# Digital Electronics and Microprocessor Systems (ELEC211)

**Dave McIntosh** and Valerio Selis

dmc@liv.ac.uk

v.selis@liv.ac.uk

Digital 1: Basic gates and function implementation

UNIVERSITY OF LIVERPOOL

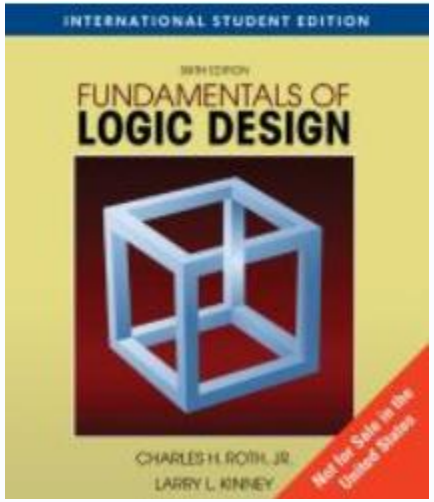
# Outline

Dave McIntosh
Room 508, EEE building
Office hour: Wednesday at 2 pm (email first)

- Digital electronics in context
- Logic gates (revision)
- Combinational logic (revision)
- Karnaugh maps (revision)
- Minterms and maxterms (revision)
- Implementing functions (revision)

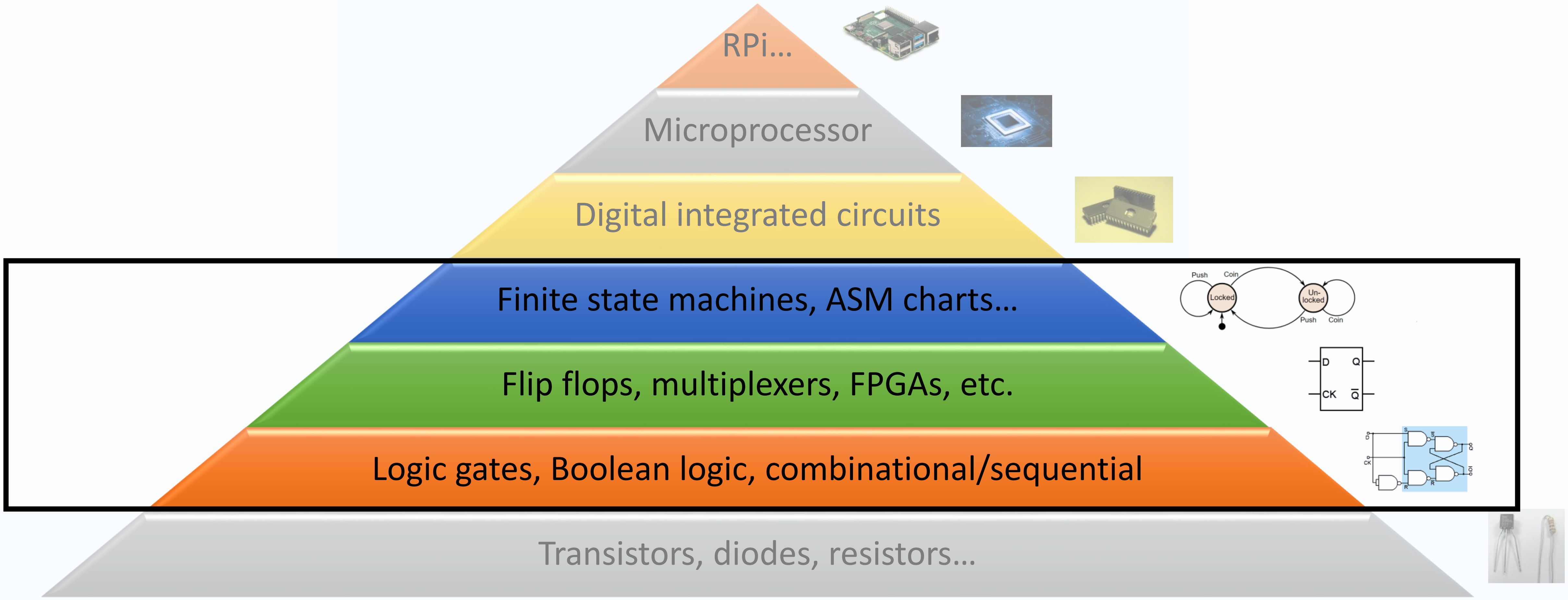
https://www.youtube.com/watch?v=Fxv3JoS1uY8

Course textbook – please borrow and use it!




