Boolean Logic & Logic Gates III

Lecture 6

Lecture 6

Boolean Logic & Logic Gates III

KARNAUGH MAPPING

Karnaugh Maps

- Shortform : **K-map**
- Another **way of minimising** Boolean equations
  - Instead of simplification using Boolean algebra,
  - **Visual way** of simplification
    - **Recognising pattern** in the equation

- Came up by Edward Veitch in 1952
  - Veitch Diagram
  - Redefined by Maurice Karnaugh in 1953

Karnaugh Mapping

- Able to simplify a boolean equation once it's in its SOP form

- Example: X = A'B'C + A'B'C' + A'BC'

Only 1 input changes between rows & columns

	C'	C
A'B'	1	1
A'B	1	
AB		
AB'		

	C'	C
A'B'	1	1
A'B	1	
AB		
AB'		

- Final Equation : X = **A'B'** + **A'C'**

Karnaugh Mapping

- Let's try this now
  - $X = A'B'C' + A'B + ABC' + AC$

Karnaugh Mapping

- Let's try this now
  - $X = A'B'C' + A'B + ABC' + AC$

	C'	C
A'B'	1	
A'B	1	1
AB	1	1
AB'		1

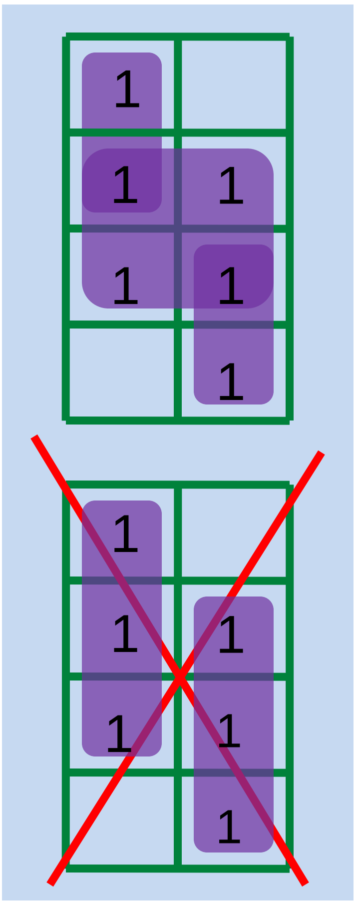
	C'	C
A'B'	1	
A'B	1	1
AB	1	1
AB'		1

	C'	C
A'B'	1	
A'B	1	1
AB	1	1
AB'		1

  - Final Equation : $X = A'C' + B + AC$

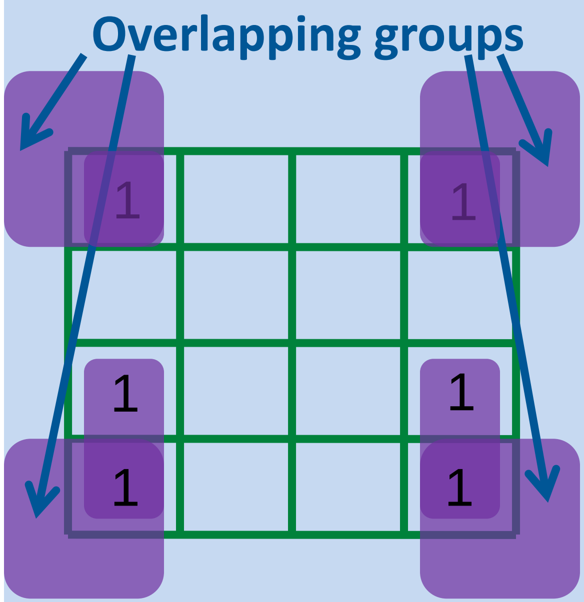
Karnaugh Mapping

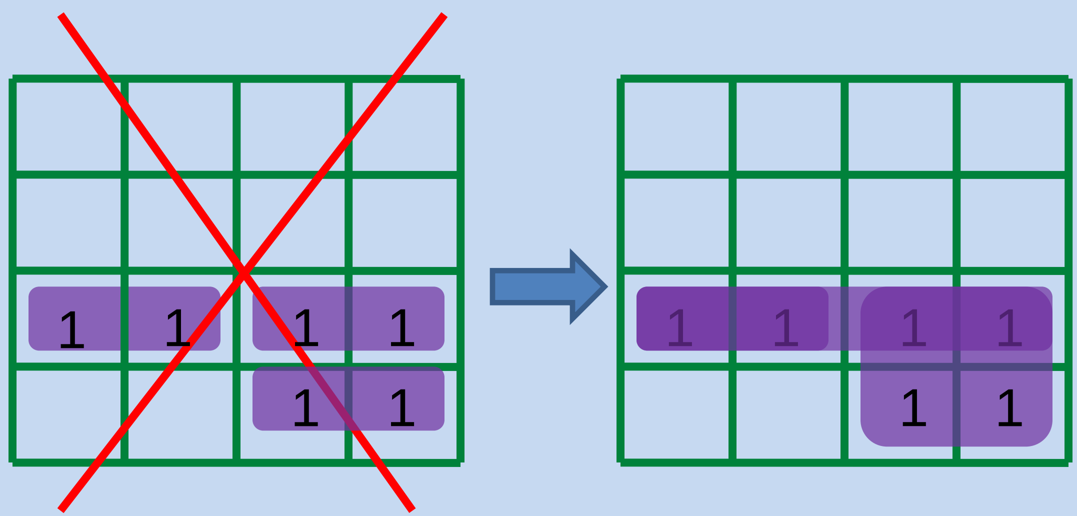
- Aim is to group together adjacent cells containing as many ones as possible according to the following rules
  1. Groups **may not contain** any cell containing a zero
  2. Groups may be **horizontal or vertical** but **not diagonal**
  3. Groups **must contain** either **1,2,4** or **8** cells
  4. Groups **may overlap**



Karnaugh Mapping

