inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture #15
Representations of Combinatorial Logic Circuits

No CPS today

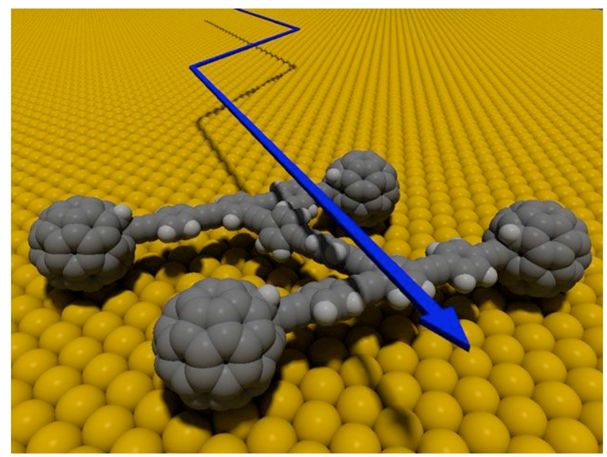
2005-10-24

There is one handout today at the front and back of the room!

Lecturer PSOE, new dad Dan Garcia

www.cs.berkeley.edu/~ddgarcia

World's Smallest Car! ⇒



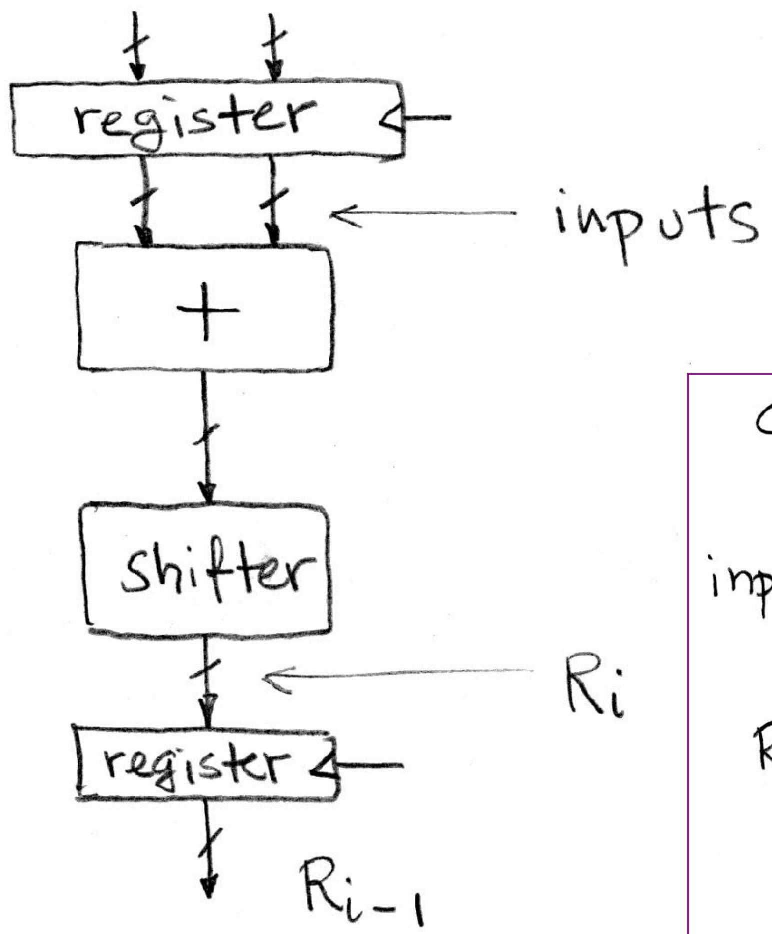
A car only 4nm across was developed by Rice U as a prototype. It actually "rolls on four wheels in a direction perp to its axes". Buckyballs for wheels!

