

Computer Systems 1
Lecture 4

Boolean Algebra and Arithmetic

Dr. John T. O'Donnell
School of Computing Science
University of Glasgow

Copyright ©2019 John T. O'Donnell

Topics

- 1 Boolean algebra
- 2 Addition and subtraction
 - Half adder
 - Full adder
 - Ripple carry adder
 - Subtraction
- 3 Circuit simulation
- 4 To do

Boolean algebra

- Elementary algebra is about the properties of operations on numbers:
 $+$ $-$ \times \div
 - ▶ Example: $x + y = y + x$
 - ▶ Elementary algebra helps you solve many problems involving numbers
- There are many different algebras, for different kinds of object
- Boolean algebra is about the operations on truth values: 0 and 1
 - ▶ Example: $x \vee y = y \vee x$
 - ▶ Boolean algebra helps you solve many problems involving truth values

Why learn about Boolean algebra?

- We will use it just a little, not very much
- But you will encounter it again and again in computer science, so a brief introduction helps!
- Boolean algebra makes it easier to understand tricky logic
 - ▶ In digital circuits
 - ▶ In programming with conditionals
- It's also important in the history and philosophy of mathematics

Constants and operations of Boolean algebra

- There are two values: 0, 1
- We can use variables to stand for a value: x, y, z, \dots
- There are three operators
- Each operator corresponds exactly to a logic gate
- Boolean algebra describes the values of signals in a digital circuit
- (There is also another operator, logical implication, but it can be expressed using the others.)

Algebra	Name	Circuit
$\neg x$	not	inv x
$x \vee y$	or	or2 x y
$x \wedge y$	and	and2 x y

Definition of the operations

x	$\neg x$
0	1
1	0

x	y	$x \wedge y$	$x \vee y$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

These are the same as the definitions of the logic gates (except we don't have a Boolean operator for the “exclusive or” gate)

Laws of Boolean algebra

- The basic rules describing logical operations
- Called laws, axioms, rules, — what you call them isn't important
- You can use these to reason about circuits
 - ▶ Simplify expressions
 - ▶ Calculate signal values
 - ▶ Prove that two circuits calculate the same value
 - ▶ Derive circuits to meet a specification
- For this course: **don't memorise the laws** (shown on the next slides), but do **know the truth tables for the logic gates**

Operations with constants

You can verify these using the truth tables for the operations

$$x \wedge 0 = 0$$

$$x \wedge 1 = x$$

$$x \vee 0 = x$$

$$x \vee 1 = 1$$

These help to work out the values of signals in a circuit

Idempotence

In general, an operation is idempotent if doing it several times is the same as doing it once.

- On some web sites, if you have an account and register again, it just puts you in your existing account. Registering twice is the same as registering once: here, registration is idempotent.
- But on some other web sites, if you register three times it gives you three separate accounts, which is annoying to deal with. Here, registration is not idempotent.

Here's the mathematical definition:

$$x \vee x = x$$

$$x \wedge x = x$$

It's straightforward to check these with truth tables: check that they hold for both cases: when $x=0$ and when $x=1$

Commutativity

Very important! These say that you can swap around the order of inputs to an and-gate or an or-gate without changing the result

$$x \vee y = y \vee x$$

$$x \wedge y = y \wedge x$$

Associativity

These give a way to compute the logical and/or of many inputs, by using several 2-input and/or gates

$$x \vee (y \vee z) = (x \vee y) \vee z$$

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

These laws are extremely important in advanced computer science: they are crucial in

- high performance circuit design
- programming massively parallel high performance computers

Distributive and absorption laws

Useful in proving correctness of circuits, but we won't need them in this course

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

$$x \wedge (x \vee y) = x$$

$$x \vee (x \wedge y) = x$$

Logical reasoning

- We want to be able to work out the consequences of assumptions
- Work with unambiguous statements that are either true or false
- Typical problem: show that two Boolean expressions always have the same value
- One approach: use truth tables. But a truth table with n variables contains 2^n lines
- Often it's easier to carry out a calculation using Boolean algebra