5. Each cell **must contain a one** must be in at least one group
6. Groups (at the edges) **may wrap around** the table
7. There should be **as few groups as possible**





- Should be as few groups as possible.

Karnaugh Mapping

- Now let’s try another example. Write it in C++ code
  - $X = A'B'C' + AC'D' + AB' + ABCD' + A'B'C$

Karnaugh Mapping

- Now let’s try another example. Write it in C++ code
  – $X = A'B'C' + AC'D' + AB' + ABCD' + A'B'C$

	C’D’	C’D	CD	CD’
A’B’	1	1	1	1
A’B				
AB	1			1
AB’	1	1	1	1

Karnaugh Mapping

- Now let’s try another example. Write it in C++ code
  - X = A’B’C’ + AC’D’ + AB’ + ABCD’ + A’B’C

	C’D’	C’D	CD	CD’
A’B’	1	1	1	1
A’B				
AB	1			1
AB’	1	1	1	1

Karnaugh Mapping

- Now let's try another example. Write it in C++ code
  - X = A'B'C' + AC'D' + AB' + ABCD' + A'B'C

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B				
AB	1			1
AB'	1	1	1	1

Karnaugh Mapping

- Now let’s try another example. Write it in C++ code
  - X = A’B’C’ + AC’D’ + AB’ + ABCD’ + A’B’C

	C’D’	C’D	CD	CD’
A’B’	1	1	1	1
A’B				
AB	1			1
AB’	1	1	1	1

Karnaugh Mapping

- Now let's try another example. Write it in C++ code
  - $X = A'B'C' + AC'D' + AB' + ABCD' + A'B'C$

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B				
AB	1			1
AB'	1	1	1	1

Karnaugh Mapping

- Now let's try another example. Write it in C++ code too.
  - $X = A'B'C' + AC'D' + AB' + ABCD' + A'B'C$

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B				
AB	1			1
AB'	1	1	1	1

  - Final Equation is $X = B' + AD'$
  - In C++, X = **!B** || (**A && !D**)