www.livescience.com/technology/051020_nanocar.html

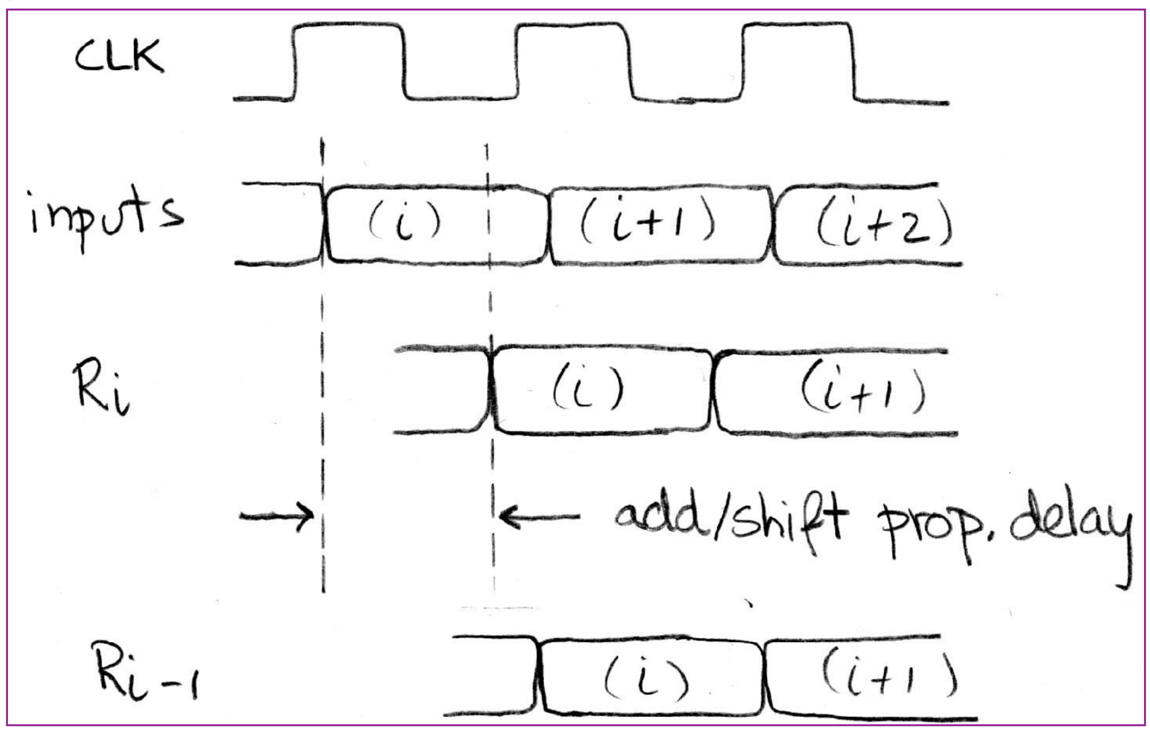
Review

- We use feedback to maintain state
- Register files used to build memories
- D-FlipFlops used to build Register files
- Clocks tell us when D-FlipFlops change
  - Setup and Hold times important
- TODAY
  - Technique to be able to increase clock speed
  - Finite State Machines
  - Representation of CL Circuits
    - Truth Tables
    - Logic Gates
    - Boolean Algebra

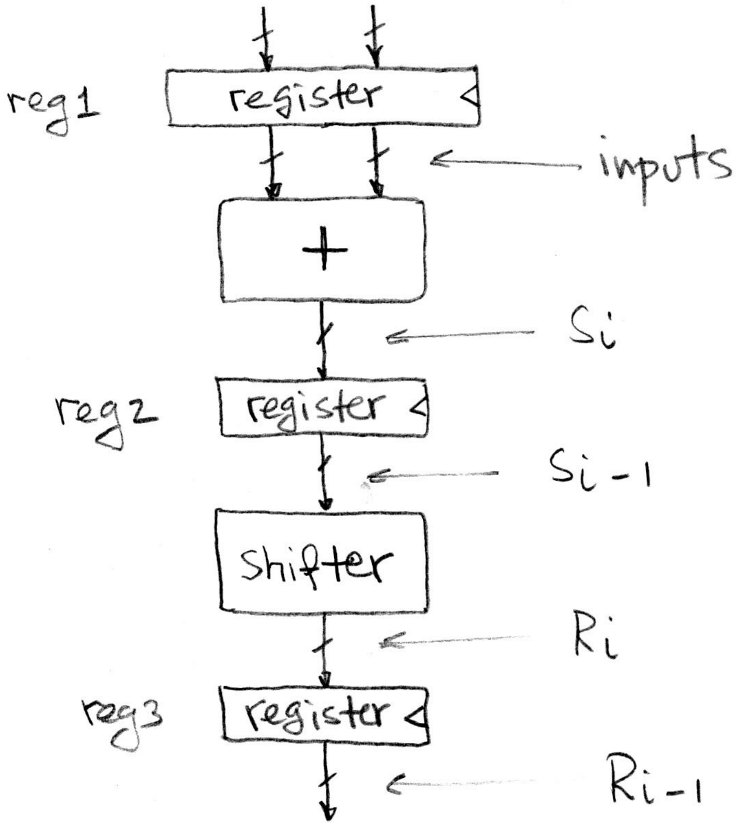
Pipelining to improve performance (1/2)



