Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Boolean Algebra

Homework #3 Review – 2.33(a)

- **Convert 12.5 to IEEE 754 single precision floating point:**
- Format requirements for single precision (32 bit total length):
  - 1 sign bit
  - 8 bit exponent (which uses a bias of 127)
  - 23 bit significant (which has an **implied 1. that is not stored in the field**)
- Convert 12.5 to binary: 1100.1 x 2^0
  - Normalize it in the IEEE way: 1.1001 x 2^3
  - Bias exponent: 3 + 127 = 130 (10000010 in binary)
- Result
  - Sign bit: **0**
  - Exponent (8 bits): **10000010**
  - Mantissa (23 bits): **10010000000000000000000**
    (padded out to 23 bits, leading 1 not shown!)
  - Thus, **0 | 10000010 | 10010000000000000000000**

Objectives

- **Chapter 3** in textbook
- Understand the relationship between **Boolean logic** and **digital computer circuits**
- Design simple logic circuits
- Understand how simple digital circuits are combined to form complex computer systems
- **Essential concepts only** – There's a whole course (ECPE 71) devoted to this topic!

Survey

- **How many people are in ECPE 71 (Digital Design) this semester?**
- **How many people have taken ECPE 71 in past semesters?**

Origin of Boolean Algebra

- "The Laws of Thought" written by George **Boole** in 1854
  - Invented symbol or Boolean logic
  - Goal: Represent logical thought through mathematical equations
- Computers today essentially implement Boole's *Laws of Thought*
  - Early computer pioneers (John Atanasoff and Claude Shannon) were among the first to see this connection

Boolean Algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values
  - Formal logic:
    - Values of “true” and “false”
  - Digital systems:
    - Values of “on”/“off”, 1 / 0, “high”/ “low”
- **Boolean expressions** are created by performing operations on Boolean variables
  - Common Boolean operators: AND, OR, NOT

AND Truth Table

Truth Table: shows **all possible inputs and outputs**

x	y	xy
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

AND: Referred to as "Boolean **Product**"

OR Truth Table

`x`	`y`	`x+y`
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

OR: Referred to as “Boolean **Sum**”

NOT Truth Table

Overbar symbol means "not"

x	$\overline{x}$
0	**1**
1	**0**

Boolean Algebra

- A Boolean function has:
  - At least one Boolean variable,
  - At least one Boolean operator, and
  - At least one input from the set {0,1}
- It produces an output that is also a member of the set {0,1}

Boolean Algebra

- Example truth table for function

$$F(x,y,z) = x\bar{z}+y$$

- The shaded column in the middle is optional
  - Make evaluation of subparts easier

$$F(x,y,z) = x\bar{z}+y$$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Function Inputs
"Show your work"
Function Output

Order of Operations

- High to low priority
  - NOT operator
  - AND operator
  - OR operator
- This is how we chose the (shaded) function subparts in our table.

$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Simplification

- **Digital** computers implement **Boolean functions in hardware**
- The simpler the Boolean function, the smaller the circuit that implements it
- **What advantages do we get from a smaller circuit?**
  - Simpler circuits are **cheaper to build**
  - Smaller circuits consume **less power**
  - Smaller circuits **run faster** than complex circuits
- Goal: reduce Boolean functions to their **simplest form**!

Boolean Identities

- Identities can help simplify Boolean functions
  - Most identities have two forms:
    AND (product) form, OR (sum) form
  - These identities are intuitive:

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\overline{x} = 0$	$x + \overline{x} = 1$

More Boolean Identities

- Are these familiar from algebra?

Identity Name	AND Form	OR Form
Commutative Law	xy = yx	x+y = y+x
Associative Law	(xy)z = x(yz)	(x+y)+z = x + (y+z)
Distributive Law	x+yz = (x+y)(x+z)	x(y+z) = xy+xz

Even More Boolean Identities

- Familiar from a formal logic class?
- These are very useful!

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

DeMorgan's Law

- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly
- DeMorgan's law makes finding the complement easy:

$$\overline{(xy)} = \overline{x} + \overline{y} \quad \text{and} \quad \overline{(x+y)} = \overline{x}\,\overline{y}$$

DeMorgan's Law

- Easy to extend DeMorgan's law to any number of variables with a **2-step process**
  1. Replace each variable by its complement
  2. Change all ANDs to ORs and ORs to ANDs
- Example: $F(X,Y,Z) = (XY) + (\bar{X}Z) + (Y\bar{Z})$

$$\bar{F}(X,Y,Z) = \overline{(XY) + (\bar{X}Z) + (Y\bar{Z})}$$

$$= \overline{(XY)}\,\overline{(\bar{X}Z)}\,\overline{(Y\bar{Z})}$$

$$= (\bar{X}+\bar{Y})(X+\bar{Z})(\bar{Y}+Z)$$

Boolean Algebra

- Example: Use Boolean identities to simplify

$$F(X,Y,Z) = (X+Y)\,(X+\overline{Y})\,(\overline{X\overline{Z}})$$

Boolean Algebra

- Simplified: $F(X,Y,Z) = (X+Y)(X+\overline{Y})\overline{(X\overline{Z})}$

$(X+Y)(X+\overline{Y})\overline{(X\overline{Z})}$	
$(X+Y)(X+\overline{Y})(\overline{X}+Z)$	DeMorgan's Law
	Double complement Law
$(XX+X\overline{Y}+YX+Y\overline{Y})(\overline{X}+Z)$	Distributive Law
$((X+Y\overline{Y})+X(Y+\overline{Y}))(\overline{X}+Z)$	Commutative and Distributive Laws
$((X+0)+X(1))(\overline{X}+Z)$	Inverse Law
$X(\overline{X}+Z)$	Idempotent and Identity Laws
$X\overline{X}+XZ$	Distributive Law
$0+XZ$	Inverse Law
XZ	Identity Law

Boolean Algebra

- Simplify

$$F(x,y) = \overline{x}(x+y) + (y+x)(x+\overline{y})$$

Canonical Forms

- Numerous ways to state the same Boolean expression
  - “Synonymous” forms are logically equivalent (have identical truth tables)
- Challenge: Confusing!
- Solution: Designers express Boolean functions in standardized or canonical form
  - Simplifies construction of circuit

Canonical Forms

- There are two canonical forms for Boolean expressions: **sum-of-products** and **product-of-sums**
  - Boolean product is the AND operation
  - Boolean sum is the OR operation.
- In the sum-of-products form, ANDed variables are ORed together

$$F(x,y,z) = xy + xz + yz$$

- In the product-of-sums form, ORed variables are ANDed together:

$$F(x,y,z) = (x+y)(x+z)(y+z)$$

Canonical Forms

- Sum-of-Products form: Easy to read off of a truth table
- Look for lines where the function is true (=1).
  - List the input values
  - OR each group of variables together

$$F(x,y,z) = x\bar{z}+y$$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Canonical Forms

- Sum-of-Products form

$$F(x,y,z) = x\bar{z}+y$$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$F(x,y,z) = (\bar{x}y\bar{z}) + (\bar{x}yz) + (x\bar{y}\bar{z}) + (xy\bar{z}) + (xyz)$$

This is *not* in simplest terms,
but it *is* in canonical sum-of-products form