Design?

Look “under the hood” of your processors
You become a better programmer when you
understand hardware your code runs on

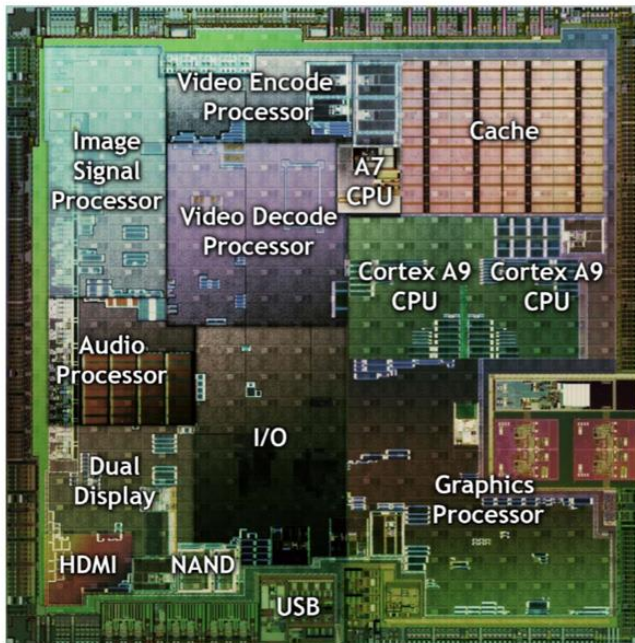


Nvidia Tegra 2 die photo

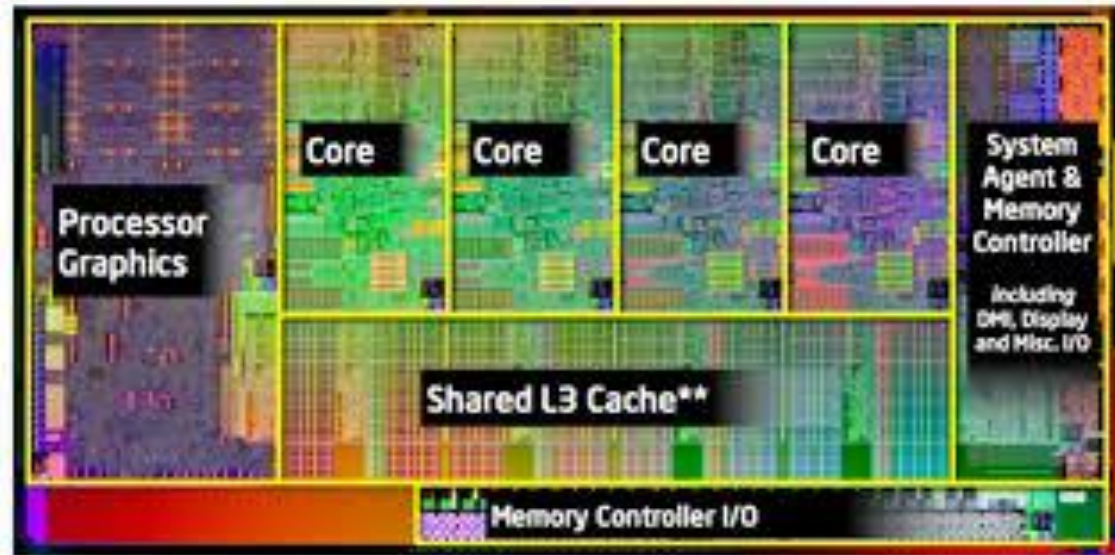


The Scope of CSE140

- We start with Boolean algebra $Y = A \text{ and } B$
- End up with the design of a simple CPU
- What's next? CSE141 – more complex CPU architectures



Nvidia Tegra 2 die photo



Lets get started!

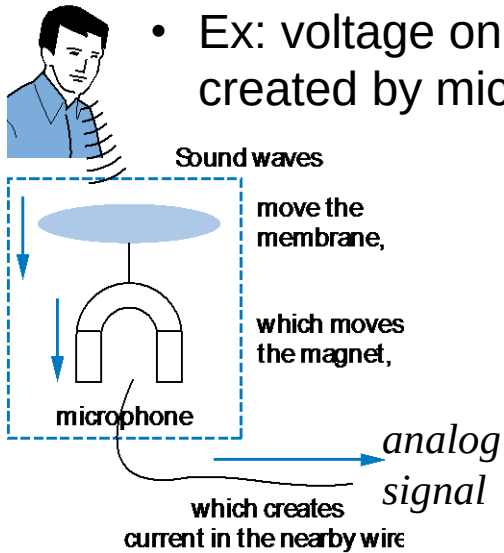
- Number representations
 - Analog vs. Digital
 - Digital representations:
 - Binary, Hexadecimal, Octal
- Switches, MOS transistors, Logic gates
 - What is a switch
 - How a transistor operates
 - Logic gates
 - Building larger functions from logic gates
- Universal gates
- Boolean algebra
 - Properties
 - How Boolean algebra can be used to design logic circuits

What Does “Digital” Mean?

