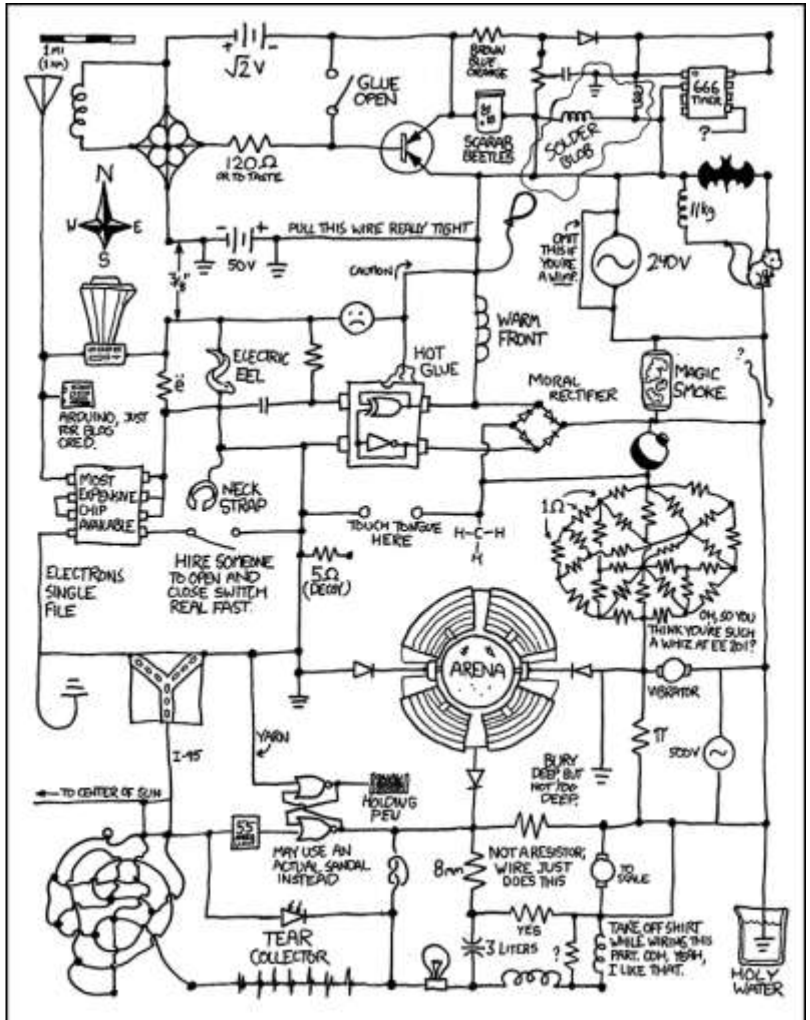
Circuits & Boolean algebra



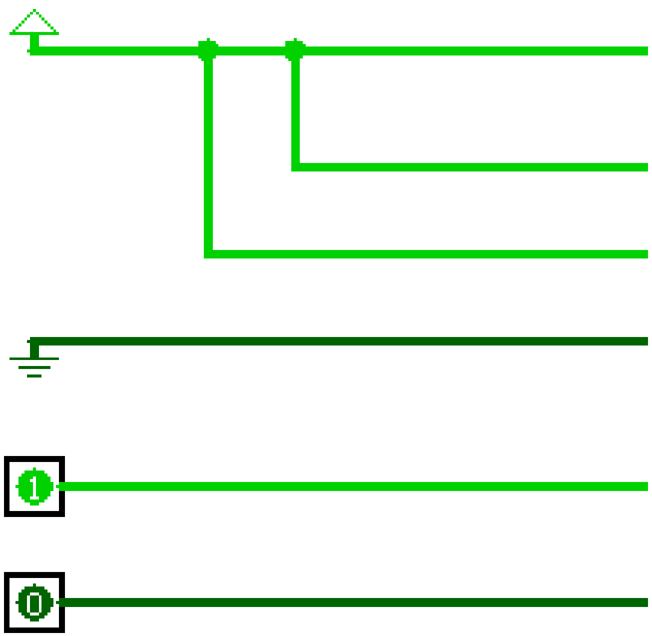
http://xkcd.com/730/

Overview

- Digital circuits
  - How a switch works
  - Building basic gates from switches
- Boolean algebra
  - Sum-of-products notation
  - Rules of Boolean algebra
  - Minimization

