1. Digital Logic Gates

Aims

- to introduce the lowest level processing circuits in digital computers;
- to show that the behaviour of a logic gate is specified by its truth table.

1.1 Introduction

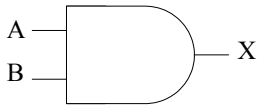
Digital computers store and process *binary* data. Binary data is represented by 1's and 0's. This simplifies the circuits that are used at the lowest level as they only have to have two stable states, one that represents a 1 and one that represents a 0.

The basis of all processing is the logic gate. The logic gate is a circuit that implements a logical function. Each gate is represented by its own symbol and the symbols can be connected together to construct circuit diagrams. The behaviour of a logic gate can be explained in words, with each gate behaving exactly the same given the same set of inputs. Truth tables can be used to reason about the behaviour of gates and circuits.

There are six basic logic gates: AND, OR, NOT, NAND, NOR, XOR.

1.2 Gates

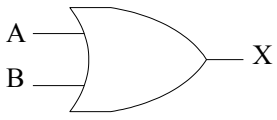
AND Gate



The AND gate outputs a 1 when all the inputs are 1 (X = 1 when both A AND B are 1).

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

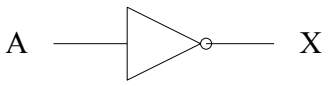
OR Gate



The OR gate outputs a 1 when *either* or *both* of the inputs is a 1 (X = 1 if A OR B = 1, OR both).

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

NOT Gate

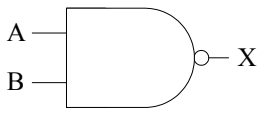


The NOT gate outputs a 1 when the input is 0. When the input is a 1 the NOT gate outputs a 0.

A	X
0	1
1	0

The NOT gate is also referred to as an inverter, as it inverts its inputs.

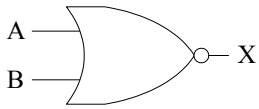
NAND Gate



The NAND gate behaves like an AND gate whose output is then passed through a NOT gate. Thus NOT AND is shortened to give NAND. Therefore the truth table is the opposite of an AND gate.

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

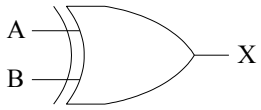
NOR Gate



The NOR gate behaves like an OR gate whose output is then passed through a NOT gate. Thus NOT OR is shortened to give NOR. Therefore the truth table is the opposite of an OR gate.

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

XOR Gate



The XOR gate (exclusive OR) outputs a 1 when *only one* of its inputs is a 1 (X = 1 if A OR B = 1, but not both).

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

The behaviour of a logic gate is independent of how it is actually constructed. This allows circuit designers to implement gates how they wish.

2. Boolean Notation for Logic Circuits

2.1 Introduction

We have already seen two distinct ways of representing a digital logic circuit. We can draw a circuit diagram, and we can construct a truth table. Both of these representations have their uses: if you are intending to actually build the circuit, the diagram would be your starting point. Alternatively, if you want to see exactly what the circuit would do in all possible situations, you would use a truth table.

However, as an approach to communicating about circuits and their functionality, both representations have their problems. Drawing a circuit diagram is complex and the diagrams can get quite unwieldy, once you go beyond very simple functionality. Furthermore, as you saw last week, it can take a certain amount of effort to see exactly what the circuit does. On the other hand, although the truth table tells you absolutely everything about the *behaviour* of a circuit, it doesn't tell you too much about how you might *construct* it using logic gates.

It seems that we need another representation for our circuit diagrams, one which:

- Is concise, allowing us to communicate it quickly and with less chance of introducing errors.
- Gives some indication of the behaviour of the circuit, if we read it properly.
- Gives us an indication of how the circuit could be constructed.

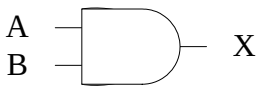
We will subsequently see that the notation we introduce in this class has other uses as well as the three that are listed above.

2.2 Boolean Notation

Boolean notation is founded on the fact that each of the basic gates has a symbol. Other symbols are built up from these.

2.2.1 AND Gate

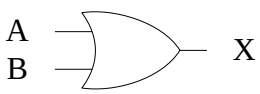
The symbol for an AND gate is a dot: $A.B$



Sometimes we omit the dot, so the symbol then becomes simply AB. Strictly speaking, the diagram above would be represented $X = A.B$ or $X = AB$.

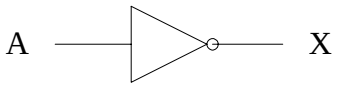
2.2.2 OR Gate

The symbol for an OR gate is a + sign: $X = A+B$



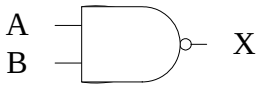
2.2.3 NOT Gate

The symbol for a NOT gate is a line over the input term: $X = \overline{A}$



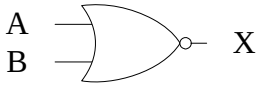
2.2.4 NAND Gate

The symbol for a NAND gate is a combination of the symbols for the AND gate and the NOT gate: $X = \overline{A.B}$ or $X = \overline{AB}$



2.2.5 NOR Gate

The symbol for a NOR gate is a combination of the symbols for the OR gate and the NOT gate: $X = \overline{A + B}$



2.2.6 XOR Gate

The symbol for an XOR gate is a + sign in a circle: $X = A \oplus B$