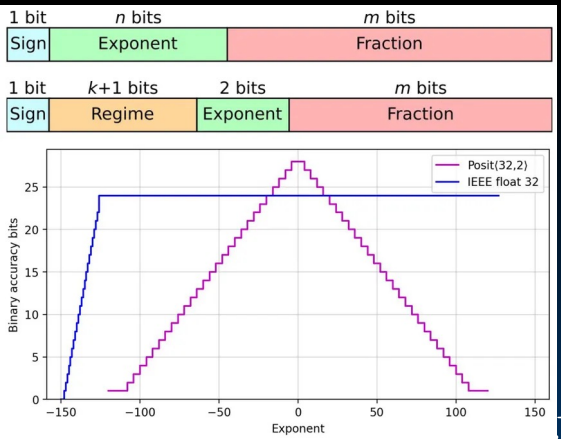
Unum

- **Everything so far has had a fixed set of bits for Exponent and Significand.**
  - What if the field widths were variable?
  - Also, add a "u-bit" to tell whether number is exact or in-between unums.
  - "Promises to be to floating point what floating point is to fixed point."
- **Claims to save power!**
- **Posits : hardware-friendly version of unums**

Dr. John Gustafson

https://en.wikipedia.org/wiki/Unum_(number_format)

Posits, a New Kind of Number, Improves the Math of AI



Scientists at Spain's Complutense University of Madrid have developed a processor core that can deploy the posit standard, a numerical representation that improves over standard floating-point arithmetic processors, in hardware. Complutense researchers synthesized the hardware in a field-programmable gate array to compare computations performed using 32-bit floats alongside 32-bit posits. Posits exhibited a four-order-of-magnitude improvement in matrix multiplication accuracy without losing computation time. "It's possible that posits will speed up [artificial intelligence] training because you're not losing as much information on the way," said Complutense's David Mallasén Quintana.

spectrum.ieee.org/floating-point-numbers-posits-processor

← This graph shows components of floating-point-number representation [top] and posit representation [middle]. The accuracy comparison shows posits' advantage when the exponent is close to 0.

Garcia, Yan

UC Berkeley
Teaching Professor
Dan Garcia

CS61C

Great Ideas in Computer Architecture

(a.k.a. Machine Structures)

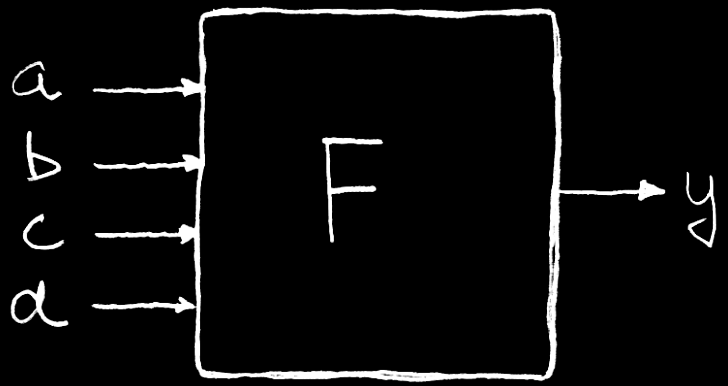
UC Berkeley
Teaching Professor
Lisa Yan

Combinational Logic

Truth Tables

真值表

Truth Tables



a	b	c	d	y
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
0	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1	1	F(1,1,1,1)

How many Fs (4-input devices) @ Fry's?

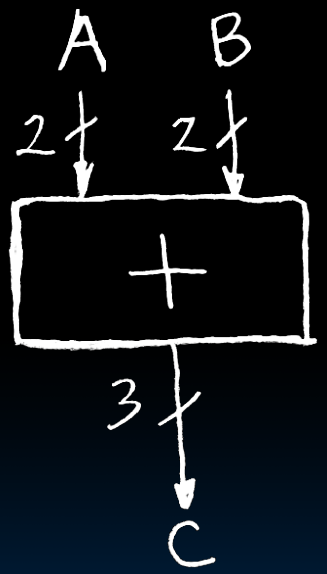
TT Example #1: 1 iff one (not both) a,b=1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{a}b + a\bar{b}$

a	y
0	b
1	$\bar{b}$

TT Example #2: 2-bit adder



A a_1a_0	B b_1b_0	C $c_2c_1c_0$
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

How Many Rows?

TT Example #3: 32-bit unsigned adder

A	B	C
000 ... 0	000 ... 0	000 ... 00
000 ... 0	000 ... 1	000 ... 01
.	.	.
.	.	.
.	.	.
111 ... 1	111 ... 1	111 ... 10

How Many Rows?

