

Boeing Recruiters tonight

sort(a, i1, i2) No

Symbol Table

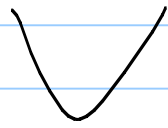
name	position in file
------	------------------

Relocation Table

position requiring fixup
based on position
where file is loaded.

.o file output of assembler

text segment
data segment
relocation table
symbol table
< debugging info >



linker



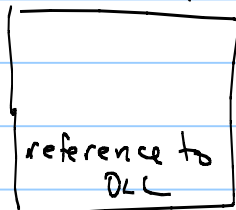
executable file

Story of static linking —
all the .o's needed are
combined into the executable

Library — collection of .o files

Dynamically linked libraries DLLs

Executable



Loader — Statically linked
— part of OS — copies from exec. file
into memory
Sets up stack segment
sets up uninitialized data segment
sets up pointers to Command line parameters
branch to your code.

DLLs w/ on-demand loading
start running your program w/o loading all library
functions
when a DLL function is called, the loader is
invoked to go get it and resolve references/relocate

How does it work —

Basically uses indirection

Review

Compiler	HLL	→	assembler input	.s	
Assembler	.s	→	object file	.o	static library
Linker	multiple .o	→	executable ;	or .o	dyn library
archiver (ar)	multiple .o	→	library	.a, or .so, or .dll	
loader	executable	→	running image in system.		

Logic Circuits

HLL, .s, executable file

ISA

electronics

Performance

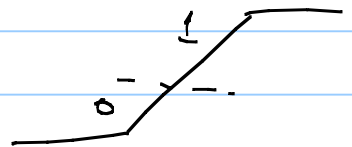
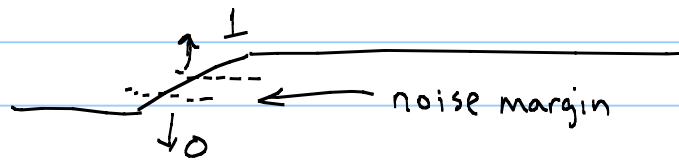
Economics of computers

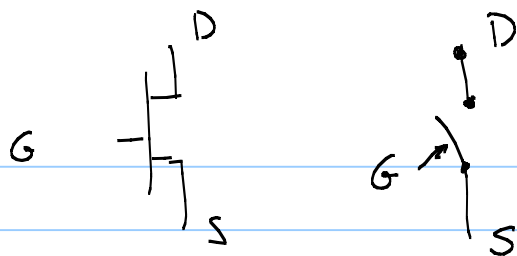
Constraints

Computers today are
Synchronous Digital Systems

↳ because circuits execute in locks step w/
a periodic clock signal

↳ because voltage signals inside are treated
as having value 0 or 1.



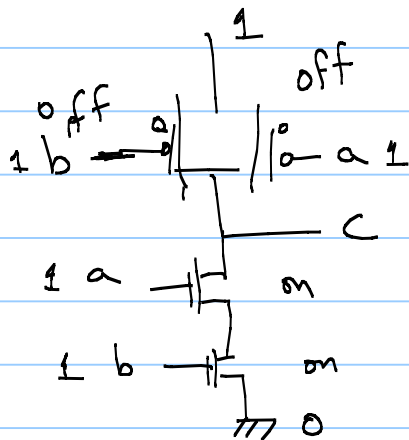


n-type MosFet — Switch

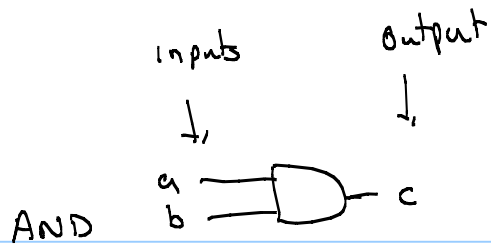
Gates — different logic ops

NAND gate

CMOS



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

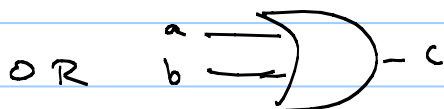


AND

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

True

	1
--	---



OR

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

False

	0
	0
	0
	0

n inputs 2^n rows in truth table

How many different truth tables

w/ 1 output and n inputs.

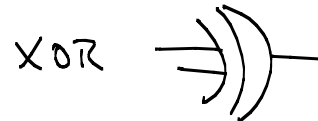
XOR

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

} 2^n rows

2^{2^n} different truth tables.

