Why are 0's and 1's all we need?

Limits of the Universal Method

- For how large a circuit can we realistically expect to use the Universal Method?
- What do we do for larger circuits?

Numeracy

- How large do circuits get?
- How large do truth tables get?

Numeracy (cont.)

X	Y
0	1
1	0

1-input Truth Table

2 rows

Numeracy (cont.)

X	Y
0	1
1	0

1-input Truth Table

2 rows

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

2-input Truth Table

4 rows

Numeracy (cont.)

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

3-input Truth Table: 8 rows

Numeracy (cont.)

Inputs	Rows in truth table
1	2
2	4
3	8
4	??
5	??
Arbitrary N	??

Numeracy (cont.)

- Each input can be 0 or 1 : 2 possibilities.
- So if there are 4 inputs, that is a total of 2 * 2 * 2 * 2 = 16 possible input values.

Numeracy (cont.)

Inputs	Rows in truth table
1	2
2	4
3	8
4	16
5	??
Arbitrary N	??

Numeracy (cont.)

Inputs	Rows in truth table
1	2
2	4
3	8
4	16
5	32
Arbitrary N	??

Numeracy (cont.)

- Each input can be 0 or 1 : 2 possibilities.
- In general, if there are n inputs,
  there will be 2^n possible input values.

Numeracy (cont.)

Inputs	Rows in truth table
1	2
2	4
3	8
4	16
5	32
Arbitrary N	2^N

Powers of 2

- 2^n comes up a lot in Computer Science.
- Numbers to memorize:
  - $2^1 = 2$, $2^2 = 4$, $2^3 = 8$,
  - $2^4 = 16$, $2^5 = 32$, $2^6 = 64$,
  - $2^7 = 128$, $2^8 = 256$, $2^9 = 512$,
  - $2^{10} = 1024$

Powers of 2 (cont.)

2^{10} = 1,024

2^{20} = 1,048,576

2^{30} = 1,073,741,824

2^{40} = 1,099,511,627,776

2^{50} = 1,125,899,906,842,624

2^{60} = 1,152,921,504,606,846,976

Powers of 2 (cont.)

- Some rough numbers:

$$2^{10} = 1{,}024 \approx 1{,}000\ (10^3)$$

$$2^{20} = 1{,}048{,}576 \approx 1{,}000{,}000\ (10^6)$$

$$2^{30} \approx 1{,}000{,}000{,}000\ (10^9)$$

$$2^{40} \approx 1{,}000{,}000{,}000{,}000\ (10^{12})$$

$$2^{50} \approx 10^{15}$$

$$2^{60} \approx 10^{18}$$

Powers of 2 (cont.)

- Some rough numbers:

$2^{10} \approx 1{,}000\ (10^3)$	kilo	
$2^{20} \approx 1{,}000{,}000\ (10^6)$	mega	RAM
$2^{30} \approx 10^9$	giga	disk
$2^{40} \approx 10^{12}$	tera	BIG disk
$2^{50} \approx 10^{15}$	peta	
$2^{60} \approx 10^{18}$	exa	knowledge

Sidebar on Exabytes

- It's taken the entire history of humanity through 1999 to accumulate 12 exabytes of information. By the middle of 2002 the second dozen exabytes will have been created
- 1 exabytes = 50,000 times the library of congress
- Floppys to make 1 exabyte would stack 24 million miles high.

Powers of 2 (cont.)

- 16 inputs means Truth Table
  has $2^{16} = 2^{10} * 2^{6} = 1{,}024 * 64$
  $\approx$ 64,000 rows
- Circuit could have 64,000 or more gates

Universal Method (cont.)

- What about 100 inputs:
- Is it reasonable to use Universal Method on Truth Table with 100 inputs?

Universal Method (cont.)

- What about 100 inputs:
- Is it reasonable to use Universal Method on Truth Table with 100 inputs?
- $2^{100} \approx 10^{30}$
- Each AND/OR/NOT gate uses at least 1 transistor.
- This is way beyond current technology, in fact . . .

Numeracy (cont.)