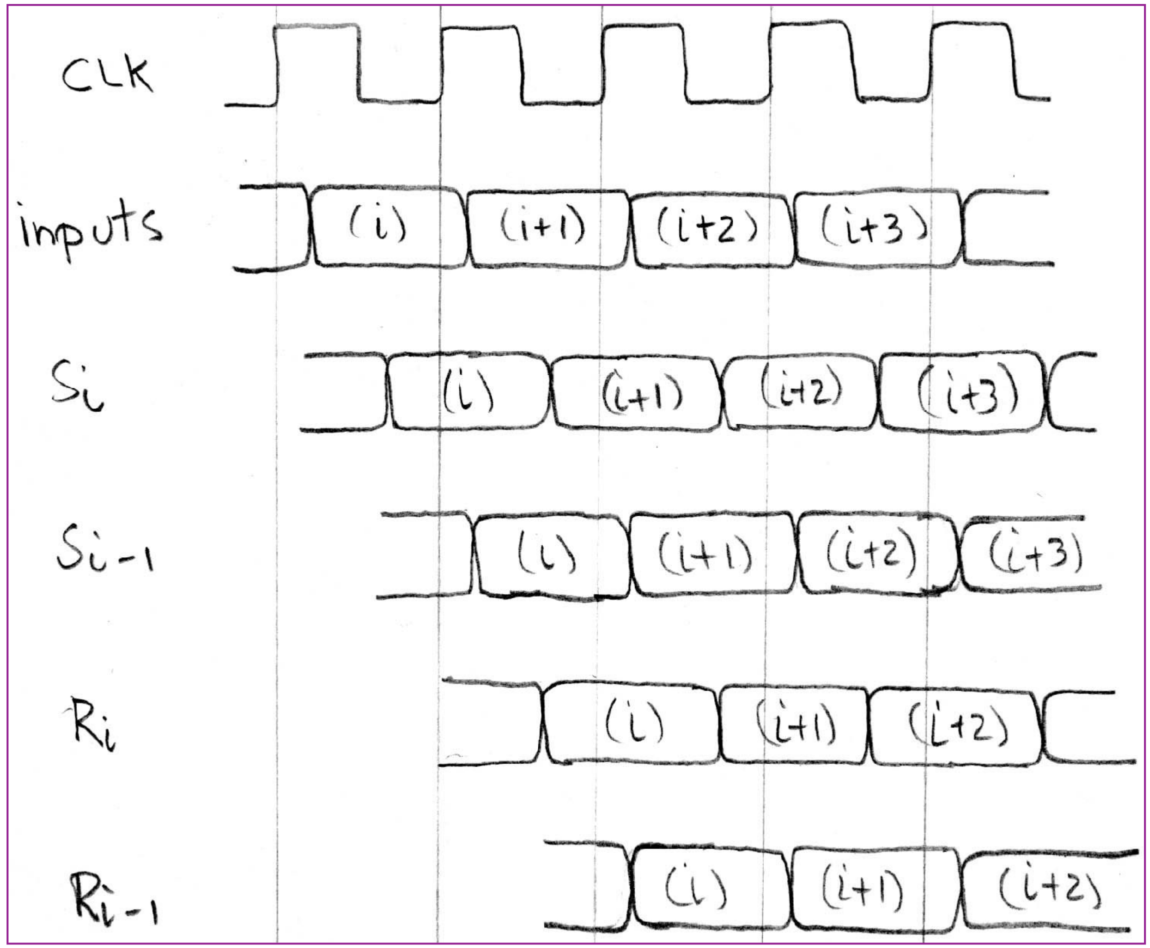
Timing…



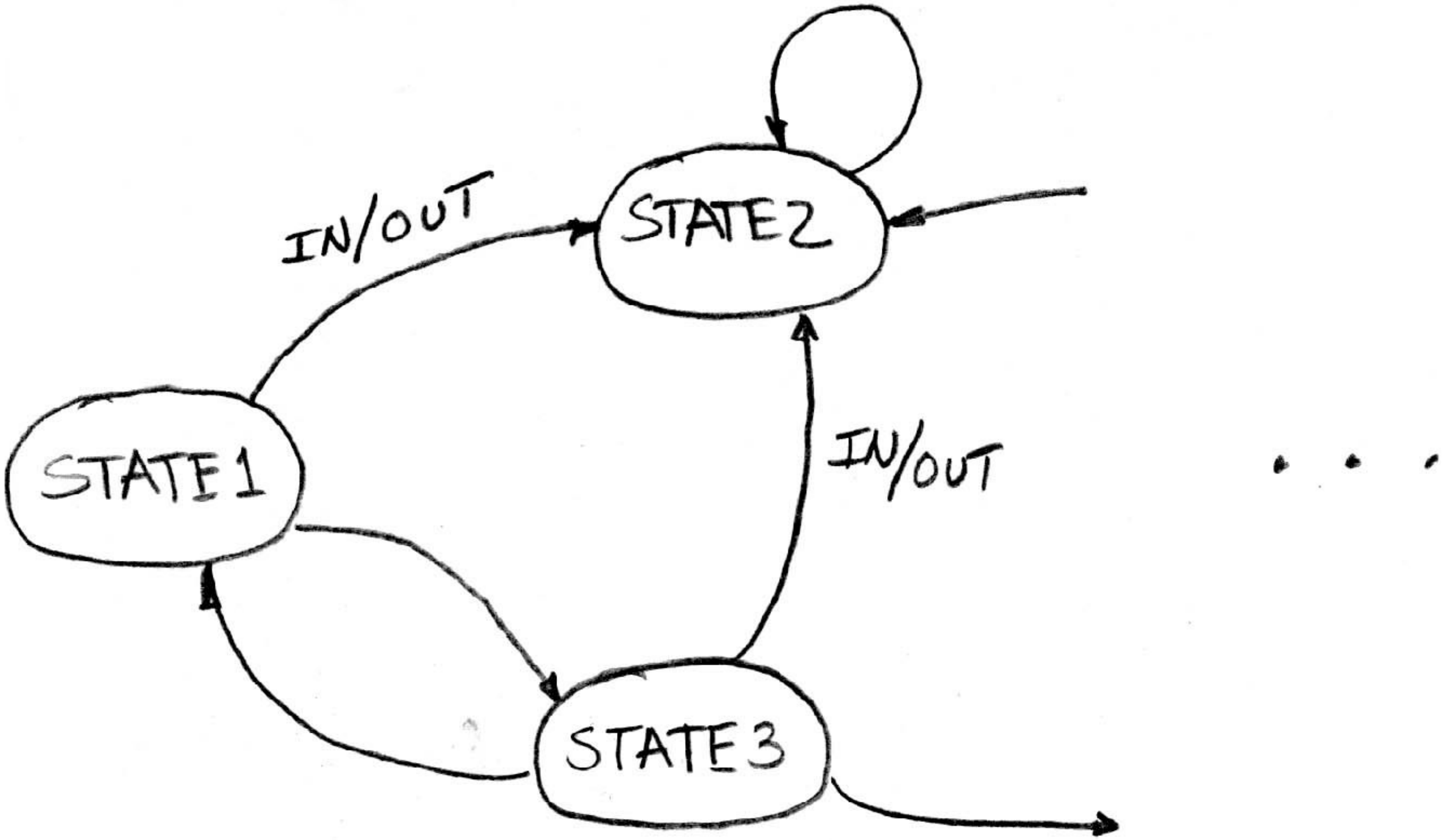
Pipelining to improve performance (2/2)