Example: equational reasoning

Equational reasoning means “substituting equals for equals” using the laws of algebra

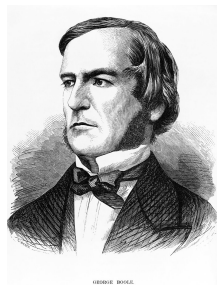
Problem: Use Boolean algebra to simplify $(0 \wedge P) \vee Q$

Each of these steps uses one of the laws:

$$\begin{aligned}
 & (0 \wedge P) \vee Q \\
 &= (P \wedge 0) \vee Q && \wedge \text{ is commutative} \\
 &= 0 \vee Q && \text{constant operation on } \wedge \\
 &= Q \vee 0 && \vee \text{ is commutative} \\
 &= Q && \text{constant operation on } \vee
 \end{aligned}$$

This may look unnecessarily picky. But for large and complicated calculations, it is easier to use the laws carefully: it helps to prevent mistakes.

George Boole, 1815–1864



- English mathematician and logician
- Applied algebra to logic
- Symbolic logic: formal reasoning instead of arguments in natural language

Addition and subtraction

- Addition, subtraction, and negation are all done by one circuit: rippleAdd, the “ripple carry adder”
- We will proceed in stages:
 - ▶ Adding two bits: `halfAdd`
 - ▶ Adding three bits: `fullAdd`
 - ▶ Adding two integers: `rippleAdd`
 - ▶ Subtracting an integer from another
- Multiplication and division are more complicated. Not too complicated, but we won't do them in this course.

Binary addition

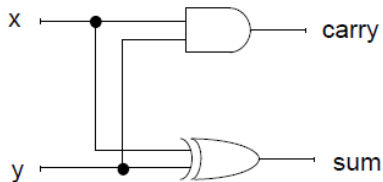
- You can add two binary numbers x and y the same way as adding decimal numbers
- Write one number above the other
- Work through each column, from right to left
- In each column, add the bit from x , the bit from y , and the carry from the column to the right.
- This gives the sum bit s for the column, and the carry output which goes to the left.
- In each column we add three bits: a carry input, a bit from x , and a bit from y

The half adder: adding two bits

- A half adder adds two bits. Since the result could be 0, 1, or 2, we need a two-bit representation of the sum, called (carry, sum).
- Specify the circuit abstractly by writing the complete addition table. Then we recognise that the carry function is just `and2`, and the sum function is just `xor2`.

x	y	result	carry	sum
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

The half adder



x	y	result	carry	sum
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

The full adder: adding three bits

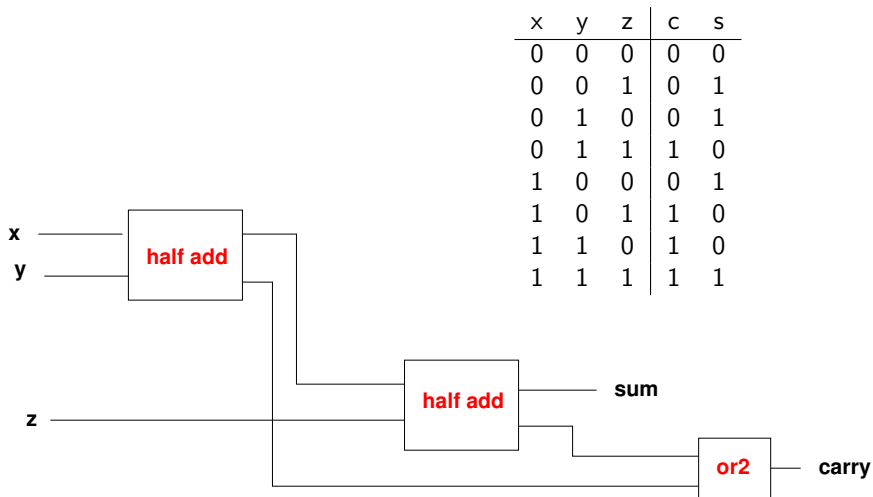
- To add two binary numbers, we must add **three** bits for each column:
 - ▶ The two data bits (one from each word)
 - ▶ The carry input from the column to the left
- Solution: the **full adder**
- A full adder adds three bits x , y , z and outputs a carry c and sum s