Lecture 6

Boolean Logic & Logic Gates III

NAND LOGIC

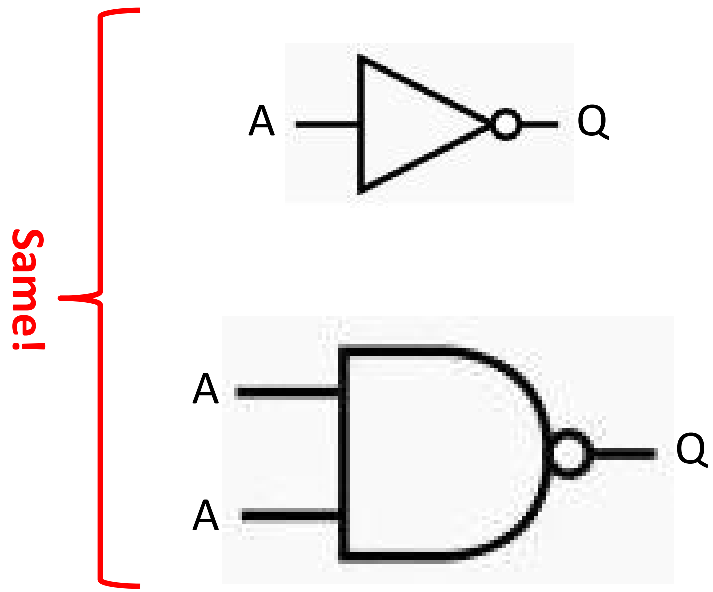
NAND Gate Logic

- **1 of 2** basic logic gates from which any other gates can be constructed
  - **Which is the other?**

- These are sometimes called "**universal gates**"
  - **However modern integrated circuits aren't solely made up of just NAND or NOR gates**
  - **i.e. for CPU design, full custom designs (transistor level) necessary to maximise performance**

Using NAND to express NOT

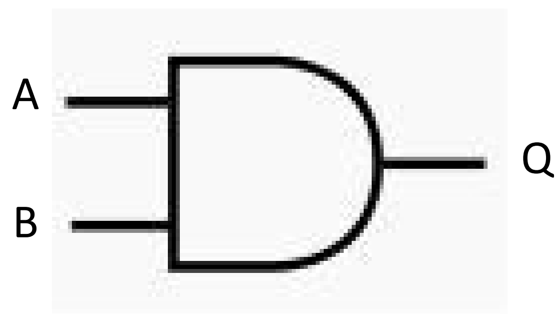
- NAND can be used to express other gate types
- Example:



NOT Gate

A	Q
0	1
1	0

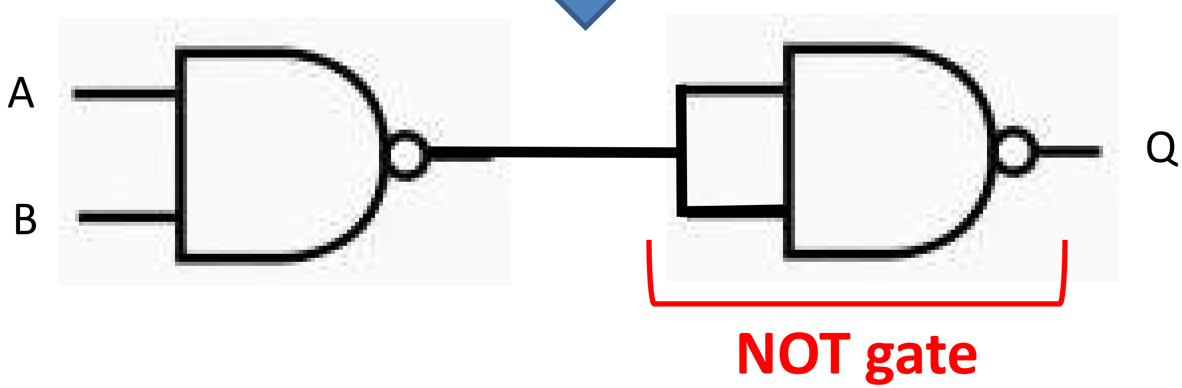
Using NAND to express AND



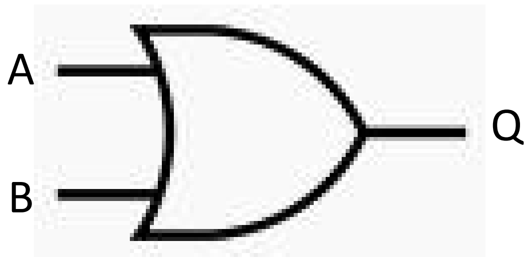
AND Gate

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Same as



Using NAND to express OR



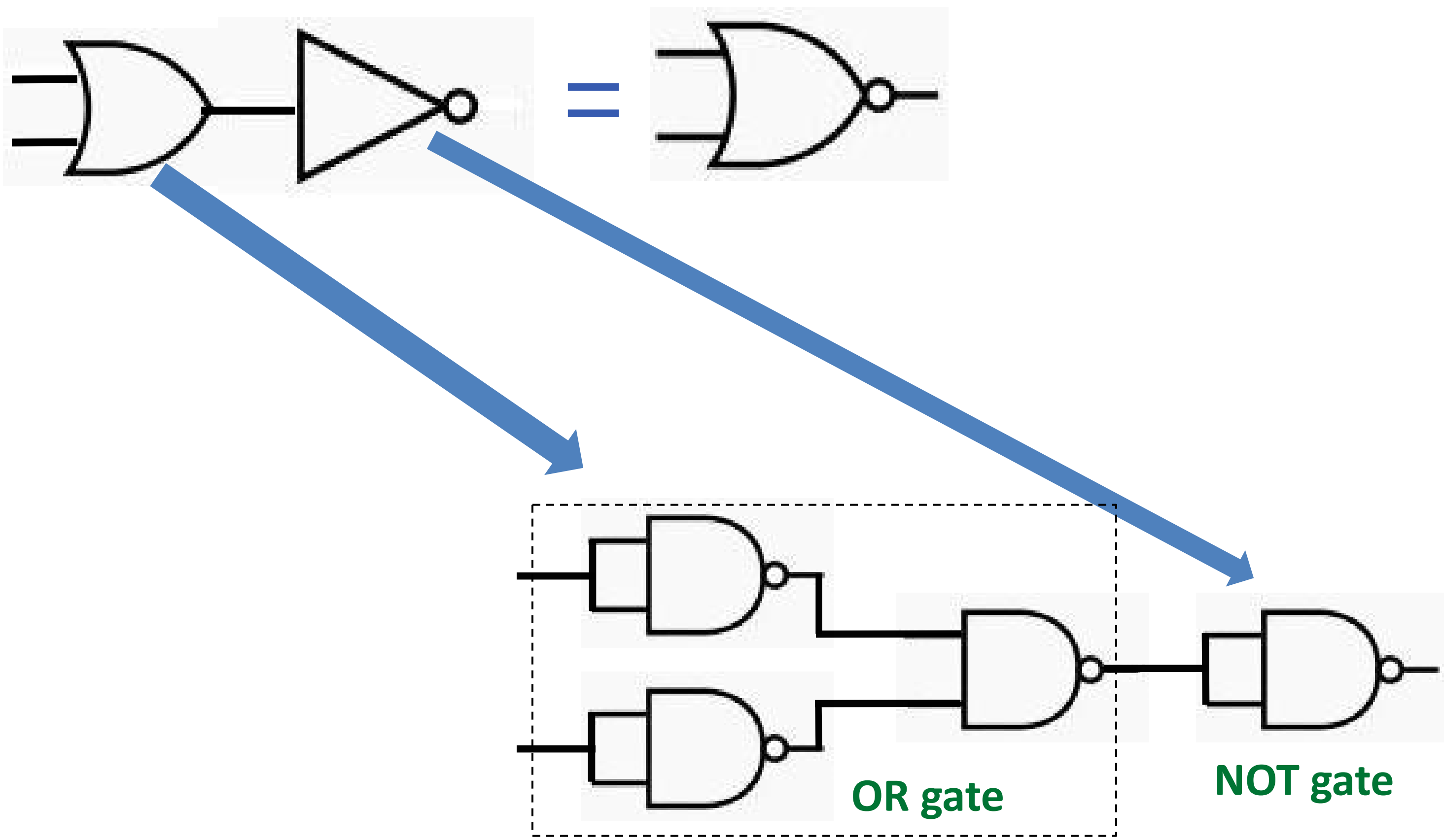
OR Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Recall from DeMorgan's theorem

Using NAND to express NOR

- NOR = **OR + NOT**



XOR & XNOR

- NAND can also express logic for **XOR or XNOR** gate

- **Do** some research online to find out how!

Lecture 7

Boolean Logic & Logic Gates III

BCD NOTATION

About BCD Notation

- Binary Coded Decimal Notation
  - Method of **encoding a decimal number** by **swapping** its digits for the **binary equivalent**
  - Example:

9356 (Decimal)

= 1001 0011 0101 0110 (BCD)

Decimal Digit	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

New Things Next Time

- End of Data Representation, Digital Logic, and Logic Gates

- Questions?