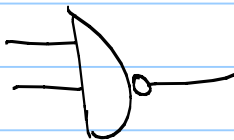
$$2^{2^2} = 2^4 = 16$$



NAND		
0	0	1
0	1	1
1	0	1
1	1	0

NOR		
0	0	1
0	1	0
1	0	0
1	1	0

NXOR		
0	0	1
0	1	0
1	0	0
1	1	1



~~Do~~

NOR

NAND
NOR

universal gate
" "

}

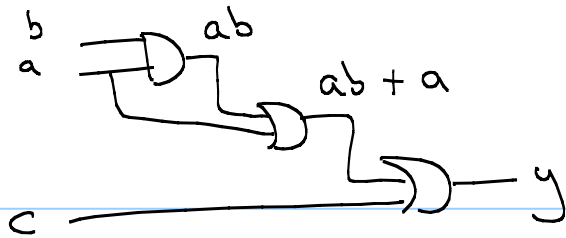
given enough NAND gates
can build any of the
16 2-input functions

Work in boolean algebra — more compact, equivalent notation.

$$y = \overset{\text{AND}}{\downarrow} ab + \overset{\text{AND}}{\downarrow} a\bar{c} + \overline{bc} \quad \leftarrow \text{NOT} \quad \leftarrow$$

↑
OR

$$y = (a \wedge b) \vee (a \wedge c) \vee \neg(b \wedge c)$$



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

True

0 — False

Laws of Boolean Algebra

$$x \bar{x} = 0$$

$$x + \bar{x} = 1$$

complements

$$x \cdot 0 = 0$$

$$x + 1 = 1$$

$$x \cdot 1 = x$$

$$x + 0 = x$$

identity rules

$$x \cdot x = x$$

$$x + x = x$$

idempotency

$$x \cdot y = y \cdot x$$

$$x + y = y + x$$

commutativity

$$(xy)z = x(yz)$$

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

associativity

$$x(y+z) = xy + xz$$

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

distributivity

$$xy + x = x$$

$$(x+y)x = x$$

$$\overline{xy} = \overline{x} + \overline{y}$$

$$\overline{(x+y)} = \overline{x} \cdot \overline{y} \quad \text{De Morgan's Laws}$$

Boolean Algebra formulas and Logic gate circuits
are in 1-1 correspondence.

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

$$x + xy + xz + yz$$

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

$$yz + x(\underbrace{1+y+z}_1)$$

Adders

a, b are one bit
value

$$\begin{array}{r} a \\ + b \\ \hline c \quad s \end{array}$$

1) Truth tables for s and c

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

ab a and b
 $a+b$ a or b
 \bar{a} not a
 $a \oplus b$ a xor b

2) Write s and c as boolean functions of a and b
english

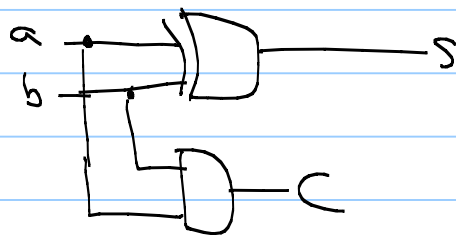
What is xor if it isn't primitive?

0 0	0
0 1	1
1 0	1
1 1	0

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

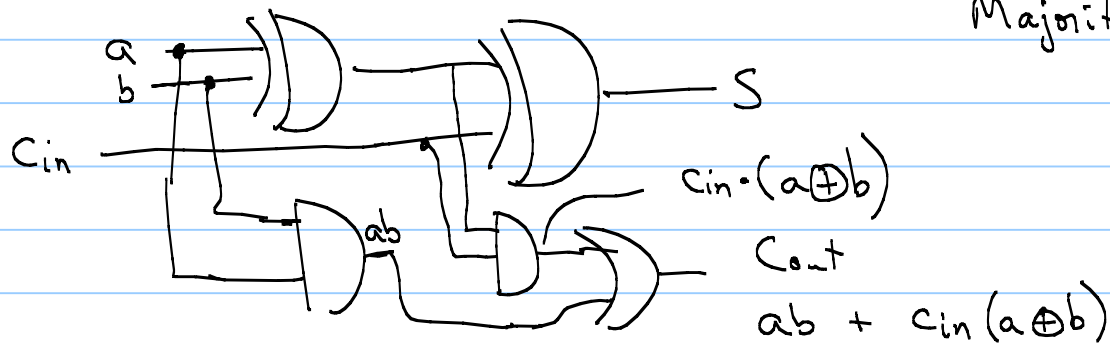
$$(a+b)\bar{a}\bar{b} = (a+b)(\bar{a}+\bar{b}) \\ = a\bar{a} + \boxed{b\bar{a} + a\bar{b}} + b\bar{b}$$

