Computer Systems

Lecture 3

Logic Gates

Dr José Cano, Dr Lito Michala
School of Computing Science
University of Glasgow

Spring 2020

Outline

- Digital Circuits
- Circuit simulation
- Boolean algebra
- Making decisions: the multiplexer
- Emergent behaviour

Lab and Quiz

- There is a weekly 2-hour lab, starting this week
- Boyd Orr 715
- Go to your scheduled lab and be sure to sit in the right section (identified as red, blue, etc.)
- There is a weekly lab sheet on Moodle, study it in advance!
- Each lab will have some paper and pencil problems, and some problems using the computers
- Please try the problems before your lab

Hardware: Digital Circuits

- We will start with primitive components
- Study their behaviour when they are connected
- Build up a family of useful circuits
- Gradually build larger and more powerful families of circuits, using the ones already defined
- We'll end up with some circuits that can do real computing

Logic gates

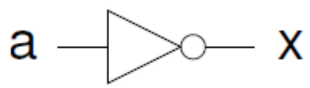
- Digital circuits are built from a very small number of primitive little machines

- Only a few are needed!
  - **Logic gates**, we'll use just four: inv, and2, or2, xor2
  - **Flip flop**, just one is needed: dff

- So we need to
  - Learn what these five components do
  - Learn how to connect them into digital circuits

The Inverter

- A basic component
- Takes an input bit a and produces an output bit x
- The bits are carried on wires
- The technical name for a wire is signal
- The output is the logical opposite of the input

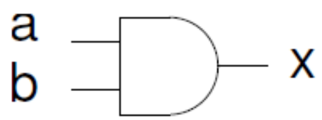


x = inv a

a	x
0	1
1	0

The 2-input and gate

- The output is 1 if all inputs are 1

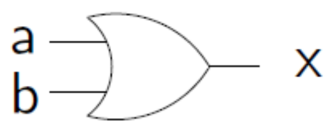


x = and2 a b

a	b	x
0	0	0
0	1	0
1	0	0
1	1	1

The 2-input inclusive or gate

- The output is 1 if any input is 1

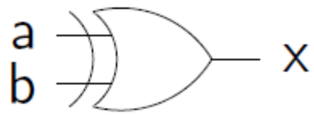


x = or2 a b

a	b	x
0	0	0
0	1	1
1	0	1
1	1	1

The 2-input exclusive or gate

- The output is 1 if either input is 1, but not both



x = xor2 a b

a	b	x
0	0	0
0	1	1
1	0	1
1	1	0

Gate delay

- A logic gate is a primitive component
- It takes a small number of input bits, and produces a result according to a fixed truth table
- As long as the inputs remain stable, the output remains stable
- If an input changes at a point in time, it will take the logic gate a small amount of time to bring the output to the new value: the gate delay
- Gate delays are on the order of 0.01 ns (ns = nanosecond, there are 10^9 ns per second)
- May be faster or slower, depending on the technology

Combinational circuits

- A combinational circuit…
  - Consists of logic gates connected together
  - The circuit has some inputs and produces some outputs
  - It contains no feedback loops
  - The outputs depend on the current input values (i.e. the circuit has no memory)

Outline

- Digital Circuits
- Circuit simulation
- Boolean algebra
- Making decisions: the multiplexer
- Emergent behaviour