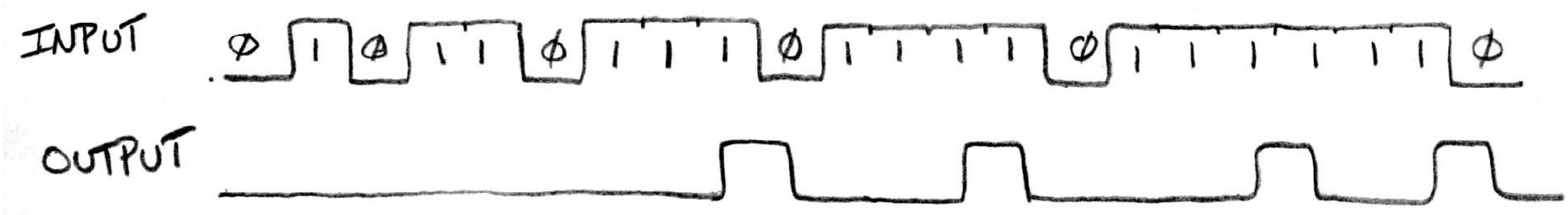
Timing…



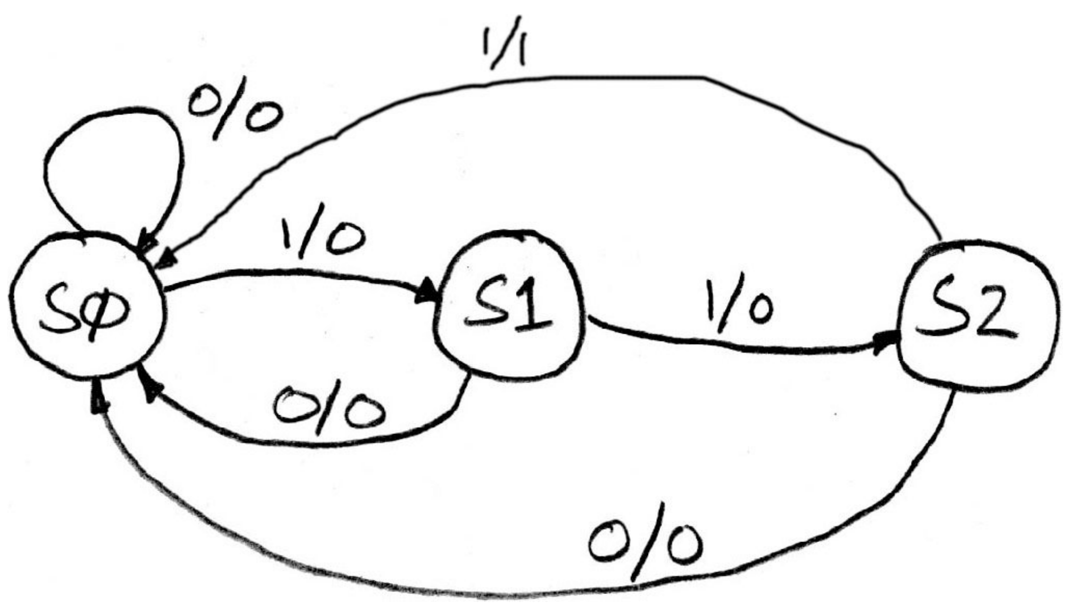
Finite State Machines Introduction



Finite State Machine Example: 3 ones…



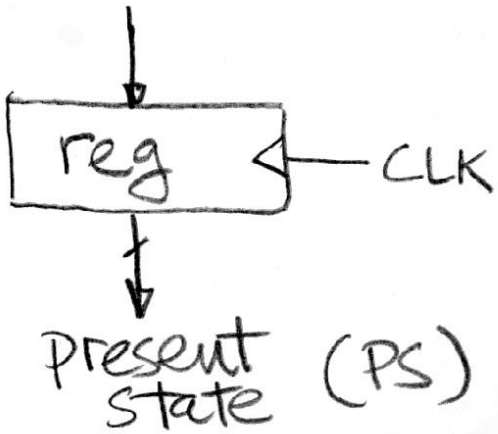
Draw the FSM…



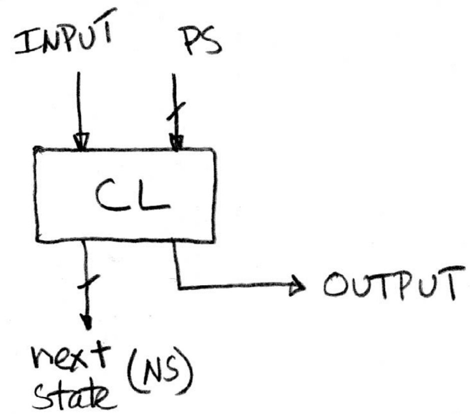
Truth table…

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

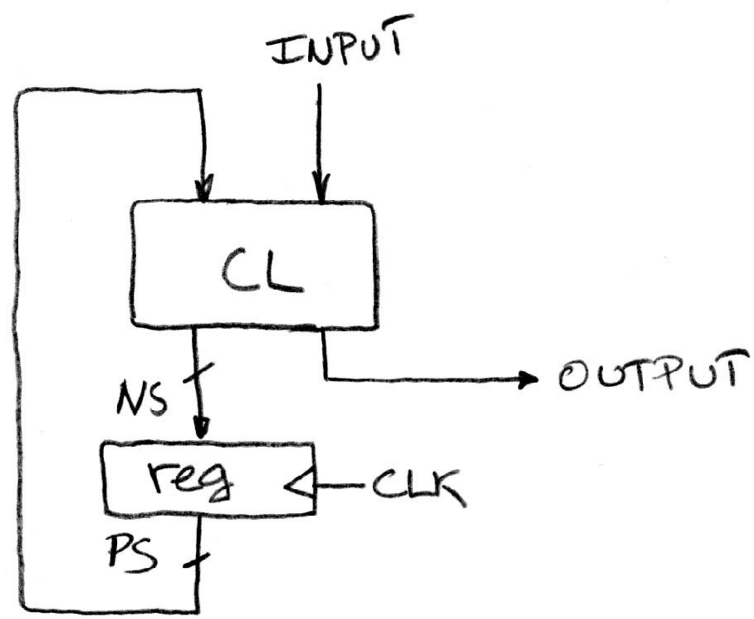
Hardware Implementation of FSM