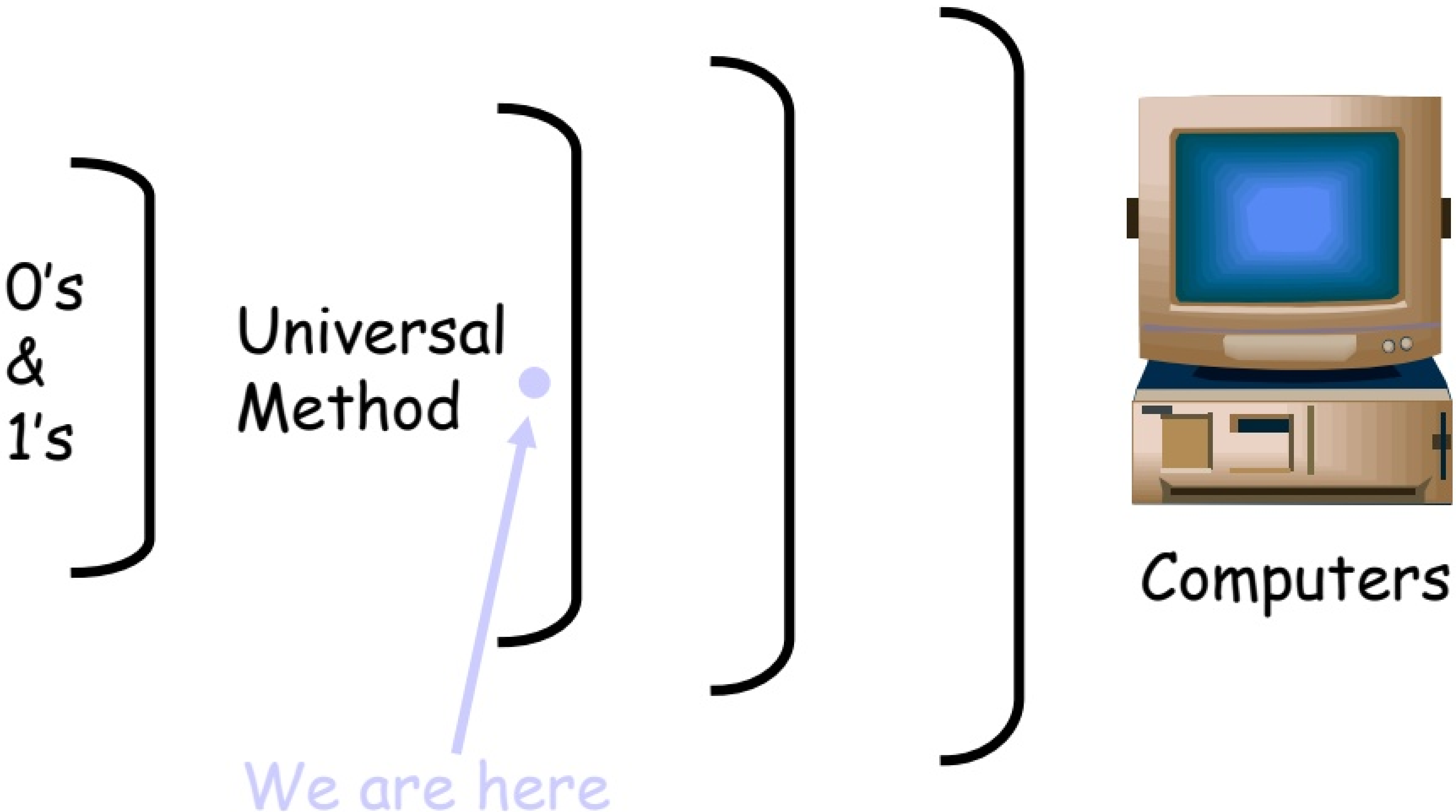
- State-of-the-art transistors are about .1 micrometer (mm) on a side:
  - Lined end-to-end, you could fit 10,000,000 transistors in 1 m. (≈ 3 f.)
  - In a 1 m. by 1 m. square, you could fit 100,000,000,000,000 transistors
  - That's still 999,999,999,999,999,
    900,000,000,000,000
    too few for 100 input Truth Table !

No good?

- How sad should we be? Not very.
- You just need to use many Truth Tables each having fewer inputs.
- Can make an entire computer using only 16-input Truth Tables and the Universal Method!
- On the other hand, must realize that in some cases, we need more efficient special purpose circuits than the Universal Method. (We won't cover these.)

Our First Abstract Tool

Universal Method: Circuits for ANY Truth Table



So what?

- We can deal with 0's and 1's now, but why should we?
- Answer: Because we can represent so many things with 0's and 1's.

