Lecture 4

From Problem Description to Circuit

In this lecture we will:

- Create circuits from Truth Tables.
- Use Karnaugh Maps to minimise circuits.
- Work through a design exercise.

Analysing and transforming circuits

So far we have studied how to analyse and to transform circuits using different forms of representation. For example:

Given a circuit

- find its truth table

Given a Boolean equation

- simplify it

etc.

Circuit Design

Circuit design brings all our analysis work together.

We start with some description of a circuit, formalise it by creating a truth table and then design the electronic circuit that implements it.

The solution is NOT unique.

We will look for the best solution.

Methodological Approaches

Given that we can create a truth table there are two approaches that always work:

1. Synthesise the circuit by using minterms.

   OR together the minterms defining 1s in the truth table.

2. Synthesise the circuit by using maxterms.

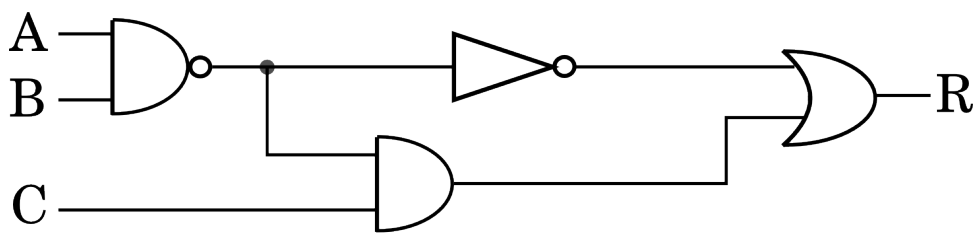
   AND together the maxterms defining 0s in the truth table.

Synthesis by Minterms

First we find the truth table which can be determined from a Boolean equation:

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

or from a circuit:



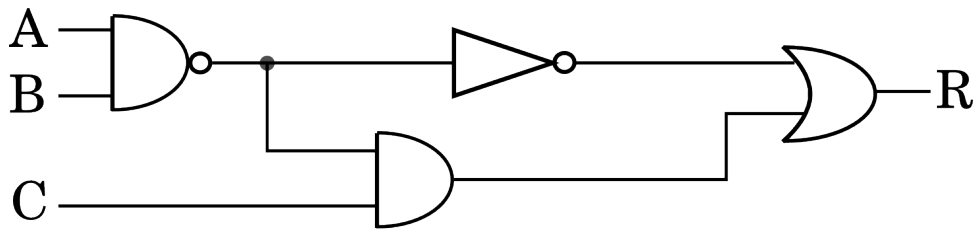
A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Synthesis by Minterms

First we find the truth table which can be determined from a Boolean equation:

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

or from a circuit:



A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Then OR together the minterms corresponding to lines of the table where the output is 1.

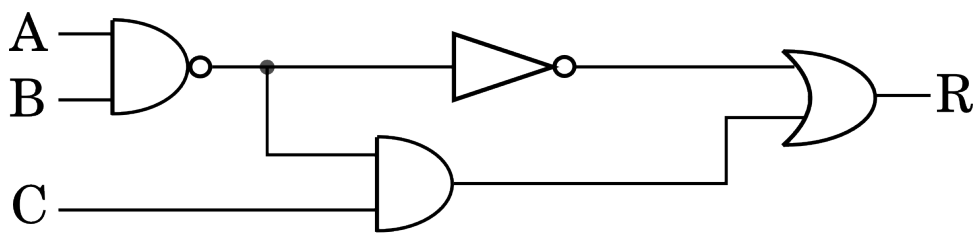
$R = A' \cdot B' \cdot C + A' \cdot B \cdot C + A \cdot B' \cdot C + A \cdot B \cdot C' + A \cdot B \cdot C$

Synthesis by Maxterms

First we find the truth table which can be determined from a Boolean equation:

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

or from a circuit:



