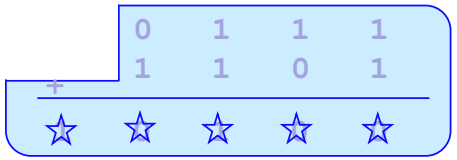
Composing circuits

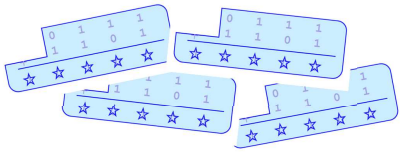
4-bit Ripple-Carry Adder



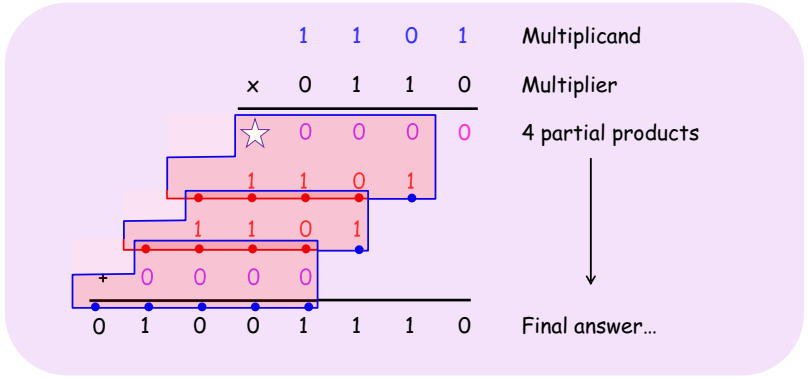
⇦ 8 input bits

⇨ 5 output bits ☆

Now let's make lots of them!!



hw6pr2: A 4-bit *multiplier*

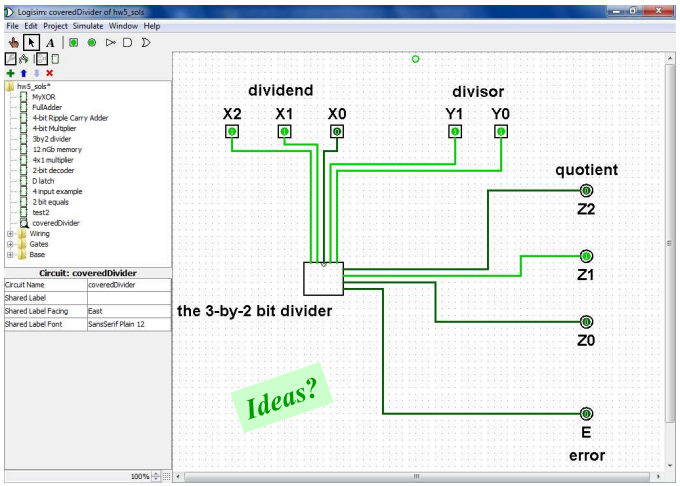


(A1) The AND gate is ***single-bit*** multiplication. **(A2)** ☆ == 0

(A1) Use a 4x1-bit helper circuit to find the four *partial products...*

(A3) You need three (3) ripple-carry adders to finish: *see above...*

Division? hw6pr3



principled design

Minterm *Division*

(0) All computation can be expressed as bits...

(1) ***Any*** function of bits can be made a truth table →

(2) Consider the output, *one bit at a time...*

(3) *The circuit will output 0 by default!*

(4) Are there *subcircuit patterns* to notice?

(5) Else, use an AND gate to **select** each input for which the output should be 1 (a minterm!)

To implement the red 1, how many inputs will its AND gate have?

How many negated?

What division is it?

(6) OR the outputs from step (5) together.

(7) optimize your circuit later -- *or never*

INPUTS					OUTPUT			
Y2 Y1 Y0 dividend			X1 X0 divisor		Z2 Z1 Z0 quotient			E error bit
anything			0	0	anything			1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0	0
0	1	1	0	1	0	1	1	0
1	0	0	0	1	1	0	0	0
1	0	1	0	1	1	0	1	0
1	1	0	0	1	1	1	0	0
1	1	1	0	1	1	1	1	0
0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	1	0
0	1	1	1	0	0	0	1	0
1	0	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	1	1	0
1	1	1	1	0	0	1	1	0
0	0	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	1	1	0	0	0	0
0	1	1	1	1	0	0	1	0
1	0	0	1	1	0	0	1	0
1	0	1	1	1	0	0	1	0
1	1	0	1	1	0	1	0	0
1	1	1	1	1	0	1	0	0

Time-optimized circuits: ***Carry lookahead adders***

The following circuit is called a **carry lookahead adder.**

By adding more hardware, we reduced the number of levels in the circuit and sped things up.

We can "cascade" carry lookahead adders, just like ripple carry adders.
We'd have to do carry lookahead *between* the adders too.

