Doc112: Hardware

Department of Computing, Imperial College London

First Year Computer Hardware Course

Lecturers

Duncan Gillies

Bjoern Schuller

Timings

Normally thursday afternoons will be organised as follows:

- 14.00 Lecture
- 15.00 Tutorial
- 16.00 Lecture

But because today is the DocSoc Sponsors Exhibition the times will be as follows:

- 14.00 Lecture
- 15.00 Tutorial
- 15.45 Lecture

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

There are two courseworks which are both practical design exercises.

The first is a combinatorial circuit design exercise and will start 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

The second exercise is a sequential circuit design which you should do 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 for the course are comprehensive and cover all the material, so you really do not need anything else.

Course Materials Online

All lecture notes, tutorial problem sheets and coursework instructions are available from the course web page:

www.doc.ic.ac.uk/~dfg

which can be found through CATE.

Tutorial solutions will be posted on the web after each tutorial.

We will have a discussion forum for the course via the piazza system.

Lecture 1

Boolean Algebra

George Boole (1815-1864)

Binary Arithmetic

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

The following two statements are propositions which may be TRUE or FALSE:

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:

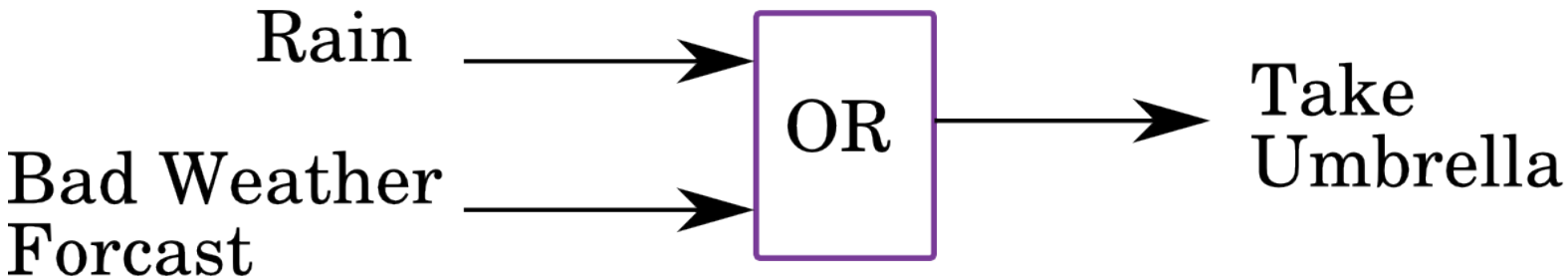
I will take an umbrella IF it is raining OR the weather forecast is bad

Means that the proposition I will take an umbrella is the result of the Boolean combination (OR) between raining and weather forecast is 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:

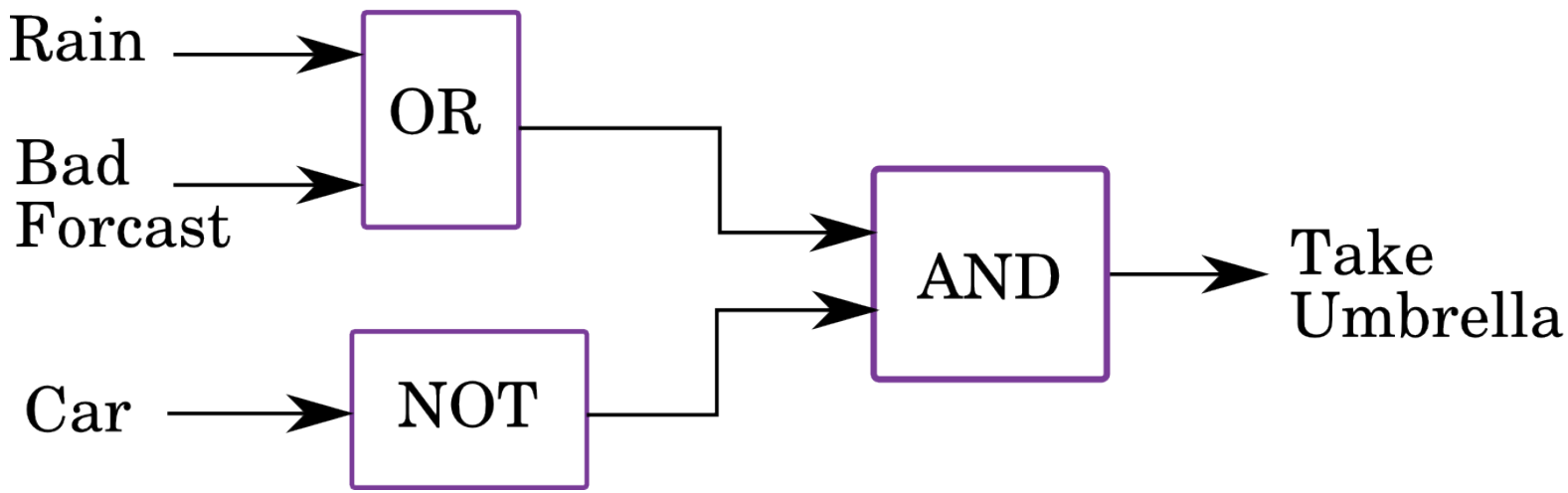
Raining	Bad Forecast	Umbrella
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

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 Lincolnshire 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 $\cdot$ = AND

Boolean Algebra has Simple Notation

Our previous long winded proposition:

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

can be represented by the concise equation:

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

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	$\cdot$	Middle
OR	$+$	Lowest

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

Note that $C' \cdot (W+R)$ is not the same as $C' \cdot W+R$

Truth Tables

The individual Boolean algebra operators can be defined by truth tables which have an entry for each possible outcome:

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 equation

Given any Boolean expression eg:

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

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 the Umbrella equation

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

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

It's time for you to wake up and do some work!

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

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

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 - usually to simplify them - without necessarily evaluating them.

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

$$(A')' = A \qquad A \cdot A' = 0 \qquad A + A' = 1$$

These rules may be verified by constructing a truth table.

There are the usual algebraic rules

Associative:

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

Commutative:

$$A \cdot B = B \cdot A$$
$$A + B = B + A$$

Distributive:

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

Simplification Rules

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

Simplification rules with single variables:

$$A \cdot A = A$$

$$A + A = A$$

Simplification Rules

Equations including 1 or 0 can often be dramatically simplified:

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A + 0 = A$$
$$A + 1 = 1$$

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

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

Further Simplification Rules

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

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

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

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

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

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

de Morgans Theorem

Two of the most used simplification rules are derived from from de Morgans theorem:

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

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

The General form of de Morgan's theorem

The theorem holds for any number of terms, so:

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

and using induction:

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

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 Morgans theorem:

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

Every simplification rule has a dual form

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

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

Note that the brackets are added to maintain the evaluation order.

Proof:

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

Complement equations

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

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

has the complement equation

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

Simplification using de Morgan's Theorem

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

Can be simplified using de Morgans theorem

$$(C' \cdot (W+R))' = C + (W+R)'$$

$$C + (W+R)' = C + W' \cdot R'$$

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!

Don't forget

Lecture 2 will begin at 15.45