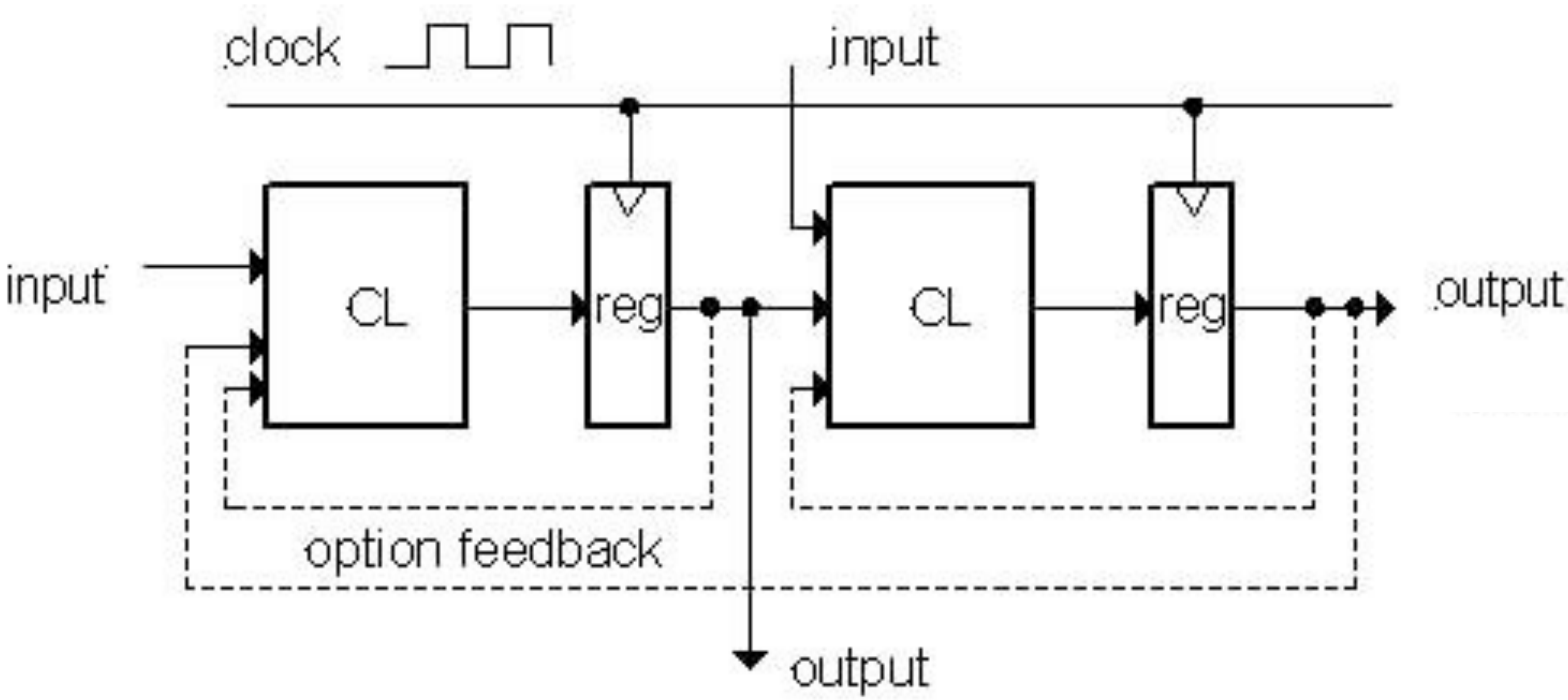
+



=



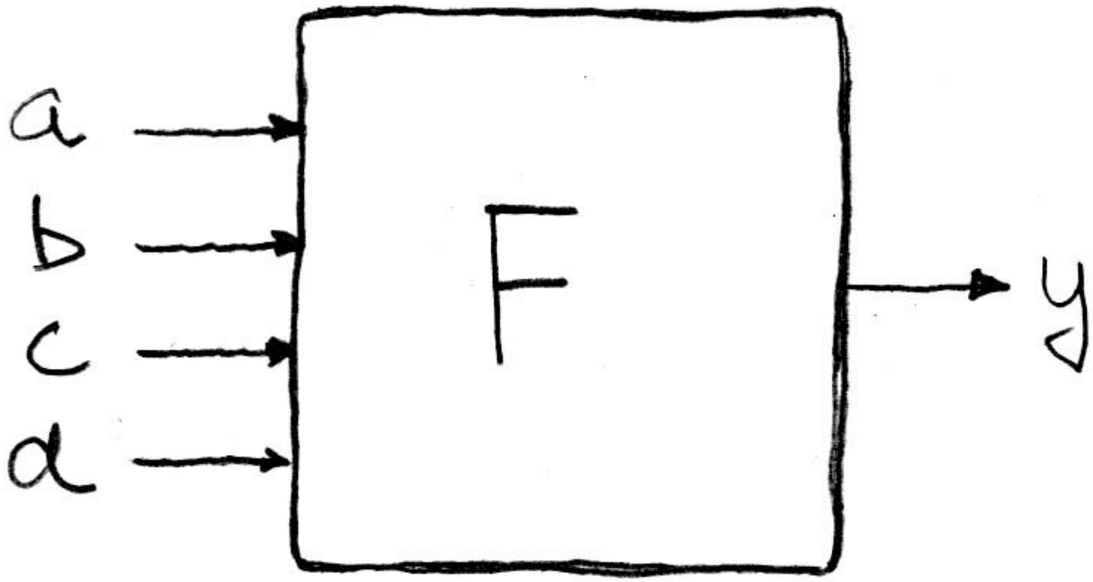
General Model for Synchronous Systems



Administrivia

- Any administrivia?

Truth Tables

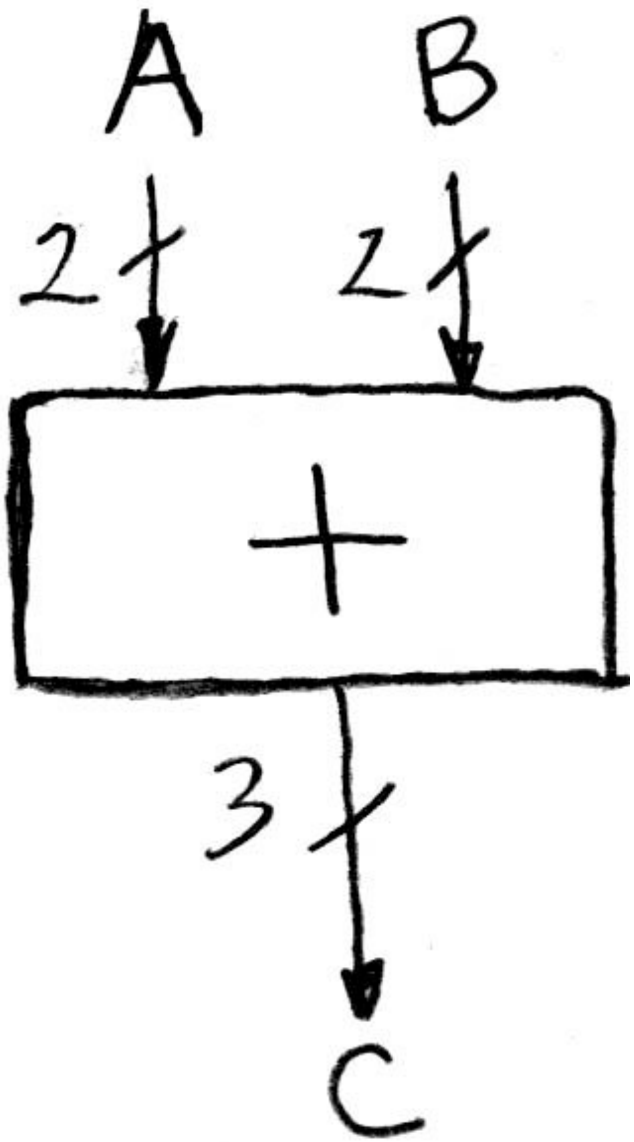


a	b	c	d	y
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
0	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1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

