Adding Decimal Numbers

```
    4  6  5
     5  4  3
==========
```

We all know that the answer is 1008 but how do we do this?

Adding Decimal Numbers

```
   4  6  5
    5  4  3
=========
```

Start from the right
5+3 = 8 and there is
no carry

Adding Decimal Numbers

```
    4  6  5
     5  4  3
==========
           8
```

Start from the right
5+3 = 8 and there is
no carry

Next, 6+4 = 10
Write the 0, carry the 1

Adding Decimal Numbers

```
 1
 4 6 5
  5 4 3
==========
     0 8
```

Start from the right
5+3 = 8 and there is
no carry

Next, 6+4 = 10
Write the 0, carry the 1

Adding Decimal Numbers

```
   1
   4 6 5
    5 4 3
==========
      0 8
```

Start from the right
5+3 = 8 and there is
no carry

Next, 6+4 = 10
Write the 0, carry the 1

Finally, 1+4+5 = 10
Write the 0, carry the 1

Adding Decimal Numbers

```
 1   1
     4  6  5
     5  4  3
==========
 1   0  0  8
```

Start from the right
5+3 = 8 and there is
no carry

Next, 6+4 = 10
Write the 0, carry the 1

Finally, 1+4+5 = 10
Write the 0, carry the 1

Adding Decimal Numbers

```
1  1
   4 6 5
   5 4 3
==========
1  0 0 8
```

Start from the right
5+3 = 8 and there is no carry
Next, 6+4 = 10
Write the 0, carry the 1
Finally, 1+4+5 = 10
Write the 0, carry the 1

At each stage, take 2 addends (5 and 3 or 6 and 4 or 4 and 5) and a bit telling whether there is a carry and produce a sum bit and a bit telling if there is a carry.

Bad news

- It is inconvenient to add decimal numbers in a computer
  - Truth tables are easier than addition tables
  - The universal method wants to work from truth tables

Bad news/Good News

- Bad news – It is inconvenient to add decimal numbers in a computer
  - Truth tables are easier than addition tables
  - The universal method wants to work from truth tables
- Good news – the same thing works for binary numbers

Adding Binary Numbers

- At each stage, take 2 addends and a bit telling whether there is a carry and produce a sum bit and a bit telling if there is a carry.

Adding Binary Numbers (cont.)

$$
\begin{array}{rcccc}
 & & 1 & 1 & \\
 & & 1 & 0 & 1 \\
+ & & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 0 & 0
\end{array}
$$

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

Sidebar – what are binary numbers

- Base 2 rather than base 10
- Decimal numbers
  - We write $(1234)_{10}$ to represent a decimal number that has 4 units, 3 tens, 2 hundreds and 1 thousand.
  - $(1234)_{10} = 1 \text{ x}10^3 + 2 \text{ x}10^2 + 3 \text{ x}10^1 + 4 \text{ x}10^0$

Sidebar – what are binary numbers

- Base 2 rather than base 10
- Decimal numbers
  - We write $(1234)_{10}$ to represent a decimal number that has 4 units, 3 tens, 2 hundreds and 1 thousand.
  - $(1234)_{10} = 1 \text{ x}10^3 + 2 \text{ x}10^2 + 3 \text{ x}10^1 + 4 \text{ x}10^0$
- Binary numbers
  - We write $(1001)_2$ to represent a decimal number that has 1 unit, 0 2's, 0 4's and 1 8.
  - $(1001)_2 = 1 \text{ x}2^3 + 0 \text{ x}2^2 + 0 \text{ x}2^1 + 1 \text{ x}2^0$