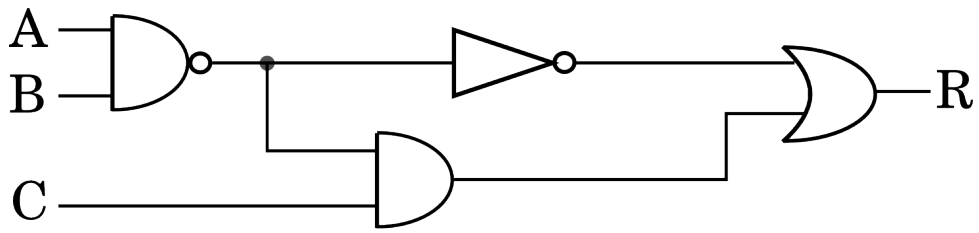
A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Synthesis by Maxterms

First we find the truth table which can be determined from a Boolean equation:

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

or from a circuit:



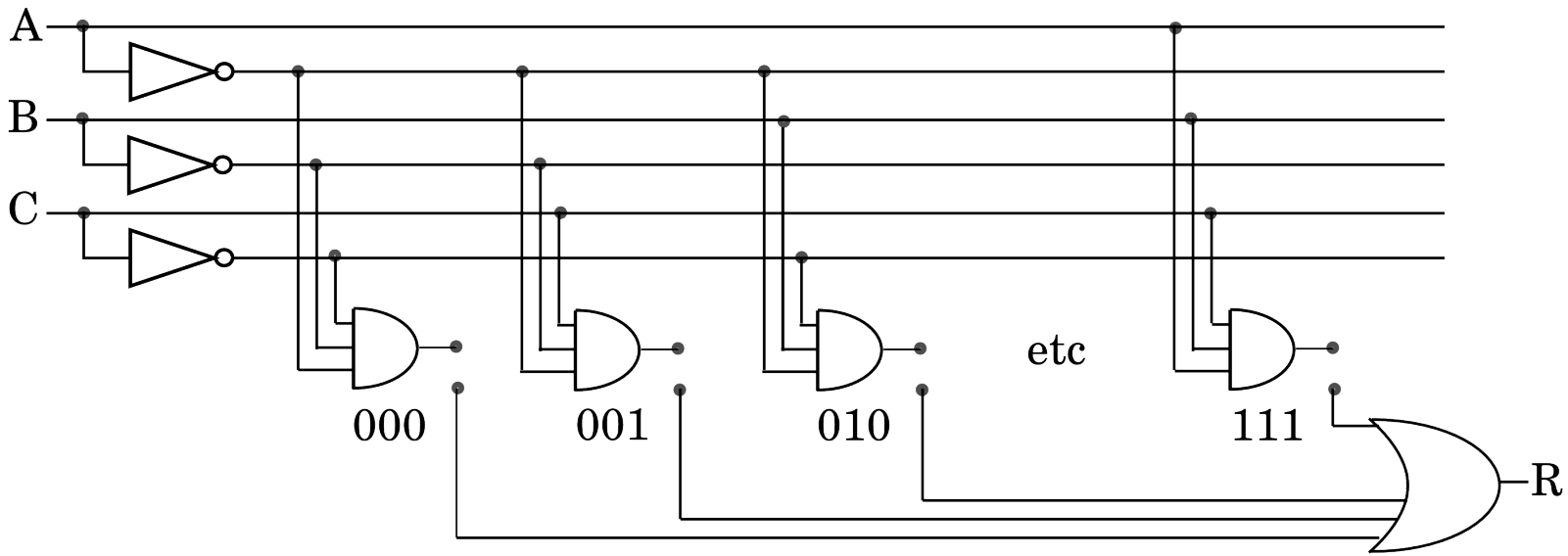
A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Then AND together the maxterms corresponding to lines of the table where the output is 0.

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

Programmable Gate Arrays

A programmable gate array can directly implement a Boolean equation in canonical form. For the minterm form its circuit would be:



Programmable links select the minterms that are linked to the output.

