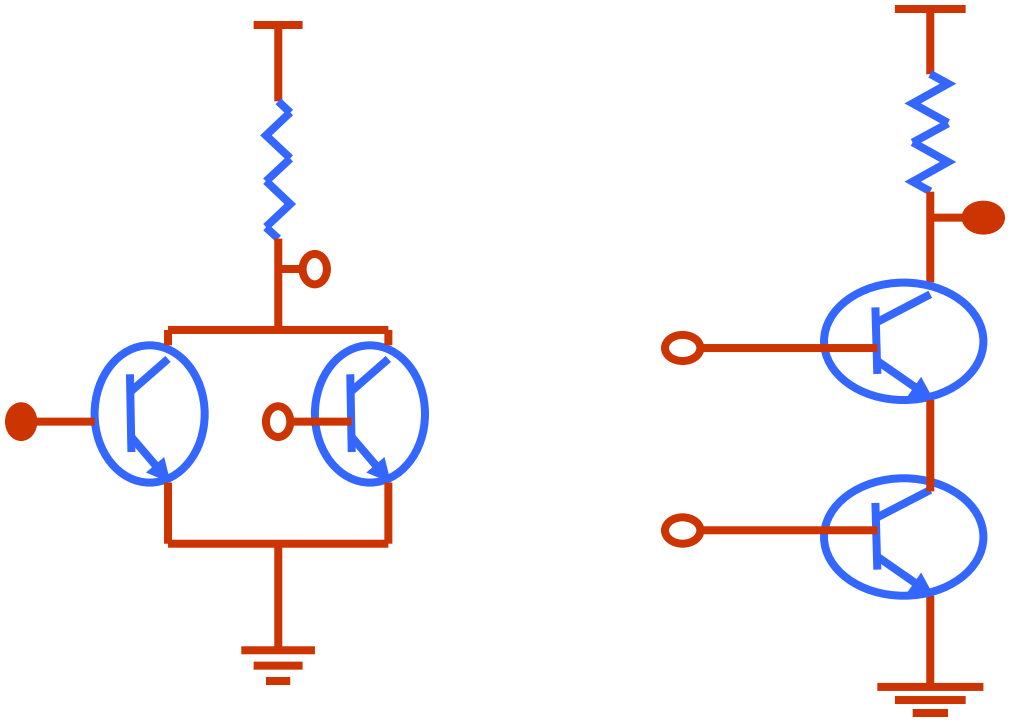
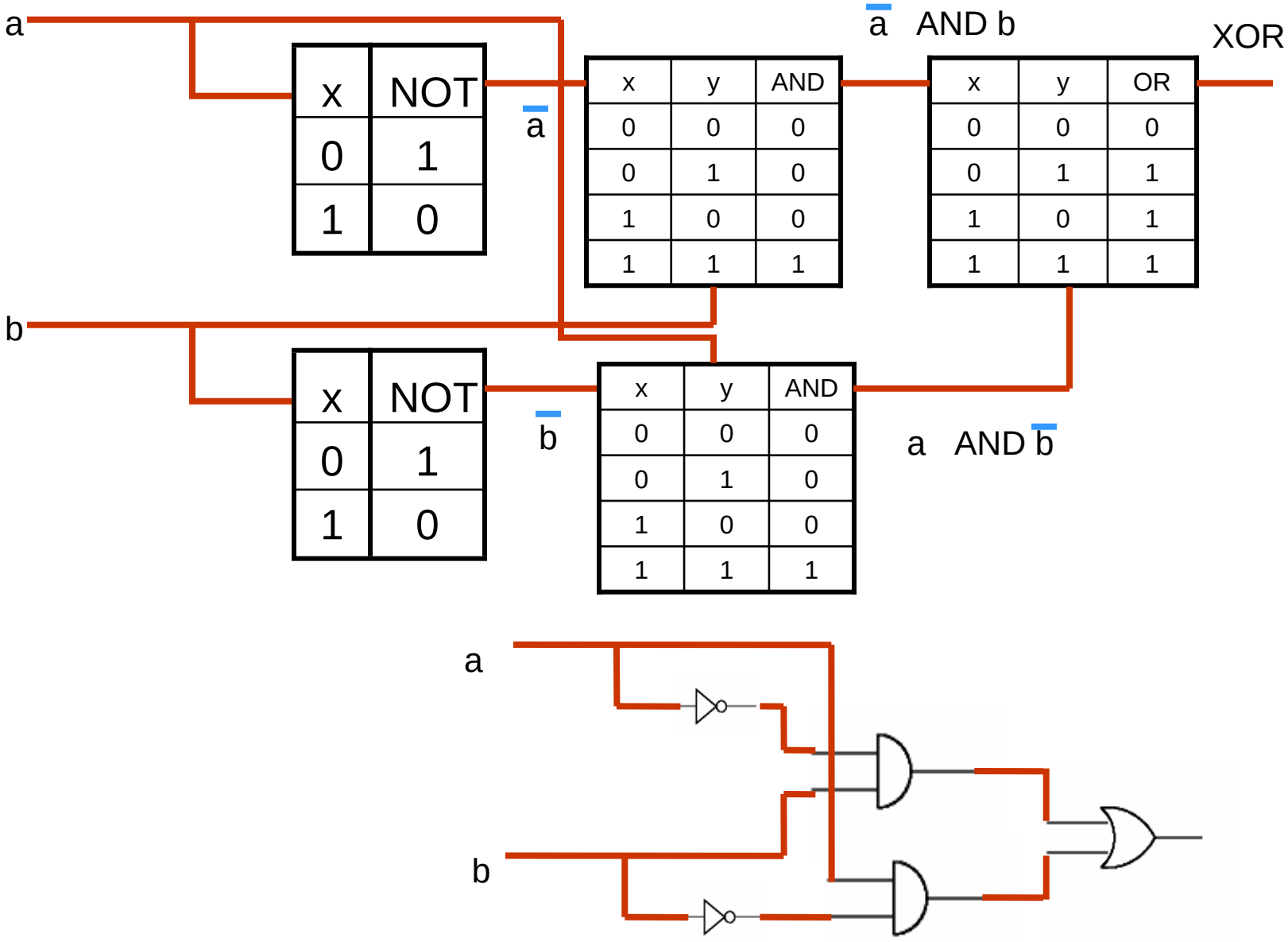
CS 130 : Computer Systems - IV

