First Year Computer Hardware Course

Lecturer

Duncan Gillies (dfg)

Lectures

The course will begin with Boolean Algebra, and will end with the design of a thirty-two bit computer.

The first twelve lectures cover techniques in the design of digital circuits.

The last six lectures will be concerned with the design of a computer.

Tutorials

Tutorials will serve three purposes:

1. They will provide design examples to reinforce understanding of the lecture material.

2. They will give valuable practice in answering examination style problems.

3. They provide a venue to discuss problems with the tutors and lecturers.

Coursework 1

A combinational circuit design exercise will be given out during week three. You will need to complete it by week five.

Submission will be in the form of a short report plus a circuit design which you can test with a simulator.

Coursework 2

A sequential design circuit exercise will be given out as part of the laboratory exercises at the start of term 2. It will involve designing and testing a circuit with a professional software system called Quartus II.

The total continuous assessment mark for the course comprises:

Combinatorial Circuit Design 60%

Sequential Circuit Design 40%

Books

The best advice about books is:

Don’t buy anything until you are sure you need it.

The notes that will be given out are comprehensive and should cover all the material.

Web based material

All lecture notes, tutorial problem sheets and coursework instructions will be made available from the course web page. (www.doc.ic.ac.uk/~dfg)

Photocopies of the notes will be given out during the lectures, so there is no need to print your own.

Tutorial solutions will be posted on the web a few days after each tutorial

Lecture 1:

Introduction to

Boolean Algebra

George Boole: 1815-1864

Binary Systems

Computer hardware works with binary numbers, but binary arithmetic is much older than computers.

Ancient Chinese Civilisation (3000 BC)

Ancient Greek Civilisation (1000 BC)

Boolean Algebra (1850)

Propositional Logic

The Ancient Greek philosophers created a system to formalise arguments called propositional logic.

A proposition is a statement that can be TRUE or FALSE

Propositions can be compounded by means of the operators AND, OR and NOT

Propositional Calculus Example

Propositions may be TRUE or FALSE for example:

it is raining

the weather forecast is bad

A combined proposition example is:

it is raining OR the weather forecast is bad

Propositional Calculus Example

We can assign values to propositions, for example:

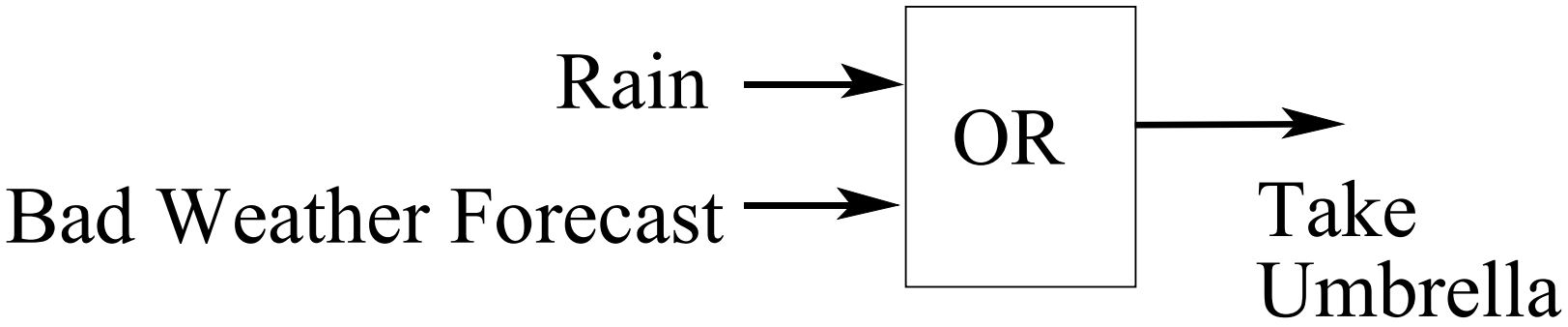
I will take an umbrella if it is raining OR the weather forecast is bad

Means that the proposition “I will take an umbrella” is the result of the Boolean combination (OR) between raining and weather forecast being bad. In fact we could write:

I will take an umbrella = it is raining OR the weather forecast is bad

Diagrammatic representation

We can think of the umbrella proposition as a result that we calculate from the weather forecast and the fact that it is raining by means of a logical OR.