Programmable Gate Arrays

Programmable gate arrays are a perfect way to create prototype designs quickly and efficiently. However they lack flexibility:

1. You only get specific patterns of gates and connections.
2. Using the canonical form means there is no optimisation of speed or circuit size.

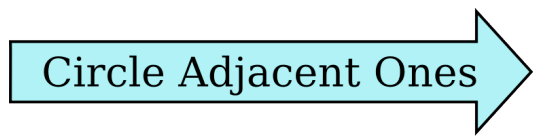
Karnaugh Map Minimisation

A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

There are algorithmic methods that start from the canonical forms and find a minimal circuit. We don't have enough time to discuss them in this course.

We will make use of the Karnaugh map method which starts from the truth table.

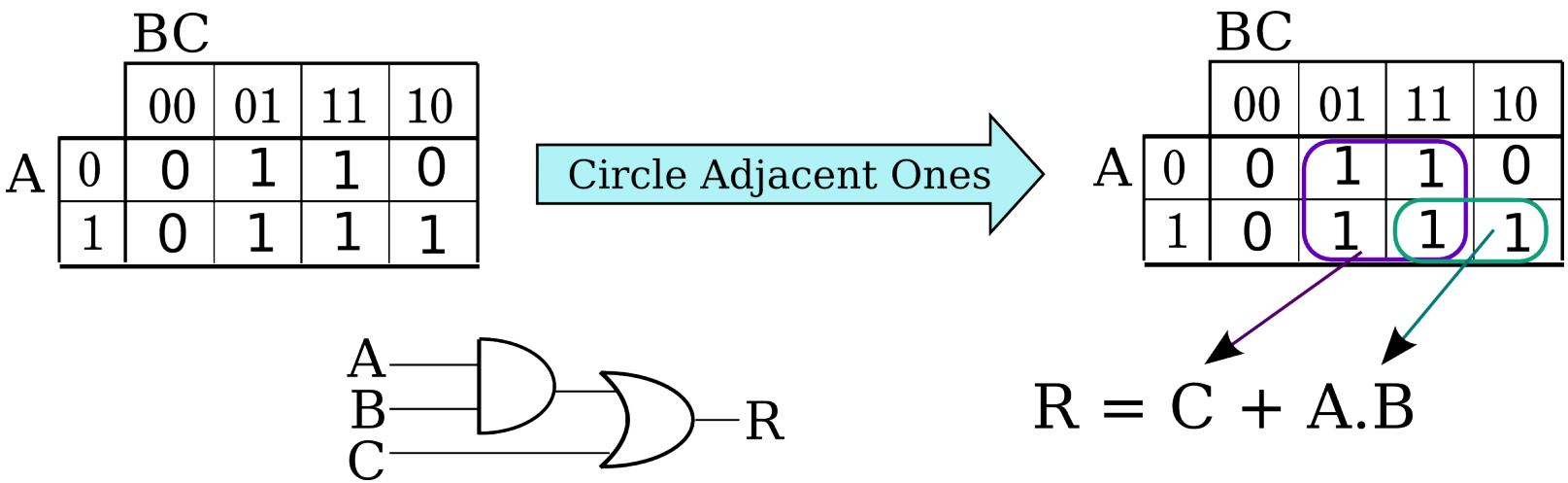
	BC			
A	00	01	11	10
0	0	1	1	0
1	0	1	1	1



	BC			
A	00	01	11	10
0	0	1	1	0
1	0	1	1	1

Karnaugh Map Minimisation

The minimised circuit is much smaller in area and faster than both the original circuit or the programmable logic array.