TT Example #4: 3-input majority circuit

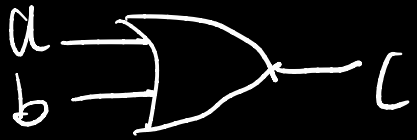
a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logic Gates

Logic Gates (1/2)

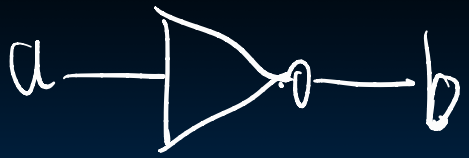
与 AND

ab	c
00	0
01	0
10	0
11	1



或 OR

ab	c
00	0
01	1
10	1
11	1

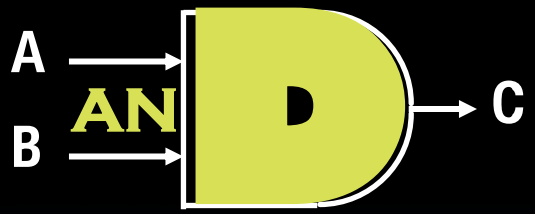


非 NOT

a	b
0	1
1	0

and vs. or … how to recall which is which

and gate symbol



a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

Logic Gates (2/2)

异或

XOR

ab	c
00	0
01	1
10	1
11	0

与非

NAND

ab	c
00	1
01	1
10	1
11	0

或非

NOR

ab	c
00	1
01	0
10	0
11	0

2-input gates extend to n-inputs

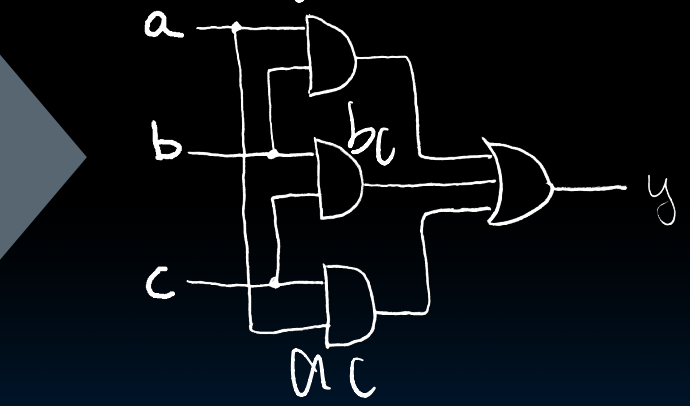
- N-input XOR is the only one which isn't so obvious
- It's actually simple…
  - XOR is a 1 iff the # of 1s at its input is odd

a	b	c	y
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

Truth Table → Gates (e.g., majority circ.)

a	b	c	y
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

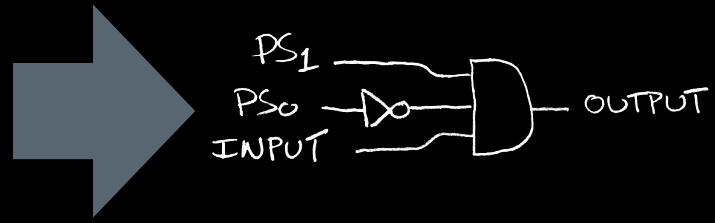
$\bar{a}bc + a\bar{b}c + ab\bar{c} + abc$
$= bc + ac + ab$



$ab + bc + ac$

Truth Table → Gates (e.g., FSM circuit)