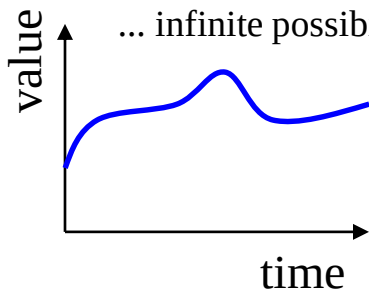
- Analog signal

- Infinite possible values

- Ex: voltage on a wire created by microphone



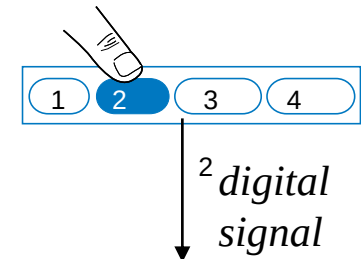
Possible values:
1.00, 1.01, 2.0000009,
... infinite possibilities



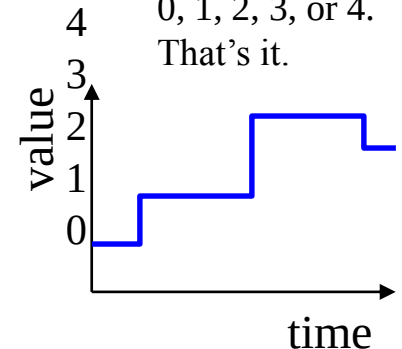
- Digital signal

- Finite possible values

- Ex: button pressed on a keypad



Possible values:
0, 1, 2, 3, or 4.
That's it.



Which is analog?

- A) Wind speed
- B) Radio Signal
- C) Clicker response
- D) A) & B)
- E) All of the above

Encoding Numbers – Base 10 & 2

- Each position represents a quantity; symbol in position means how many of that quantity
 - Base ten (*decimal*)
 - Ten symbols: 0, 1, 2, ..., 8, and 9
 - More than 9 -- next position
 - So each position power of 10
 - Nothing special about base 10 -- used because we have 10 fingers
 - Base two (*binary*)
 - Two symbols: 0 and 1
 - More than 1 -- next position
 - So each position power of 2

Decimal to binary:

- Divide by two
- Report the remainder
- Concatenate the reminders

NUMBER	REMINDER
523	1
261	1
130	0
65	1
32	0
16	0
8	0
4	0
2	0
1	1

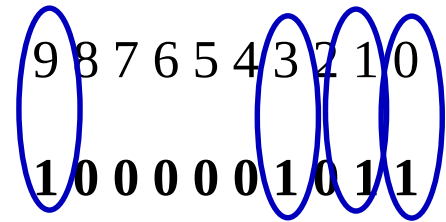
$$523 = 1000001011$$

Encoding Numbers – Base 10 & 2

- Each position represents a quantity; symbol in position means how many of that quantity
 - Base ten (**decimal**)
 - Ten symbols: 0, 1, 2, ..., 8, and 9
 - More than 9 -- next position
 - So each position power of 10
 - Nothing special about base 10 -- used because we have 10 fingers
 - Base two (**binary**)
 - Two symbols: 0 and 1
 - More than 1 -- next position
 - So each position power of 2

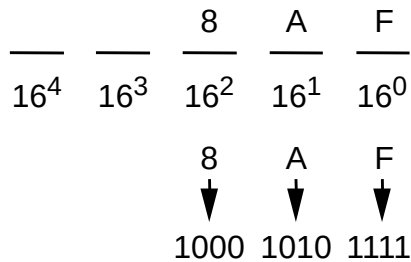
Binary to decimal:

- Each position is a power of 2
- Sum up all the powers of 2 where you have a "1"



$$2^0 + 2^1 + 2^3 + 2^9 = 523$$

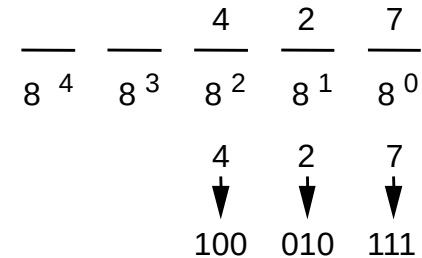
Bases Sixteen & Eight



hex	binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

hex	binary
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

- Base sixteen
 - Used as compact way to write binary numbers
 - Basic digits: 0-9, A-F
 - Known as *hexadecimal*, or just *hex*
- Base eight
 - Basic digits: 0-7
 - Known as *octal*



octal	binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Bases Sixteen & Eight

hex	binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

hex	binary
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Hexadecimal to binary:

- Expand each hexadecimal digit into the corresponding binary string

8 A F =
1000 1010 1111

Binary to hexadecimal:

- Pad the binary string with zeros on the left until you have a number of digits multiple of 4
- Group digits 4-by-4 starting from the right and convert into the corresponding hexadecimal symbol