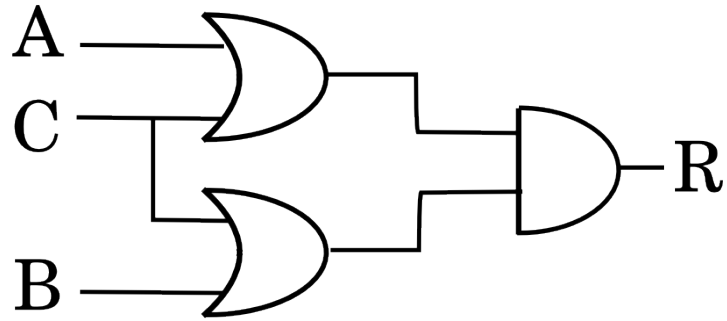


Karnaugh Map Minimisation

We can also try minimising using the maxterms. In this case we circle the zeros.

A \ BC	00	01	11	10
0	0	1	1	0
1	0	1	1	1

$R = (B+C).(A+C)$



However, in this case the final result is not as good as using minterms.

A Design Exercise

We will now work through a design exercise. The informal specification is as follows.

Build a combinatorial digital circuit with three inputs and one output. Inputs A and B are for data, C is for control.

if C = 0: $R = A \cdot B$

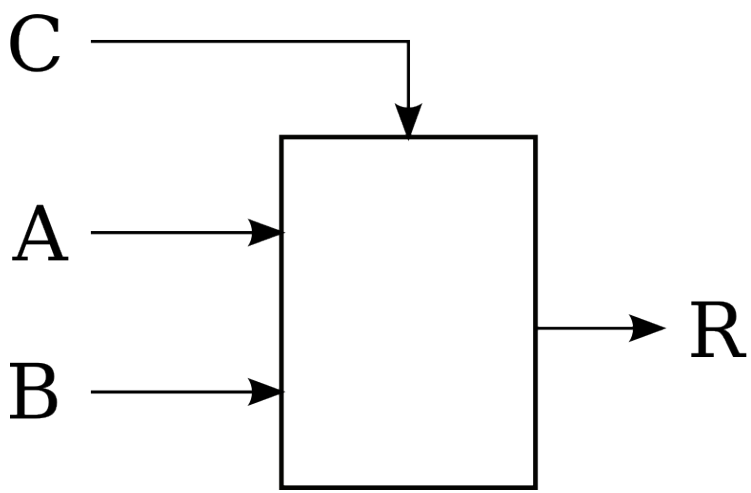
if C = 1: $R = A + B$

Design the circuit to be as small as possible.

Formalising the Specification

The first stage is formalising the specification. We can do this by finding a canonical form, or equivalently finding the truth table.

It often helps to draw a block diagram.



Control	R
0	A.B
1	A+B

Formalising the Specification

The truth table is found by expanding our function table:

C	R
0	A·B
1	A+B

$\Rightarrow$

C	A	B	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Karnaugh Map Minimisation (not again!)

C	A	B	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Yes - yet again:

- The truth table is used to fill up the Karnaugh Map.
- We circle the ones and find the minimum equation.

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Circle Adjacent Ones

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$R = B.C + A.B + A.C$

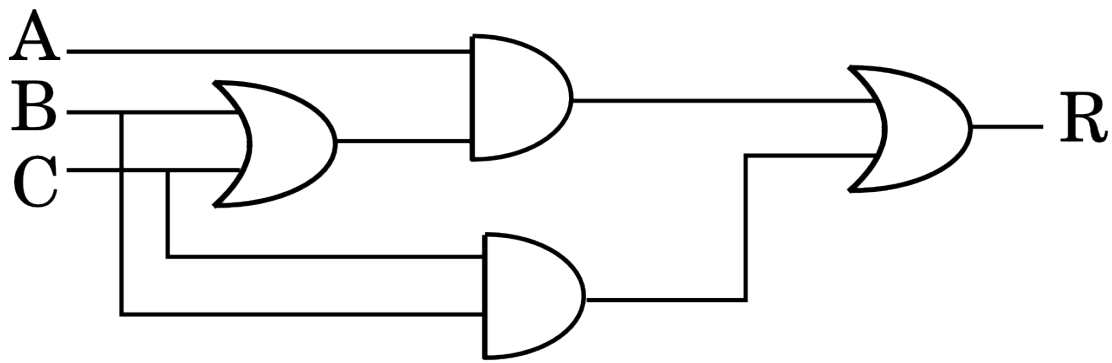
Further Factorisation

The Karnaugh Map isn't perfect - in this case there is another factorisation:

$$R = B \cdot C + A \cdot B + A \cdot C$$

$$R = B \cdot C + A \cdot (B + C)$$

We can now draw the final circuit.



Problem Break

Use maxterms to find the minimum circuit, starting with the same Karnaugh Map.

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

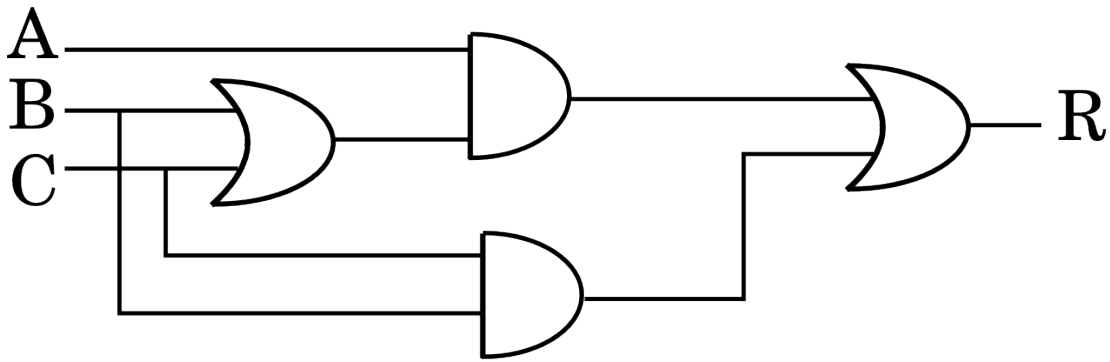
Minimising the Circuit Size

Different gates have different sizes when manufactured as part of a silicon chip. The approximate relative sizes are:

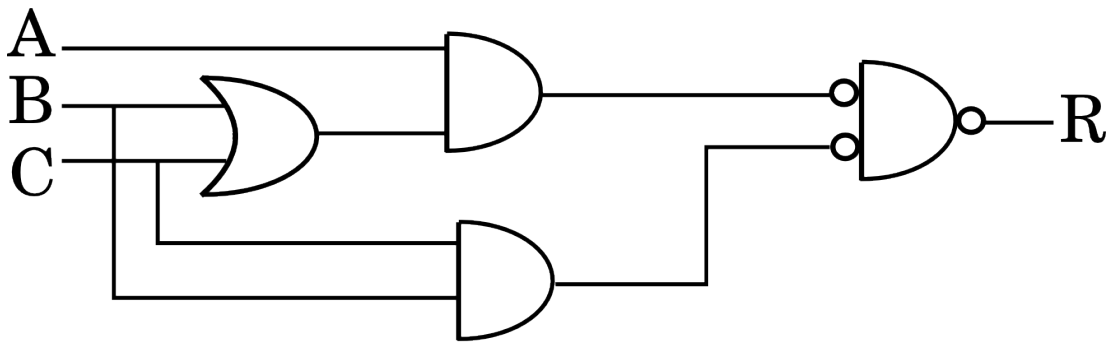
Gate	Nominal Size
INVERTER	3
NAND, NOR	4
AND, OR	6
XOR, XNOR	8

Our circuit design uses AND and OR gates, so we should be able to reduce it's size by changing them to NAND and NOR.