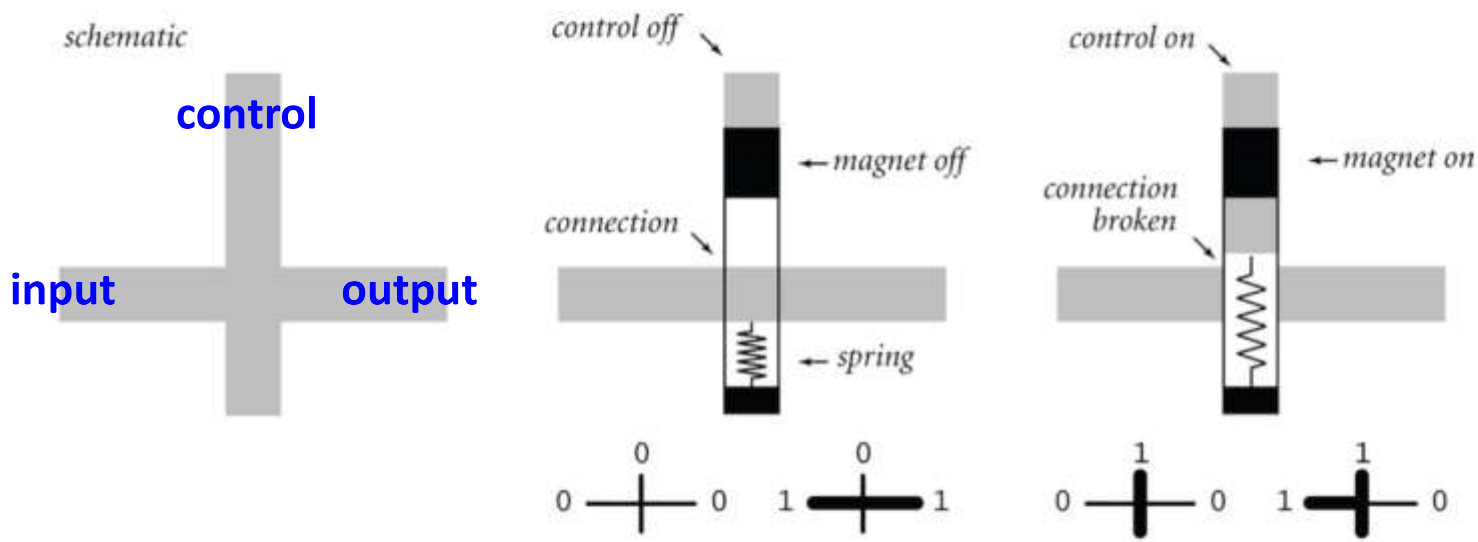
Digital circuits

- Building blocks:
  - Wires
    - Propagate an ON/OFF value
    - 1 = connected to power
    - 0 = not connect to power
    - Any wire connected to a wire that is on is also on
  - Switches
    - Controls propagation of an ON/OFF value through a wire



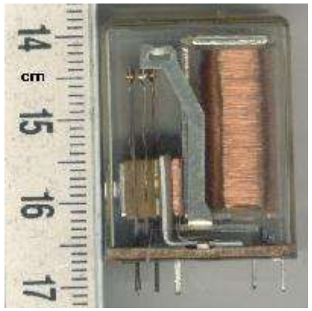
Controlled switch

- Building a switch
  - 3 connections: input, output, control
  - control = OFF, input connected to output



Anatomy of a relay (controlled switch)

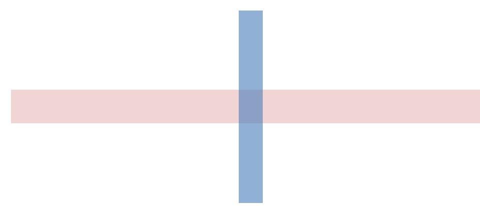
Switch types



Relay

Vacuum tube

Transistor



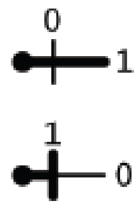
Pass transistor

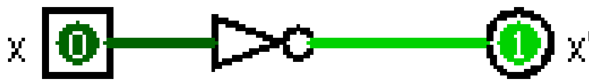
Logic gates

- Build NOT, OR, AND gates from switches

NOT = x'