1 0 1 1 0 1

0 0 1 0	1 1 0 1
---------	---------

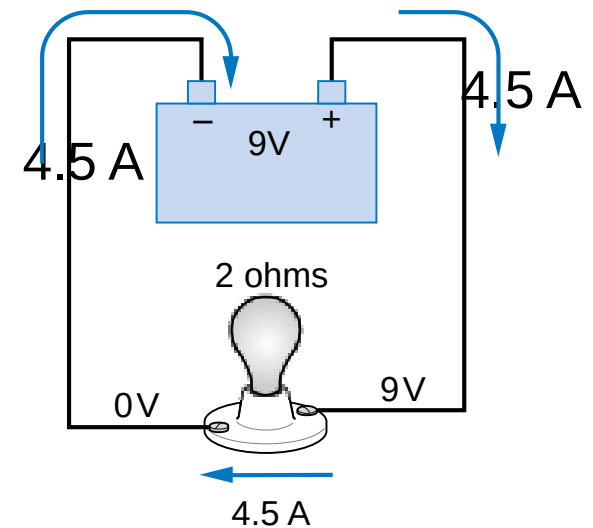
2 D



Combinational circuit building blocks: Switches & CMOS transistors

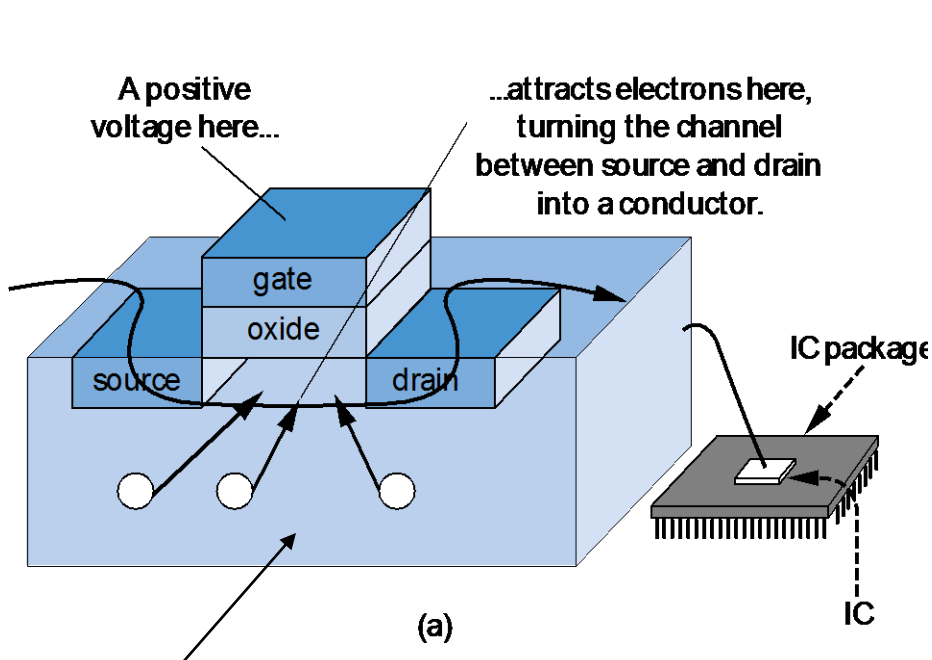
Switches

- Electronic switches are the basis of binary digital circuits
 - Electrical terminology
 - **Voltage**: Difference in electric potential between two points
 - Analogous to water pressure
 - **Current**: Flow of charged particles
 - Analogous to water flow
 - **Resistance**: Tendency of wire to resist current flow
 - Analogous to water pipe diameter
 - $V = I * R$ (Ohm's Law)



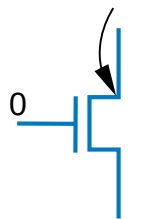
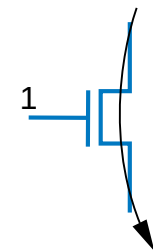
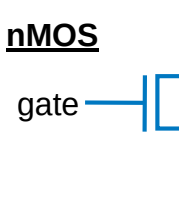
CMOS Switches

- CMOS circuit
 - Consists of N and PMOS transistors
 - Both N and PMOS are similar to basic switches



Silicon -- not quite a conductor or insulator:
Semiconductor

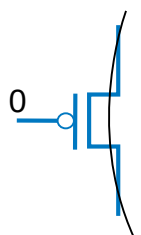
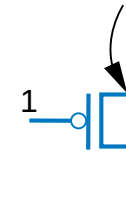
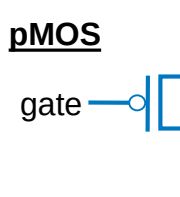
nMOS