Half-adder

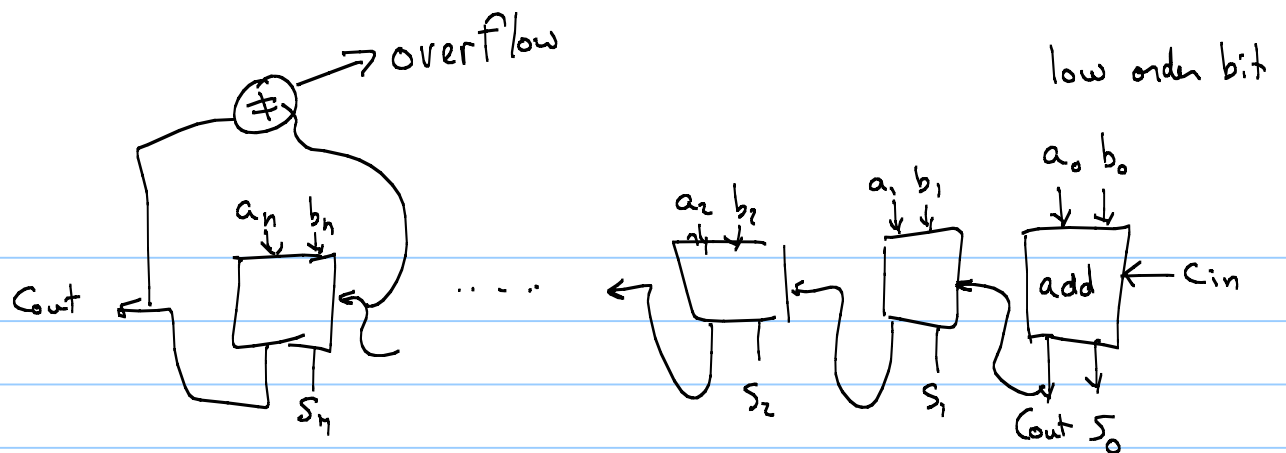


	1
0	1
1	0
<hr/>	
	1

Majority test on 3 bits



$$\begin{aligned}
 C_{out} &= \boxed{ab} + ac + bc = ab + (c + \bar{c})ab \\
 &= \boxed{c_{in}ab} + \boxed{\bar{c}_{in}ab} + c_{in}(a \oplus b) \\
 &= \boxed{c_{in}ab} + \bar{c}_{in}ab + c_{in}(\bar{a}b + a\bar{b}) \\
 &= c_{in}b + c_{in}a + ab
 \end{aligned}$$

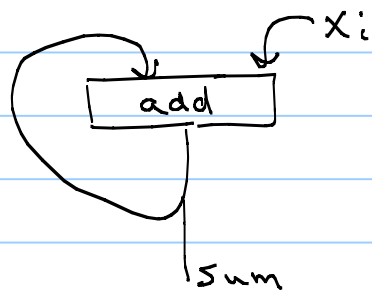


Logic operates at finite speed.

ripple-carry adder - takes time proportional to number of stages.

Look ahead carry adders

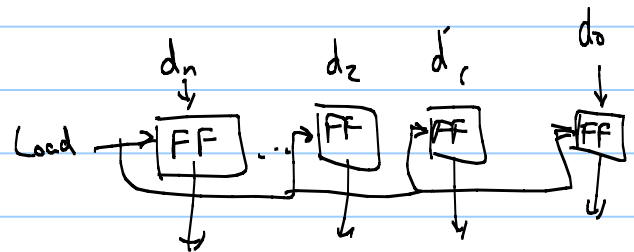
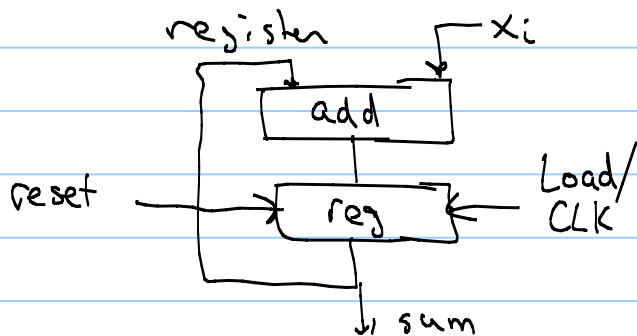
Suppose we want to add a sequence of numbers in hardware.