How much faster is this?

For a 4-bit adder, not much. There are 4 gates in the longest path of a carry lookahead adder, versus 9 gates for a ripple carry adder.

But if we do the cascading properly, a 16-bit carry lookahead adder could have only 8 gates in the longest path, as opposed to 33 for a ripple carry adder.

Newer CPUs these days use 64-bit adders. **That's 12 vs. 129 gates or 10x speedup!**

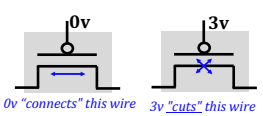
The delay of a carry lookahead adder grows *logarithmically* with the size of the adder, while a ripple carry adder's delay grows *linearly*.

The thing to remember about this is the **trade-off between complexity and performance.** Ripple carry adders are simpler, but slower. Carry lookahead adders are faster but more complex.

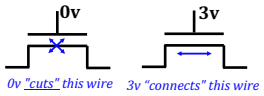
Building a NOT gate

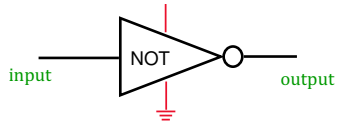
Transistors are current ***switches***:

switch-off-type (pmos)

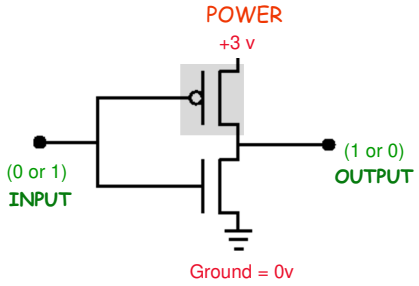


switch-on-type (nmos)

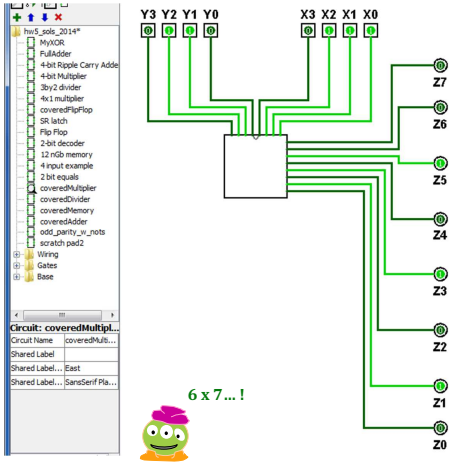




Building a **NOT** gate from transistors…



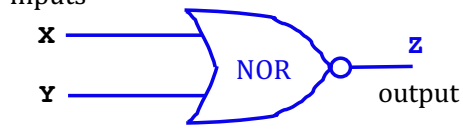
Half a computer: the **CPU**



transistors
↓
gates
↓
arithmetic

For systems, a face-lift is to add an edge that ***creates a cycle***, not just an additional node.

- also Alan Perlis



NOR's Truth Table

NOR		
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

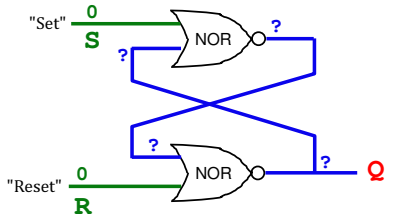
Let's engineer this into 1 bit of memory!

For systems, a face-lift is to add an edge that ***creates a cycle***, not just an additional node.

- also Alan Perlis

Thought experiment

- The circuit starts with **R** being **0 and S** being **0**
- The ***internal wires*** may be undefined, to start (this is ok) ...



Q is the output

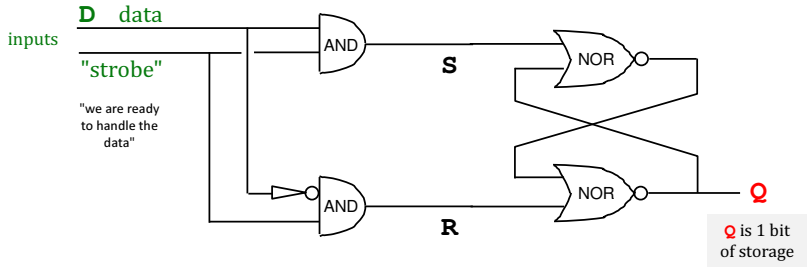
NOR's Truth Table

NOR		
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

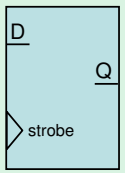
- What happens if **R** is set to **1**? (and **S** stays **0**)
- What happens if **R** is set back to **0**? (and **S** stays **0**)
- What if, then, **S** is set to **1**? (and **R** stays **0**)
- What happens if, then, **S** is set back to **0**? (and **R** stays **0**)
- Why *does "S" stand for "**Set**" and R for "**Reset**" ?*

The flip-flop

Demo!