UNIVERSITY OF
LIVERPOOL

# Digital electronics in context

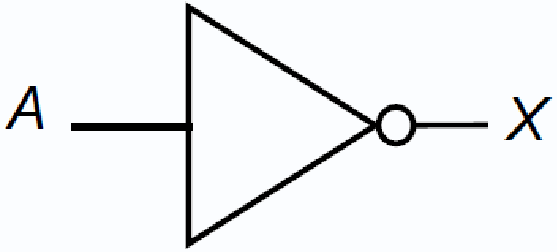


# Logic gates: basic operations (revision)

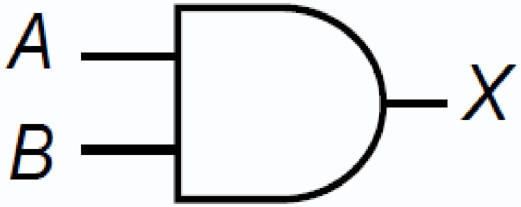
Logic gates use Boolean algebra

Basic operations are:

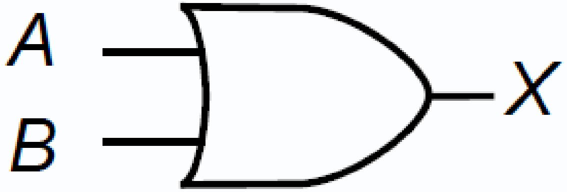
NOT




AND

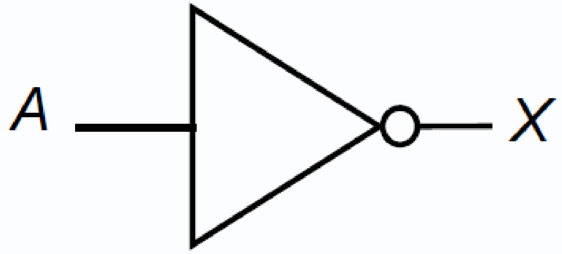



OR



# The NOT gate or inverter

The circuit symbol is




The **truth table** is

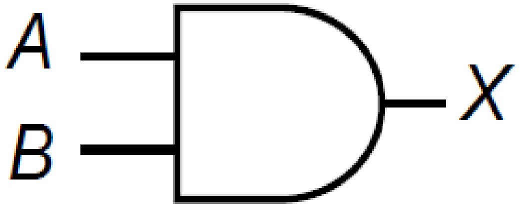
| $A$ | $\boldsymbol{X}$ |
|---|---|
| 0 | **1** |
| 1 | **0** |

The notation is

$$\boldsymbol{X = \overline{A}}$$

$$\boldsymbol{X = A'}$$

## AND

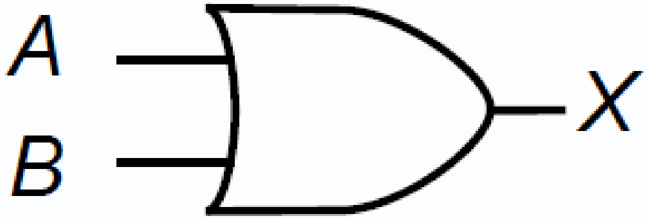



$$X = A \cdot B$$

$$X = A \wedge B$$

| $A$ | $B$ | $\mathbf{X}$ |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **0** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