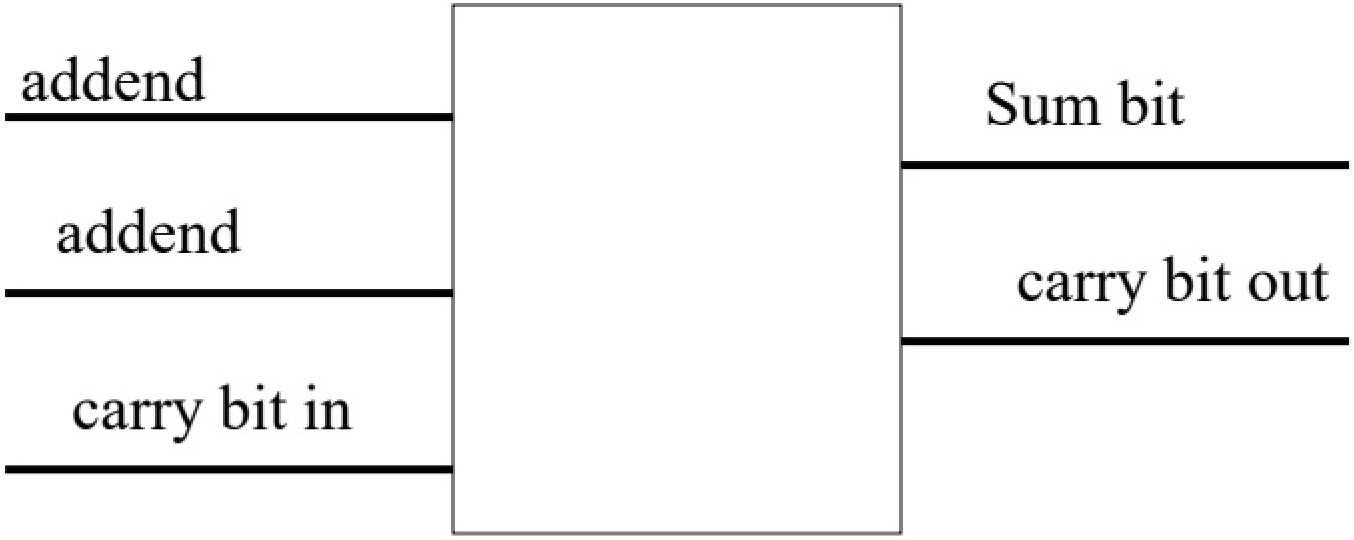
Adding Binary Numbers

- At each stage, take 2 addends and a bit telling whether there is a carry and produce a sum bit and a bit telling if there is a carry.

Adding Binary Numbers

At each stage, take 2 addends and a bit telling whether there is a carry and produce a sum bit and a bit telling if there is a carry.

We can build a black box to do this



Adding Binary Numbers

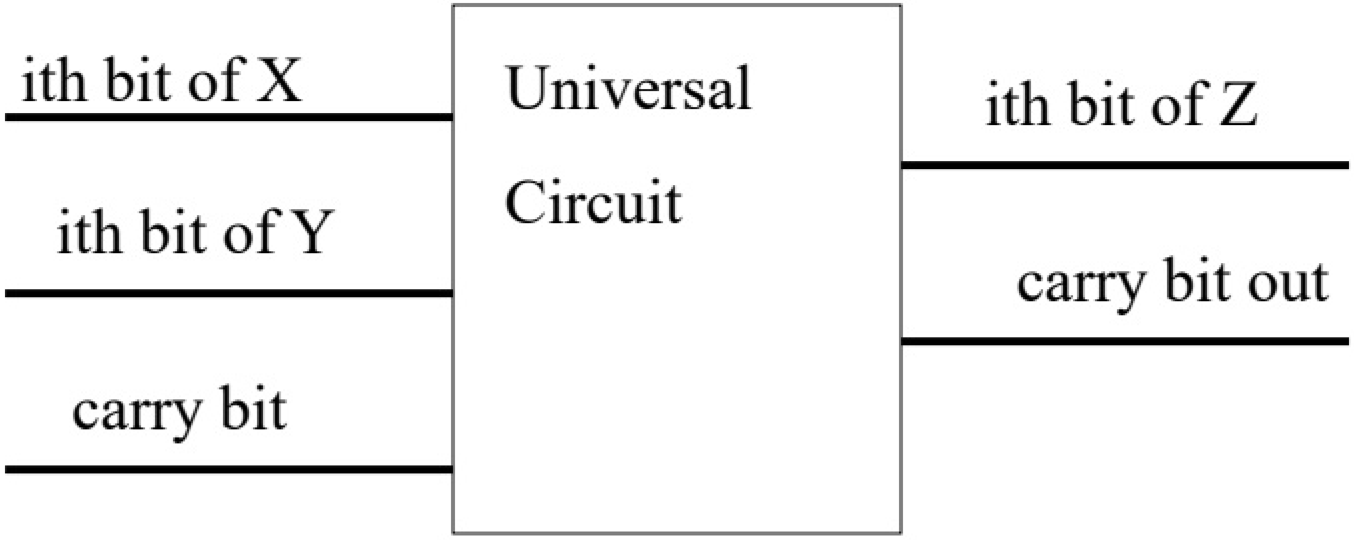
- Write the numbers as
  - X4 X3 X2 X1 X0
  - Y4 Y3 Y2 Y1 Y0
- Represent the sum as
  - C Z4 Z3 Z2 Z1 Z0