Q is 1 bit of storage



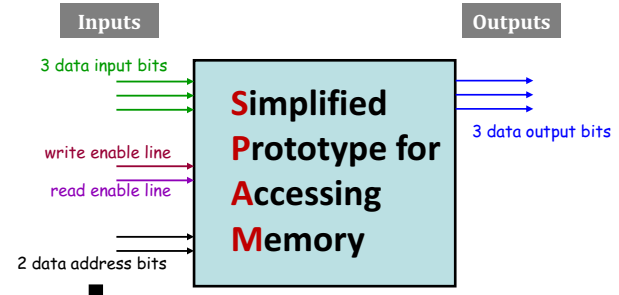
the flip-flop's diagram

1 bit of memory!

Random Access Memory

Demo!

Extra this week: ***Design 12nGbits of RAM***



3 bits stored at location 00
3 bits stored at location 01
3 bits stored at location 10
3 bits stored at location 11

Extra Credit

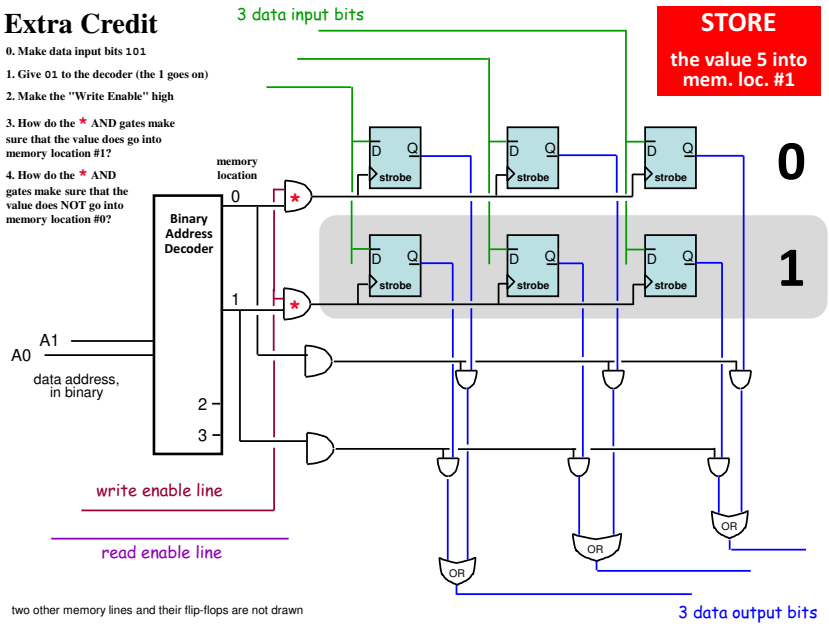
0. Make data input bits `101`

1. Give `01` to the decoder (the 1 goes on)

2. Make the "Write Enable" high

3. How do the * AND gates make sure that the value does go into memory location #1?

4. How do the * AND gates make sure that the value does NOT go into memory location #0?



0. Suppose 101 is in Location #1

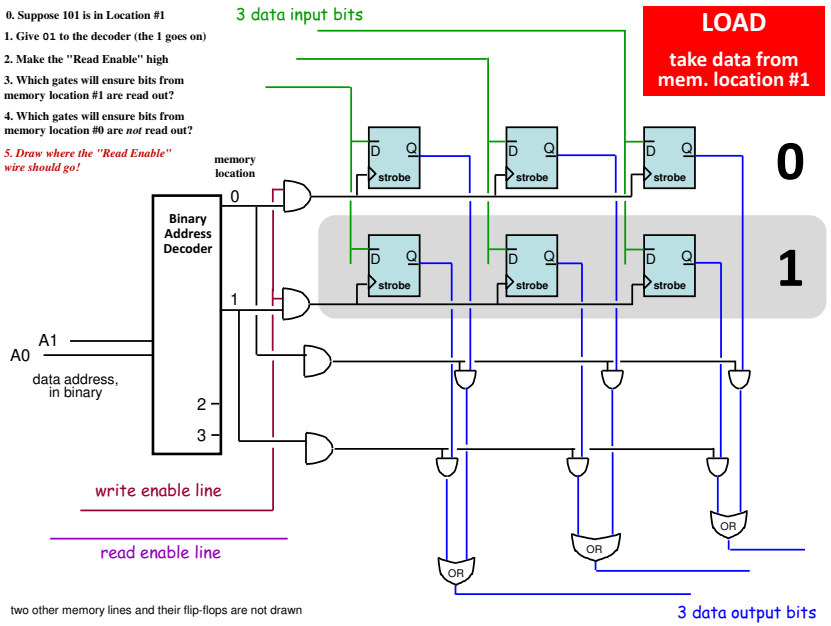
1. Give `01` to the decoder (the 1 goes on)