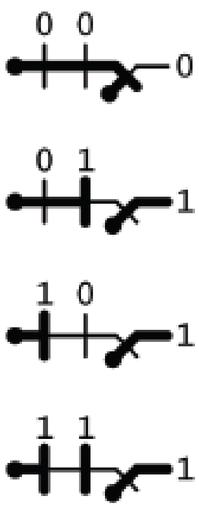
x	NOT
0	1
1	0

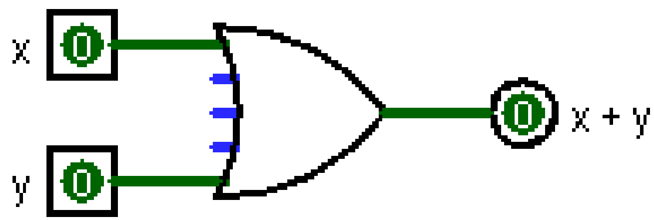




OR = x + y

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

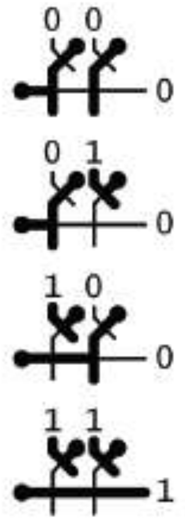


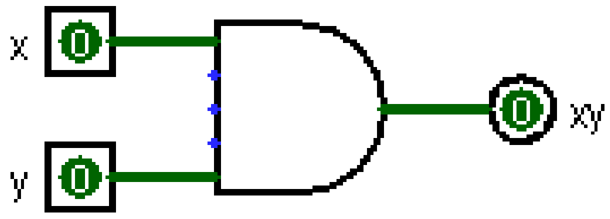


Logic gates

AND = xy

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

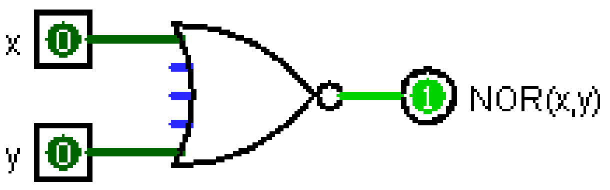




Inverted gate variants

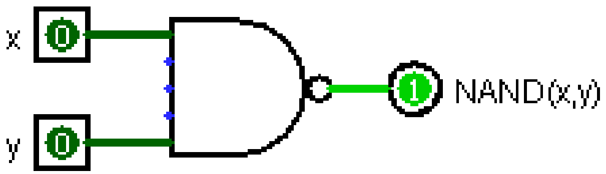
NOR(x,y)

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



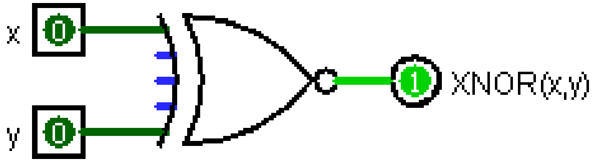
NAND(x,y)

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0



XNOR(x,y)

x	y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1



Boolean algebra

- Boolean algebra
  - Every variable is either 0 or 1
  - Functions whose inputs and outputs are 0 or 1
- Relationship to circuits:
  - Boolean variable = signal (ON/OFF)
  - Boolean function = circuit made of gates & wires
- Relationship to truth tables:
  - Systematic way to represent any Boolean function
  - One row for any input combination

2 variable truth tables

- Given 2 variables, how many possible Boolean functions?

x	y	function 1	function 2	…	function N
0	0				
0	1				
1	0				
1	1				

All 2 variable Boolean functions

x	y	ZERO	AND		x		y	XOR	OR
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

x	y	NOR	EQ	y'		x'		NAND	ONE
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

3 variables truth tables

- Given 3 variables, how many total possible Boolean functions?

x	y	z	function 1	function 2	…	function N
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Representing Boolean functions

- 16 $= 2^4$ Boolean functions of 2 variables
- 256 $= 2^8$ Boolean functions of 3 variables
- 65536 $= 2^{16}$ Boolean functions of 4 variables
- 2^{2^n} Boolean functions of n variables!
  - We need a more compact representation

Sum-of-products

- Universality: any Boolean function can be expressed using {AND, OR, NOT}
  - Also universal:

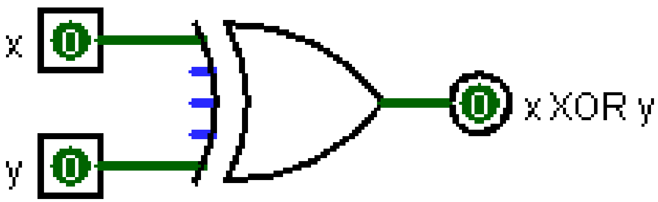
    {AND, NOT}, {OR, NOT},

    {NAND}, {NOR}
- Sum-of-products
  - Create Boolean expression from truth table
  - Form AND term for each 1 in table
  - OR terms together

Sum-of-products: XOR

XOR = x ⊕ y

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0



XOR(x,y) = x'y + xy'