conducts

does not conduct

pMOS



does not conduct

conducts

Transistor Circuit Design

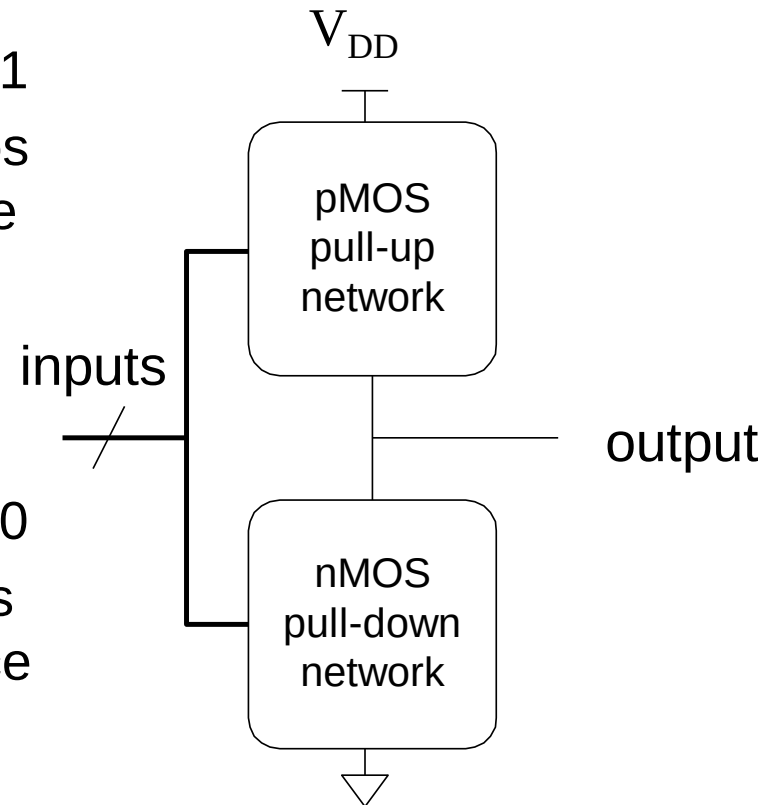
- **nMOS:**

- Turns on when gate is connected to 1
- When turned on, nMOS passes zeros well, but not ones, so connect source to GND
- nMOS forms a pull-down network

- **pMOS:**

- Turns on when gate is connected to 0
- When turned on, pMOS passes ones well, but not zeros, so connect source to V_{DD}
- pMOS forms a pull-up network

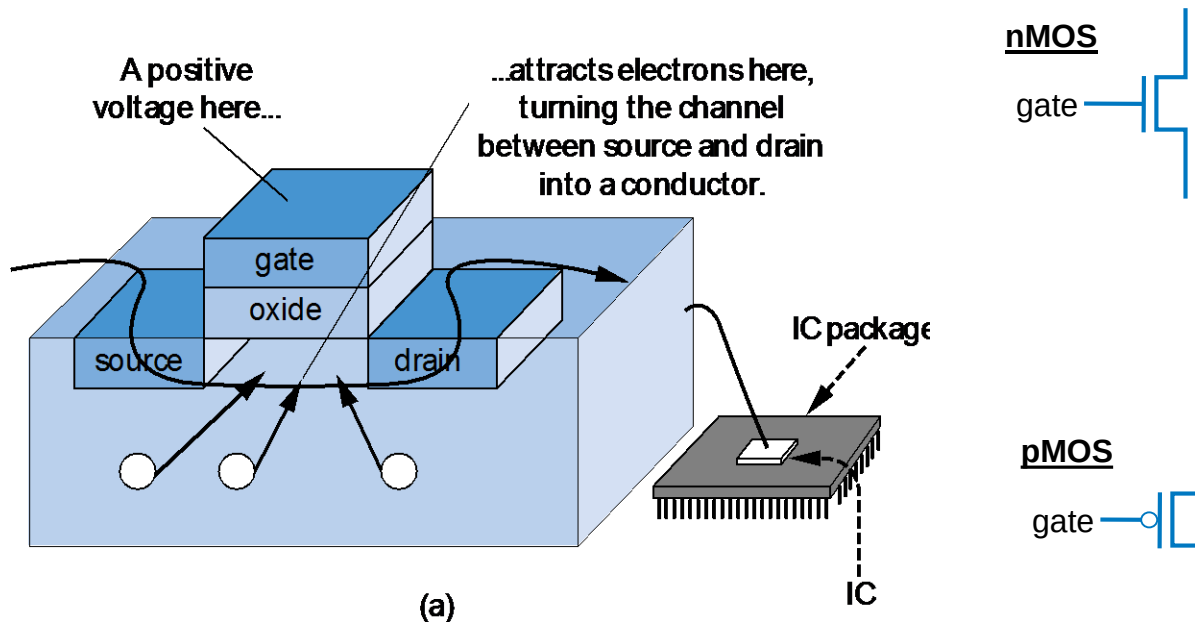
- **Note:** Vahid's textbook shows some circuits with pMOS connected to GND and nMOS to Vdd: this is NOT normally done in practice!



