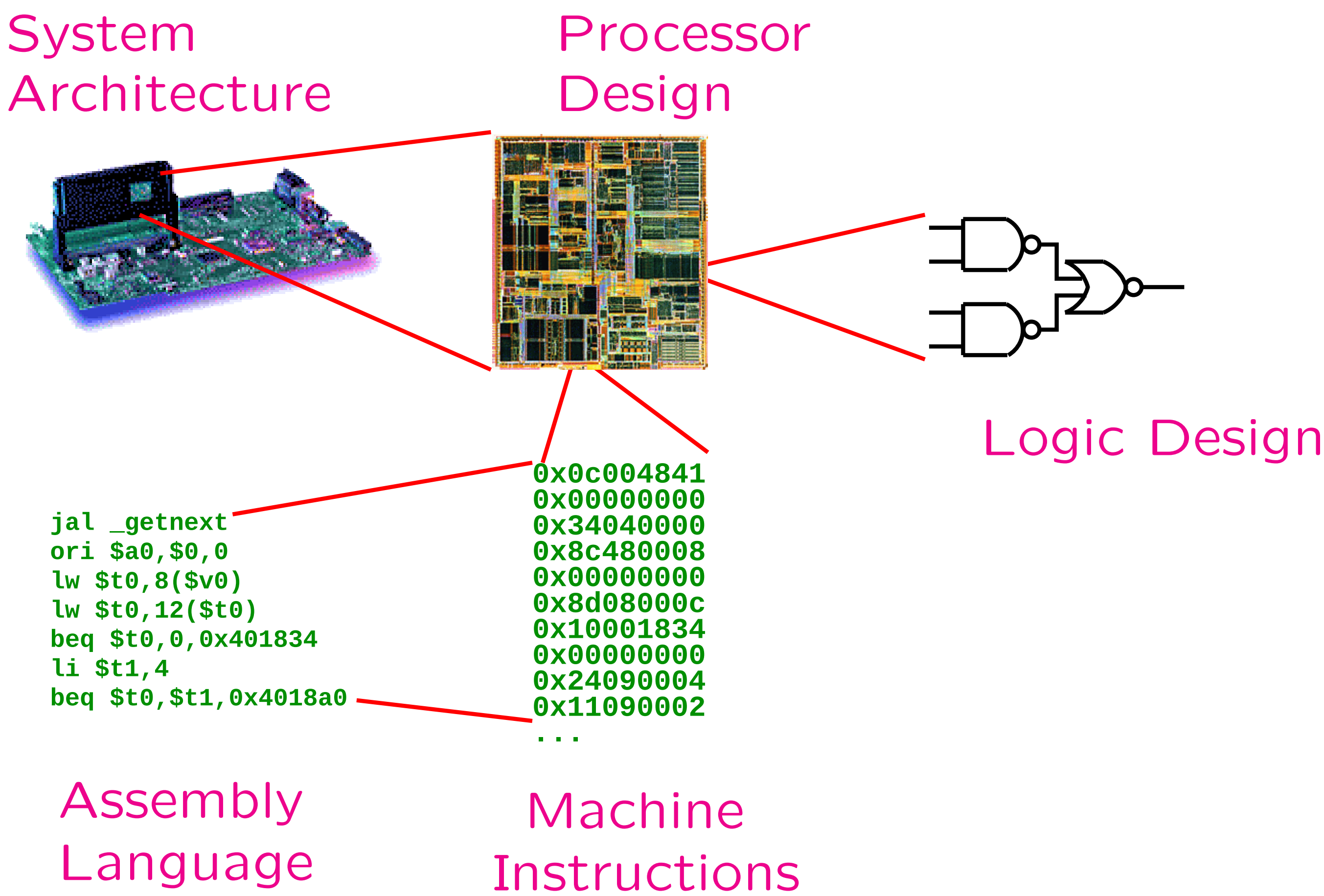
What 314 Is About



Building A Computer

Information is encoded with bits: 0's and 1's.

(we've already seen 2's complement numbers)

These are encoded using *voltages*...

- \+ well understood
- \+ easy to generate, detect
- \- affected by environment

But why 1's and 0's only?