– Form AND term for each 1 in table

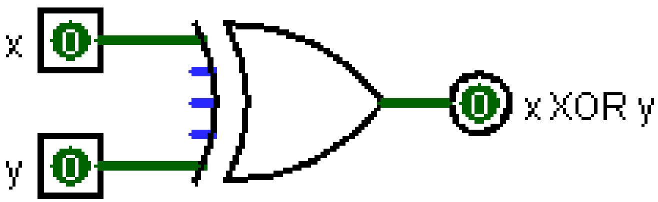
– OR terms together

– Easy to convert to circuit using only AND, OR, NOT

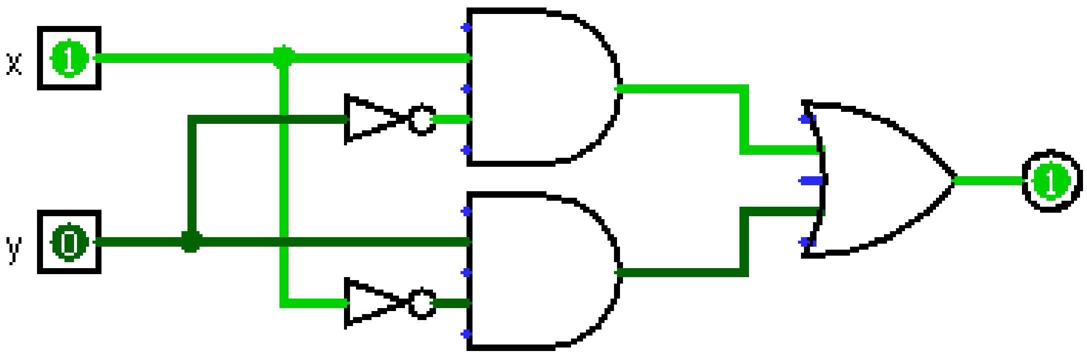
Sum-of-products: XOR

XOR = x ⊕ y

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0



XOR(x,y) = x'y + xy'



Majority function

- Majority function
  - 1 if majority of bits are 1, 0 otherwise

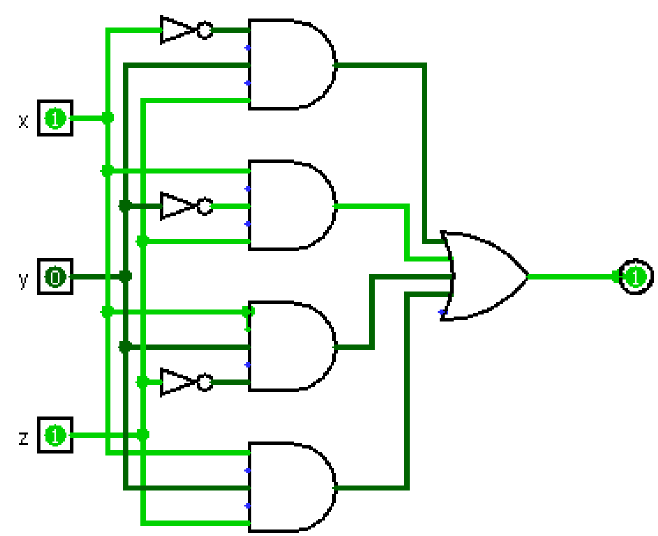
x	y	z	MAJ(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Majority function