CMOS Switches

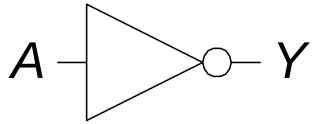
The following is true for CMOS switches:

- A. nMOS turns on when gate is connected to logic 1
- B. pMOS is an open switch when gate is connect to logic 1
- C. All of the above
- D. None of the above



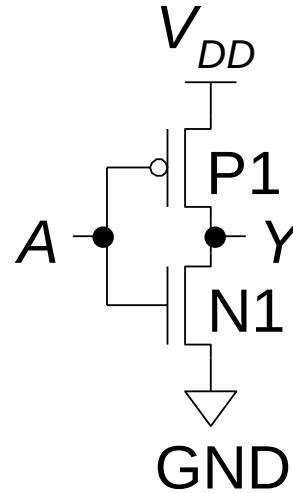
Logic gates: CMOS NOT Gate

NOT



$$Y = \overline{A}$$

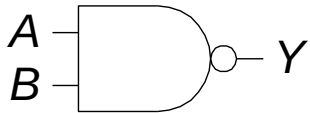
A	Y
0	1
1	0



A	P1	N1	Y
0			
1			

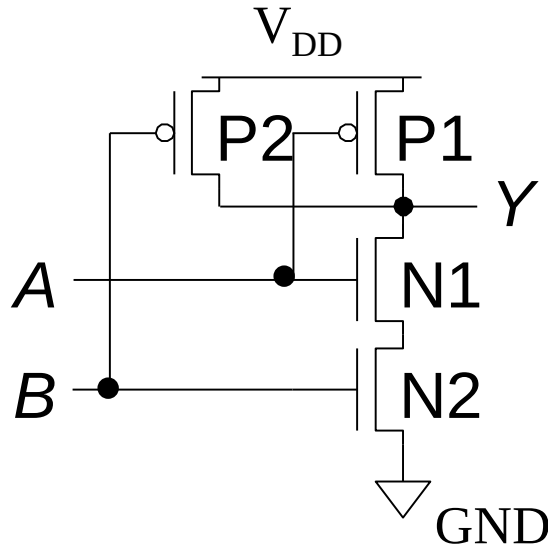
CMOS Two Input NAND Gate

NAND



$$Y = \overline{AB}$$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

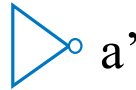
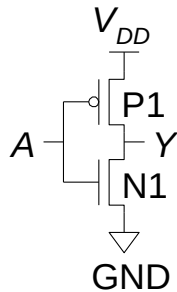


A	B	P1	P2	N1	N2	Y
0	0					
0	1					
1	0					
1	1					

BREAK !



Common Logic Gates



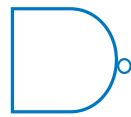
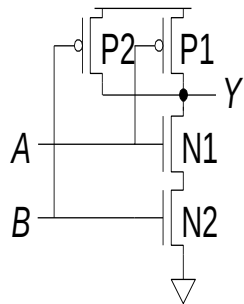
a'

a	NOT
0	1
1	0



a

a	BUF
0	
1	



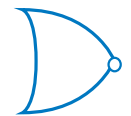
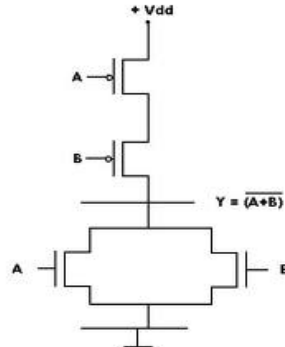
$(a \cdot b)'$

a	b	NAND
0	0	1
0	1	1
1	0	1
1	1	0



$a \cdot b$

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



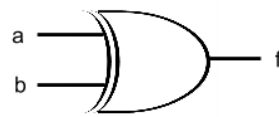
$(a+b)'$

a	b	NOR
0	0	1
0	1	0
1	0	0
1	1	0



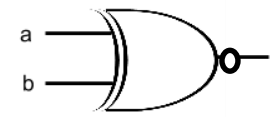
$a+b$

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



$ab' + a'b$

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



$ab + a'b'$

a	b	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

Boolean algebra

- $B = \{0, 1\}$
- Variables represent 0 or 1 only
- Operators return 0 or 1 only
- Basic operators
 - Intersection: $\cdot$ is logical **AND**: $a \text{ AND } b$ returns 1 only when $a=1$ & $b=1$
 - Union: $+$ is logical **OR**: $a \text{ OR } b$ returns 1 if either $a=1$ or $b=1$ (or both)
 - Complement: $'$ is logical **NOT**: $\text{NOT } a$ returns the opposite of a

AND



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

$a \cdot b$

OR



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

$a + b$

NOT



a'

a	NOT
0	1
1	0

BUFFER



a

a	BUF
0	0
1	1

– Derived operators:

NAND



a	b	NAND
0	0	1
0	1	1
1	0	1
1	1	0

$(a \cdot b)'$

NOR



a	b	NOR
0	0	1
0	1	0
1	0	0
1	1	0

$(a + b)'$

XOR

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

XNOR

a	b	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra

X and Y are Boolean variables with $X=1$, $Y=0$

What is $X+X+Y$?