## OR

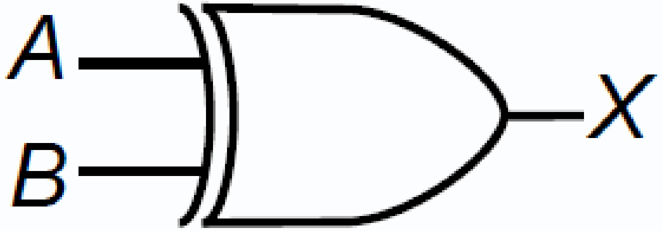



$$X = A + B$$

$$X = A \vee B$$

| $A$ | $B$ | $\mathbf{X}$ |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **1** |

## XOR



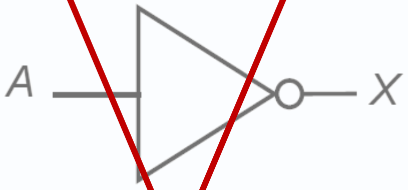

***Exclusive OR***

$$X = A \oplus B$$

| $A$ | $B$ | $\mathbf{X}$ |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |

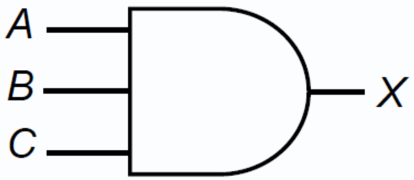
# 3+ inputs?

## NOT?




| A | X |
|---|---|
| 0 | **1** |
| 1 | **0** |

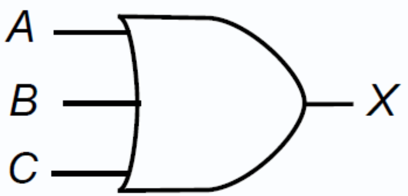
## AND




| $A$ | $B$ | $C$ | $\boldsymbol{X}$ |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **0** |
| 0 | 1 | 0 | **0** |
| 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **0** |
| 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **0** |
| 1 | 1 | 1 | **1** |

$$X = A \cdot B \cdot C$$

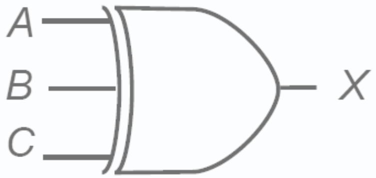
## OR




| $A$ | $B$ | $C$ | $\boldsymbol{X}$ |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **1** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **1** |

$$X = A + B + C$$

## XOR ...?




| $A$ | $B$ | $C$ | $\boldsymbol{X}$ |
|---|---|---|---|
| 0 | 0 | 0 | **0** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **0** |
| 1 | 1 | 1 | **0** |

Rarely seen

This interpretation: "one and only one"

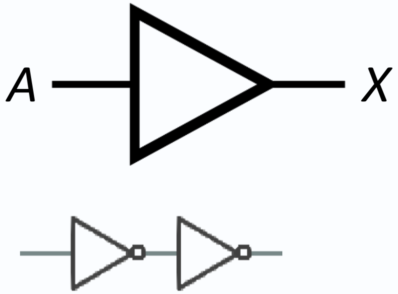
Alternative: "odd parity"

$$X = A \oplus B \oplus C$$

UNIVERSITY OF LIVERPOOL

# Inverted versions of these gates…

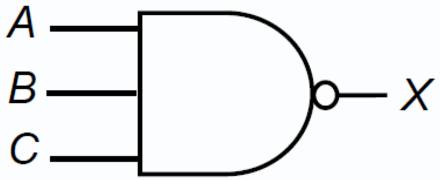
## "Buffer"




| A | X |
|---|---|
| 0 | **0** |
| 1 | **1** |

$$X = A$$

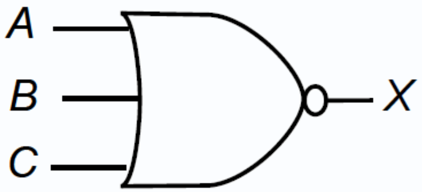
## NAND




| *A* | *B* | *C* | ***X*** |
|---|---|---|---|
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **1** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **0** |