Meaning

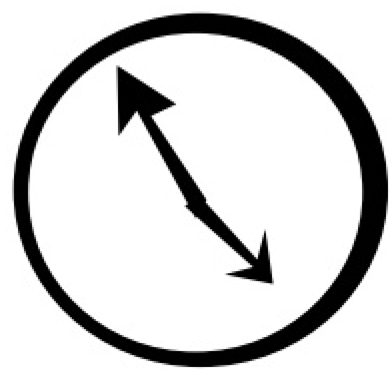
- In Logic, we thought of 0 and 1 as meaning True and False.
- Now, we remove these connotations.
- Definition: a bit is just a single variable that can be 0 or 1.

Representing Information

- Information in the world comes in many forms – letters, numbers, pictures, sounds…
- How can we represent these things with 0's and 1's?
- Start with numbers.

Representing Numbers

- Before computers, in devices, numbers typically represented continuously.
- e.g. Clocks:



Binary Numbers

- How do we count normally?
- 0, 1, 2, 3, 4, 5, 6, 7, 8,
  9, 10, …
  19, 20, …
  99, 100, … 999, 1000, …
- Suppose we only had two numerals – 0 and 1.
- Then how would we count?
  0,
  1, 10,
  11, 100, 101, 110, …

Binary Numbers (cont.)

0	0
1	1
2	1 0
3	1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1
8	1 0 0 0

Binary Numbers (cont.)

- Addition :

```
      1 0 1
+     1 1 1
___________
```

Binary Numbers (cont.)

- Addition – Our "basic" addition table is really easy now:
- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 10

Binary Numbers (cont.)

- Addition – just like usual:

$$
\begin{array}{r}
101 \\
+\ 111 \\
\hline
\end{array}
$$

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Binary Numbers (cont.)

- Addition – just like usual:

```
      1
    1 0 1
+   1 1 1
---------
        0
```

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Binary Numbers (cont.)

- Addition – just like usual:

	1	1	
	1	0	1
+	1	1	1
		0	0

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$

Binary Numbers (cont.)

- Addition – just like usual:

```
        1 1
        1 0 1
+       1 1 1
_______________
      1 1 0 0
```

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Binary Numbers (cont.)

- If $(10)_{10}$ is called 10, what do we call $(10)_2$?
- Can we convert between binary and decimal?

$(123)_{10} = 2*61 + 1$

$= 2^2*30 + 2^1 *1 + (2^0 *)1$

$= 2^3*15 + 0* 2^2 + 2^1 *1 + (2^0 *)1$

$= 2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0$

……. $= (1111011)_2$

$(101101)_2 = 2^5 + 2^3 + 2^2 + 2^0 = 32 + 8 + 4 + 1 = 45$

Binary Numbers (cont.)

- Can also do:
  - Subtraction, Multiplication, etc
  - Negative Numbers
  - Fractions

- Great, where do Logic Circuits fit in?

Binary Numbers (cont.)

- For Addition on small numbers, can make a truth table:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	10

Problem: Output can be 2 bits sometimes

Binary Numbers (cont.)

- Addition Truth Table with Multiple Outputs:

X	Y	A	B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Binary Numbers (cont.)

- Same as 2 Truth Tables:

X	Y	A
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	B
0	0	0
0	1	1
1	0	1
1	1	0

Binary Numbers (cont.)

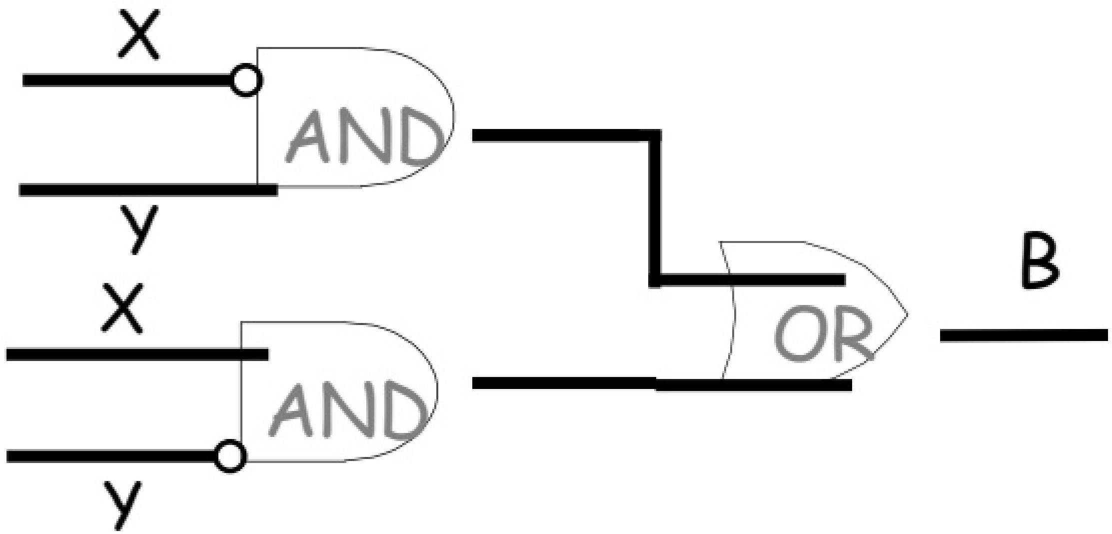
- Same as 2 Truth Tables:

X	Y	A
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	B
0	0	0
0	1	1
1	0	1
1	1	0

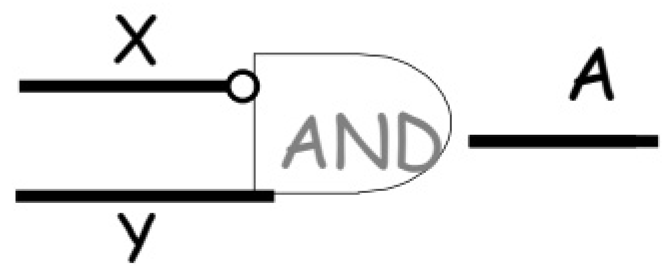
- Now we can convert into 2 circuits!

Binary Numbers (cont.)

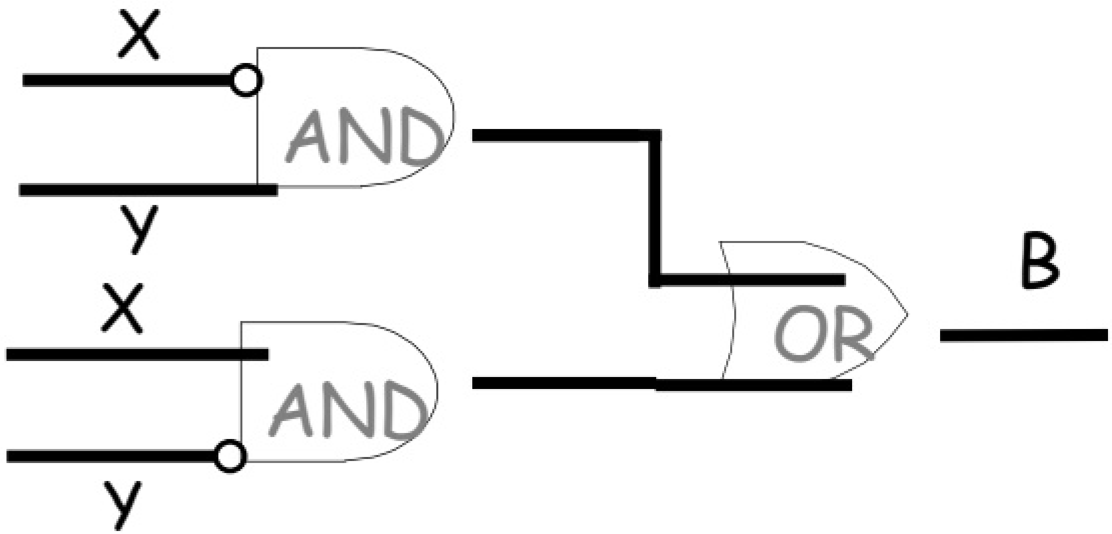


X	Y	B
0	0	0
0	1	1
1	0	1
1	1	0

X	Y	A
0	0	0
0	1	0
1	0	0
1	1	1



Binary Numbers (cont.)



X	Y	B
0	0	0
0	1	1
1	0	1
1	1	0

This is often called the eXclusive OR (XOR) circuit
We say that B is the exclusive OR of X and Y.

We write $B = X \oplus Y$.

A Logic Puzzle?

- Bob will go to the party if
  Ed goes OR Dan goes.
- Dan will go if Xena does NOT go AND
  Yanni goes.
- Ed will go if Xena goes AND
  Yanni does NOT go.