A. 0

B. 1

C. 2

D. None of the above

Universal Gate: NAND

Any logic function can be implemented using just **NAND** gates. Boolean algebra's basic operators are AND, OR and NOT



$$Q = \text{NOT}(A \text{ AND } B)$$

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

Desired NOT Gate



$$Q = \text{NOT}(A)$$

NAND Construction



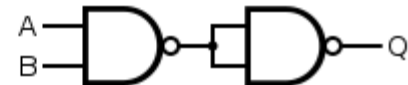
$$= \text{NOT}(A \text{ AND } A)$$

Desired AND Gate



$$Q = A \text{ AND } B$$

NAND Construction



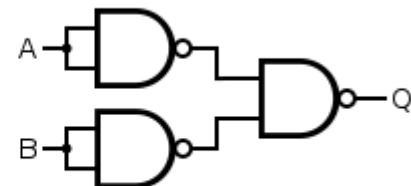
$$= \text{NOT}[\text{NOT}(A \text{ AND } B) \text{ AND } \text{NOT}(A \text{ AND } B)]$$

Desired OR Gate



$$Q = A \text{ OR } B$$

NAND Construction



$$= \text{NOT}[\text{NOT}(A \text{ AND } A) \text{ AND } \text{NOT}(B \text{ AND } B)]$$

Universal Gate: NOR

Any logic function can be implemented using just **NOR gates**.
Boolean algebra needs AND, OR and NOT



Truth Table

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

Desired Gate



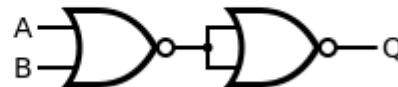
NOR Construction



Desired Gate



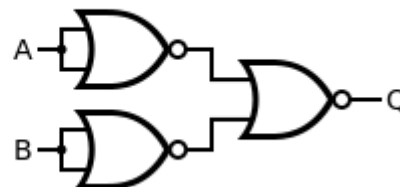
NOR Construction



Desired Gate



NOR Construction



Boolean Axioms & Theorems

Axiom		Dual		Name
A1	$B = 0 \text{ if } B \neq 1$	A1'	$B = 1 \text{ if } B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2'	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3'	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4'	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5'	$1 + 0 = 0 + 1 = 1$	AND/OR

Theorem		Dual		Name
T1	$B \bullet 1 = B$	T1'	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2'	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3'	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5'	$B + \overline{B} = 1$	Complements

Boolean theorems of multiple variables

Theorem		Dual		Name
T6	$B \bullet C = C \bullet B$	T6'	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7'	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + B \bullet D = B \bullet (C + D)$	T8'	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9'	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10'	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D)$ $= B \bullet C + \overline{B} \bullet D$	T11'	$(B + C) \bullet (\overline{B} + D) \bullet (C + D)$ $= (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \dots}$ $= (\overline{B_0} + \overline{B_1} + \overline{B_2} \dots)$	T12'	$\overline{B_0 + B_1 + B_2 \dots}$ $= (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2})$	De Morgan's Theorem

Boolean Duality

- Derived by replacing $\cdot$ by $+$, $+$ by $\cdot$, 0 by 1 , and 1 by 0 & leaving variables unchanged

$$X + Y + \dots \Leftrightarrow X \cdot Y \cdot \dots$$

- Generalized duality:

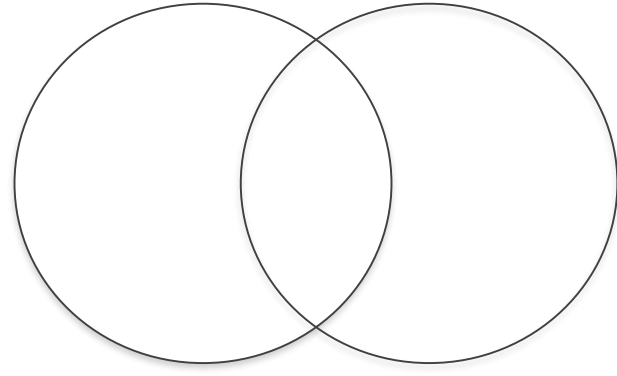
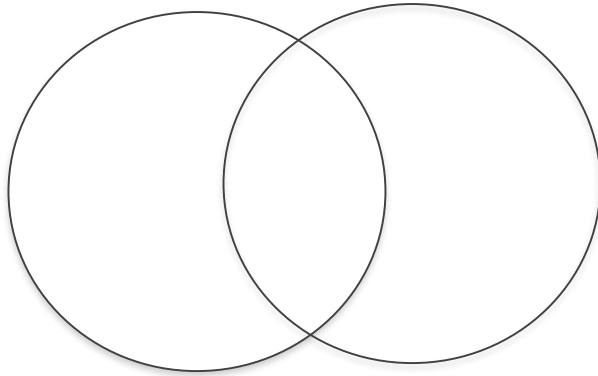
$$f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot) \Leftrightarrow f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$$

- Any theorem that can be proven is also proven for its dual! Note: this is NOT deMorgan's Law