$$X = \overline{A \cdot B \cdot C}$$

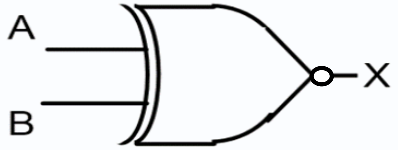
## NOR




| *A* | *B* | *C* | ***X*** |
|---|---|---|---|
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | **0** |
| 0 | 1 | 0 | **0** |
| 0 | 1 | 1 | **0** |
| 1 | 0 | 0 | **0** |
| 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **0** |
| 1 | 1 | 1 | **0** |

$$X = \overline{A + B + C}$$

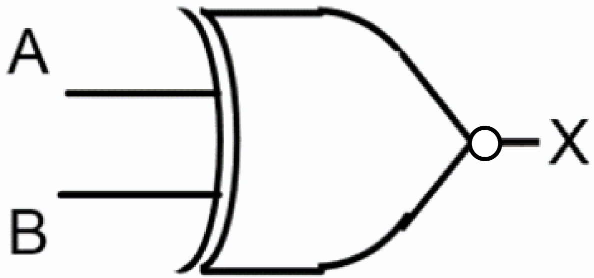
## XNOR *(2 input)*



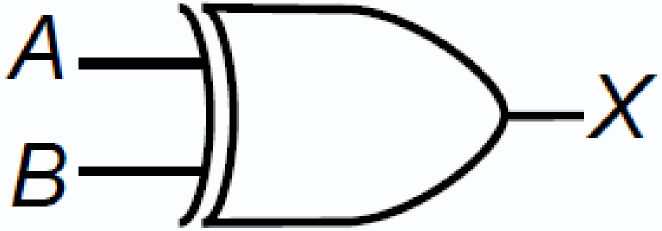

| *A* | *B* | ***X*** |
|---|---|---|
| 0 | 0 | **1** |
| 0 | 1 | **0** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

$$X = \overline{A \oplus B}$$

XNOR




XOR




***Exclusive NOR***

$$X = \overline{A \oplus B}$$

| $A$ | $B$ | $\boldsymbol{X}$ |
|---|---|---|
| 0 | 0 | **1** |
| 0 | 1 | **0** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

***Exclusive OR***

$$X = A \oplus B$$

| $A$ | $B$ | $\boldsymbol{X}$ |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |

# Notations

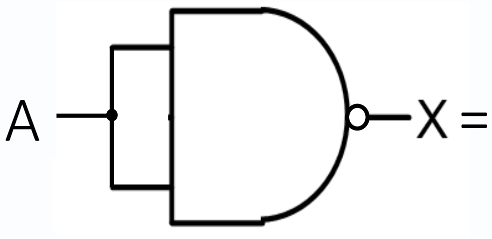
| | | |
|---|---|---|
| AND | $A.B$ | $(A \wedge B)$ |
| OR | $A + B$ | $(A \vee B)$ |
| Inversion ("complementation") | $\overline{X}$ | $(X')$ |

# Universal gates

NAND and NOR are ‘universal gates’

Each one can implement any Boolean function without using other gates…

e.g. NAND as NOT




$X = (A.A)' = A'$

- NAND and NOR gates are cheaper to produce than other gates
- NAND gates are routinely used to build the other gates in circuits
- Particularly if buying in bulk, many NAND gates may be bought instead of purchasing all types

*Which logic gate is "(A NAND B) AND (A OR B)" equivalent to?*

$$X = (\overline{A \cdot B}) \cdot (A + B)$$

***Groups of 2 or 3***

***1. Draw circuit diagram***

***2. Use the individual truth tables to construct a combined truth table***

UNIVERSITY OF LIVERPOOL

***Extension task: Can you make the same equivalent gate using four NAND gates?***

# Combinational and sequential circuits

- **Combinational circuits**:
  - Output depends only on **present input** (easier)
- **Sequential circuits:**
  - Output values depend on **present *and* past** values
  - Circuit ‘remembers’ its own **state**
- **Combinational logic** uses **Boolean algebra.**