Truth Tables

Since propositions can only take two values, we can express all possible outcomes of the umbrella proposition by a table:

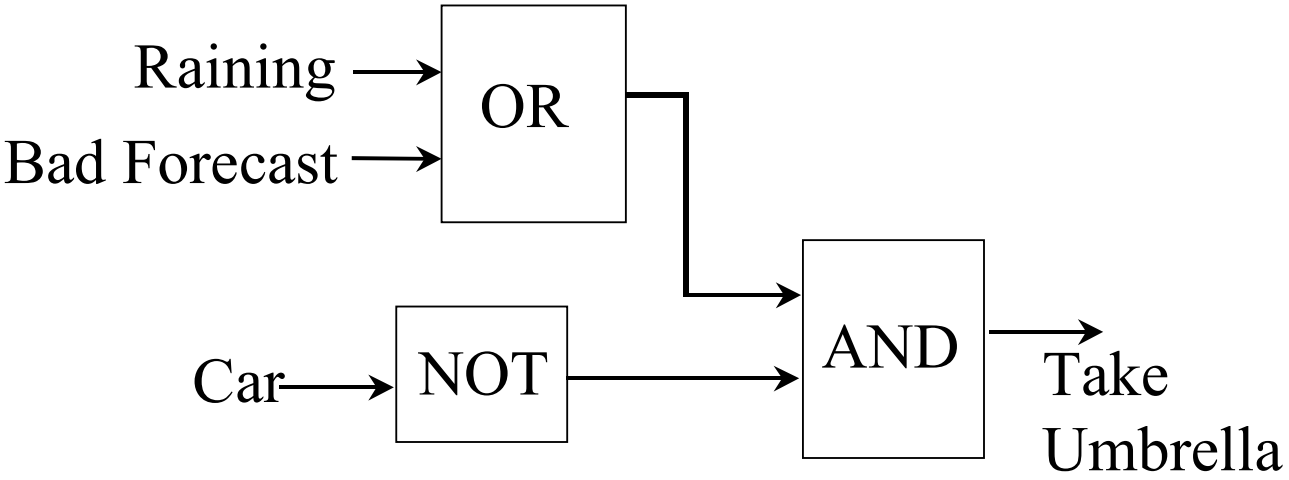
Raining	Bad Forecast	Umbrella
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

More complex propositions

We can make our propositions more complex, for example:

(Take Umbrella) = (NOT (Take Car)) AND
((Bad Forecast) OR (Raining))

and as before represent this diagrammatically



Boolean Algebra

To perform calculations quickly and efficiently we can use an equivalent, but more succinct notation than propositional calculus.

We also need a to have a well-defined semantics for all the “operators”, or connectives that we intend to use.

The system we will employ is called Boolean Algebra (introduced by the English mathematician George Boole in 1850) and satisfies the criteria above.

Fundamentals of Boolean Algebra

The truth values are replaced by 1 and 0:

1 = TRUE 0 = FALSE

Propositions are replaced by variables:

R = it is raining W = The weather forecast is bad

Operators are replaced by symbols

' = NOT + = OR • = AND

Simplifying Propositions

Our previous complex proposition:

Take Umbrella = (NOT (Take Car)) AND
(Bad Forecast OR Raining)

can be formalised by the simpler equation:

U = (C')•(W+R)

Precedence

Further simplification is introduced by defining a precedence for the evaluation of the operators. The highest precedence operator is evaluated first.

OPERATOR	SYMBOL	PRECEDENCE
NOT	'	Highest
AND	•	Middle
OR	+	Lowest

U = (C')•(W+R) = C' •(W+R)

Note that C' •(W+R) is not the same as C' •W+R

Truth Tables

All possible outcomes of the operators can be written as truth tables.