Shankar Balachandran
Dept. of Computer Science & Engineering
IIT Madras

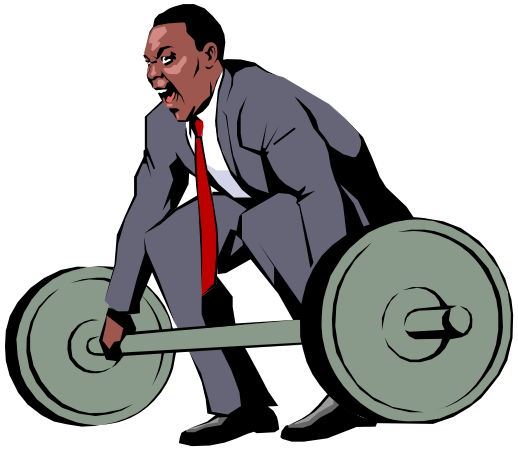
Recap

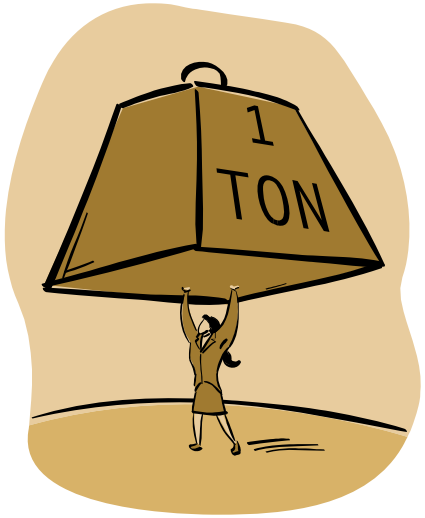


Recap


a
b
x NOT
0 1
1 0
$\bar{a}$
x NOT
0 1
1 0
$\bar{b}$
x y AND
0 0 0
0 1 0
1 0 0
1 1 1
$\bar{a}$ AND b
x y OR
0 0 0
0 1 1
1 0 1
1 1 1
XOR
x y AND
0 0 0
0 1 0
1 0 0
1 1 1
a AND $\bar{b}$
a
b