# Laws of Boolean Algebra

| | | |
|---|---|---|
| Operations with 0 | $X+0=X$ | $X \cdot 0=0$ |
| Operations with 1 | $X+1=1$ | $X \cdot 1=X$ |
| Idempotent laws | $X+X=X$ | $X \cdot X=X$ |
| Involution law | $(X')'=X$ | |
| Laws of complementarity | $X+X'=1$ | $X \cdot X'=0$ |
| Commutative laws | $X+Y=Y+X$ | $X \cdot Y=Y \cdot X$ |
| Associative laws | $(X+Y)+Z=X+(Y+Z)=X+Y+Z$ | $(XY)Z=X(YZ)=XYZ$ |
| Distributive laws | $X(Y+Z)=XY+XZ$ | $X+YZ=(X+Y)(X+Z)$ |

# Laws / theorems of Boolean Algebra

| | | |
|---|---|---|
| Simplification theorem (1) | $X \cdot Y + X \cdot Y' = X$ | $(X+Y) \cdot (X+Y') = X$ |
| Simplification theorem (2) | $X + X \cdot Y = X$ | $X \cdot (X+Y) = X$ |
| Simplification theorem (3) | $(X+Y') \cdot Y = X \cdot Y$ | $X \cdot Y' + Y = X + Y$ |
| DeMorgan's laws | $(X+Y+Z+\ldots)' = X' \cdot Y' \cdot Z' \ldots$ | $(X \cdot Y \cdot Z \ldots)' = X' + Y' + Z' \ldots$ |
| Multiplying and factoring | $(X+Y) \cdot (X'+Z) = X \cdot Z + X' \cdot Y$ | $X \cdot Y + X' \cdot Z = (X+Z) \cdot (X'+Y)$ |
| Consensus theorem | $X \cdot Y + Y \cdot Z + X' \cdot Z = X \cdot Y + X' \cdot Z$ | $(X+Y) \cdot (Y+Z) \cdot (X'+Z)$<br>$= (X+Y) \cdot (X'+Z)$ |

**BUT KARNAUGH MAPS ARE USUALLY MORE EFFICIENT THAN SIMPLIFYING ALGEBRAICALLY!**

UNIVERSITY OF LIVERPOOL

(To be continued....)

# Minterms

| $A$ | $B$ | $C$ | Minterms | Maxterms |
|---|---|---|---|---|
| 0 | 0 | 0 | $A'B'C'=m_0$ | $A+B+C=M_0$ |
| 0 | 0 | 1 | $A'B'C=m_1$ | $A+B+C'=M_1$ |
| 0 | 1 | 0 | $A'BC'=m_2$ | $A+B'+C=M_2$ |
| 0 | 1 | 1 | $A'BC=m_3$ | $A+B'+C'=M_3$ |
| 1 | 0 | 0 | $AB'C'=m_4$ | $A'+B+C=M_4$ |
| 1 | 0 | 1 | $AB'C=m_5$ | $A'+B+C'=M_5$ |
| 1 | 1 | 0 | $ABC'=m_6$ | $A'+B'+C=M_6$ |
| 1 | 1 | 1 | $ABC=m_7$ | $A'+B'+C'=M_7$ |

**Minterm: a product term which outputs a logic high for one combination of A, B and C and their inverses.**

**Example:**

For the combination ***A*=1**, ***B*=0, *C*=1**, the only product term worth 1 is $A.B'C$.

*[Let $A = 1$, $B = 0$, $C = 1$, then:*
*$A.B'C = 1.1.1 = 1$]*

The binary number 101 is 5 as a decimal, so we call this minterm 5, $m_5$

# Maxterms

| $A$ | $B$ | $C$ | Minterms | Maxterms |
|---|---|---|---|---|
| 0 | 0 | 0 | $A'B'C'=m_0$ | $A+B+C=M_0$ |
| 0 | 0 | 1 | $A'B'C=m_1$ | $A+B+C'=M_1$ |
| 0 | 1 | 0 | $A'BC'=m_2$ | $A+B'+C=M_2$ |
| 0 | 1 | 1 | $A'BC=m_3$ | $A+B'+C'=M_3$ |
| 1 | 0 | 0 | $AB'C'=m_4$ | $A'+B+C=M_4$ |
| 1 | 0 | 1 | $AB'C=m_5$ | $A'+B+C'=M_5$ |
| 1 | 1 | 0 | $ABC'=m_6$ | $A'+B'+C=M_6$ |
| 1 | 1 | 1 | $ABC=m_7$ | $A'+B'+C'=M_7$ |

**Maxterm: a <u>sum</u> term which outputs a logic LOW for one combination of A, B and C and their inverses.**

**<u>Example:</u>**

For the combination ***A*=1**, ***B*=0, *C*=1**, the only sum term worth 0 is $A' + B + C'$.