Covering Theorem Explained

- Covering Theorem:

$$A^*(A+B) = A+A^*B = A$$

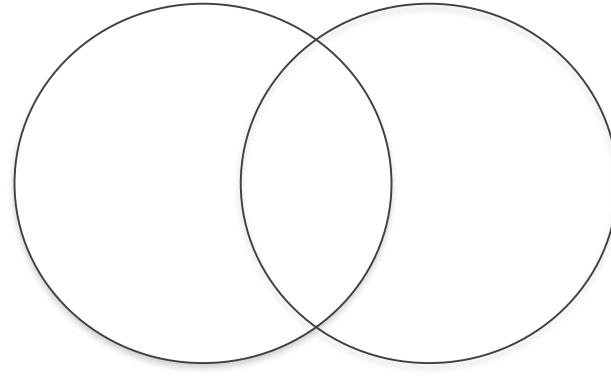
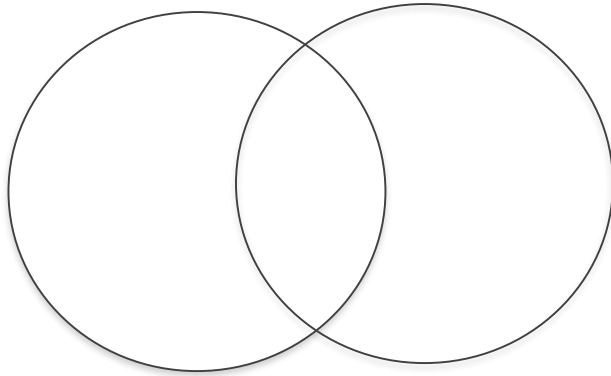


Venn Diagrams

Combining Theorem Explained

- Combining Theorem:

$$AB + AB' = (A+B)(A+B') = A$$



Proving theorems with Boolean Algebra

- Using the axioms of Boolean algebra:
 - e.g., prove the consensus theorem: $X \cdot Y + X \cdot Y' = X$

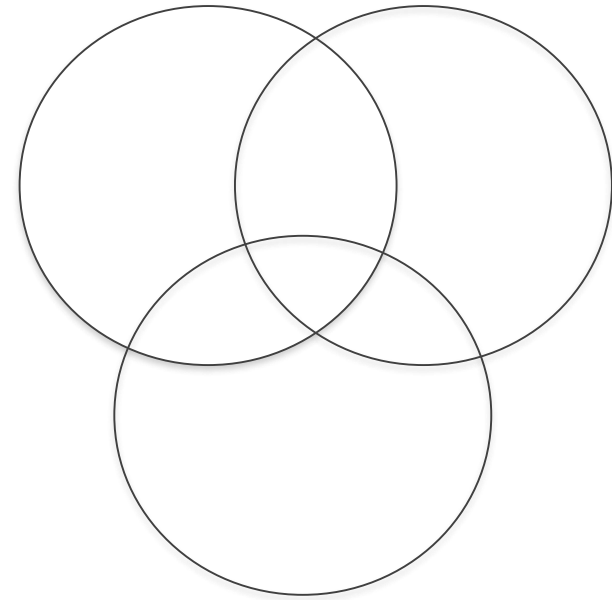
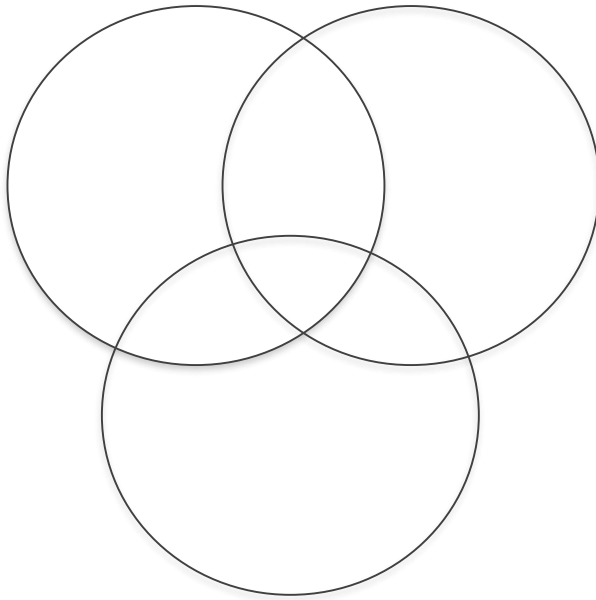
distributivity	$X \cdot Y + X \cdot Y'$	$=$	$X \cdot (Y + Y')$
complementarity	$X \cdot (Y + Y')$	$=$	$X \cdot (1)$
identity	$X \cdot (1)$	$=$	$X \checkmark$

- e.g., prove the covering theorem: $X + X \cdot Y = X$

identity	$X + X \cdot Y$	$=$	$X \cdot 1 + X \cdot Y$
distributivity	$X \cdot 1 + X \cdot Y$	$=$	$X \cdot (1 + Y)$
identity	$X \cdot (1 + Y)$	$=$	$X \cdot (1)$
identity	$X \cdot (1)$	$=$	$X \checkmark$