- This is addition without the carry
- Bob goes if the XOR of Xena and Yanni go.

A Simple Breakthrough

- We can represent information by bits.
  (So we interpret bits to mean things like numbers.)

- Then we re-interpret those bits as
  Logical True/False values.

- Finally we use Universal Method to construct
  circuits for operations on information
  (like Addition).

Intermission

- Questions??

- How are we going to build a circuit for addition?

Addition

X1 X2
Y1 Y2
==========
C Z1 Z2

C is the carry bit

X1	X2	Y1	Y2	Z1	Z2	C
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	0	1

Addition

X1 X2
Y1 Y2
=========
C Z1 Z2

C is the carry bit

Could have 3 circuits
16 inputs each

2 w/ 8 ANDs, 1 OR
1 w 6 ANDs, 1 OR

25 gates

X1	X2	Y1	Y2	Z1	Z2	C
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	0	1

Addition

```
     X1 X2
     Y1 Y2
==========
C   Z1   Z2
```

Rewrite as

```
  X2              X1
  Y2              Y1
====              C2
C2 Z2         ========
               C   Z1
```

Addition

X2

Y2

====

C2 Z2

X1

Y1

C2

====

C Z1

X1	Y1	C2	C	Z1
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

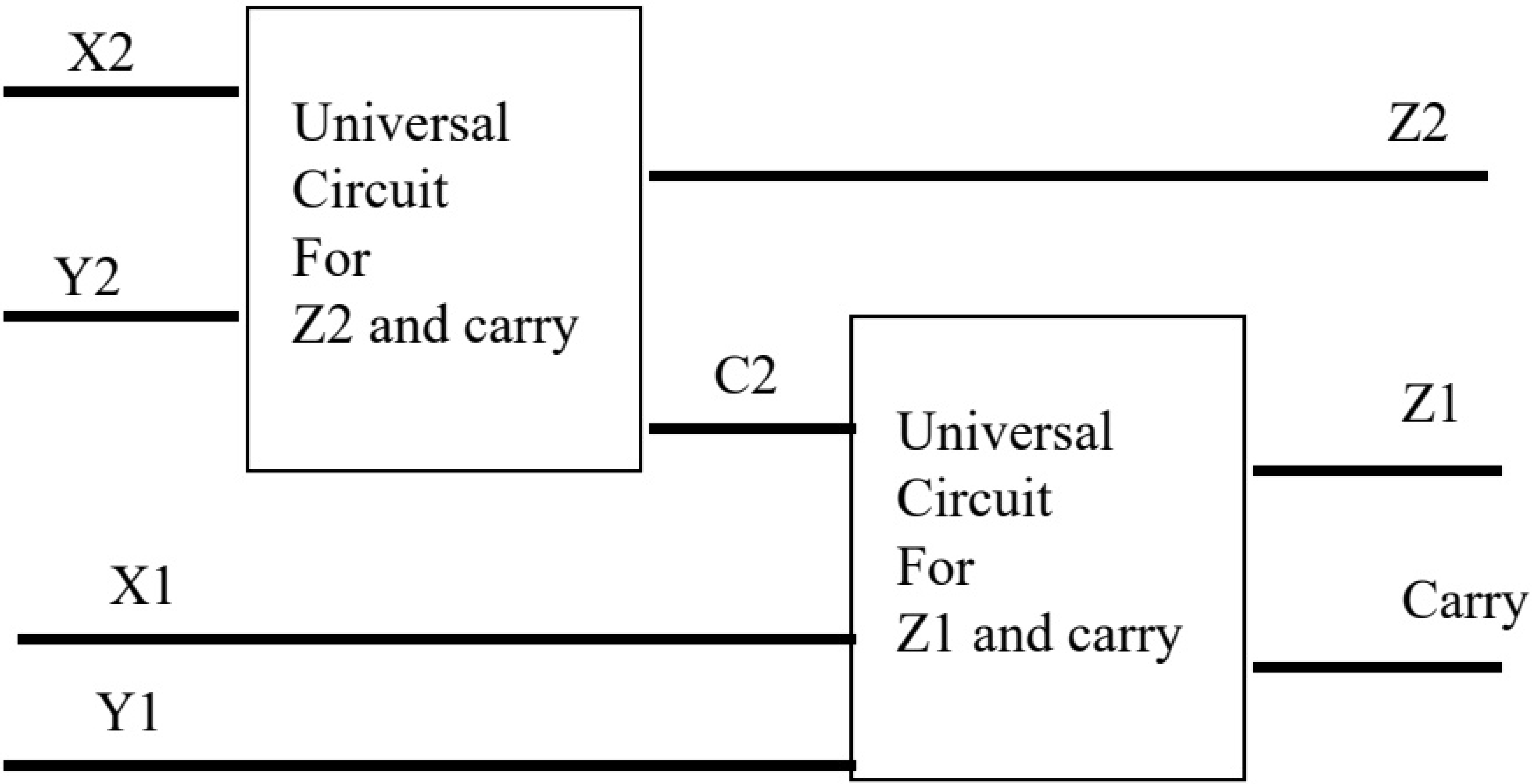
2 circuits

total of 4 gates

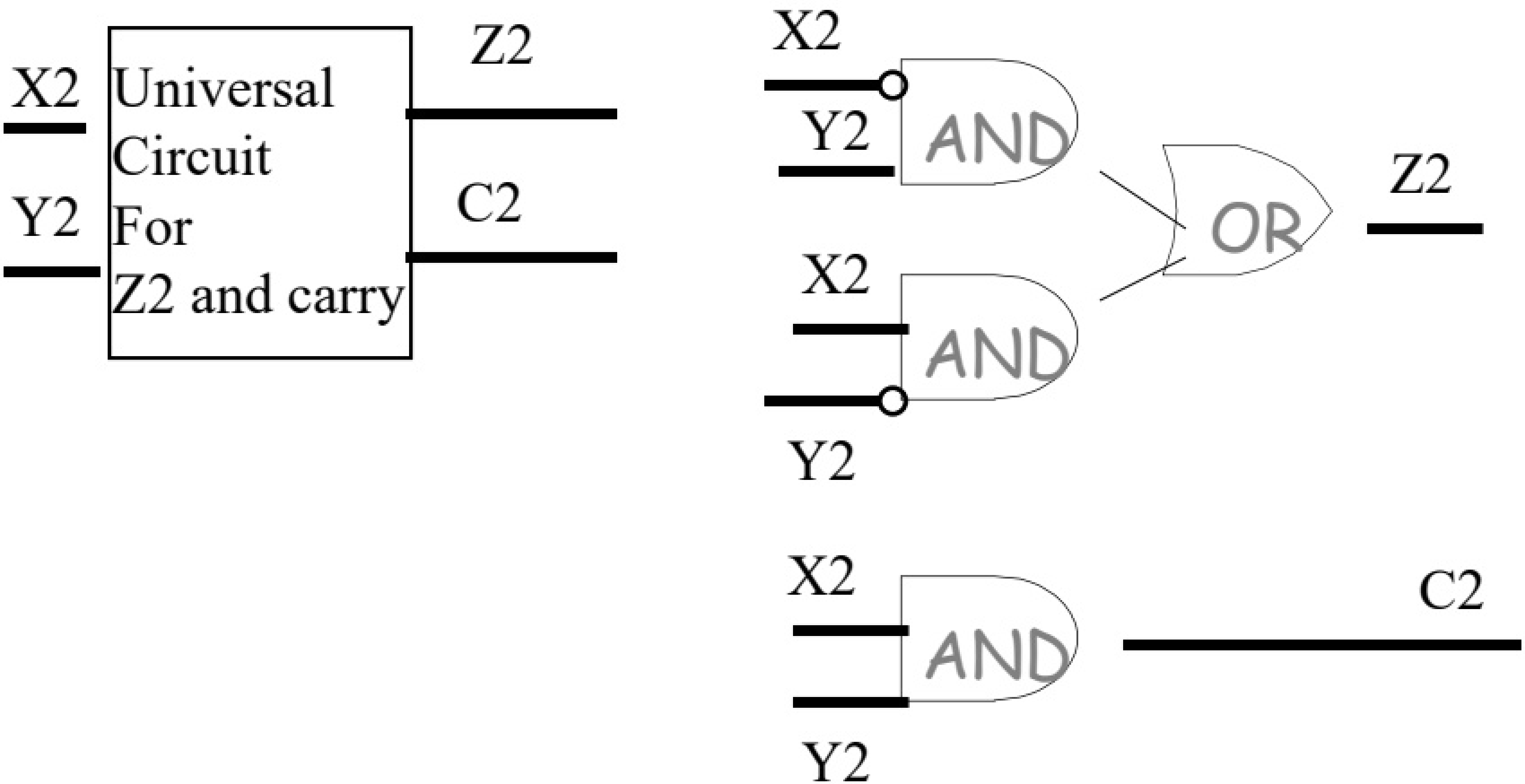
3 circuits

total of 10 gates

Addition


X2
Y2
Universal
Circuit
For
Z2 and carry
Z2
C2
Universal
Circuit
For
Z1 and carry
Z1
Carry
X1
Y1