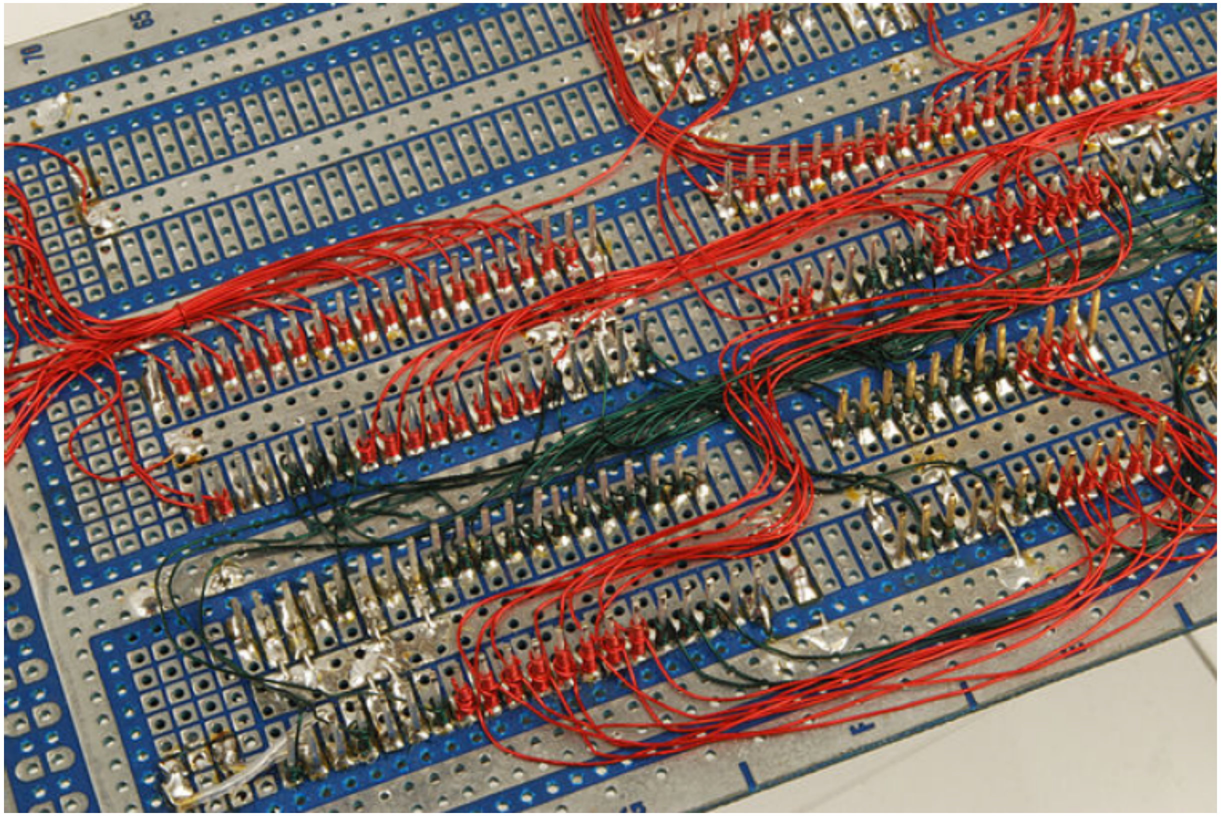
Circuit simulation

- When you learn to program, you learn how to “execute a program by hand”

