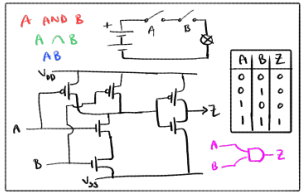
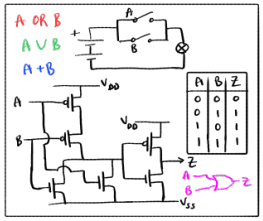
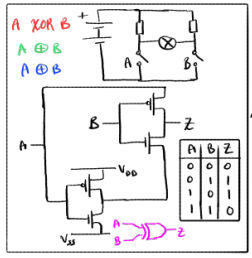
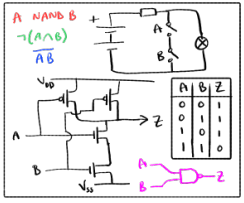
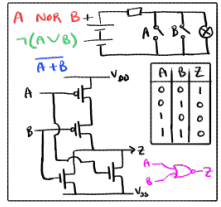


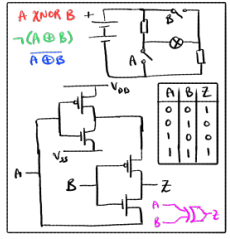






NOTATION
Plain English
Propositional logic
Boolean logic
Many use this
...but I prefer this (Hunter)





"Never leave wires undriven (Make sure there's always a path to V_{DD} or GND"
-Univ. of Texas at Austin

Circuits can be expressed in boolean algebra. Complex circuits can then be simplified hence using less gates, therefore saving resources.

Transistor types

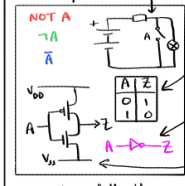
Diagrams use Enhancement-mode MOSFETs

Definition
"Trans" = Change
"Resistor" = Resistance
∴ "Change resistor"
∴ Electrically controlled switches

- Transistor
  - BJTs
    - NPN
    - PNP
  - FETs ("Field Effect Transistors" } To do w/ electric field ∴ Voltage-controlled
    - JFETs
      - Depletion mode
        - N-Channel
        - P-Channel
    - MOSFETs
      - Depletion mode
        - N-Channel
        - P-Channel
      - Enhancement mode
        - N-Channel
        - P-Channel
    - IGBTs

Computers also tend to use MOSFETs

V_{DD} = Drain
V_{SS} = Source



Truth table
Boolean circuit
Transistor circuit

Using full adders, binary numbers of any length can be added.

How the transistor circuits form due to boolean operations.

Create NMOS & PMOS through DOPING
↳ topic for more research

NMOS
If A is "on" then NMOS turns on it is "active-high" Allows current if input is high.

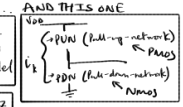
PMOS
If A is "on" then PMOS turns off it is "active-low" Allows current if input is low.

CMOS LOGIC

HOW THEY BASICALLY WORK

KNOW THIS TABLE ... AND THIS ONE

	NMOS	PMOS
+/OR	Parallel	Series
/AND	Series	Parallel



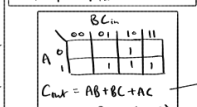
FULL ADDER TRUTH TABLE

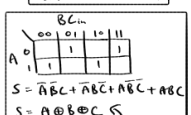
A	B	C_{in}	S	C_{out}	$\bar{C}_{out}$
0	0	0	0	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	0

$C_{in}\bar{C}_{out}$
$B\bar{C}_{out}$
$A\bar{C}_{out}$
ABC_{in}

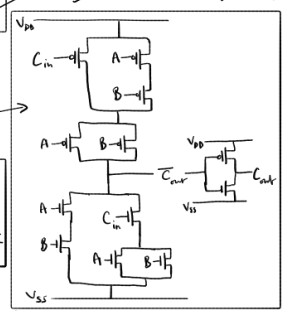
∴ A method for less transistors (saves money)

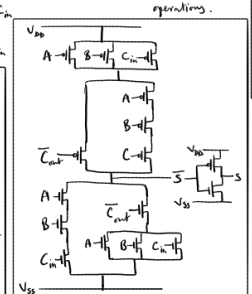
$S = C_{in}\bar{C}_{out} + B\bar{C}_{out} + A\bar{C}_{out} + ABC_{in}$
$= \bar{C}_{out}(A+B+C_{in}) + ABC_{in}$



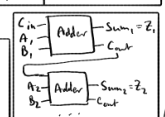


But more transistors needed (See method for less transistors above)





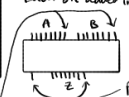
Topics for further research
- Binary decoders
- Registers & Assembly language
- CMOS
- Karnaugh maps



Adding 2 bytes
0 1 0 1 1 1 0 1
eg: + 1 0 0 0 1 0 1 0
1 1 1 0 0 1 1 1 1
Sum
carry

This is done through transistors; changing inputs (A & B) quickly produces an output (because electricity is fast).

Each bit added individually



For overflow

8 Bit Adders

Through similar logic, subtractors, incrementors and other operations can be carried out.

Such components tend to be stored in the ALU

Boolean Algebra (To simplify circuits)

Commutative
- $AB = BA$
- $A+B = B+A$

Associative
- $(A+B)+C = A+(B+C)$
- $(AB)C = A(BC)$

Distributive
- $A(B+C) = AB+AC$
- $A+BC = AB+AC$

De Morgan's
- $\overline{A+B} = \bar{A}\bar{B}$
- $\overline{AB} = \bar{A}+\bar{B}$

Identity
- $A+0 = A$
- $A(1) = A$

Annulment
- $A+1 = 1$
- $A(0) = 0$

Idempotent
- $A+A = A$
- $AA = A$

Complement
- $A+\bar{A} = 1$
- $A\bar{A} = 0$

Involution
- $\overline{(\bar{A})} = A$

Absorption
- $A+AB = A$
- $A(A+B) = A$

Consensus Theorem
- $AB+\bar{A}C+BC = AB+\bar{A}C$
- $(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$