# CS1026 – Digital Logic Design

## Boolean Algebra I

Alistair Morris [1]

[1]Distributed Systems Group
Trinity College Dublin

January 19, 2016

# Today's Overview

# Basic Binary Operators

- $+$ called OR
  - E.g. $Z = X + Y$
  - See 74LS32 datasheet
- $.$ called AND
  - E.g. $Z = X.Y$
  - See 74LS08 datasheet
- $'$ called NOT
  - E.g. $Z = X'$
  - Negates/finds the complement
  - See 74LS04 datasheet

# Order of Operation Precedence

- Same as in decimal arithmetic
  - I.e. $=, (), \prime, ., +$
    - Parentheses forces operation order
    - Note: $=$ used for assignment

## An Expression $E$

- A combination of variables and binary operators
- E.g. $Z = (X + Y).X$

## Number of Literals

- Total occurrences of all variables in expression
- E.g. $f(X, Y, Z) = X + Y.X.Z + X'.Y'.Z$ has 7

# Additional Logic Gates

- NAND
  - $(A.B)'$
  - See 74LS02 datasheet
- NOR
  - $(A+B)'$
  - See 74LS00 datasheet

## XOR – $\oplus$

- "Exclusive OR" or Mod 2 addition
- $X \oplus Y = X'.Y + X.Y'$
- $X \oplus Y \oplus Z = (X \oplus Y) \oplus Z$

# What makes an Axiom?

- Self-evident mathematical statements
    - We can state them *without* proof

**Why we use them?**

It allows us/Dr. Boole to develop Boolean Algebra

# Huntington's First Set of Postulates

Given a bag $B$ with at least two elements:

1. If $X, Y \in B$, then $X + Y \in B$
   - If $X, Y \in B$, then $X.Y \in B$
2. $\forall_{x \in B} : X + 0 = X$
   - $\forall_{x \in B} : X + 1 = X$
3. $X + Y = Y + X$
   - $X.Y = Y.X$
4. $X + Y.Z = (X + Y).(X + Z)$
   - $X.(Y + Z) = X.Y + X.Z$
5. $\forall_X : X + X' = 1, X + X' = 0$

# Finding Duals

The dual of an expression gained by:

- Changing 0 with 1
- Changing . with +

$E^D$ gives the dual of:

- $(X + 0)^D = X.1$
- $(X + Y.Z)^D = X.(Y + Z)$

Axioms also works for duals!

# Pure and mixed forms

Pure form:

- $X.Y.Z$
  - Product of terms
- $X + Y + Z$
  - Sum of terms

Mixed form:

- (X+Y).(Z+Y+X)
  - Product of sums (POS)
- X.Y + Y.Z
  - Sum of products (SOP)

# Truth Tables I

| $X$ | $Y$ | $Z$ | $F(X, Y, Z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

How to use?

1. Find all possible combos of “1s” and “0s”
2. Evaluate the output for each set of input values

# Truth Tables II

| $X$ | $Y$ | $Z$ | $F(X,Y,Z)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** |

## A nice easy example

$$F(X,Y,Z) = X.Y + Y.Z + Z'.Y$$

# Why use theorems?

- Axioms and theorems reduce number of literals
  - Less gates needed to implement expression
  - Easier to design and build hardware
- Sometimes handy to just rearrange an expression
  - Allows us to better use available gates

**Example**

$(X + Y).(X + Y') = X$

# How to prove theorems?

1. Use Boolean Algebra [Brown, 2012]
   - Show equality using Axioms
2. Use Truth Tables
   - Show equality using I/O values

Option 2 only works for a *small* number of variables

# Theorems and proofs I

- Double Negation Theorem
  - $X'' = X$
- Idempotency Theorem
  - $X + X = X$
  - $X.X = X$
- Identity Element Theorem
  - $X + 1 = 1$
  - $X.0 = 0$
- Absorption Theorem
  - $X + X.Y = X$
  - $X.(X + Y) = X$
- Associative Theorem
  - $X + (Y + Z) = (X + Y) + Z$

# Theorems and proofs II

- Adjacency Theorem
  - $X.Y + X.Y' = X$
  - $(X + Y).(X + Y') = X$
- Consensus Theorem
  - $X.Y + X'.Z + Y.Z = X.Y + X'.Z$
  - $(X + Y).(X' + Z).(Y + Z) = (X + Y).(X' + Z)$
- Simplification Theorem
  - $X + X'.Y = X + Y$
  - $X.(X' + Y) = X.Y$
- DeMorgans Theorem (General form)
  - $(X_1 + X_2 + \ldots + X_n)' = (X_1)'.(X_2)'.(\ldots).(X_n)'$
  - $(X_1.X_2.\ldots.X_n)' = (X_1)' + (X_2)' + (\ldots) + (X_n)'$

# DeMorgan's, NAND and NOR

Using DeMorgans's Law..

- NANDs and NORs can represent each other
- Transforming from one to another easy:
  1. Invert/complement every input and output
  2. Swap OR and ANDs

## Real World Implication

We can build everything using only NOR or NAND gates

- Cheap mass production

# That's it (for now)

Thanks.. Any Questions?

## You can ask later at:

*morrisa5@scss.tcd.ie*

## Useful links

- Notes/Slides: bitbucket.com/morrisa5/DLD
- LinkedIn: ie.linkedin.com/in/alistair-morris-9712b247

# References (Homework) I

Brown, F. M. (2012).
*Boolean reasoning: the logic of Boolean equations*.
Springer Science & Business Media.