2. Make the "Read Enable" high

3. Which gates will ensure bits from memory location #1 are read out?

4. Which gates will ensure bits from memory location #0 are *not* read out?

5. Draw where the "Read Enable" wire should go!



Quiz

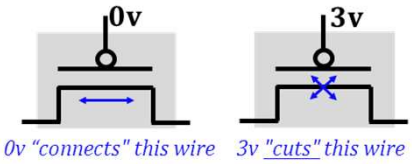
Name(s) _____

X	Y	Z
0	0	
0	1	
1	0	
1	1	

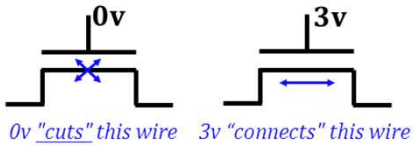
(1) Find the circuit's truth table

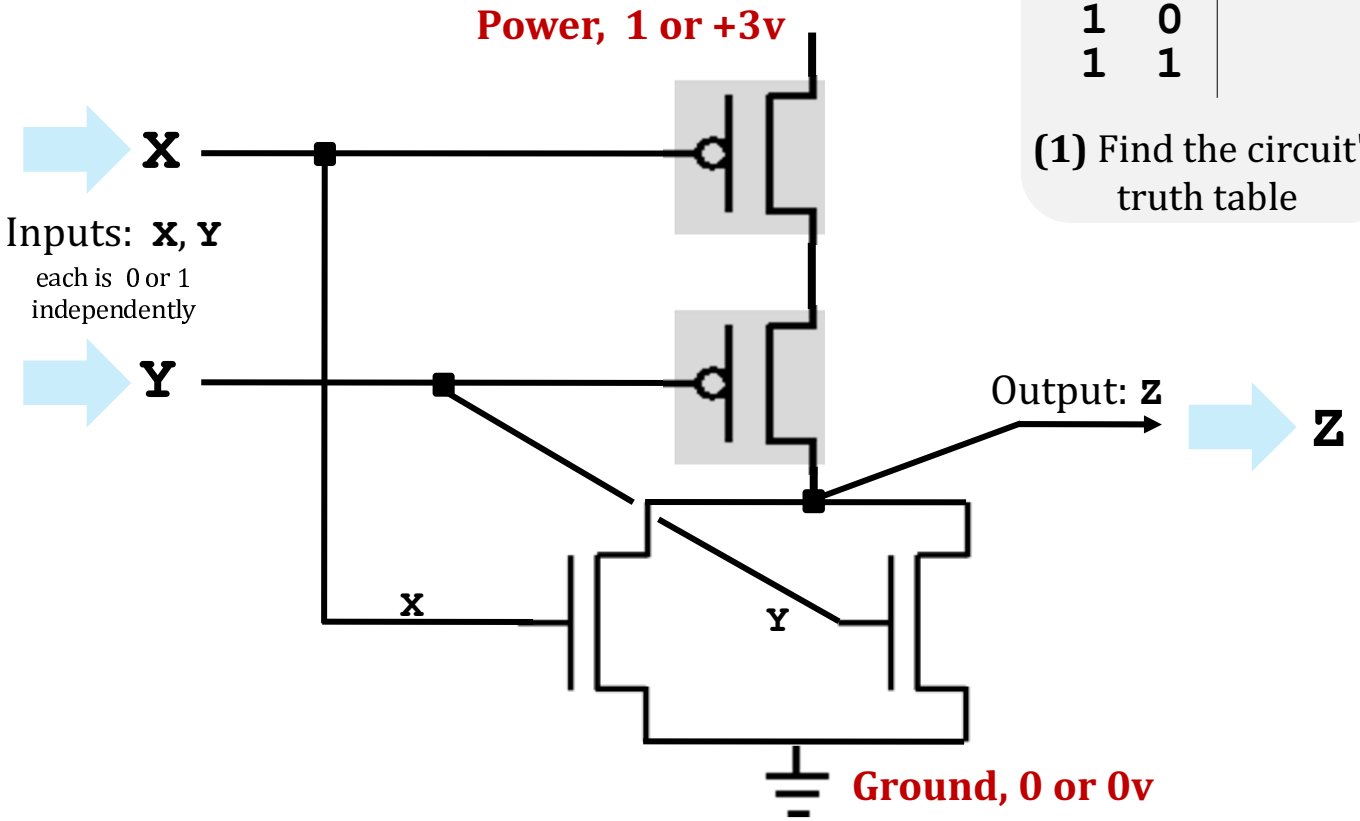
Transistors are current ***switches***:

switch-off-type (pmos)

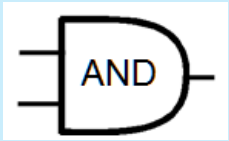


switch-on-type (nmos)





(3) Extra! How could you alter the transistor-level design to make an AND?



(2) the above circuit is one of these gates – ***which one***?