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

TT Example #2: 2-bit adder



A	B	C
a_1a_0	b_1b_0	$c_2c_1c_0$
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

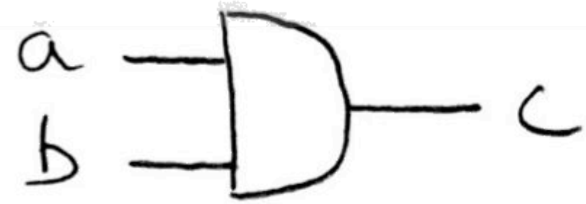
How Many Rows?

TT Example #3: 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

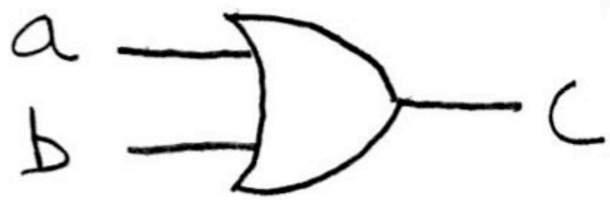
Logic Gates (1/2)

AND



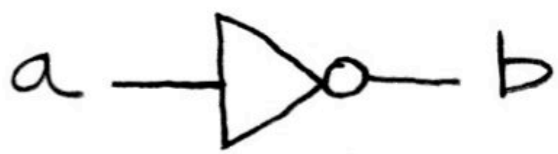
ab	c
00	0
01	0
10	0
11	1

OR



ab	c
00	0
01	1
10	1
11	1

NOT

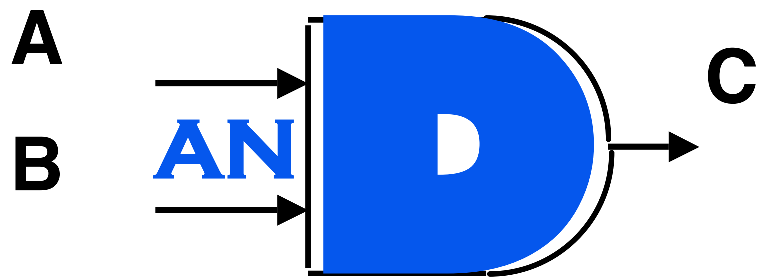


a	b
0	1
1	0

And vs. Or review – Dan's mnemonic

AND Gate

Symbol

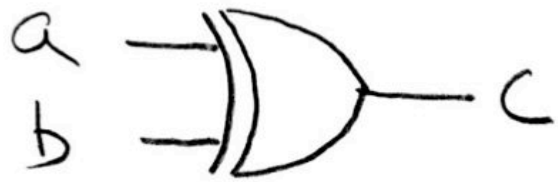


Definition

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

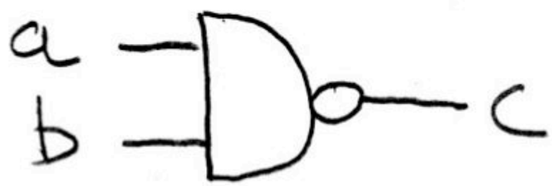
Logic Gates (2/2)

XOR



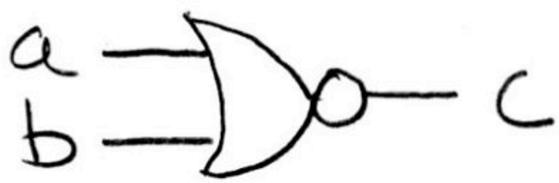
ab	c
00	0
01	1
10	1
11	0

NAND



ab	c
00	1
01	1
10	1
11	0

NOR



ab	c
00	1
01	0
10	0
11	0

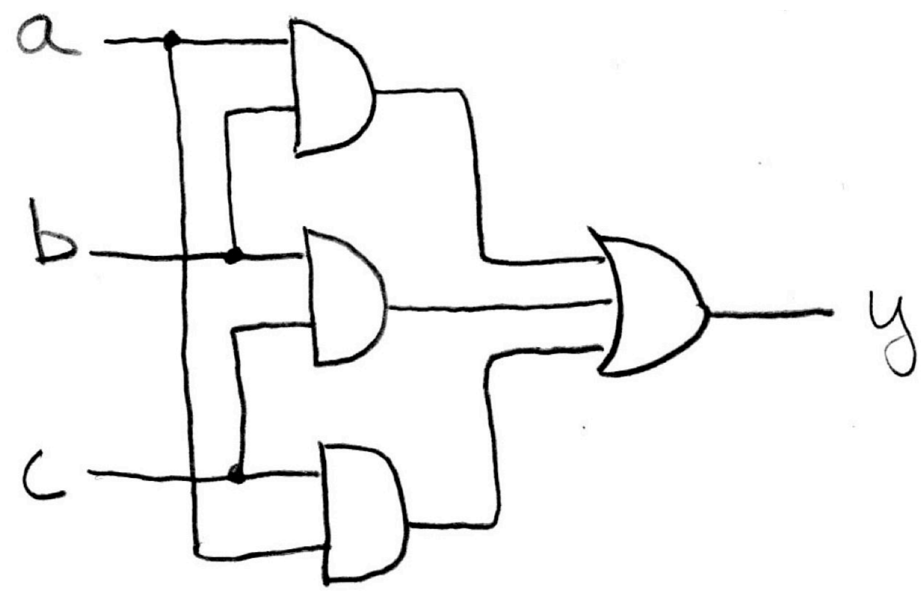
2-input gates extend to n-inputs

- N-input XOR is the only one which isn't so obvious
- It's 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