PS	Input	NS	Output
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1

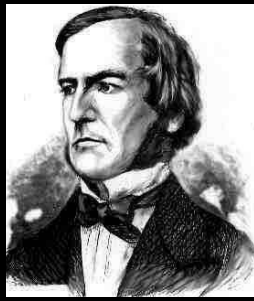


or equivalently…

Boolean Algebra

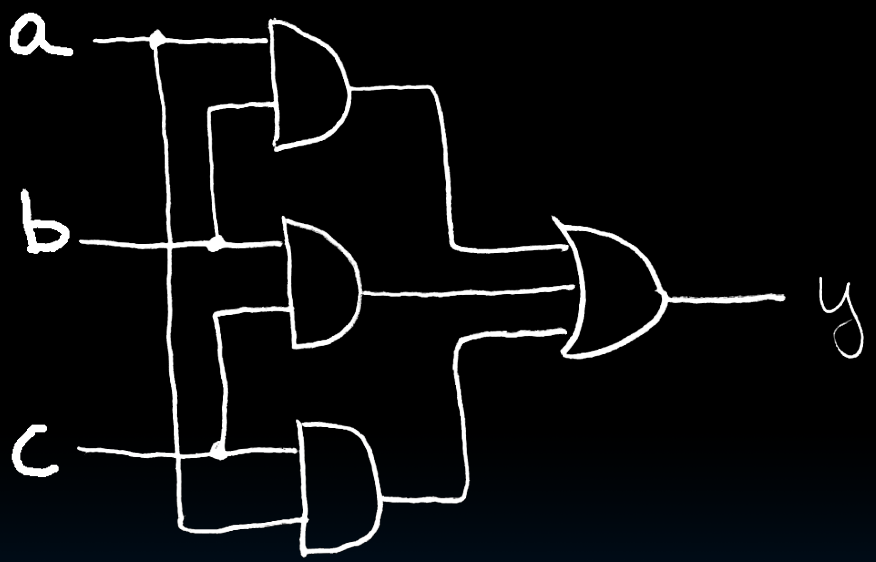
布尔代数

Boolean Algebra



- George Boole, 19th Century mathematician
- Developed a mathematical system (algebra) involving logic
  - later known as "Boolean Algebra"
- Primitive functions: AND, OR and NOT
- Power of Boolean Algebra
  - there's a one-to-one correspondence between circuits made up of AND, OR and NOT gates and equations in BA
- + means OR, • means AND, $\overline{x}$ means NOT

Boolean Algebra (e.g., for majority fun.)

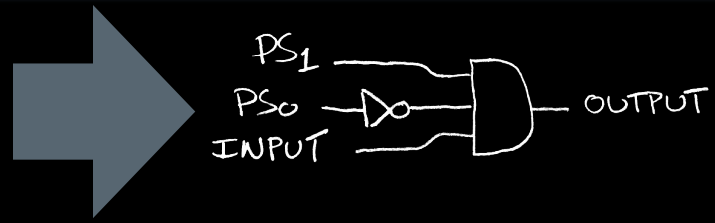


$$y = a \cdot b + a \cdot c + b \cdot c$$

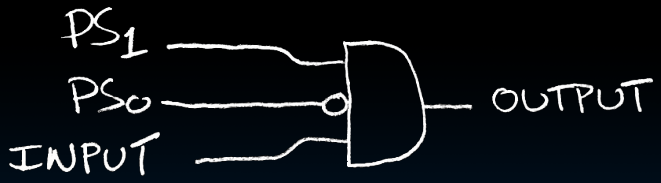
$$y = ab + ac + bc$$

Boolean Algebra (e.g., for FSM)

PS	INPUT	NS	OUTPUT
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1

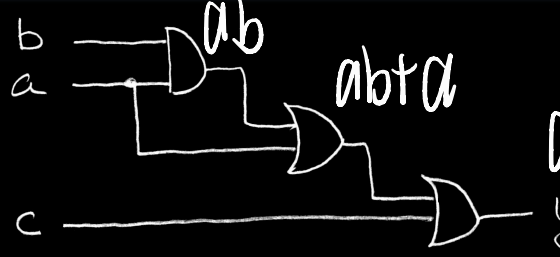


or equivalently…



$$\text{OUTPUT} = PS_1 \bullet \overline{PS_0} \bullet \text{INPUT}$$

BA: Circuit & Algebraic Simplification



$ab + a + c = a(b+1) + c$
$= a + c$

original circuit

$$\downarrow$$

$$y = ab + a + c$$

equation derived from original circuit

$$\downarrow$$

$$ab + a + c$$
$$= a(b+1) + c$$
$$= a(1) + c$$
$$= a + c$$

algebraic simplification

$$\downarrow$$

simplified circuit

BA also great for circuit verification
Circ X = Circ Y? Use BA to prove!

Laws of Boolean Algebra

Laws of Boolean Algebra

$x + yz = (x+y)(x+z)$

$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$	complementarity
$x \cdot 0 = 0$	$x + 1 = 1$	laws of 0's and 1's
$x \cdot 1 = x$	$x + 0 = x$	identities
$x \cdot x = x$	$x + x = x$	idempotent law
$x \cdot y = y \cdot x$	$x + y = y + x$	communitive law
$(xy)z = x(yz)$	$(x + y) + z = x + (y + z)$	associativity
$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$	distribution
$xy + x = x$	$(x + y)x = x$	uniting theorem
$\overline{x \cdot y} = \overline{x} + \overline{y}$	$\overline{(x + y)} = \overline{x} \cdot \overline{y}$	DeMorgan's Law

Boolean Algebraic Simplification Example

$$
\begin{aligned}
y &= ab + a + c \\
&= a(b + 1) + c && \textit{distribution, identity} \\
&= a(1) + c && \textit{law of 1's} \\
&= a + c && \textit{identity}
\end{aligned}
$$

Canonical Forms

Canonical forms (1/2)