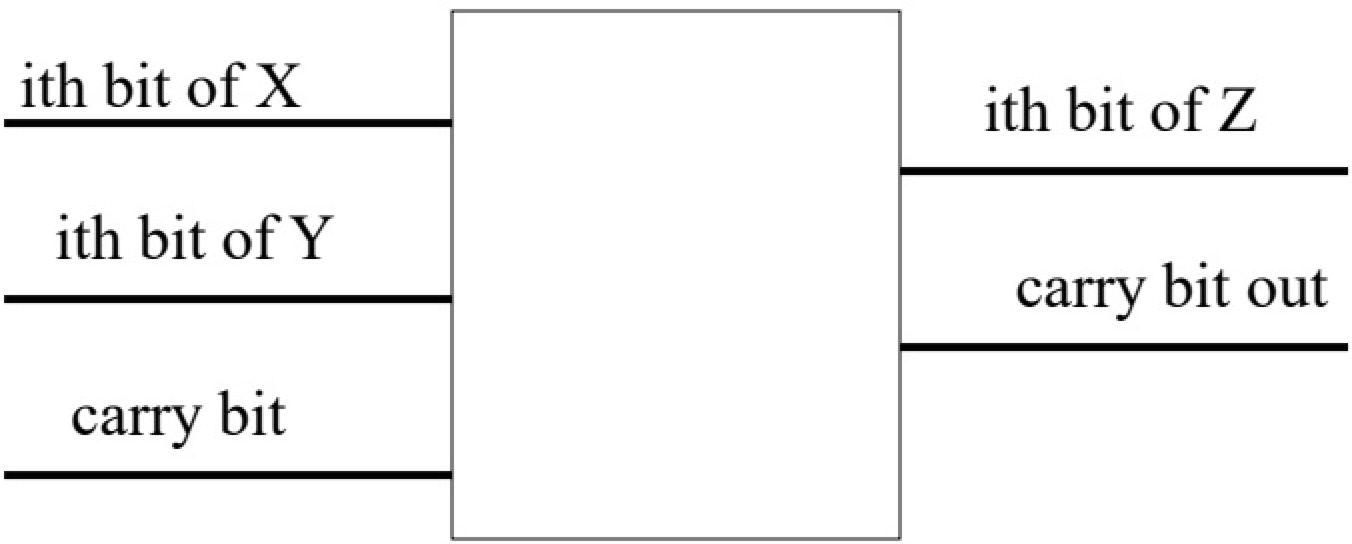
Inside the box



Addition

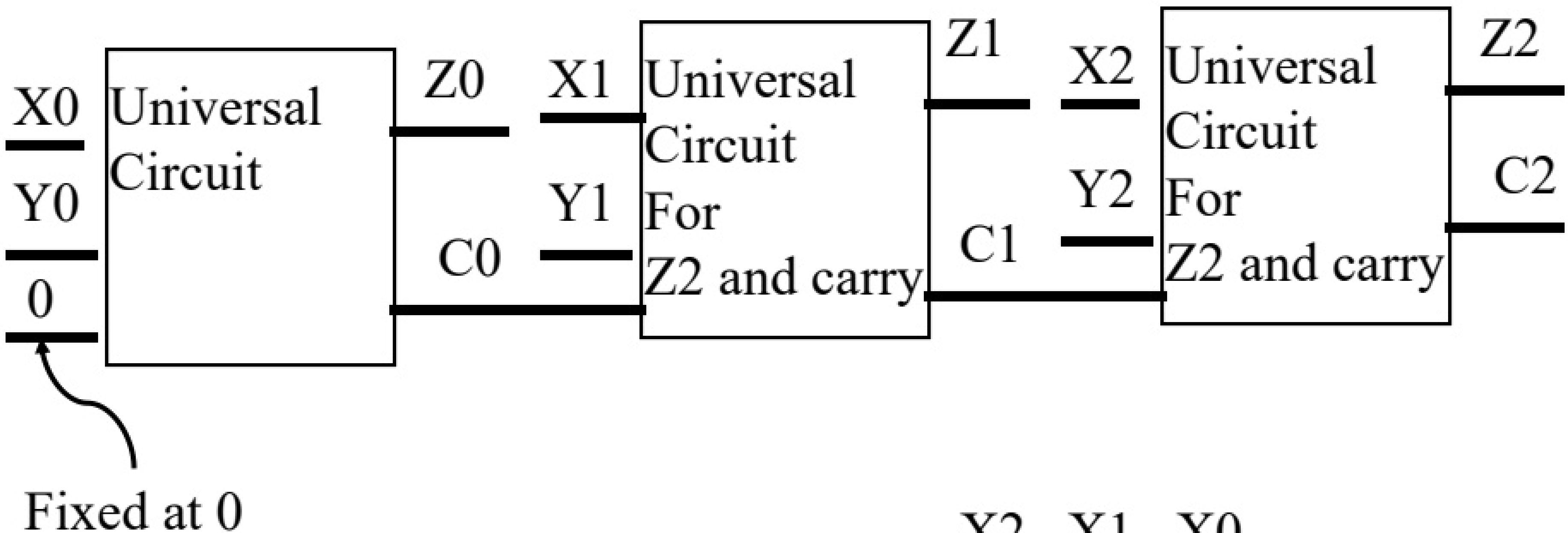
We can use 2 building blocks to add numbers of any length.

Actually we need only 1 building block



Abstraction in action -- This is a piece of a carry-ripple adder

Carry-Ripple Adder



X2 X1 X0
Y2 Y1 Y0
============
C2 Z2 Z1 Z0

Representing information

- How do we represent characters?
  - How many characters might we want to represent?
  - What characters might we want to represent?

Representing information

- How do we represent characters?
  - How many characters might we want to represent?
  - What characters might we want to represent?

• A-Z	26
• A-Z and a-z	52
• All the keys on my keyboard	104
• Maybe a power of 2?	128
• Maybe an even power of 2?	256
• Maybe an even bigger power of 2?	65536

Representing characters

- ASCII is the American Standard Code for Information Interchange. It is a 7-bit code.
- Many 8-bit codes contain ASCII as their lower half
- The ASCII standard was published by the United States of America Standards Institute (USASI) in 1968.

Unicode

- Universal Character Set (UCS) contains all characters of all other character set standards. It also guarantees round-trip compatibility, i.e., conversion tables can be built such that no information is lost when a string is converted from any other encoding to UCS and back.