Digital Representation

Example: representing a B&W picture:

- Black = 0 V
- White = 1 V
- 80% grey = 0.8 V
- ...

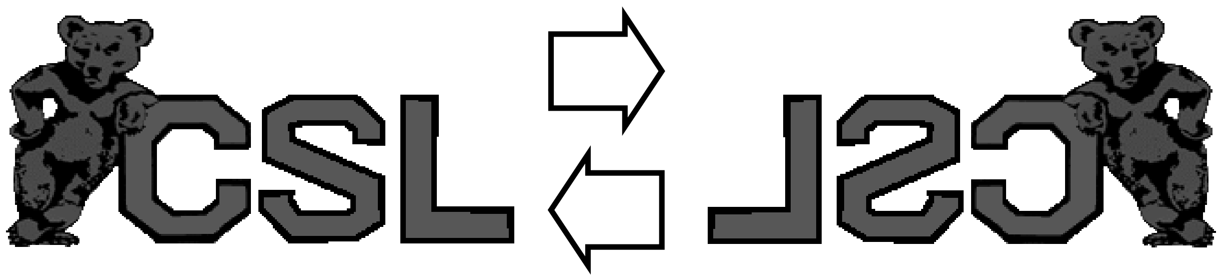
Represent by scanning picture in fixed order.

Let's try doing some computation with the voltages...

Digital Representation

Flip image:

Flip back and forth...



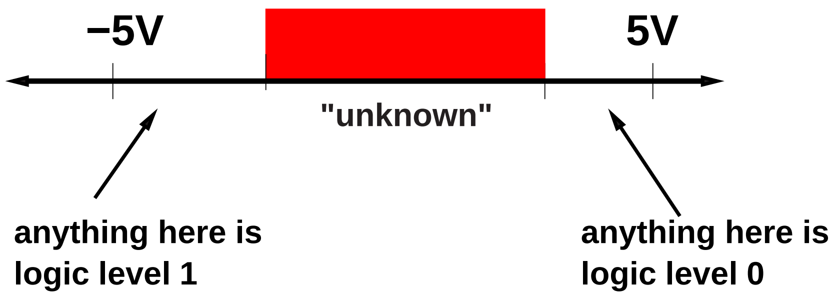
What really happens...

Have to build system to tolerate some error (noise).