Recap





- Issues with scale
  - Computation
  - Storage
- Dealing with complexity
  - Break down the problem

Computers Also Have Constraints

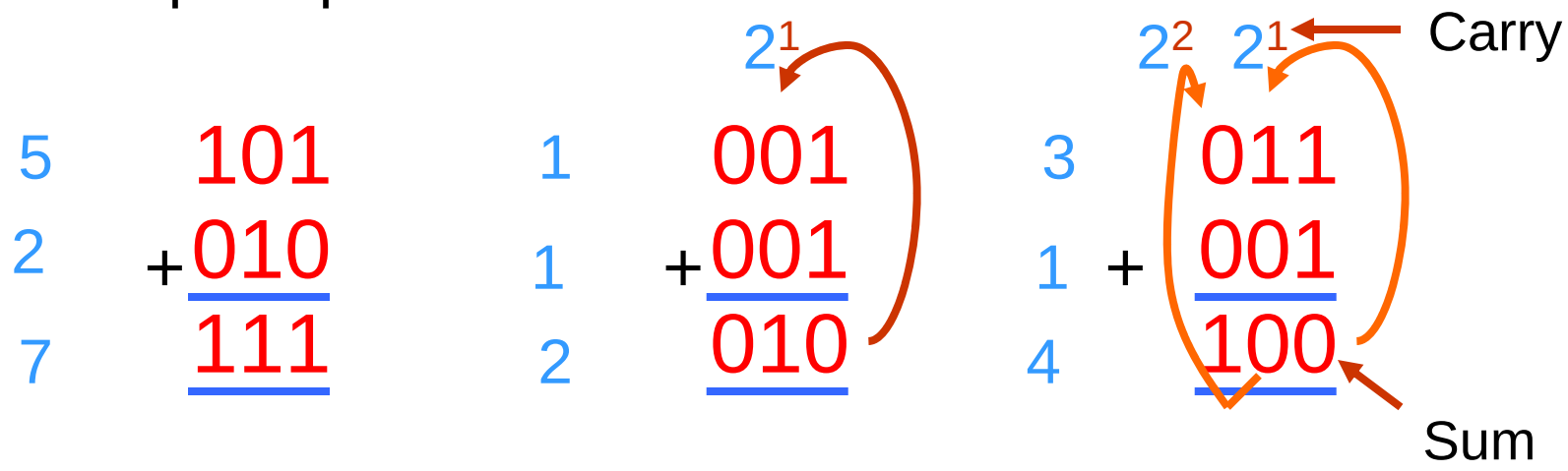
- Computation
  - How fast can a computer run?
  - What kind of computations can it do?
- Storage
  - How much can it store?
  - How does it represent data?
- This is what engineering is about :
  - Solving problems under constraints
  - Usual mantra :
    - Better, cheaper, quicker

How Do Engineers Deal with Complexity?

- Design small sub-systems and put them together
- Deal with complexity, one subsystem at a time
- When you integrate subsytems
  - Make sure the whole assembly works fine too
- Use tools
- Current day hardware and software systems are both complex
  - Many ways to deal with complexity
  - Many are tested, many are not