*[Let $A = 1$, $B = 0$, $C = 1$, then:*
*$A' + B + C' = 0 + 0 + 0 = 0$]*

The binary number 101 is 5 as a decimal, so we call this maxterm 5, $M_5$

# Implement a function using minterms & maxterms

Consider the function, *f*, represented by this truth table.

We only want *f* to take a logic 'high' value for the combinations of *A, B, C* in the final five rows *(ABC* = 011, 100, 101, 110, 111*)*.

How can we implement this using logic gates?

| *A* | *B* | *C* | *f* |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# Minterms and the Sum of Products

We can derive an expression for the output, *f*, by ORing together the product terms that make *f* = 1 for the desired combinations:

$f=(A'BC)+(AB'C')+(AB'C)+(ABC')+(ABC)$

| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**This is called a SUM OF PRODUCTS (SOP)**

# Maxterms and the Product of Sums

We can also write *f* by ANDing together the product terms that make *f* = 0 for the desired combinations:

$f = (A+B+C)(A+B+C')(A+B'+C)$

| $A$ | $B$ | $C$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**This is called a PRODUCT OF SUMS (POS)**

UNIVERSITY OF LIVERPOOL

# Minimum SOP

Before implementing using logic gates, it makes sense to minimise (simplify) the sum of products expression using Boolean laws:

$f=(A'BC)+(AB'C')+(AB'C)+(ABC')+(ABC)$

$= A'BC+A\ (B' \ldots '+B'C+BC'+BC \ldots$

$= A'\ BC+A \ldots )+B(C \ldots BC+A = BC+A.$

The above e … simplified using:

- 1st: Distrib … d out)
  - 2nd … tored out)
- 3rd: … $C'+C=1$)
- … n ($B$ … appears)
- 5th: … n theorem … $X+Y$)

| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

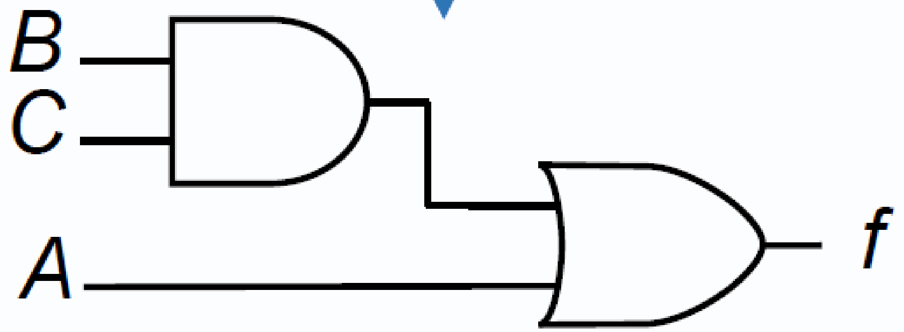
**What is a simpler way to minimise this?**

# Minimum SOP and implementation

| $A$ | $B$ | $C$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f=(A'\cdot B\cdot C)+(A\cdot B'\cdot C')+(A\cdot B'\cdot C)+(A\cdot B\cdot C')+(A\cdot B\cdot C)$$

| $f$ | $A'B'$ | $A'B$ | $AB$ | $AB'$ |
|---|---|---|---|---|
| $C'$ | | | **1** | **1** |
| $C$ | | **1** | **1** | **1** |