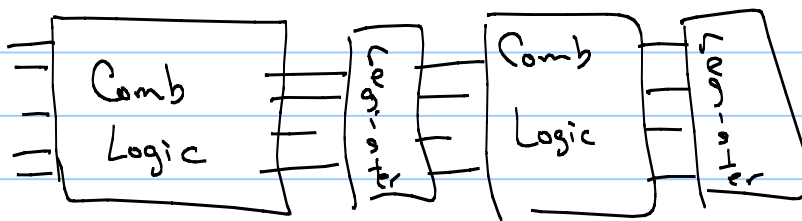
When do we stop?
When do we accept a new x_i ?

How do we start w/ $sum = 0$?

Solution: a clock and a storage

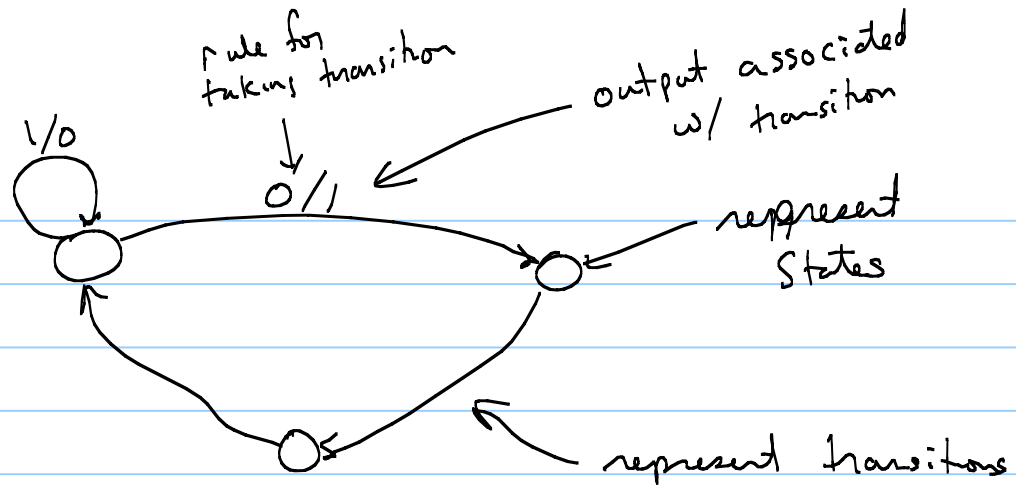


Synchronous digital systems!

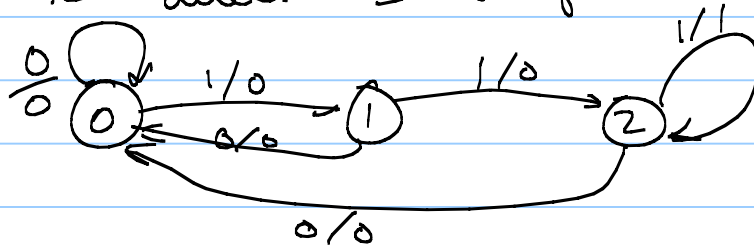


A Basic Synch Dig. System is
a finite state machine

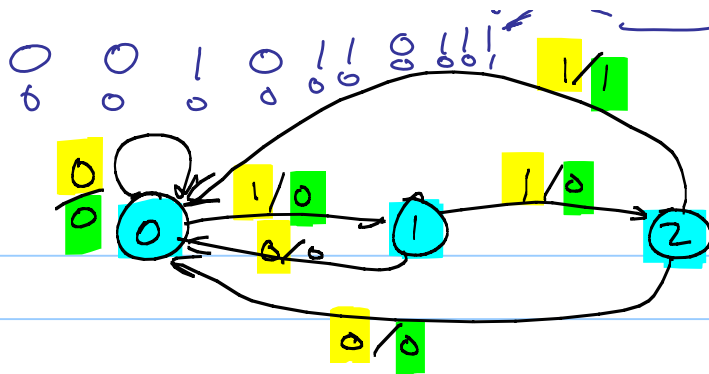
a number of states / rules to transition between
them.