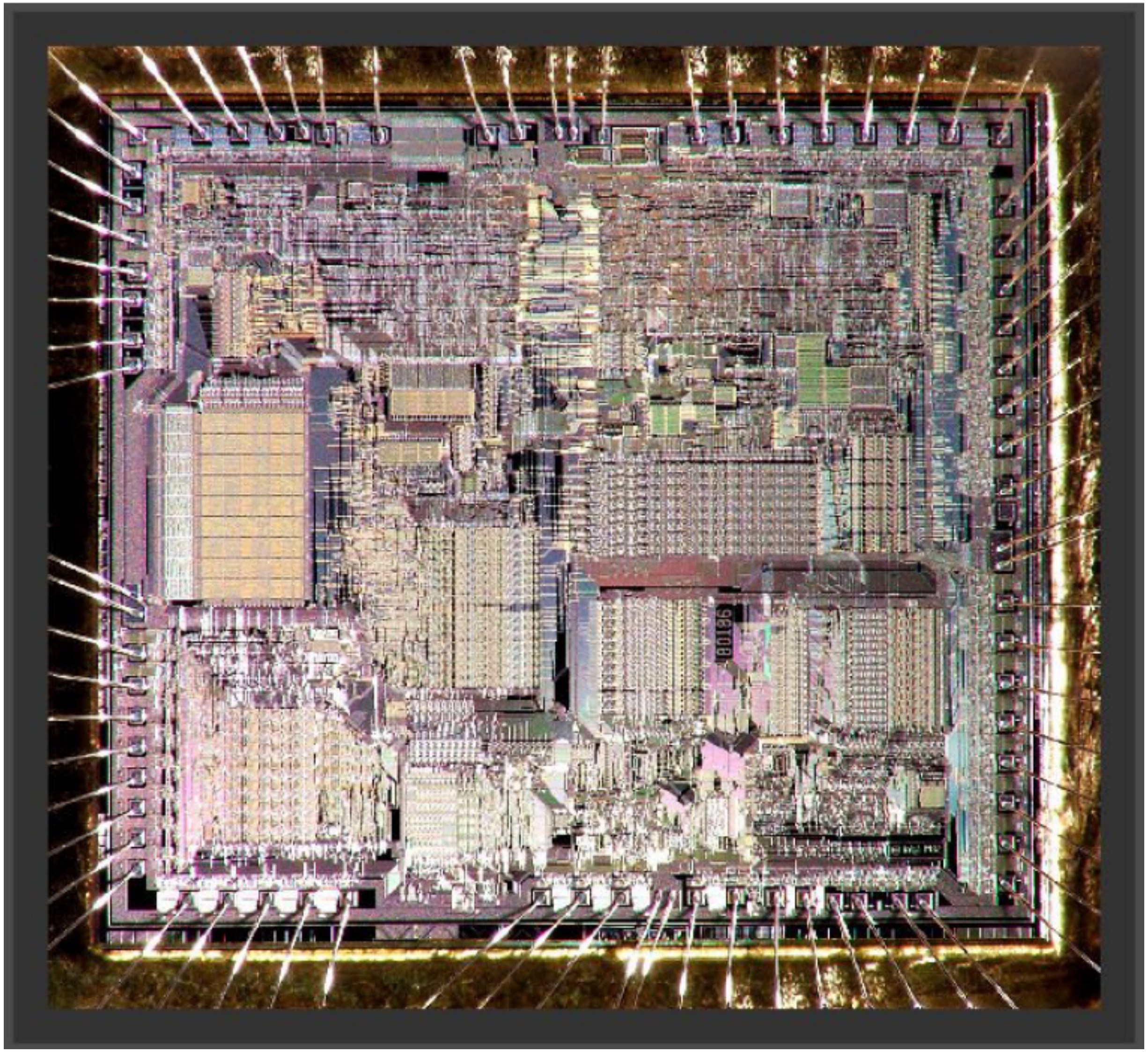
- This is an essential skill
  - It helps to understand how the computer executes the program
  - It enables you to understand bugs

- For a digital circuit, the equivalent skill is circuit simulation
  - You're given the inputs to a circuit
  - You calculate what its outputs should be, by doing the same calculations the hardware would make

Connecting components with wires



An integrated circuit: Intel A80186 CPU

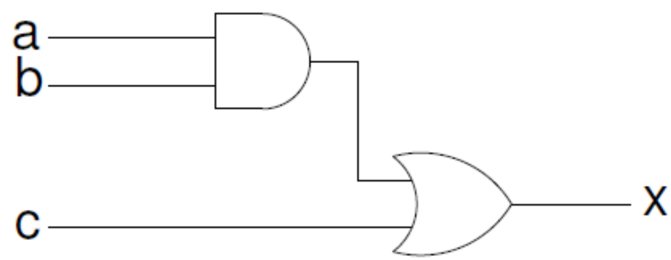


How to simulate a combinational circuit

- You're given the inputs to the circuit
  - The other signal values are unknown
- Find an output of a logic gate where all the inputs are known, and write down the output value
  - Using the truth table for the logic gate
- Repeat until all signals are known

Example of circuit simulation

- Inputs are a, b, c (each is a bit)
- Output is x (a bit)

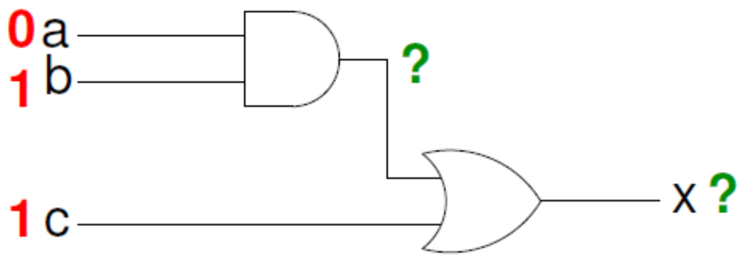


- Given specific values of a, b, c, what is the output x?