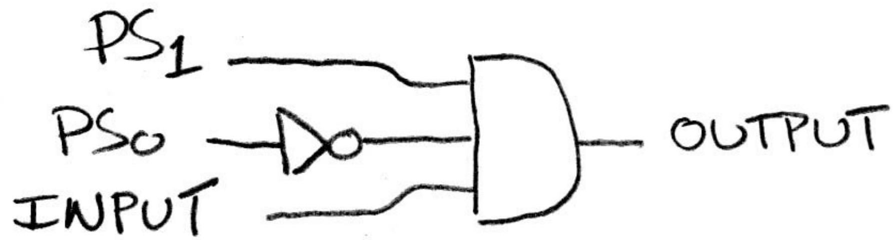
Truth Table ⇒ Gates (e.g., majority circ.)

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

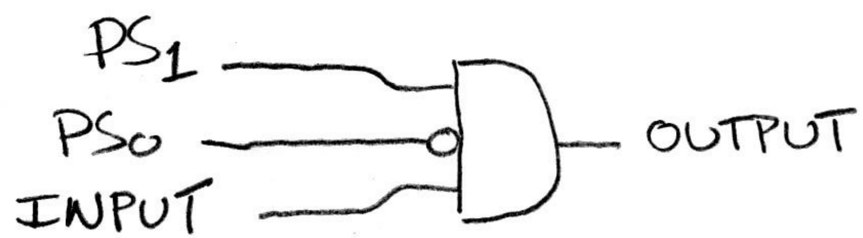


Truth Table ⇒ Gates (e.g., FSM circ.)

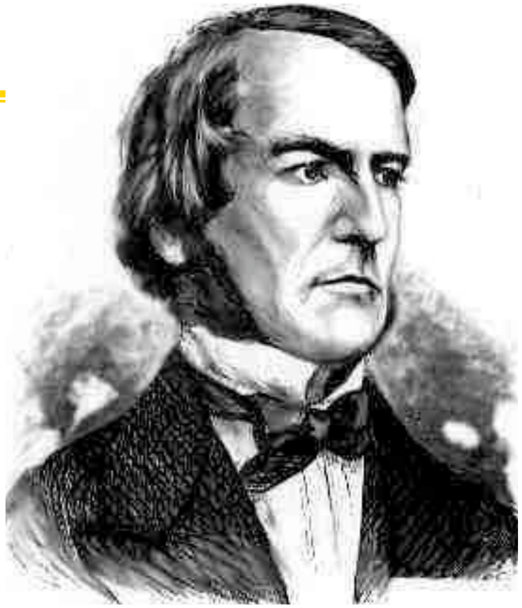
PS	Input	NS	Output
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1



or equivalently…



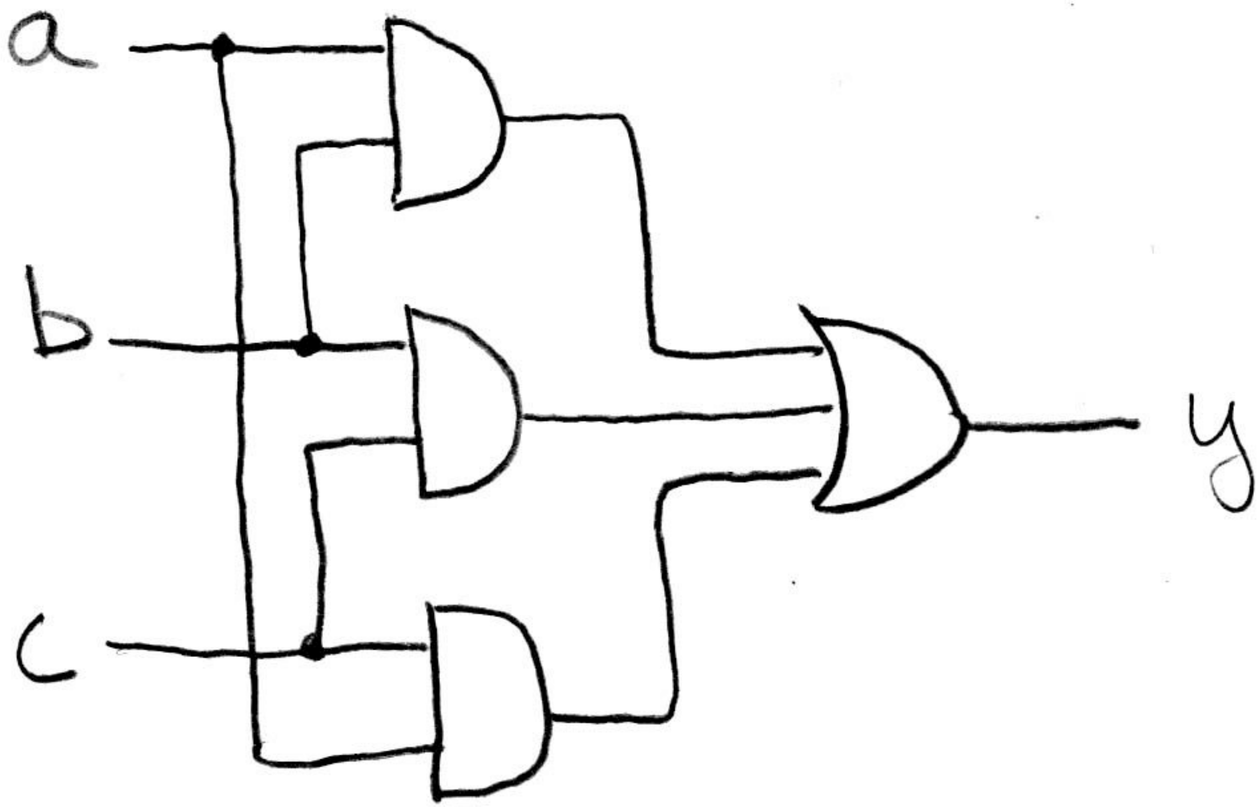
Boolean Algebra



- George Boole, 19th Century mathematician
- Developed a mathematical system (algebra) involving logic
  - later known as "Boolean Algebra"
- Primitive functions: AND, OR and NOT
- The power of BA is 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.)