x	y	z	MAJ(x,y,z)
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**

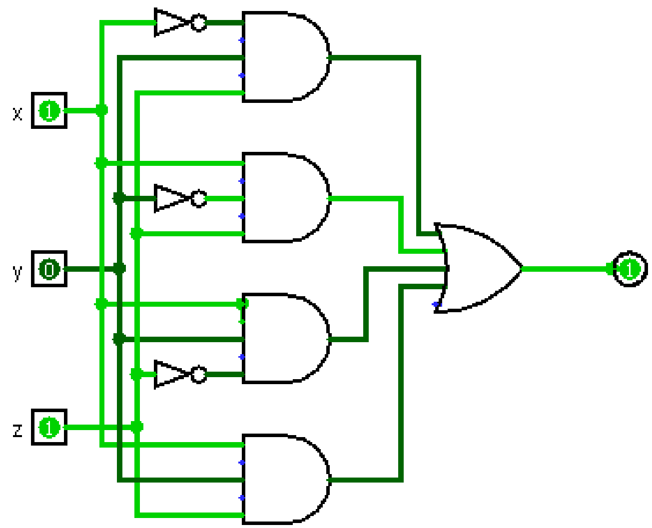
MAJ(x,y,z) = x'yz + xy'z + xyz' + xyz

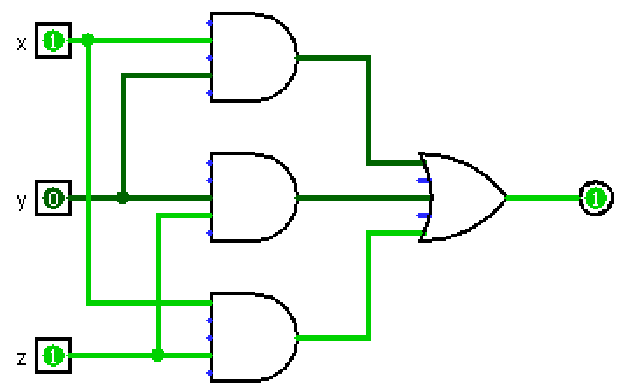
Can we do better?



Minimizing MAJ(x,y,z)

MAJ(x,y,z) = x'yz + xy'z + xyz' + xyz = xy + yz + xz





4, 3-input AND gates
3, NOT gates
1, 4-input OR gate

3, 2-input AND gates
1, 3-input OR gate

Products-of-sums

- Products-of-sums
  - Create Boolean expression from truth table
  - Form OR term for each 0 in table
    - Use X in OR term if X = 0, X' is X = 1
  - AND terms together

Product of sums: Majority

x	y	z	MAJ(x,y,z)
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

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

Comparing POS vs. SOP

- Products-of-sums (POS)
  - Create Boolean expression from truth table
  - Form OR term for each 0 in table
    - Use X in OR term if X = 0, X’ is X = 1
  - AND terms together
- Sum-of-products (SOP)
  - Create Boolean expression from truth table
  - Form AND term for each 1 in table
    - Use X in AND term if X = 1, use X’ if X = 0
  - OR terms together

Rules of Boolean algebra

	(a)	(b)
1. Commutative law	x + y = y + x	xy = yx
2. Associate law	(x + y) + z = x + (y + z)	(xy)z=x(yz)
3. Distributive law	x(y + z) = xy + xz	(x + y)(x + z) = x + yz
4. Identity law	x + x = x	xx = x
5.	xy + xy' = x	(x + y)(x + y') = x
6. Redundance law	x + xy = x	x(x + y) = x
7.	0 + x = x	0x = 0
8.	1 + x = 1	1x = x
9.	x' + x = 1	x'x = 0
10.	x + x'y = x + y	x(x' + y) = xy
11. **De Morgan's Theorem**	**(x + y)' = x'y'**	**(xy)' = x' + y'**

Minimizing MAJ(x, y, z)

MAJ(x, y, z)	$= x'yz + xy'z + xyz' + xyz$	
	$= x'yz + xy'z + xy(z' + z)$	[3a] distributive
	$= x'yz + xy'z + xy$	[9a]
	$= x'yz + xy'z + xy(1 + z)$	[8a]
	$= x'yz + xy'z + xy + xyz$	[3a] distributive
	$= yz(x' + x) + xy'z + xy$	[3a] distributive
	$= yz + xy'z + xy$	[9a]
	$= yz + xy'z + xy(1 + z)$	[8a]
	$= yz + xy'z + xy + xyz$	[3a] distributive
	$= yz + xz(y' + y) + xy$	[3a] distributive
	$= yz + xz + xy$	[9a]

Problem 1: Odd parity function

- Odd parity
  - 1 if odd number of bits are 1
  - Find sum-of-products
  - Draw the circuit

x	y	z	ODD(x,y,z)
0	0	0	0
0	0	1	**1**
0	1	0	**1**
0	1	1	0
1	0	0	**1**
1	0	1	0
1	1	0	0
1	1	1	**1**

Problem 2: Absolute value

- 3-bit number in two's complement
  - Bits = xyz
- Create a truth table for ABS(x,y,z) >= 2
  - ABS() is 1 if and only if absolute value is 2 or more
- Find sum-of-products
- Minimize the Boolean expression
- Draw the circuit

Problem 3:

- Show that {NAND} is universal
  – Hint: show you can build AND, OR, NOT from 1-3 NAND gates
- Show that {NOR} is universal
- Show that {AND, NOT} is universal
  – Hint: Use De Morgan's on sum-of-products to eliminate OR
- Show that {OR, NOT} is universal
  – Hint: Use De Morgan's on products-of-sums to eliminate AND

Summary

- Wires + switches → gates {AND, OR, NOT}
- Truth table → sum-of-products Boolean expression
- Sum-of-products → circuit
- Simplification via rules of Boolean algebra