Let's Revisit Binary Systems

- 0's and 1's
- Simple Operation : Addition



Let's Build A Circuit For Adder

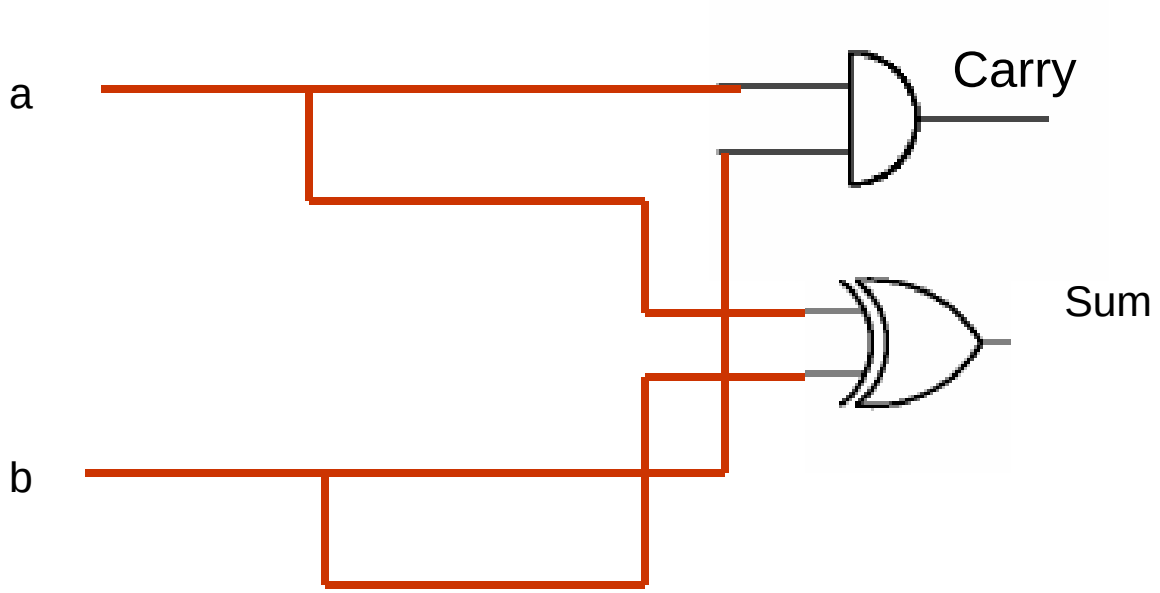
- Two binary variables
- What is the maximum value of sum?
- How many bits do we require to represent the sum?
- Let's design the circuit
  - Incoming data form inputs A and B
  - Outgoing data : Sum and Carry

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

You recognize the tables?

Sum and Carry

- Carry := A AND B
- Sum := A XOR B



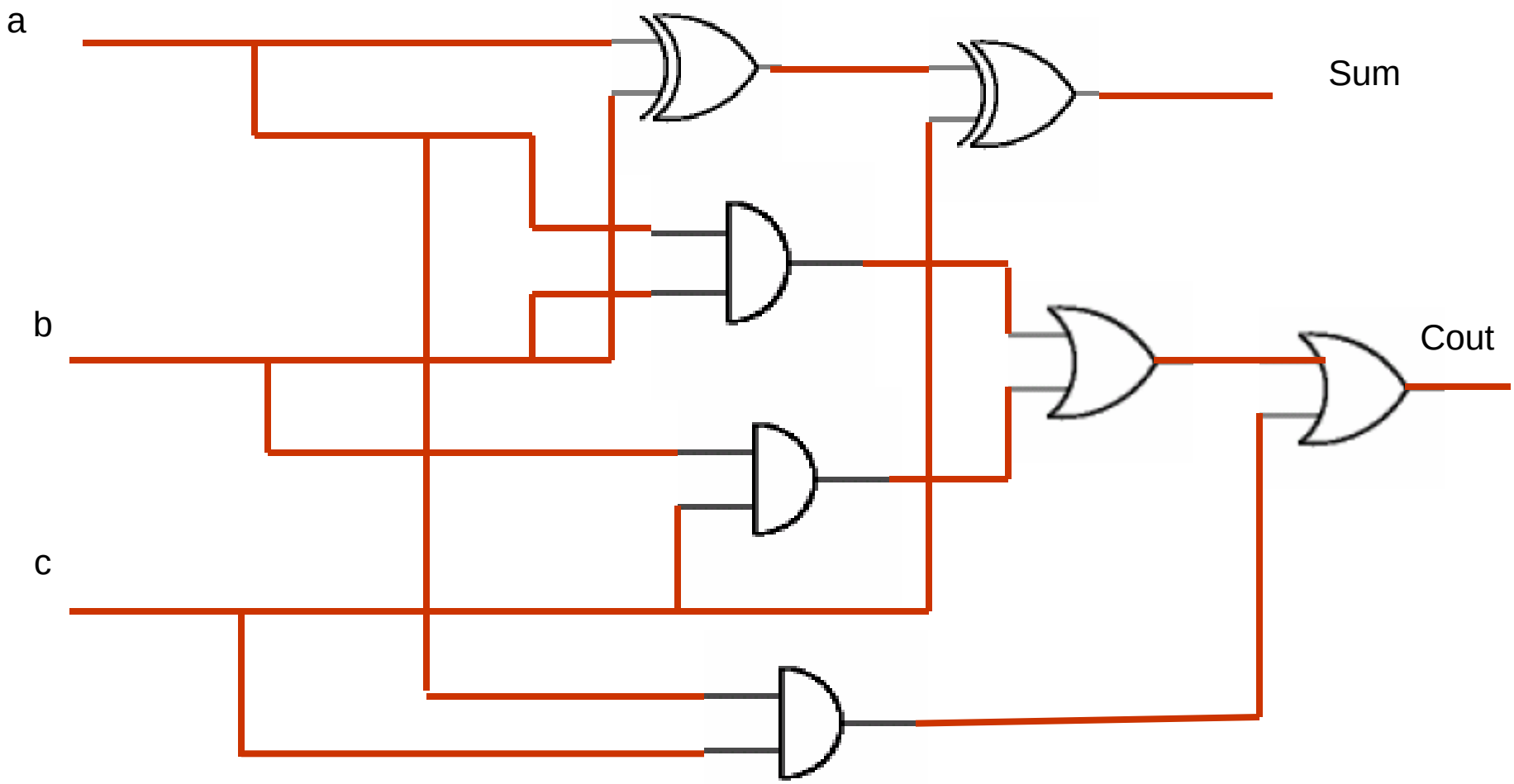
Want to Add Three Numbers?