Example of circuit simulation

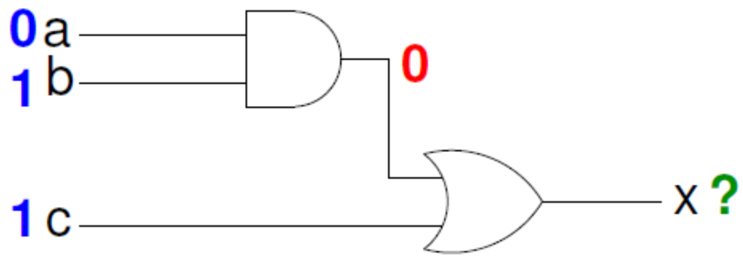
- First, we're given the inputs to the circuit

- Other signals are unknown



After one gate delay

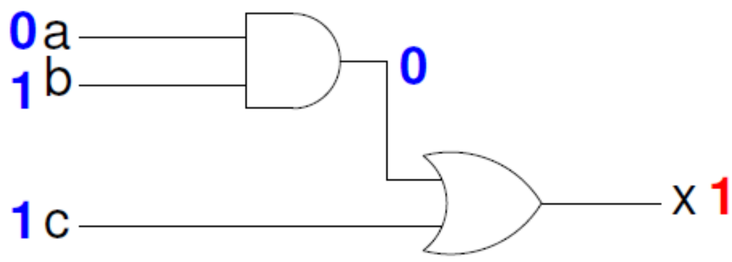
- Look for a component where all the inputs are known, and note the value of its output sigma



- In this example, we can't simulate the or2 gate, because we don't yet know its top input
  - But we can simulate the and2 gate, and its output is 0

After two gate delays

- Look for a component where all the inputs are known, and note the value of its output signal



- Continue the process until all signals are known

Outline

- Digital Circuits
- Circuit simulation
- Boolean algebra
- Making decisions: the multiplexer
- Emergent behaviour

Introduction to Boolean algebra

- Boolean algebra is a good notation for logic gates
  - There are two values: 0, 1
  - We can use variables to stand for a value: x, y, z, …
  - There are three operators
  - Each operator corresponds exactly to a logic gate
  - Boolean algebra describes the values of signals in a digital circuit

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

Laws of Boolean algebra

- Ordinary algebra has laws that you can use to simplify expressions, examples:
  - $x + y = y + x$
  - $x + 0 = x$

- Boolean algebra does too! Here are a few of them:

$$
\begin{aligned}
x \wedge 0 &= 0 \\
x \wedge 1 &= x \\
x \vee 0 &= x \\
x \vee 1 &= 1
\end{aligned}
$$

- These are very important, and we will shortly use them to calculate how a circuit works

Building block circuits

- To do “useful stuff” a circuit usually needs to be fairly big
- But it's hard to understand or to design a really big circuit
- The solution is abstraction
  - We will put several components together to build a black box circuit
  - Then several of those will combine to make a bigger one
  - And then a still-larger one…
- We're going to look at several example circuits that are really important
  - They are critically important in computers

Outline

- Digital Circuits
- Circuit simulation
- Boolean algebra
- Making decisions: the multiplexer
- Emergent behaviour

Multiplexer

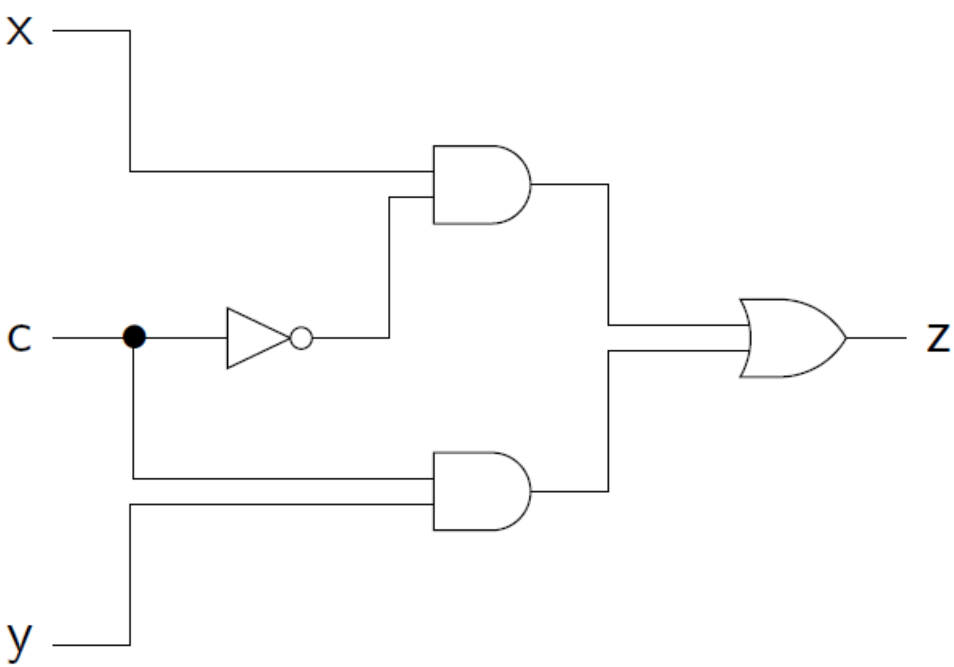
- The multiplexer is the fundamental circuit used to make decisions