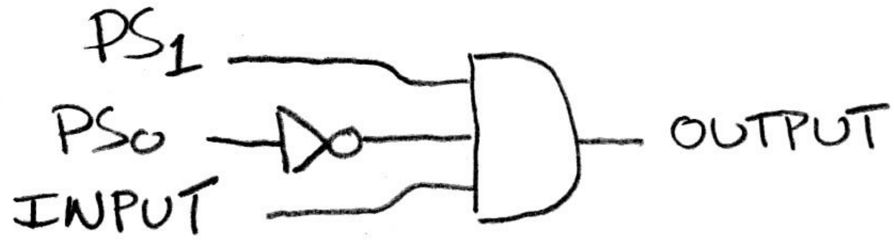


$$y = a \cdot b + a \cdot c + b \cdot c$$

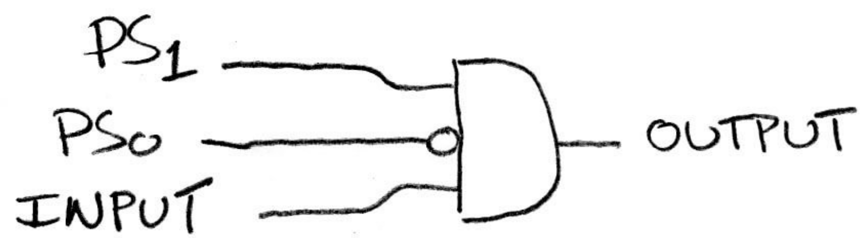
$$y = ab + ac + bc$$

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

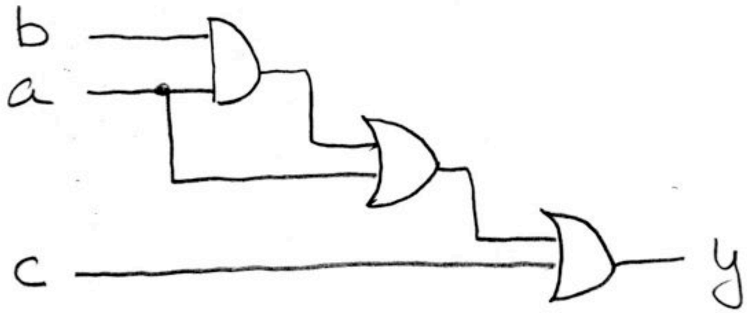


or equivalently…



$$y = PS_1 \cdot \overline{PS_0} \cdot \text{INPUT}$$

BA: Circuit & Algebraic Simplification



original circuit

$$\downarrow$$

$$y = ((ab) + a) + c$$

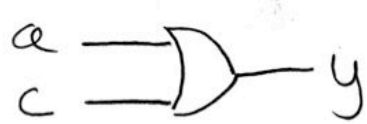
equation derived from original circuit

$$\downarrow$$

$$\begin{aligned} &= ab + a + c \\ &= a(b+1) + c \\ &= a(1) + c \\ &= a + c \end{aligned}$$

algebraic simplification

$$\downarrow$$



simplified circuit

BA also great for circuit <u>verification</u>
Circ X = Circ Y? use BA to prove!

Laws of Boolean Algebra

$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$	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
$x \cdot y = y \cdot x$	$x + y = y + x$	commutativity
$(xy)z = x(yz)$	$(x + y) + z = x + (y + z)$	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

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

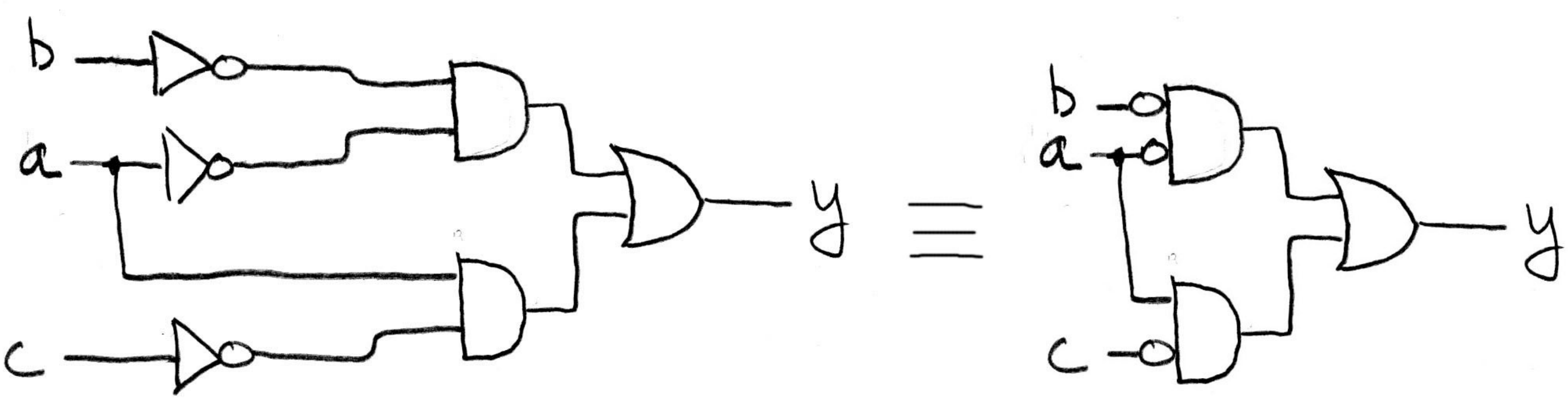
Canonical forms (1/2)

	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)

Canonical forms (2/2)

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



"And In conclusion…"

- Pipeline big-delay CL for faster clock
- Finite State Machines extremely useful
  - You'll see them again in 150, 152 & 164
- Use this table and techniques we learned to transform from 1 to another