Adding Binary Numbers

- Write the numbers as
  - X4 X3 X2 X1 X0
  - Y4 Y3 Y2 Y1 Y0
- Represent the sum as
  - C Z4 Z3 Z2 Z1 Z0
- Start with X0, Y0, 0 in, Z0 and C0 out
- Then X1,Y1,C0 in and Z1 and C1 out.
- And so on

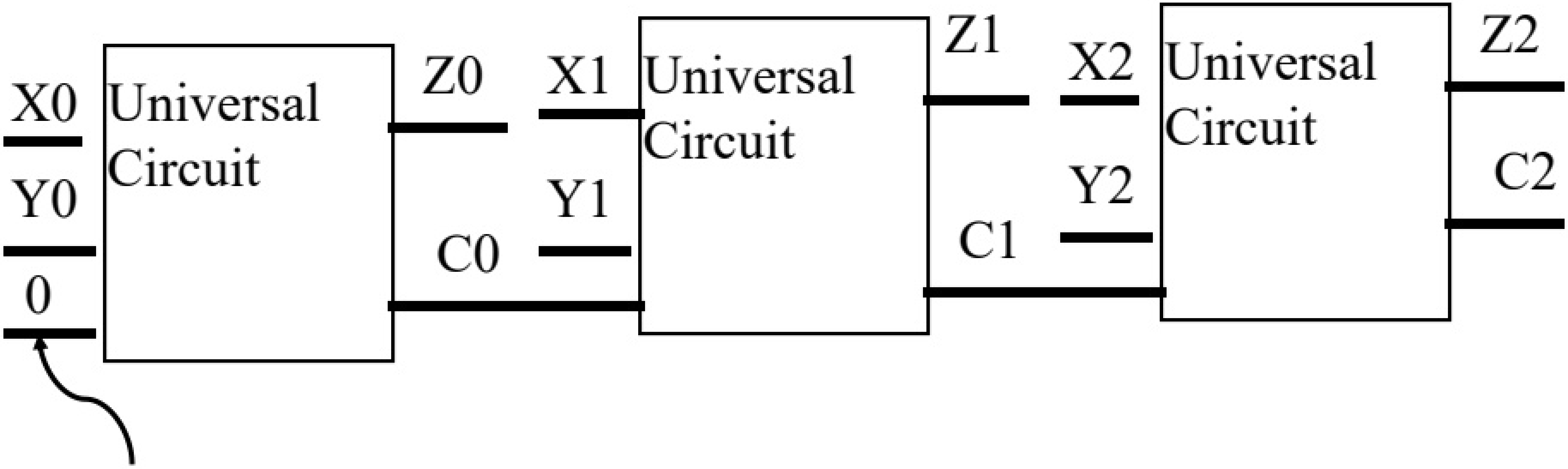
Adding Binary Numbers

At the ith stage



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

Carry-Ripple Adder



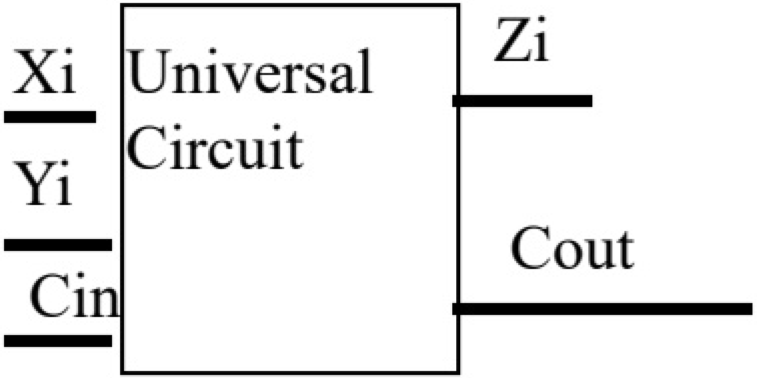
Fixed at 0

X2 X1 X0
Y2 Y1 Y0

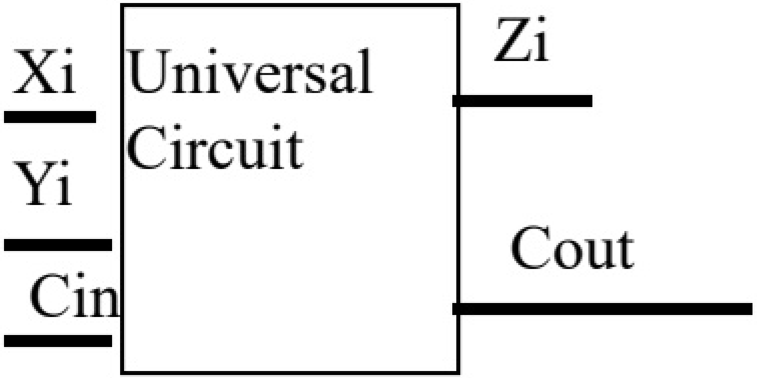
============

C2 Z2 Z1 Z0

What’s inside the box?



What’s inside the box?



Do the problem set and find out

Carry ripple adders

- Useful for some applications
- Too slow for other
  - Delay while waiting for the carry to ripple

- One solution – add larger blocks
  - Details on the homework assignment

Arithmetic Logical Unit (ALU)

- This is the part of the CPU that does arithmetic and comparison operations
- We’ve built a piece of the ALU
- With abstraction, we could build the other parts
  - Multiplication/subtraction/division
  - Comparison
- Then we would write a language to let the user talk to the ALU (programming)

Pause

- Questions??

- How to build the other parts of the ALU?
  - Use abstraction

- Back to representing information

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

Sidebar – what are hexadecimal numbers

- Base 16 rather than base 10
- Decimal numbers
  - We write $(1234)_{10}$ to represent a decimal number that has 4 units, 3 tens, 2 hundreds and 1 thousand.
  - $(1234)_{10} = 1 \text{ x}10^3 + 2 \text{ x}10^2 + 3 \text{ x}10^1 + 4 \text{ x}10^0$
- Hexadecimal numbers
  - We write $(2AC4)_{16}$ to represent a decimal number that has 1 units, 12 16's, 10 256's and 2 65,536's
  - $(2AC4)_{16} = 1 \text{ x}16^3 + 0 \text{ x}16^2 + 0 \text{ x}16^1 + 1 \text{ x}16^0$

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

Memory

Memory

- Logic circuits we've seen so far have no memory. They just transform input to output.

- Can we make logic circuits that remember?