AND

•

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

OR

+

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

NOT

'

A	R
0	1
1	0

Truth tables can be constructed for any function

Given any Boolean expression eg:

U = C' •(W+R)

We can calculate a truth table for every possible value of the variables on the right hand side.

For n variables there are 2^n possibilities.

The truth table for “Umbrella”

$$U = C' \bullet (W+R)$$

R	W	C	X1=R+W	X2=C'	U=X1•X2
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0
Inputs			Partial Results		Outputs

Problem Break

Using Boolean algebra construct a truth table to enumerate all possible values of:

$R = A \bullet B + C'$

A	B	C	A•B	C'	R
0	0	0			
0	0	1			
0	1	0			
etc					

Algebraic Manipulation

One purpose of an algebra is to manipulate expressions without necessarily evaluating them.

Boolean algebra has many manipulation rules, for example, concerning negation and its interaction with and/or:

$(A')' = A$ $A \bullet A' = 0$ $A + A' = 1$

These may be verified by construction a truth table.

The normal laws of algebra apply

Associative

$(A \bullet B) \bullet C = A \bullet (B \bullet C)$

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

Commutative

$A \bullet B = B \bullet A$

$A + B = B + A$

Distributive

$A \bullet (B + C) = A \bullet B + A \bullet C$

$A + (B \bullet C) = (A + B) \bullet (A + C)$ (strange but true!)

Simplification Rules

Simplification rules allow us to reduce the complexity of a Boolean expression:

The first type of simplification rule concerns single variables:

$A \bullet A = A$

$A + A = A$

Simplification rules with 1 and 0

$A \bullet 0 = 0$

$A \bullet 1 = A$

$A + 0 = A$

$A + 1 = 1$

Note that here (as in all simplification rules) A and B can be any boolean expression. Thus

$(P \bullet (Q+R') + S) \bullet 0 = 0$

More general simplification rules

There are many possible simplification rules involving more than one variable. We will look at just one:

$A + A \bullet (B) = A$

It is possible to prove this by construction a truth table, or by direct proof as follows:

$A + A \bullet B = A \bullet (1+B)$ (Distributive law)

$A \bullet (1+B) = A \bullet 1$ (Simplification rule with 1 and 0)

$A \bullet 1 = A$ (Simplification rule with 1 and 0)

de Morgan's Theorem

Two of the most used simplification rules come from de Morgan's theorem:

$(A+B)' = A' \bullet B'$

$(A \bullet B)' = A' + B'$

Note that (as before) A and B can be any Boolean expression.

Generalising de Morgan's theorem:

The theorem holds for any number of terms, so:

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

$$((A+B)+C)' = ((A+B)') \bullet C'$$

$$((A+B)') \bullet C' = A' \bullet B' \bullet C'$$

and similarly:

$$(A \bullet B \bullet C.... \bullet X)' = A' + B' + C' + ...+ X'$$

The principle of Duality

Every Boolean equation has a dual, which is found by replacing the AND operator with OR and vice versa:

We saw one example in de Morgan's theorem:

$$(A+B)' = A' \bullet B'$$

$$(A \bullet B)' = A' + B'$$

The dual of a simplification rule

The simplification rule: $A + A \bullet B = A$

Has a dual equation: $A \bullet (A + B) = A$

Note that the brackets are added to maintain the evaluation order

Proof:

$A \bullet (A + B) = A \bullet A + A \bullet B$

$A \bullet A + A \bullet B = A + A \bullet B$

$A + A \bullet B = A$

Complement expressions

Boolean equations also have a complement found by negating both sides:

$U = C' \bullet (W+R)$

has the complement equation

$U' = (C' \bullet (W+R))'$

Simplification with de Morgan

U' = (C' •(W+R))'

Can be simplified using de Morgan's theorem

(C' •(W+R))' = C + (W+R)'

C + (W+R)' = C + W' • R'

or, in propositional logic:

I will not take the umbrella = I am going by car OR the weather forecast is good AND it is not raining!