A	B	C	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum = A XOR B XOR C

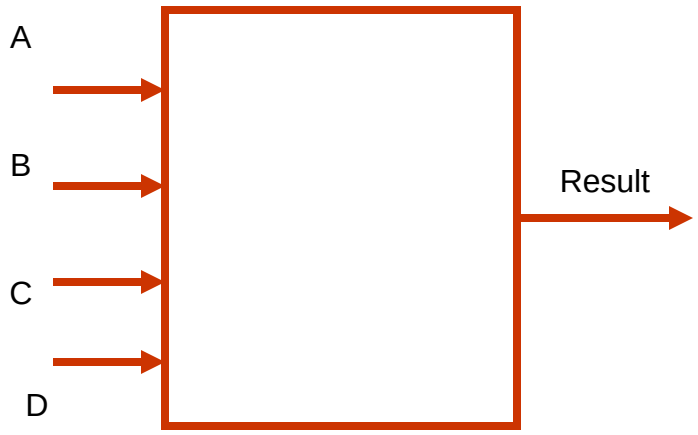
COUT = A.B + B.C + C.A

Circuit For This Binary Adder

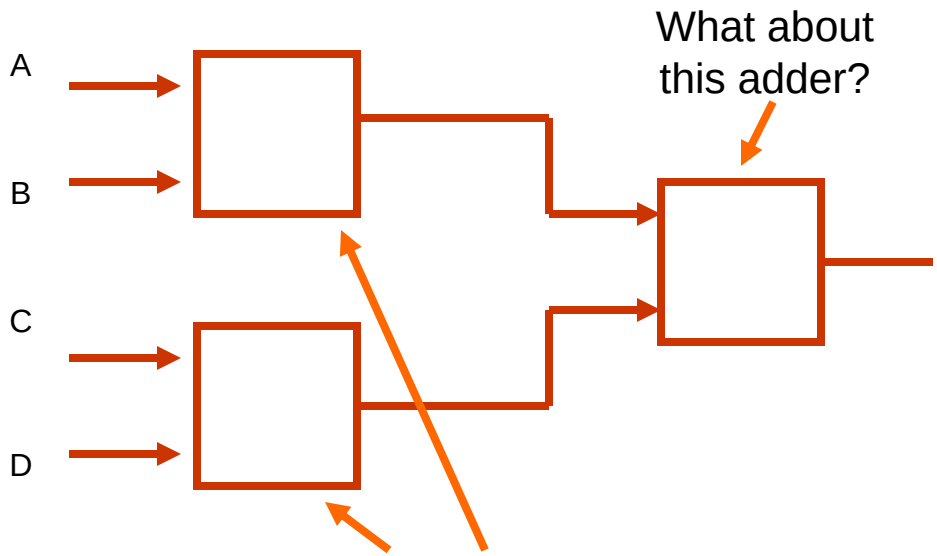


What if You Want to Add More Binary Values?