Logic Levels

- Store just one bit on a wire...
- Gain reliability

Different conventions are possible

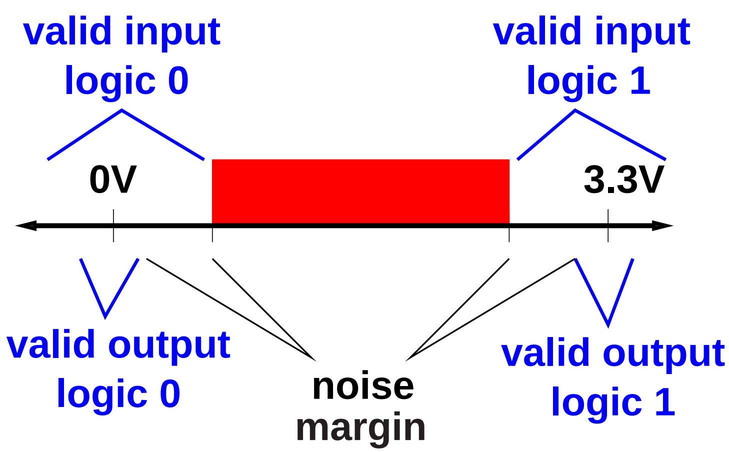


Combinational Devices

A *combinational device:*

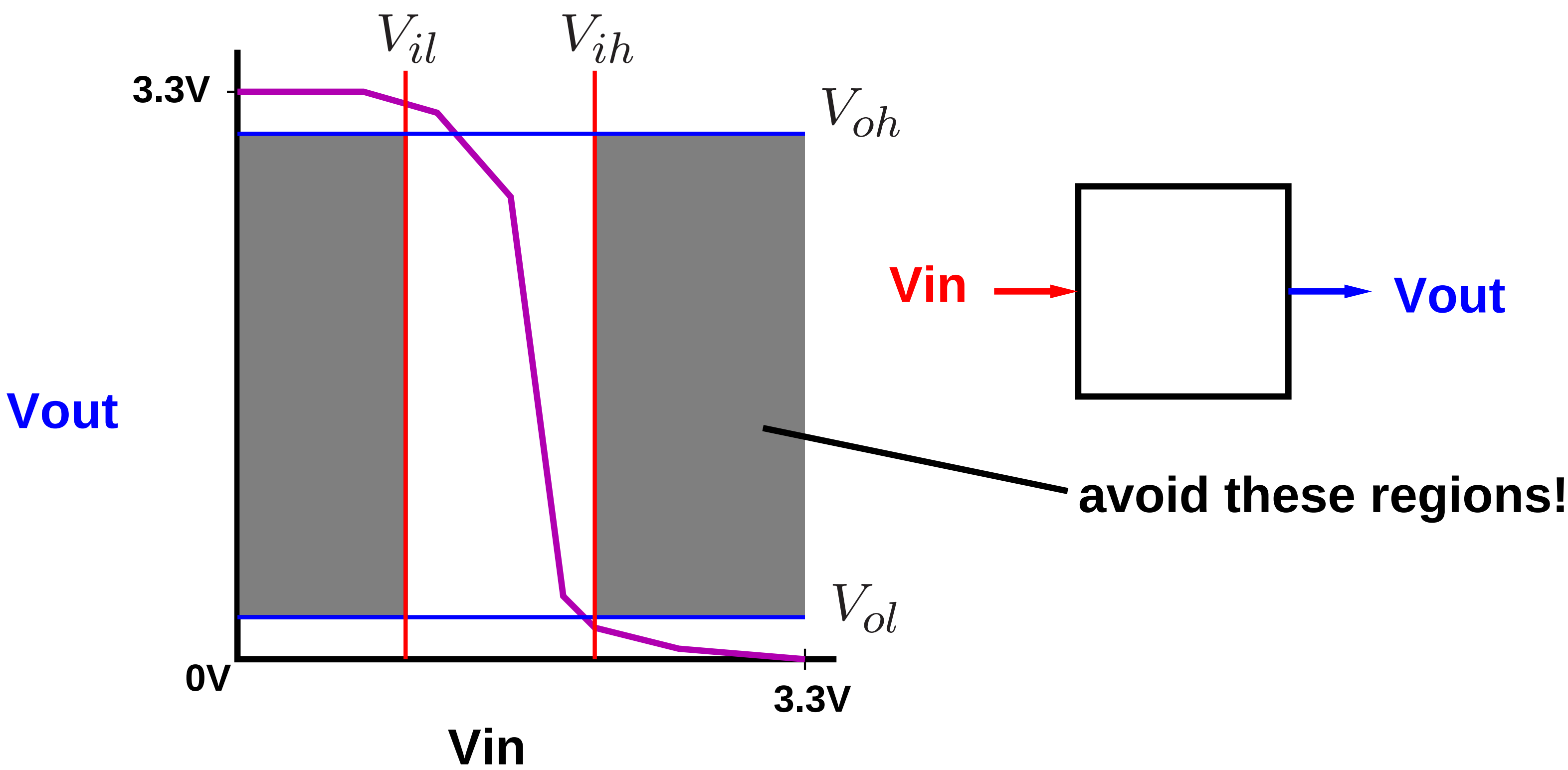
- Output is a function of inputs only ("memoryless")
- Takes input to valid, stable outputs

Combinational devices *restore* marginally valid signals!



Example

- Input: logic 0 if $< V_{il}$, logic 1 if $> V_{ih}$
- Output logic 0 if $< V_{ol}$, logic 1 if $> V_{oh}$



Digital View

- If input is 0, output is 1
- If input is 1, output is 0

Normally written in a table, like this:

In	Out
0	1
1	0

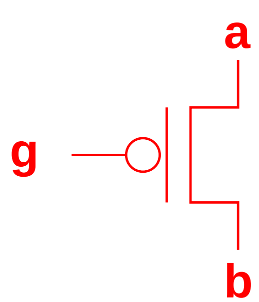
Called a "truth-table".