FSM to detect 3 conseq ones in its input

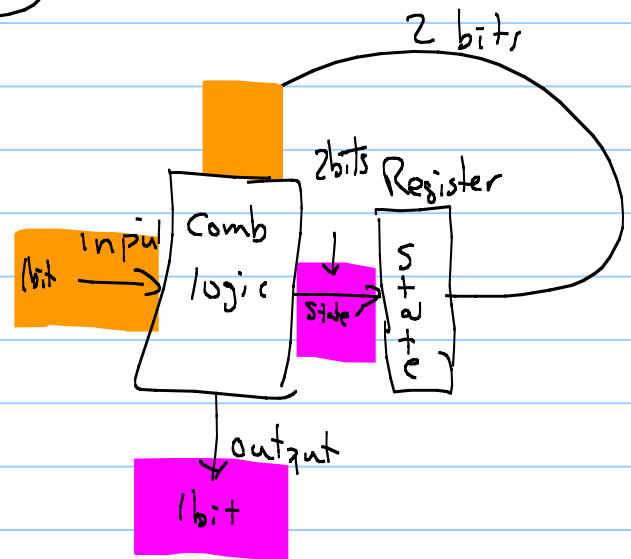


input
output



Output 1 when see 3 1 bits
in input
Recognize exactly
3 ones in a
row in input

state	in	state'	out
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1



In processor (CPU) have to execute instructions.

Logic for execute instruction?

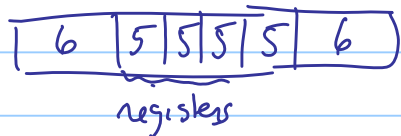
No. Several steps

1) • Instruction Fetch — same for all instructions.
— increment program counter (registers)

2) • Instruction Decode

- Based on opcode

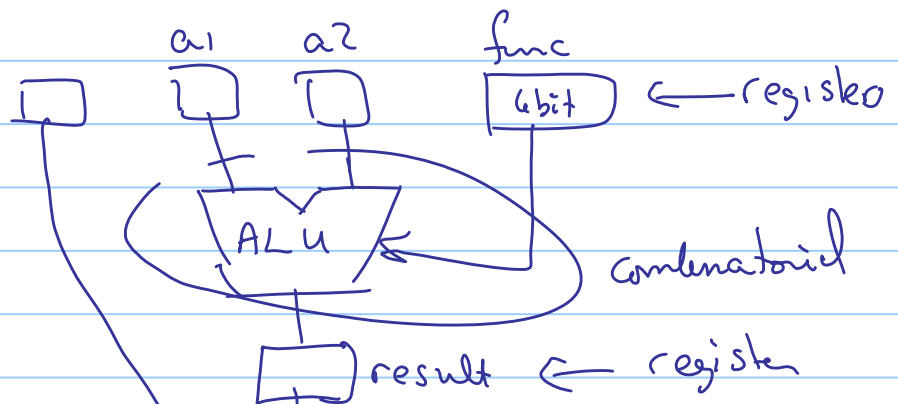
- read data from designated registers



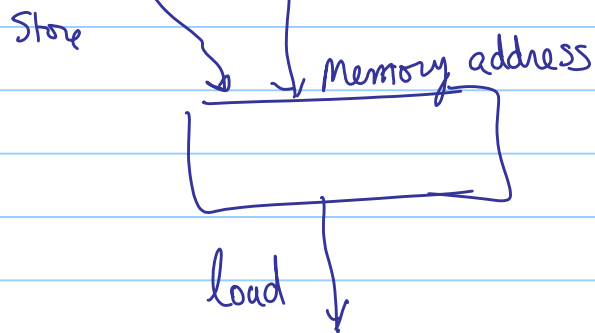
add instruction values from
registers (0-32) are stored in
temp. logic register.

How fast can it go?

3) ALU operation



4) Memory Access
• loads and stores only



5) Register Write
either result from ALU
or value from memory (load only)

Five Stages?

2nd Midterm 10/31/08