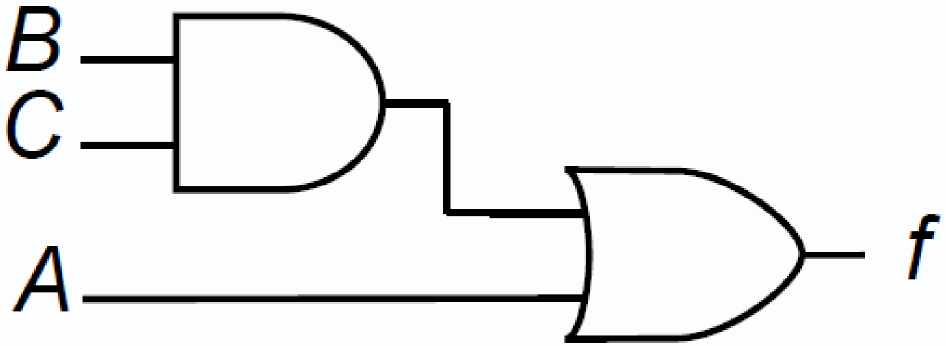
# Maxterms – minimum POS

- As noted, we can also write *f* by ANDing together the combinations of values that make it 0 (maxterms):

$$f = (A+B+C)(A+B+C')(A+B'+C)$$

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | **0** | **0** | 1 | 1 |
| 1 | **0** | 1 | 1 | 1 |

$$f = BC+A$$




| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# Combinational logic: minterms

Combinational logic expressions can be expressed as a sum of minterms ('canonical sum of products') as follows:

$$f(C,B,A) = m_2 + m_5 + m_7$$

$$f(C,B,A) = \sum m(2,5,7)$$

$m_2$, $m_5$ and $m_7$ are minterms; we can substitute them in:

$$m_2 = C'BA' \quad m_5 = CB'A \quad m_7 = CBA$$

$$f = C'BA' + CB'A + CBA = C'BA' + CA$$

# Minterms & Maxterms

| $A$ | $B$ | $C$ | Minterms | Maxterms |
|---|---|---|---|---|
| 0 | 0 | 0 | $A'B'C'=m_0$ | $A+B+C=M_0$ |
| 0 | 0 | 1 | $A'B'C=m_1$ | $A+B+C'=M_1$ |
| 0 | 1 | 0 | $A'BC'=m_2$ | $A+B'+C=M_2$ |
| 0 | 1 | 1 | $A'BC=m_3$ | $A+B'+C'=M_3$ |
| 1 | 0 | 0 | $AB'C'=m_4$ | $A'+B+C=M_4$ |
| 1 | 0 | 1 | $AB'C=m_5$ | $A'+B+C'=M_5$ |
| 1 | 1 | 0 | $ABC'=m_6$ | $A'+B'+C=M_6$ |
| 1 | 1 | 1 | $ABC=m_7$ | $A'+B'+C'=M_7$ |

**Sum of Products (SOP)**

$$m_2 + m_5 + m_7 = \Sigma m(2,5,7)$$

$$A'BC' + AB'C + ABC = A'BC' + AC$$

**Product of Sums (POS)**

$$M_2 \cdot M_5 \cdot M_7 = \Pi M(2,5,7)$$

$$(A + B' + C)(A' + B + C')(A' + B' + C')$$

UNIVERSITY OF LIVERPOOL

# Question

- Substitute in the minterms to give the following 'canonical sum of products' as a sum of product terms, and simplify (if possible).

$$f(C,B,A) = \sum m(0,3,6)$$

# Question

- By substituting in the minterms and simplifying, rewrite the following 'canonical sum of products' expression as a minimum sum of products (minimum SOP).

$$f(D,C,B,A) = \sum m(1,4,8,9,14)$$

# Summary and suggested reading

- Context
- Logic gates (Section 2.2)
- Minterms & maxterms (Section 4.3)
- Boolean algebra (Sections 2.4 – 2.8)
- Karnaugh maps (Sections 5.2 – 5.3)

Next lecture is this **Wednesday at 12.00 in the Chadwick Barkla** lecture theatre: Multiplexers, decoders, encoders…

**Roth and Kinney** *Fundamentals of Logic Design*