Things seem to get messy around here...

For every function, a circuit...

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(truth table)

Composing circuits ~ hw6

4-bit "Ripple-Carry" Adder

4-bit Multiplier

3x2-bit Divider

12 nGbits of memory (RAM)
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

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
\times & 0 & 1 & 1 \\
\hline
\star & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 \\
\hline
+ & 0 & 0 & 0 & 0 \\
\hline
0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Final answer...

(Q3) How could THREE 4-bit ripple-carry adders help here?

(Q2) What bit would be correct for the starred spot \(\star\)?

(Q1) What circuits could you use to create the four "partial products"?

Division? hw6pr3

Minterm Division

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

(1) \textbf{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

\[
\begin{array}{cccccccccc}
\text{INPUTS} & Y2 & Y1 & Y0 & X1 & X0 & \text{OUTPUT} & Z2 & Z1 & Z0 & E \\
\text{Anything} & 0 & 0 & 0 & 0 & 0 & \text{Anything} & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]
Circuit Optimization?

Perhaps artistically optimized!

using a genetic algorithm

16 gates

Optimize for what?!

try E85!

Time-optimized circuits: Carry lookaheads

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.

A 4-bit carry-lookahead adder circuit

Details!

Sum bits

"carry-out", not "c-zero"

Carry bits

A 4-bit carry-lookahead adder circuit

A 4-bit ripple-carry adder circuit

What information is needed? Where? How?

speed vs. complexity tradeoffs ~ cs facets of engineering
What's inside gates?

What's the other half of computation?

Silicon-based switches (transistors)

Transistors are current switches:

- switch-off-type (pmos)
  - 0v "open" this wire
  - 5v "closes" this wire

- switch-on-type (nmos)
  - 0v "closes" this wire
  - 5v "open" this wire

Transistors are single-electron tunneling, or SET transistor.

Today's gates?

https://www.youtube.com/watch?v=Fxv3Jo51uY8
Building a NOT gate

Transistors are current switches:

<table>
<thead>
<tr>
<th>Switch-off-type (PMOS)</th>
<th>Switch-on-type (NMOS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0v “opens” this wire</td>
<td>0v “cuts” this wire</td>
</tr>
<tr>
<td>5v “cuits” this wire</td>
<td>5v “opens” this wire</td>
</tr>
</tbody>
</table>

Building a NOT gate from transistors...

POWER +5v

(0 or 1) INPUT

Ground = 0v

(1 or 0) OUTPUT

Quiz

Transistors are current switches:

<table>
<thead>
<tr>
<th>Switch-off-type (PMOS)</th>
<th>Switch-on-type (NMOS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0v “opens” this wire</td>
<td>0v “cuits” this wire</td>
</tr>
<tr>
<td>5v “cuits” this wire</td>
<td>5v “opens” this wire</td>
</tr>
</tbody>
</table>

Power, 1 or +5v

Inputs: X, Y

Output: Z

X Y Z
0 0 0
0 1 1
1 0 1
1 1 1

(1) Find this circuit’s truth table

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

NAND: “not and”

OR: “not or”

NOR: “exclusive or”

Reading this week: H₂O computing!

Implemented!

Implemented!
Electromechanical "gates" (relays)  

Signal Input, A  

External Power (6v)  

NOT gate  

Signal Output Q  

Transistors as disruptive technology  

point contact transistor  

1947: Bell Labs  
seeking amplifiers for phone lines  
team of physicists: W. Brattain, W. Shockley, and J. Bardeen  

1948: junction transistor  
much more robust design  

1956: Shockley Semiconductor Co.  
in hometown of Palo Alto...  

the "traitorous eight" – left - ICs   

... and so begins the valley's siliconization

The Mark 1  
an early, relay-based computer  

Grace Hopper + Howard Aiken, Harvard ~ 1944  

ran at 0.00001 MHz  

5 tons  
530 miles of wiring  
765,299 distinct parts!  

Addition: 0.6 seconds  
Multiplication: 5.7 seconds  
Division: 15.3 seconds

Half a computer: the CPU  

transistors  

↓ gates  

↓ arithmetic  

6 x 7...!
What's *inside* gates?

What's the *other half* of computation?

Make no mistake... computers process numbers - not symbols.

*We can only automate what we can arithmetize.*

True! But it misses 99% of what computers do! *What?*

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

- Alan Perlis

<table>
<thead>
<tr>
<th>NOR's Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
</tr>
<tr>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
</tr>
<tr>
<td>1 0</td>
</tr>
<tr>
<td>1 1</td>
</tr>
</tbody>
</table>

Let's engineer this into 1 bit of memory!

- also Alan Perlis
Random Access Memory

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

**Simplified Prototype for Accessing Memory**

12 bits of RAM

Inputs

- 3 data input bits
- 2 data address bits
- write enable line
- read enable line

Outputs

- 3 data output bits

### Ex Cr

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?

### Demo!

- Make data input bits 101
- Give 01 to the decoder (the 1 goes on)
- 001 goes on the decoder
- 010 goes on the decoder
- 111 goes on the decoder

**STORE the value 5 into mem. loc. #1**

0.1.0

---

**The flip-flop**

The flip-flop's diagram

1 bit of memory!

---

**What happens if \( S \) stays 0 and \( R \) is set back to 0?**

**What happens if \( R \) is 0 and \( S \) is set to 1?**

**Why does "S" stand for "Set" and \( R \) for "Reset"?**

• What happens if \( S \) stays 0 and \( R \) is set back to 0?

   S "sets" \( Q \) to 1; \( R \) "resets" it back to 0.

   "we are ready to handle the data"

   \( Q \) starts at 0

   Q is a single bit of storage

   **the "strobe" is 1?**

   **How does the 'strobe' bit help store the bit D into \( Q \)?**

   "we are ready to handle the data"

   **to store a single bit we want to store (either a 0 or a 1).**

   \( Q \) starts at 0

   Q is a single bit of storage

   • Random Access Memory

   **Demo!**

   3 data input bits

   3 data output bits

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

<table>
<thead>
<tr>
<th>A0</th>
<th>A1</th>
<th>Memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

   3 data address bits

   data address, in binary

   write enable line

   read enable line

   **Binary Address Decoder**

   **STORE the value 5 into mem. loc. #1**

   0.1.0
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!

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!

Happy Wiring! Animusic's Fiber Bundles

Office hrs: Today @ ~2 Tutoring hrs: lots!
Quiz

Name(s) ________________________________

(1) Find this circuit’s truth table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- **NAND**: “not and”
- **OR**: “not or”
- **NOR**: “exclusive or”
- **XOR**: “exclusive or”

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

Transistors are current switches:

- **Switch-off-type (pmos)**: 0v "opens" this wire, 5v "cuts" this wire
- **Switch-on-type (nmos)**: 0v "cuts" this wire, 5v "opens" this wire

Inputs: X, Y each is 0 or 1 independently

Output: Z

Power, 1 or +5v

Ground, 0 or 0v