- A multiplexer is a hardware version of the if-then-else expression

- The idea is to choose between two values (x, y) based on another value (c)
  - Given three inputs: c, x, y
  - Produces one output
    - If c = 0 the output is x
    - If c = 1 the output is y

- In digital hardware circuits, there are no statements
  - So, there are no “if statements” to determine whether other statements should be executed
  - Instead, we have signals whose values depend on other signals

Multiplexer: z = (if c=0 then x else y)



Simulating the multiplexer

To find out what the circuit does, simulate it!

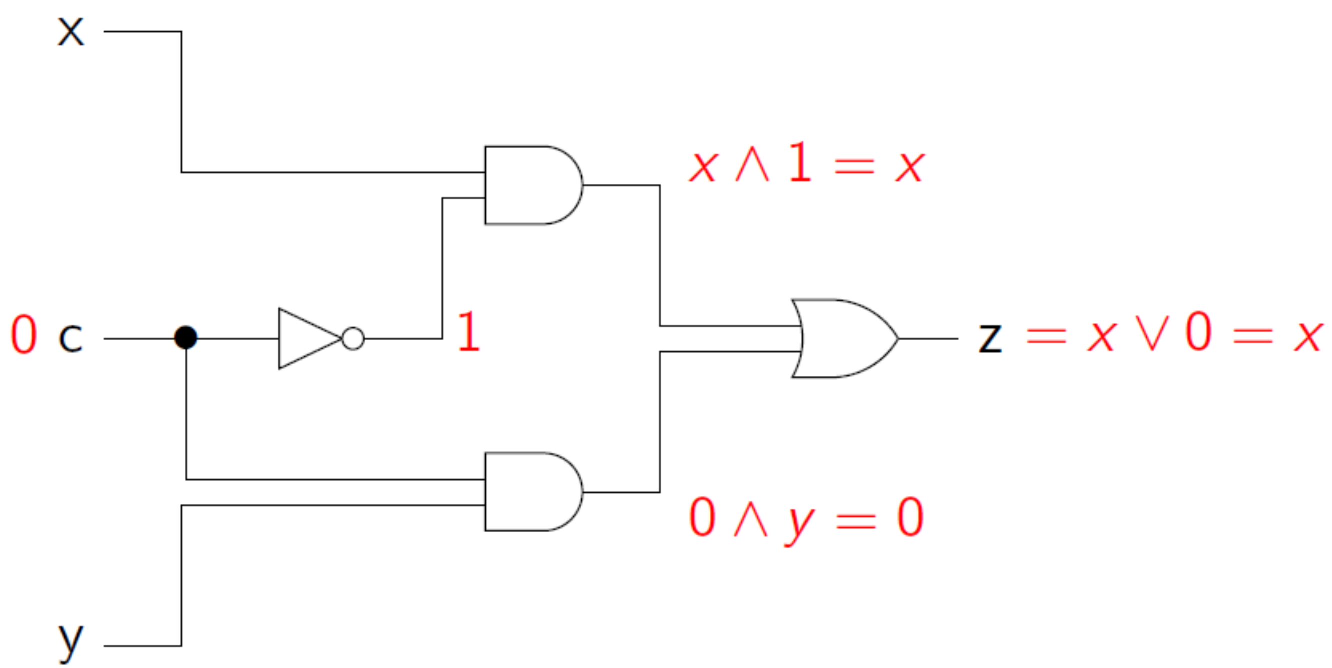
- Given a set of input values, you can calculate the corresponding output