Consensus Theorem of 3 Variables

- Consensus Theorem: $AB + B'C + AC = AB + B'C$



Proof of Consensus Theorem with Boolean Algebra

- Consensus Theorem:

- $(X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = X \cdot Y + X' \cdot Z$

$$(X \cdot Y) + (Y \cdot Z) + (X' \cdot Z)$$

identity

$$(X \cdot Y) + (1) \cdot (Y \cdot Z) + (X' \cdot Z)$$

complementarity

$$(X \cdot Y) + (X' + X) \cdot (Y \cdot Z) + (X' \cdot Z)$$

distributivity

$$(X \cdot Y) + (X' \cdot Y \cdot Z) + (X \cdot Y \cdot Z) + (X' \cdot Z)$$

commutativity

$$(X \cdot Y) + (X \cdot Y \cdot Z) + (X' \cdot Y \cdot Z) + (X' \cdot Z)$$

factoring

$$(X \cdot Y) \cdot (1 + Z) + (X' \cdot Z) \cdot (1 + Y)$$

null

$$(X \cdot Y) \cdot (1) + (X' \cdot Z) \cdot (1)$$

identity

$$(X \cdot Y) + (X' \cdot Z) \checkmark$$

Applying Boolean Algebra Theorems



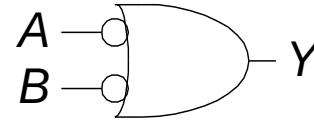
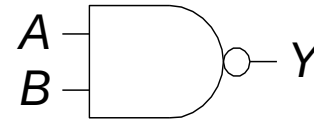
Which of the following is $CB+BA+C'A$ equal to?

- A. $AB+AC'$
- B. $BC+AC'$
- C. $AB+BC$
- D. None of the above

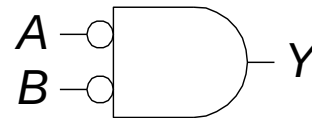
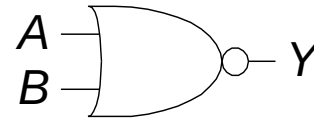
DeMorgan's Theorem

(Bubble Pushing)

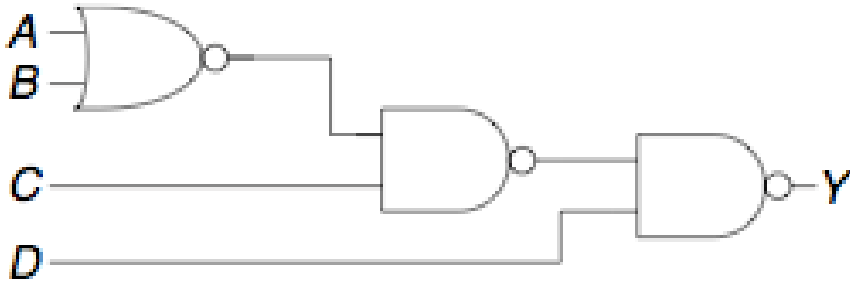
- $Y = \overline{AB} = \overline{A} + \overline{B}$



- $Y = \overline{A + B} = \overline{A} \cdot \overline{B}$



Example of Transforming Circuits with Bubble Pushing



Implement using only NORs

- Hint: DeMorgan's Theorem

$$F = X'Y + Z$$

Proving Theorems with Perfect Induction

- Using perfect induction = complete the truth table:
 - e.g., de Morgan's:

$(X + Y)' = X' \cdot Y'$
NOR is equivalent to AND
with inputs complemented

X	Y	X'	Y'	$(X + Y)'$	$X' \cdot Y'$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

$(X \cdot Y)' = X' + Y'$
NAND is equivalent to OR
with inputs complemented

X	Y	X'	Y'	$(X \cdot Y)'$	$X' + Y'$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

Summary



- What we reviewed thus far:
 - Number representations
 - Switches, logic gates
 - Universal gates
 - Boolean algebra
 - Using Boolean algebra to simplify Boolean equations
 - There is an easier way!
- What is next:
 - Combinational logic:
 - Minimization (the easier way)
 - Designs of common combinational circuits
 - Adders, comparators, subtractors, multipliers, etc.

Historical Note

Claude Shannon:

- In 1936, Shannon began his graduate studies in electrical engineering at MIT
- While studying the complicated *ad hoc* circuits of this analyzer, Shannon designed switching circuits based on Boole's concepts.
- In 1937, he wrote his master's degree thesis, *A Symbolic Analysis of Relay and Switching Circuits*,^[9]
- A paper from this thesis was published in 1938.^[10] In this work, Shannon proved that his switching circuits could be used to simplify the arrangement of the electromechanical relays that were used then in telephone call routing switches. Next, he expanded this concept, proving that these circuits could solve all problems that Boolean algebra could solve. In the last chapter, he presents diagrams of several circuits, including a 4-bit full adder.^[9]
- Using this property of electrical switches to implement logic is the fundamental concept that underlies all electronic digital computers

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

<http://www.maa.org/press/periodicals/convergence/applications-of-boolean-algebra-claude-shannon-and-circuit-design>