Truth table of full adder

x	y	z	c	s
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

- The sum is 1 if an odd number of inputs are 1
- The carry is 1 if two or more inputs are 1
- Think of carry, sum as a 2-bit binary number giving the result

The full adder circuit



Binary word addition

- Use a full adder circuit for each bit position
- To add two 16-bit words x and y , we have 16 separate full adders
- Each full adder receives
 - ▶ a bit from x and a bit from y
 - ▶ the carry output from the full adder to the right

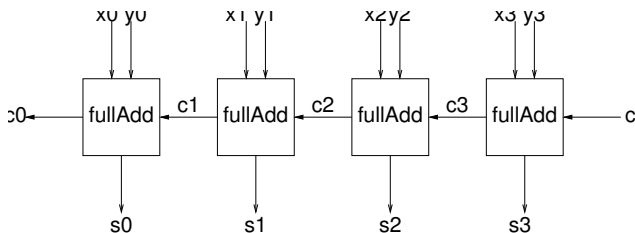
(From Lecture 2) Add the weight 4 column

In the middle of an addition:

		128	64	32	16	8	4	2	1
c						1	0	0	0
x		0	0	1	0	1	1	0	1
y		0	1	0	0	1	1	1	0
s							0	1	1

A full adder will calculate the carry and sum in every column

4-Bit Ripple Carry Adder



- Inputs: word x , word y , carry input bit c
- Outputs: sum s , carry output
- Each bit position receives a bit from x , a bit from y , and a carry input from the position to the right
- A full adder circuit adds these three bits

Pascal's adder



- Carry propagation using gears!
- Designed by the French philosopher and mathematician (and early computer scientist!) Blaise Pascal (1623–1662)

Subtraction

- Work with two's complement numbers
- Note that $x - y = x + (-y)$
- Recall that to negate a number, you invert the bits and add 1
- To invert the bits of y , just put an inverter on each bit of y
- To add 1, just set the carry input to the entire ripple carry adder to 1
- So: use a ripple carry adder, with each bit of y inverted, and with carry input = 1, and the output will be $x - y$
- Modern computers use the adder circuit to perform subtraction as well as addition

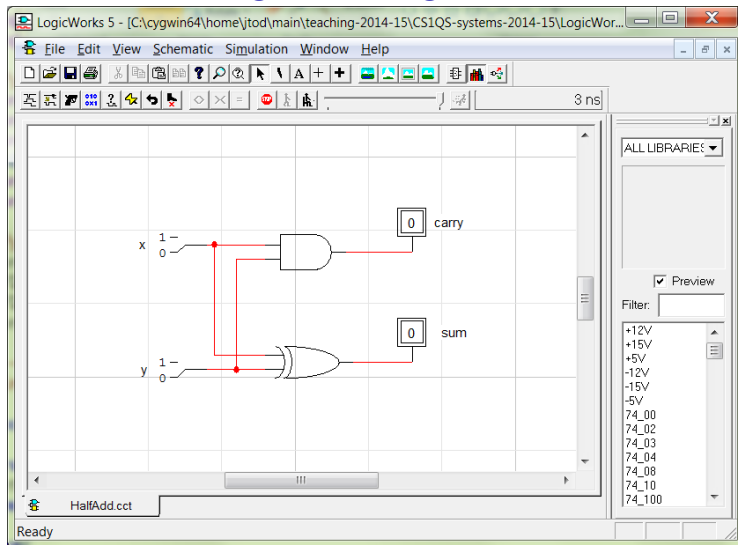
Circuit simulation

- After designing a circuit, **how do we find out whether it works?**
- One way: build the circuit out of transistors, and test it.
- To fabricate a design on a chip *takes a long time* and *costs a lot of money*.
- Better way: use a **circuit simulator**.