- UCS contains the characters required to represent almost all known languages. This includes apart from the many languages which use extensions of the Latin script also the following scripts and languages: Greek, Cyrillic, Hebrew, Arabic, Armenian, Gregorian, Japanese, Chinese, Hiragana, Katakana, Korean, Hangul, Devangari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayam, Thai, Lao, Bopomofo, and a number of others. Work is going on to include further scripts like Tibetian, Khmer, Runic, Ethiopian, Hieroglyphics, various Indo-European languages, and many others.

- It's intended to use 31 bits (32768 possible characters)

What do we do in practice

- Problems
  - Bits represent too little – too many are needed
  - Decimal numbers don’t translate well into bits
- So,
  - Group into blocks of 4 and 8 bits
    - 8 bits = 256 characters, holds ASCII
    - 8 bits make 1 byte – things are organized into bytes
    - 4 bits make 1 nibble

Shorthand: Hexadecimal

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1

8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

Hexadecimal

- We can add numbers
  - 1+1=2, 2+2=4, 4+4=8, 4+8=C, 2+8=A, …
- We can combine 2 hexadecimal numbers to make a byte.
- It’s easier to read than 0’s and 1’s
- In ASCII
  - hex 41 through 5A represent A to Z
  - Hex 61 through 7A represent a to z

Summary

- Review of gates and the Universal Method
- Show that the universal method can lead to very big circuits
- Fix the problem
- Demonstrate the fix
  - Carry-ripple adder (a meaty example)
- Representing characters
  - Hexadecimal
  - Bytes, nibbles

Next Time:
Memory

Now that we can represent it, how do we store it??