- **Example**: Suppose c = 1; x = 0; y = 1
  - The inverter outputs 0, the first and gate has inputs 0 and 0 so it outputs 0
  - The second and gate has inputs 1 and 1 so it outputs 1
  - The or2 gate has inputs 0 and 1 so it outputs 1, which is the output of the entire circuit

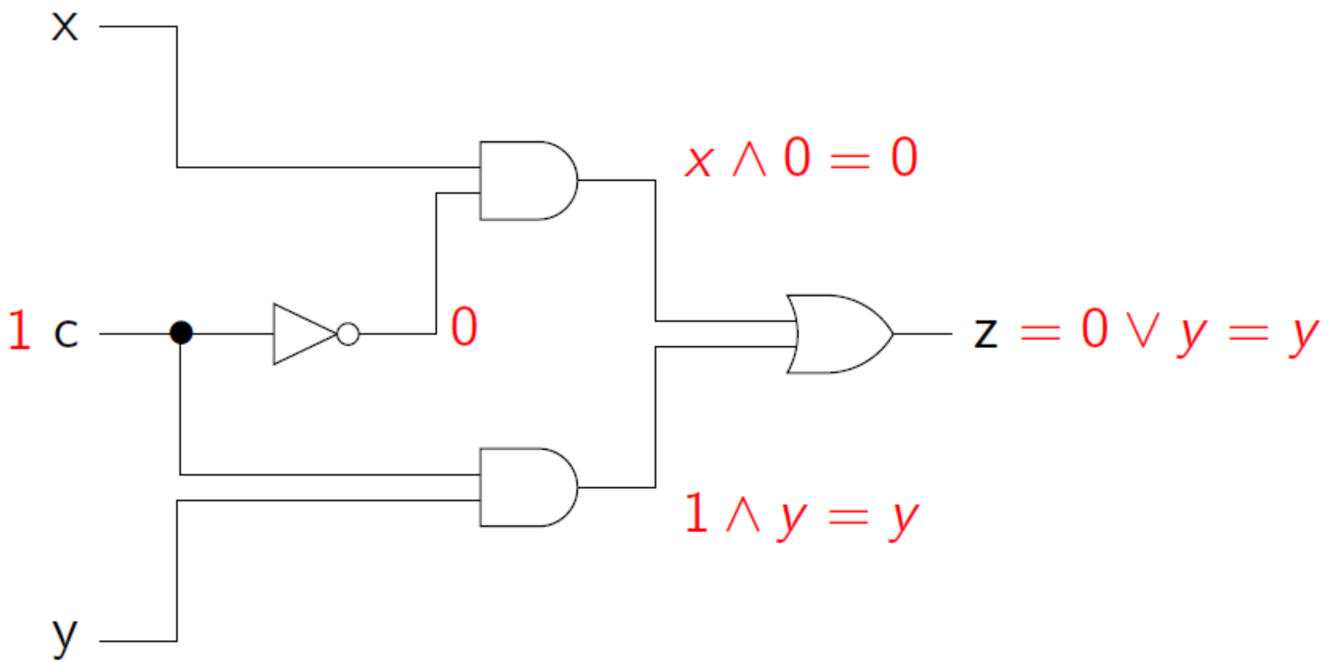
- By doing this for all 8 sets of input values, you can fill in the truth table for the circuit, which fully describes its behaviour

Multiplexer when c=0: z = (if c=0 then x else y) = x

- We can calculate the output z using Boolean algebra



Multiplexer when c=1: z = (if c=0 then x else y) = y



Truth table for mux1

c	x	y	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Meaning of the multiplexer

- mux1 c x y = if c is zero then x else y

c	x	y	out
0	**0**	0	**0**
0	**0**	1	**0**
0	**1**	0	**1**
0	**1**	1	**1**
1	0	**0**	**0**
1	0	**1**	**1**
1	1	**0**	**0**
1	1	**1**	**1**

Multiplexers are important!