Software tools for circuit design

- “Real world” hardware is designed using special languages called **computer hardware description languages**
- An easier approach (though less powerful) is to use a **schematic capture** application.
 - ▶ You draw the circuit interactively.
 - ▶ A diagram of a circuit is called a **schematic diagram**.
 - ▶ The software then simulates it.
- We will use a schematic capture and simulation tool called LogicWorks.

A schematic diagram in LogicWorks



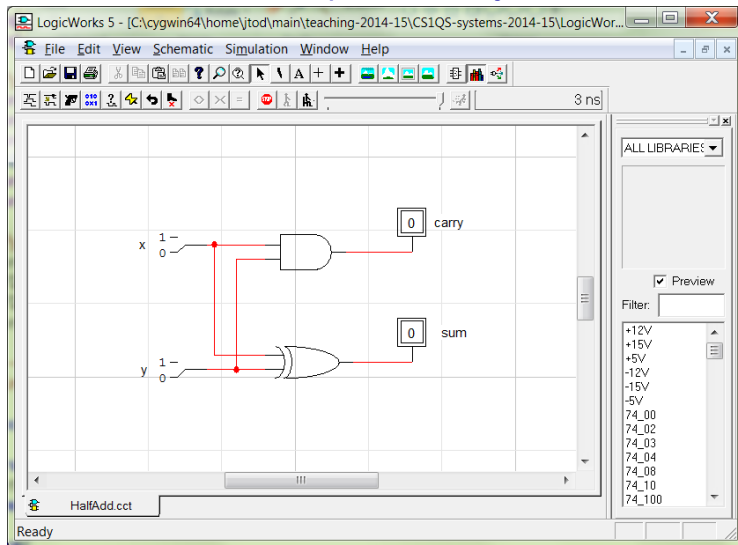
Drawing a circuit

Some basic points:

- Select a component from combo box in the lower right
- You'll see the component in a little window on upper right
- Click it, and drag to where you want to place the component
- To draw a wire, click the + icon on toolbar
- For input, use a binary switch
- For output, use a binary probe

See document on Moodle for more about how to use LogicWorks.