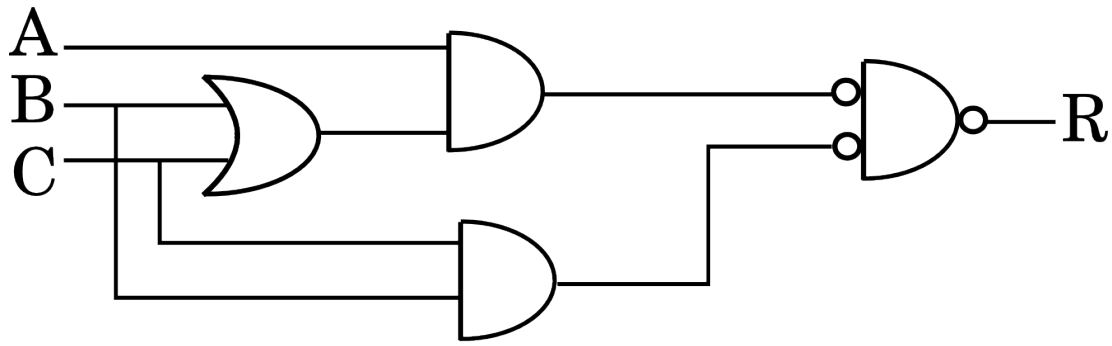
Replacing OR with NAND



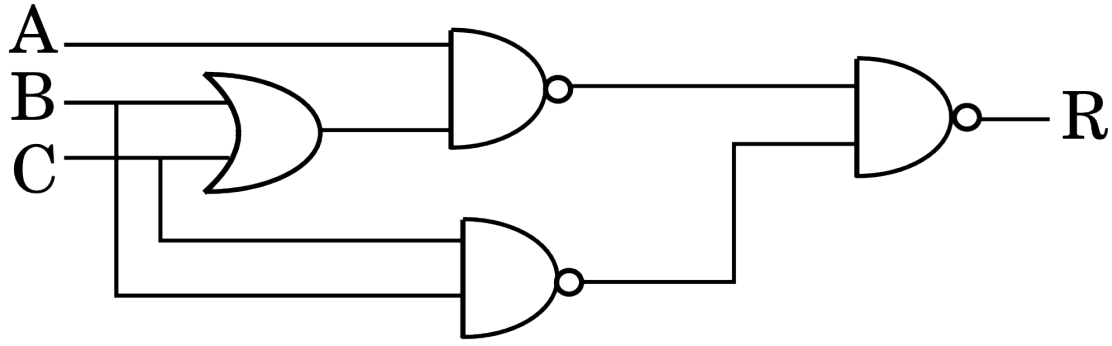
Applying de Morgan's theorem to the last OR gate we re-draw the circuit as:



Regrouping the Inverters



We can move the two inverters on the last gate back creating 2 more NAND gates



The size count has reduced from 24 to 18.

Testing the final Design

Unfortunately debugging hardware is difficult. The final circuit may not implement the original specification for a few - difficult to identify - inputs

For example the original Intel Pentium P5 (1993) had a floating division bug. Clearly:

4195835*3145727/3145727 = 4195835,

but on a P5 you would get

4195835*3145727/3145727 = 4195579 !

The chip was withdrawn and re-manufactured, incurring a high rectification cost.

Systematic Testing

- Feed a large range of inputs to the circuit and observe its behaviour.
- This is done using simulation before manufacture, and after manufacture with real tests.
- The problem is that there are too many possible input combination to test a large circuit exhaustively. Thus is is not feasible to identify all bugs.

Formal Techniques

- This approach aims to verify the chip at design stage (before silicon)
- Abstraction, simulation equivalence, and bounded model checking are used extensively to identify bugs
- However: it's very hard to reason about all possible behaviours because of the very large state spaces.
- The Intel Core i7 EXE module was validated by formal techniques.

Now it's your turn - Coursework 1

- The first coursework is a design exercise like the one we have just done.
- Everybody has their own circuit to design.
- Testing is done with a simple in house simulator called DIGISIM.
- The handout (on the course webpage) contains a full example of a worked solution.