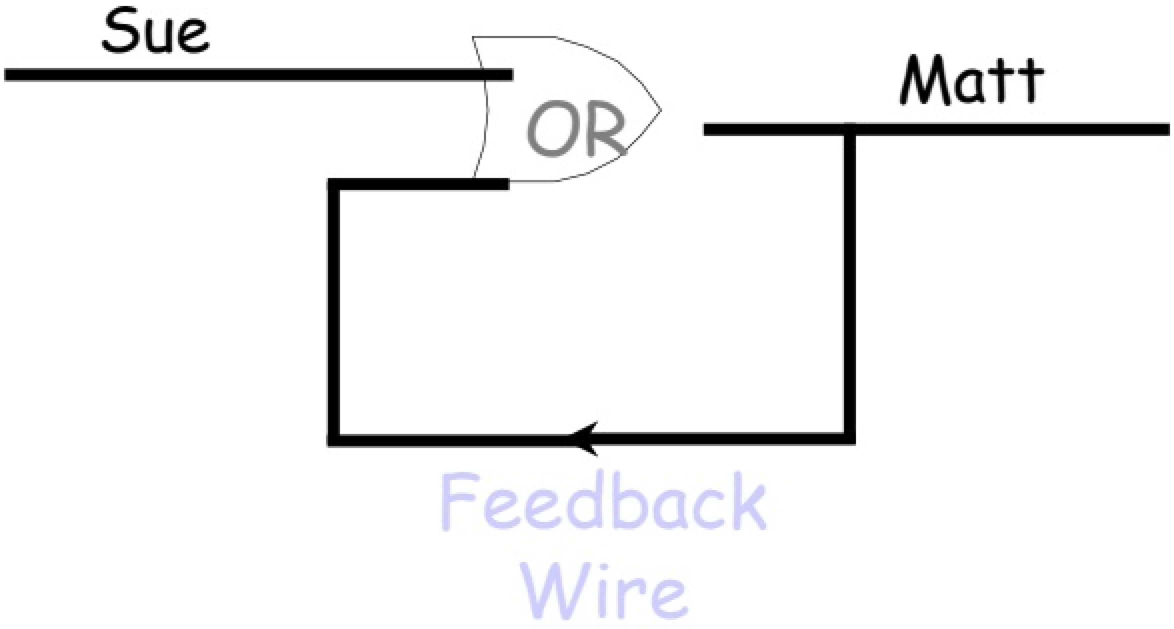
The Stubborn Guy

- Matt really likes Sue, but he doesn't like changing his mind… so:

- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.

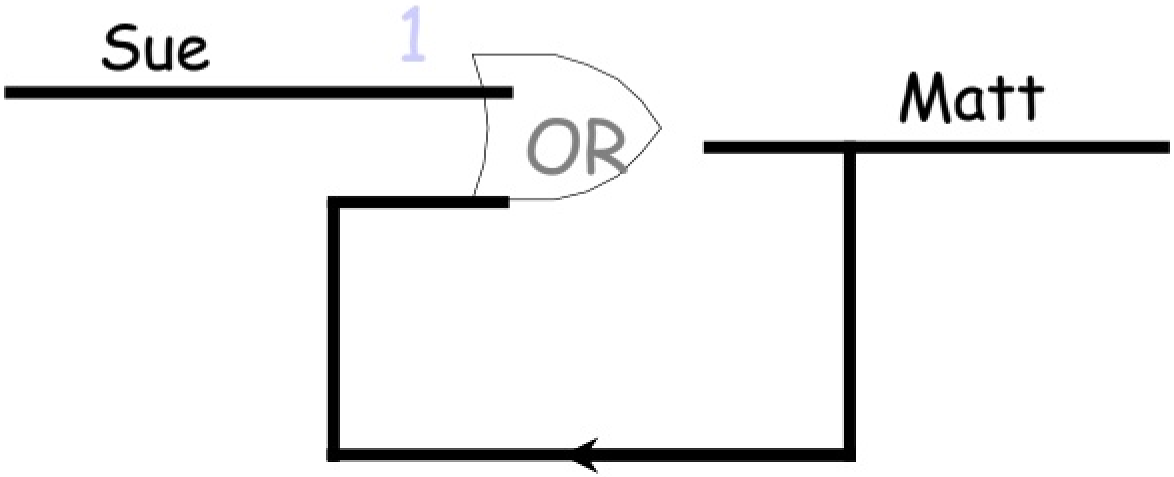
The Stubborn Guy: Feedback

- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.



The Stubborn Guy: Feedback

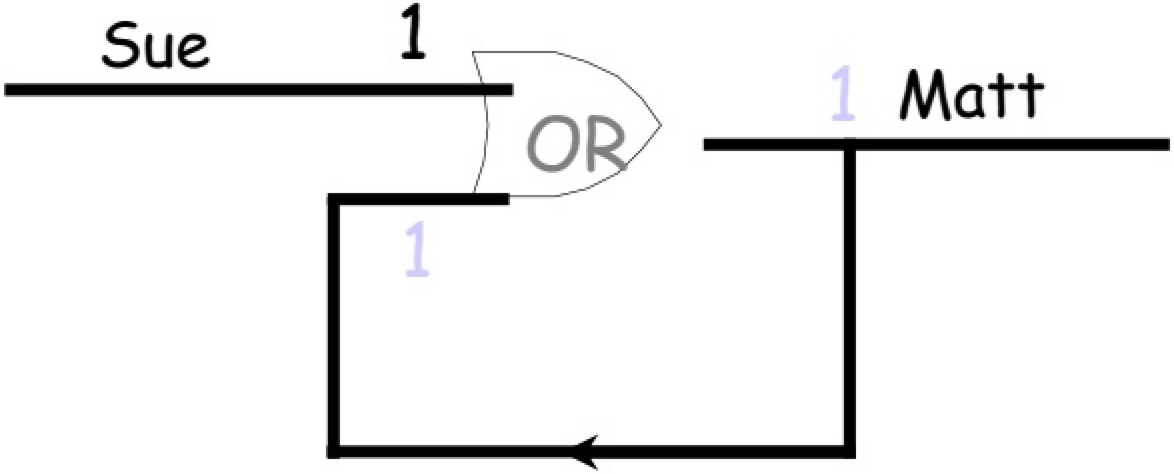
- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.