- Multiplexers are a central black box which we will use again and again
- One way to understand the circuit is to calculate its truth table by simulating it for each set of input values
- Most conditional operations in digital circuits are implemented, at the lowest level, by a multiplexer
- In a circuit, we don't "execute if-then-else statements", there are no statements!
  - Instead, we define signals using if-then-else expressions, implemented with multiplexers

Outline

- Digital Circuits
- Circuit simulation
- Boolean algebra
- Making decisions: the multiplexer
- Emergent behaviour

Emergent behaviour

- Some systems have these characteristics
  - They comprise parts with relatively simple behaviour
  - The whole system has complicated behaviour
  - This complex behaviour emerges from the interaction of the simple parts
- Examples in science, sociology, politics, philosophy, …

Complex computing from simple logic gates

- Each of our five primitive devices has extremely simple behaviour
- You can understand completely what each of them does
- None of them does anything remotely like computing
  - They can't do arithmetic
  - They can't execute statements
  - They can't make decisions
  - They can't do much at all!
- So how can a computer work?
  - All the interesting things that a computer can do are emergent behaviours
  - Emergent means that a system whose components are all simple can exhibit complex behaviour

Examples of emergent behaviour

- **Chemistry**: The ideal gas law
- **Biology**: Motion of a flock of birds
- **Computing**: Addition of binary numbers

Decisions

- Ultimately, all decisions become a choice of two alternative values on a signal (wire)

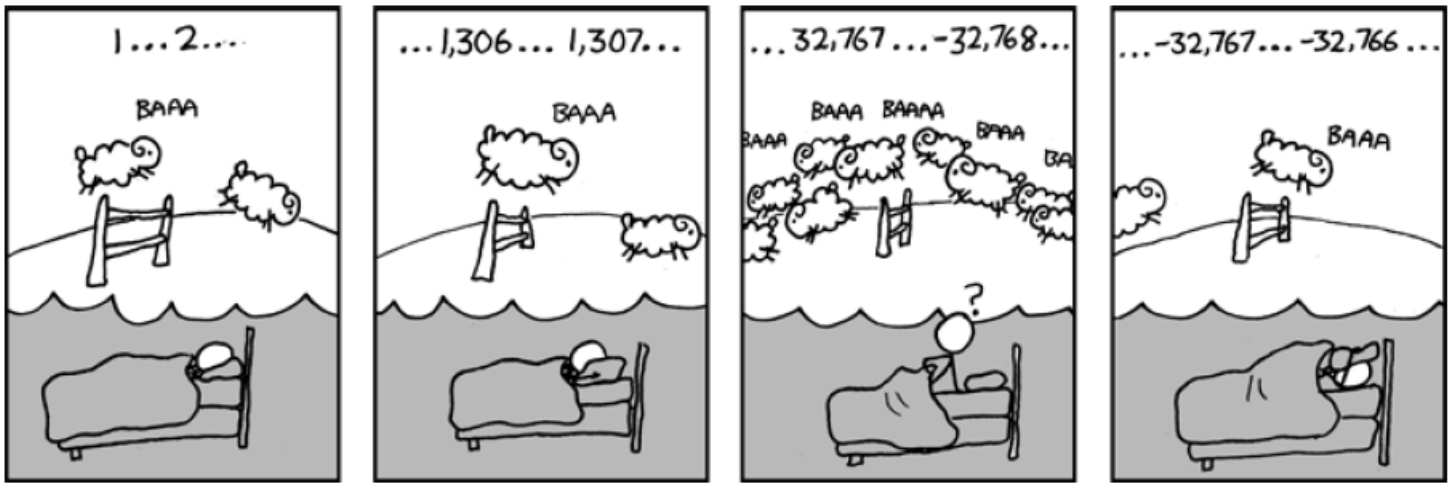
- All of these choices are made by the multiplexer circuit
  - Its behaviour is clearly conditional: the output is x if c = 0, otherwise it's y
  - Yet the implementation of the multiplexer just uses a few logic gates

- This is the fundamental point where conditional behaviour emerges from a lower level behaviour of logic gates!

- Every decision that a computer system makes ultimately comes down to a multiplexer!

To do

- Revise the lecture slides
- Solve the tutorial exercises (they are on Moodle)
- Check MyCampys->My Student Centre->My Calendar to find your Lab Group.
  - Remember that attendance will be taken at the lab
  - It is your responsibility to find your Tutor
- Go to the lab and discuss the exercises
- Ask questions about any of the lecture material



https://xkcd.com/571/