Half adder circuit: inputs $x=0$, $y=0$

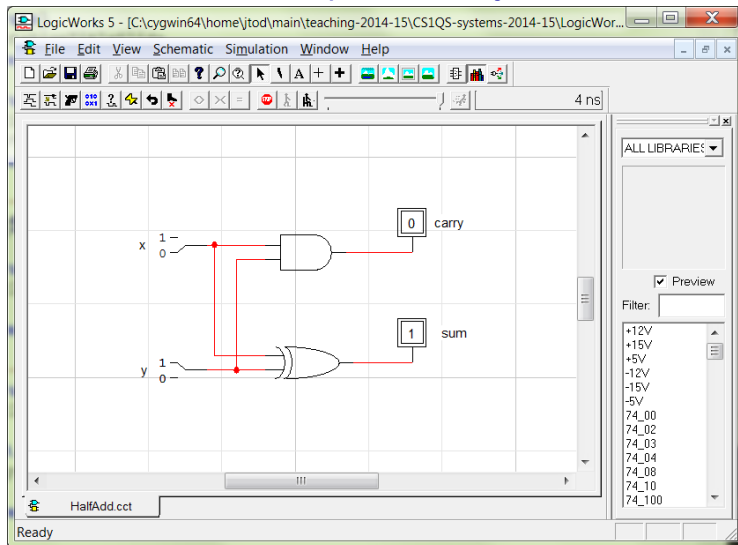


Aim: find out whether the circuit works

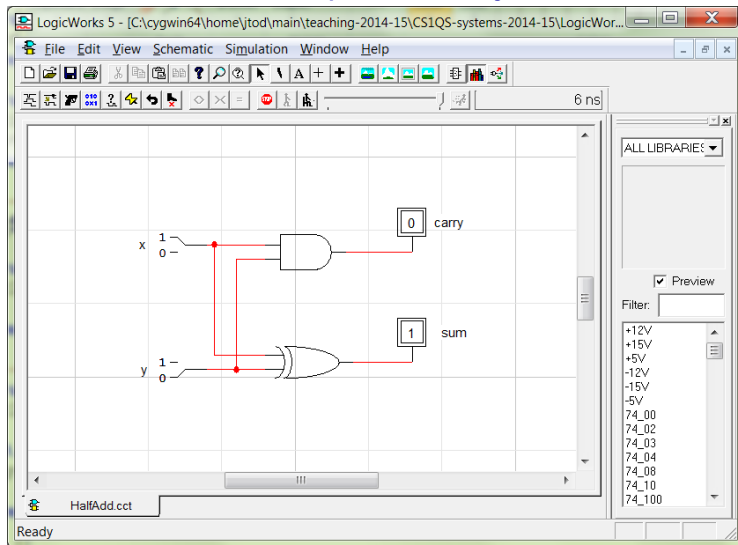
- The circuit has to give the correct outputs for every possible values of the inputs
- So there are four separate simulation problems.
- For each line, set x and y using the switches, and record the outputs c and s in the table.

x	y	c	s
0	0	?	?
0	1	?	?
1	0	?	?
1	1	?	?

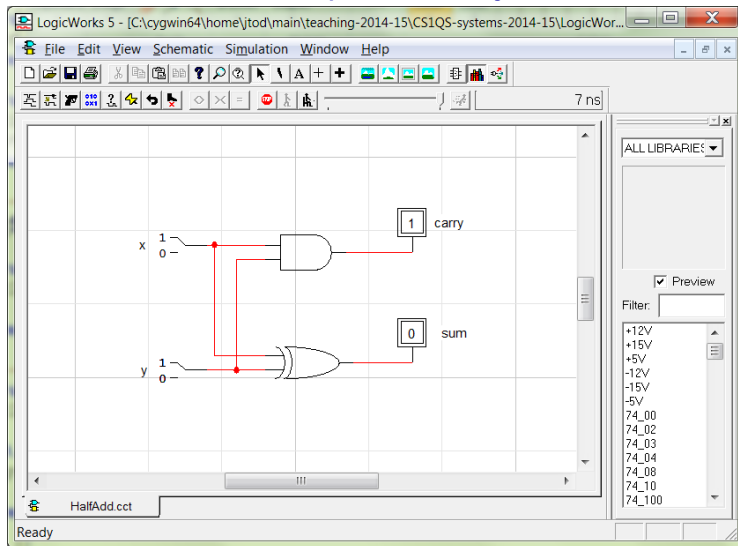
Half adder circuit: inputs $x=0$, $y=1$



Half adder circuit: inputs $x=1$, $y=0$



Half adder circuit: inputs $x=1$, $y=1$



Record simulation results in a truth table

For each row, set the input switches for x , y , observe the outputs c , s and fill in the truth table.

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The table gives the correct carry and sum outputs, so the circuit works correctly.

To do

- Revise the lecture slides
- Finish Quiz 1 by Friday night
- Quiz 2 available
- Check the solutions to Week 2 Lab (on Moodle, at the weekend). Always read the solutions, even if you know the answer: sometimes the model solution will give additional information
- Study the ripple carry adder circuit. Understand how it's doing the same thing you do when you add numbers by hand, with carry where needed



<https://xkcd.com/302/>