- You cannot use this table based approach
- Use the engineering approach
  - Break down the problem



Need to add four numbers!



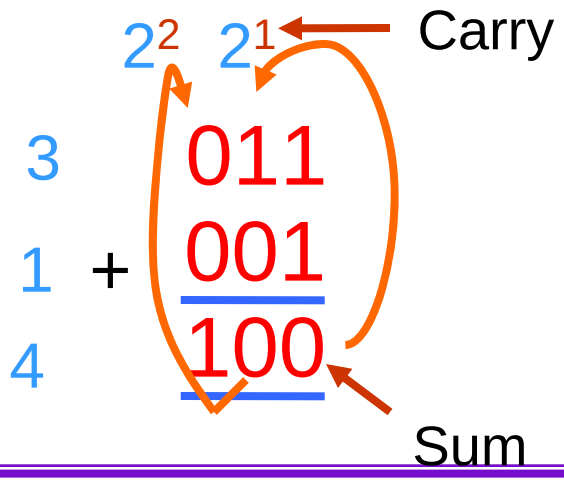
These adders take two inputs

Let's Add Two Numbers Again

- Now the numbers that we are going to add are multi-bit numbers
  - Like the examples that we saw before
- How do we attack this problem?
  - Go back to how you do decimal addition.
- You should ask me a few questions now ☺
  - Hint : Define multi
  - How big is multi?
- What if the numbers are two-bit numbers?
  - Lookup?
  - Iterative?

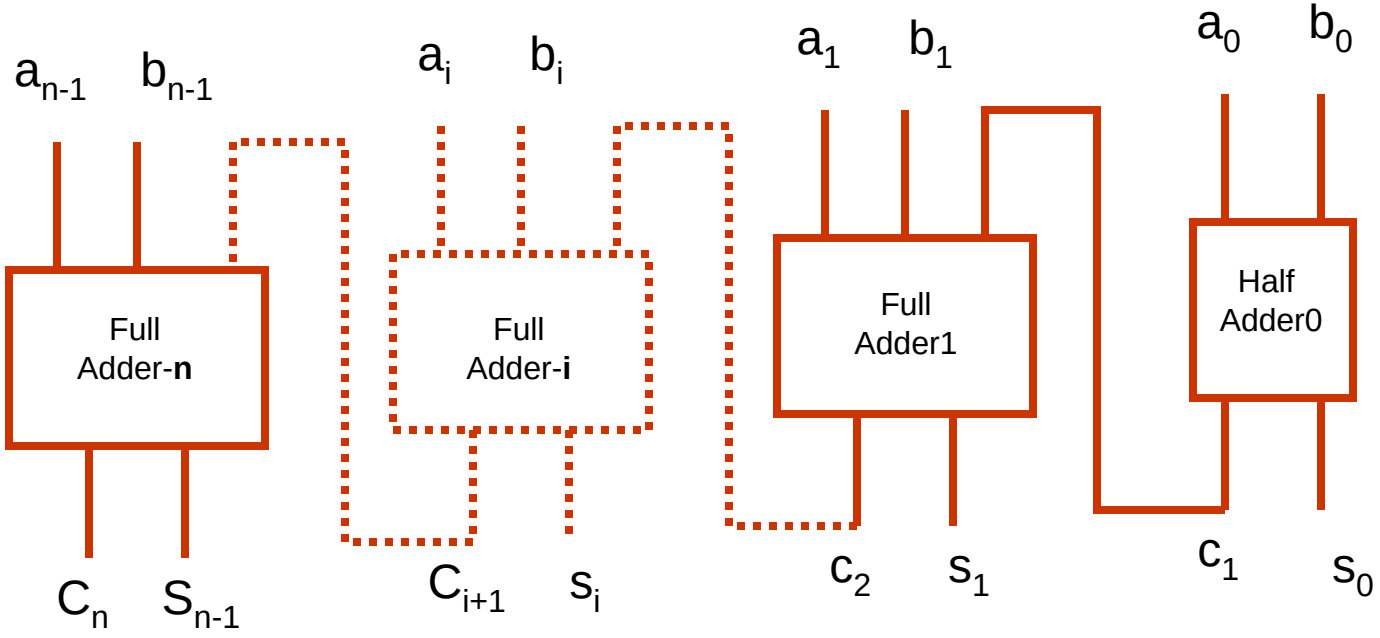
Two Binary Adders So Far

- Half Adder
  - Takes two binary inputs and adds them up
  - Gives sum and carry as output
- Full Adder
  - Takes three binary inputs and adds them up
  - Gives sum and carry as output
- Can we use half-adders and full-adders and do multi-bit addition?
  - YES!