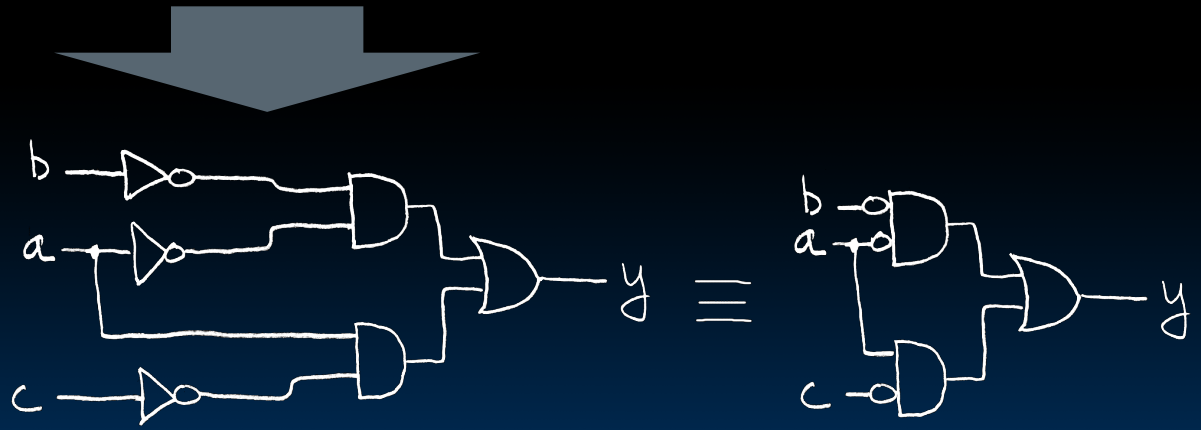
	abc	y
$\overline{a} \cdot \overline{b} \cdot \overline{c}$	000	1
$\overline{a} \cdot \overline{b} \cdot c$	001	1
	010	0
	011	0
$a \cdot \overline{b} \cdot \overline{c}$	100	1
	101	0
$a \cdot b \cdot \overline{c}$	110	1
	111	0

Sum-of-products (ORs of ANDs)

$$y = \overline{a}\overline{b}\overline{c} + \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c}$$

Canonical forms (2/2)

$$
\begin{aligned}
y &= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + a\bar{b}\bar{c} + ab\bar{c} \\
&= \bar{a}\bar{b}(\bar{c} + c) + a\bar{c}(\bar{b} + b) && \textit{distribution} \\
&= \bar{a}\bar{b}(1) + a\bar{c}(1) && \textit{complementarity} \\
&= \bar{a}\bar{b} + a\bar{c} && \textit{identity}
\end{aligned}
$$



L16 Three questions

1) $(a+b) \cdot (\bar{a}+b) = b$
2) N-input gates can be thought of cascaded 2-input gates. I.e.,
   $(a \Delta b \Delta c \Delta d \Delta e) = a \Delta (b \Delta (c \Delta (d \Delta e)))$
   where Δ is one of AND, OR, XOR, NAND
3) You can use NOR(s) with clever wiring to simulate AND, OR, & NOT

F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

Answer

1) $(a+b)\cdot(\overline{a}+b) = a\overline{a}+ab+b\overline{a}+bb = 0+b(a+\overline{a})+b = b+b = b$ TRUE

2) (next slide)

3) You can use NOR(s) with clever wiring to simulate AND, OR, & NOT.

$NOR(a,a) = \overline{a+a} = \overline{a}\cdot\overline{a} = \overline{a}$

Using this NOT, can we make a NOR an OR? An And? TRUE

1) $(a+b)\bullet(\overline{a}+b) = b$

2) N-input gates can be thought of cascaded 2-input gates. I.e.,
$(a \Delta b \Delta c \Delta d \Delta e) = a \Delta (b \Delta (c \Delta (d \Delta e)))$
where Δ is one of AND, OR, XOR, NAND

3) You can use NOR(s) with clever wiring to simulate AND, OR, & NOT

123
FFF
FFT
FTF
FTT
TFF
TFT
TTF
TTT

Answer (for question 2)

1)

2) N-input gates can be thought of cascaded 2-input gates. I.e.,
(a Δ bc Δ d Δ e) = a Δ (bc Δ (d Δ e))
where Δ is one of AND, OR, XOR, NAND…FALSE

- Let's confirm!

CORRECT 3-input

XYZ	AND	OR	XOR	NAND
000	0	0	0	1
001	0	1	1	1
010	0	1	1	1
011	0	1	0	1
100	0	1	1	1
101	0	1	0	1
110	0	1	0	1
111	1	1	1	0

CORRECT 2-input

YZ	AND	OR	XOR	NAND
00	0	0	0	1
01	0	1	1	1
10	0	1	1	1
11	1	1	0	0

"And In conclusion…"

- Pipeline big-delay CL for faster clock
- Finite State Machines extremely useful
  - You'll see them again in (at least) 151A, 152 & 164
- Use this table and techniques we learned to transform from 1 to another