Implementation: Switching Networks

Lots of ways to build switches...

- relays
- vacuum tubes
- transistors
- ...

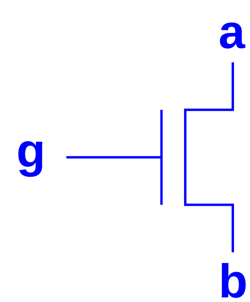
P−transistor



Connect a and b

if g = 0.

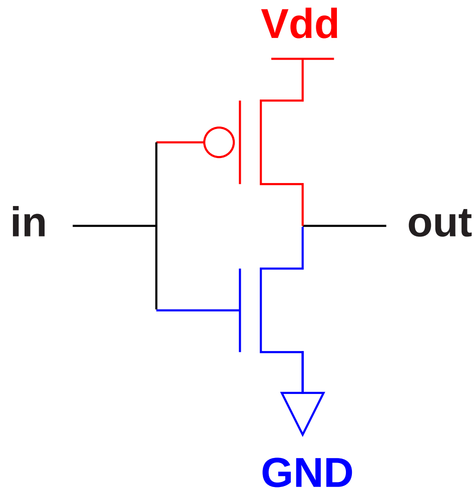
N−transistor



Connect a and b

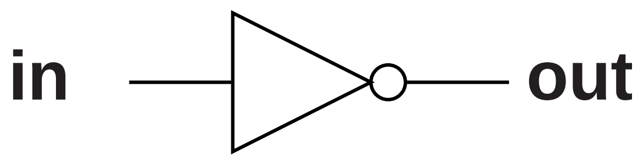
if g = 1.

Switching Networks: Inverter

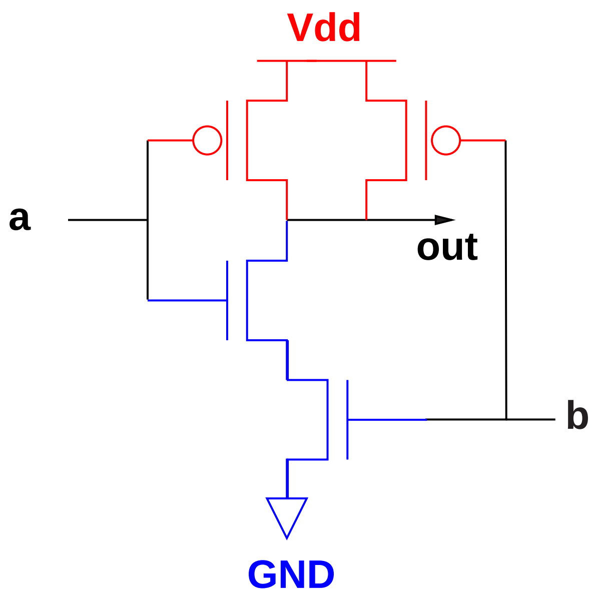


in	out
0	1
1	0

- Function: NOT
- Called an inverter
- Symbol:

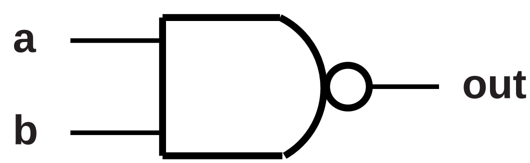


Switching Networks: NAND

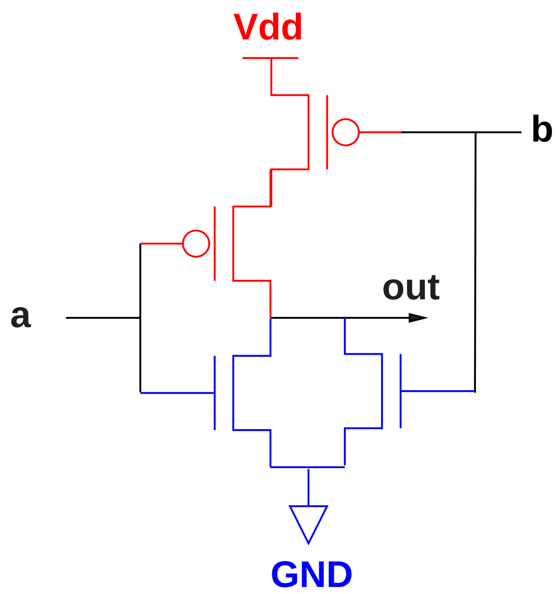


a	b	out
0	0	1
1	0	1
0	1	1
1	1	0

- Function: NAND
- Symbol:

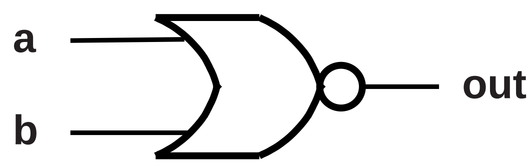


Switching Networks: NOR



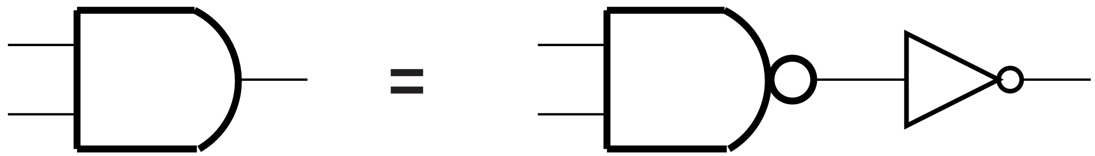
a	b	out
0	0	1
1	0	0
0	1	0
1	1	0

- Function: NOR
- Symbol:

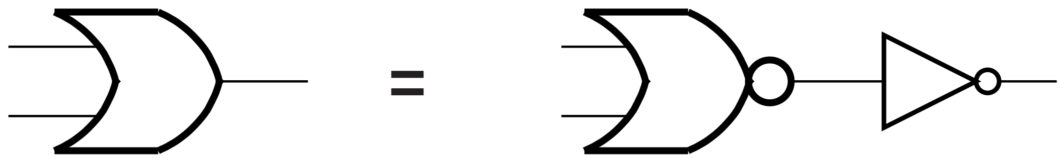


Building Functions From Gates

- AND:



- OR:



Can specify function by describing gates, truth table, or logic equations.

Logic Equations

AND:

$$
\begin{aligned}
out &= a \cdot b \\
out &= ab \\
out &= a \wedge b
\end{aligned}
$$

OR:

$$
\begin{aligned}
out &= a + b \\
out &= a \vee b
\end{aligned}
$$

NOT:

$$
\begin{aligned}
out &= \neg in \\
out &= \overline{in}
\end{aligned}
$$

Logic Equations

Fun with identities:

$$a + \overline{a} = 1$$
$$a + 0 = a$$
$$a + 1 = 1$$

$$a\overline{a} = 0$$
$$a \cdot 0 = 0$$
$$a \cdot 1 = a$$

$$a(b + c) = ab + ac$$
$$\overline{(a + b)} = \overline{a} \cdot \overline{b}$$
$$\overline{(a \cdot b)} = \overline{a} + \overline{b}$$
$$a + \overline{a}b = a + b$$

Check by writing truth tables, or by manipulating logic equations.

Let's Build An Adder

Write down function:

- Two 1-bit inputs, a and b
- Two 1-bit outputs, sum and $carry$

Truth-table:

a	b	carry	sum
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

CSL

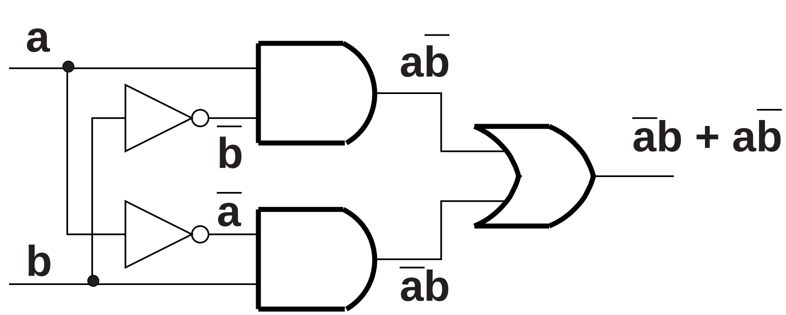
Let's Build An Adder

Sum output:

a	b	sum	Logic term
0	0	0	$\overline{a} \cdot \overline{b}$
1	0	1	$a \cdot \overline{b}$
0	1	1	$\overline{a} \cdot b$
1	1	0	$a \cdot b$