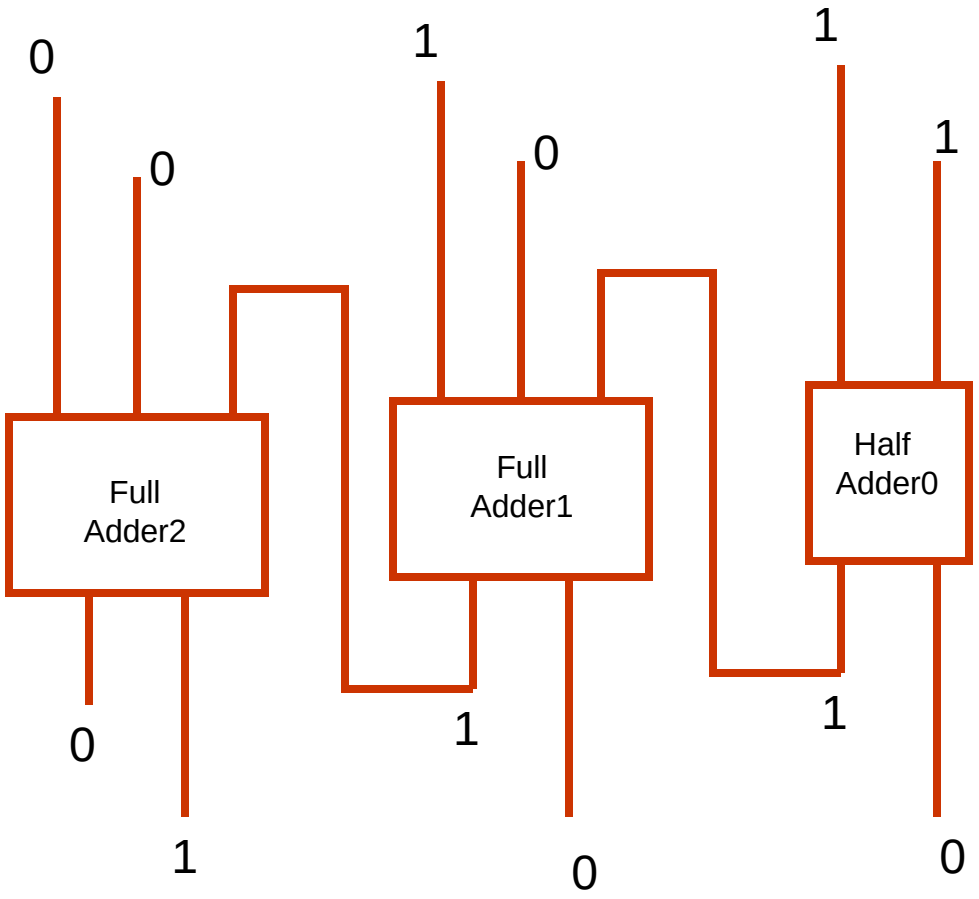


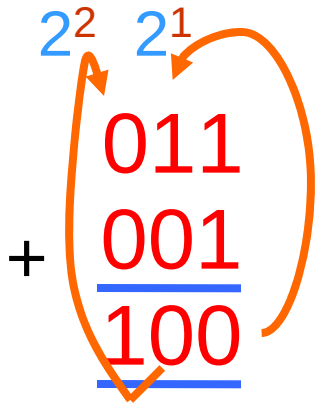
Multi-Bit Adder

- Let A = $a_{n-1}\ldots a_2 a_1 a_0$
- Let B = $b_{n-1}\ldots b_2 b_1 b_0$
- How many bits in Sum?
- How many bits in carry?



Example


0
0
1
0
1
1
Full
Adder2
Full
Adder1
Half
Adder0
0
1
1
0
1
0


2^2 2^1
011
+ 001
100