- If Sue decides to go...

The Stubborn Guy: Feedback

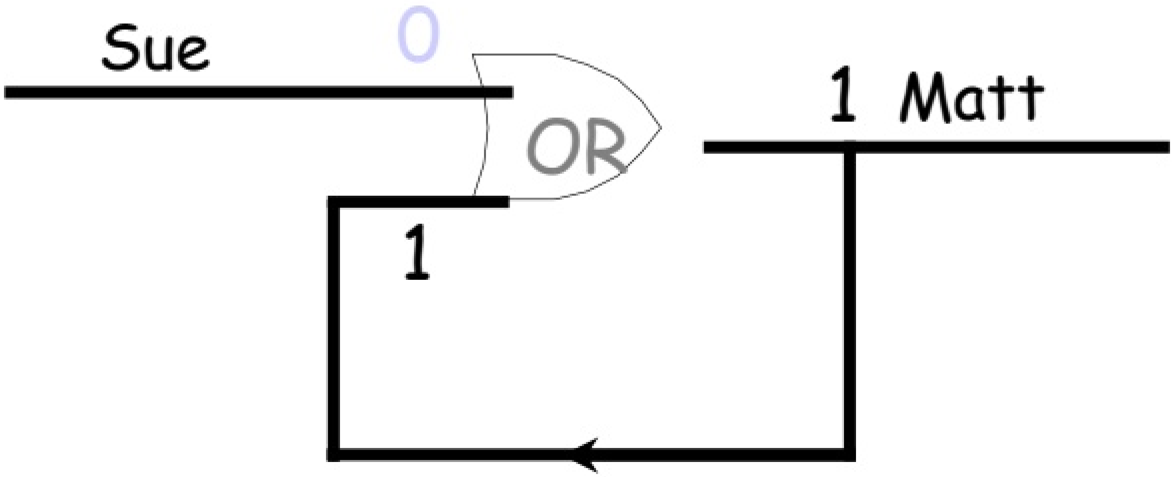
- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.



- If Sue decides to go, Matt will go.

The Stubborn Guy: Feedback

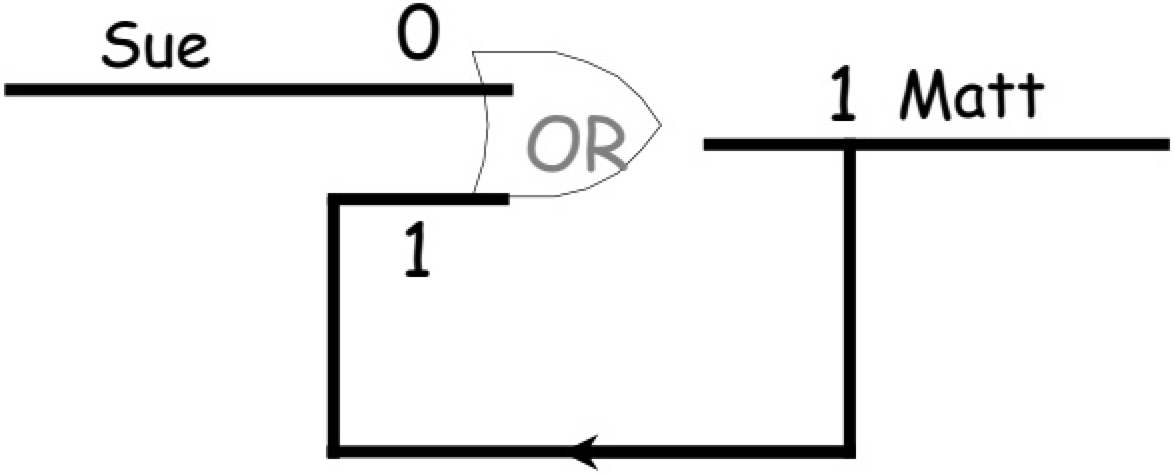
- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.



- If Sue changes her mind…

The Stubborn Guy: Feedback

- Matt decides to go to the party if Sue decides to go OR if he (Matt) already feels like going.



- If Sue changes her mind, Matt will still go!
- Once he decides to go, we can't ever get Matt to change his mind.