Logic equation: $a \cdot \overline{b} + \overline{a} \cdot b$

Circuit:



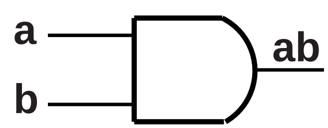
Let's Build An Adder

Carry output:

a	b	carry	Logic term
0	0	0	$\overline{a} \cdot \overline{b}$
1	0	0	$a \cdot \overline{b}$
0	1	0	$\overline{a} \cdot b$
1	1	1	$a \cdot b$

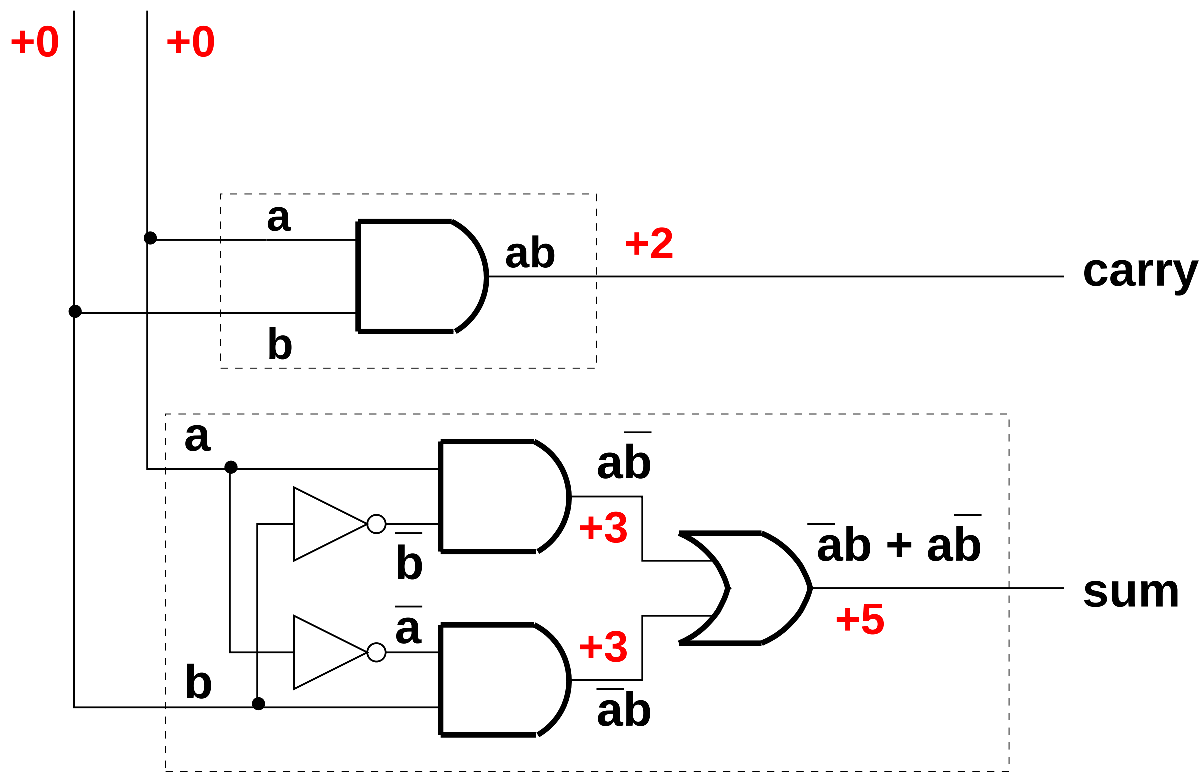
Logic equation: $a \cdot b$

Circuit:



Let's Build An Adder

Final Circuit:



Numbers indicate the number of sequential steps from input to output (worst-case).