EE334
Boolean Algebra

Last time you were introduced to the concept of Boolean Algebra. Today, we go a bit deeper by learning how to manipulate and simplify Boolean expressions and how to translate Boolean expressions into circuit diagrams and truth tables and back again. We'll tackle these topics in a moment.

Today's lesson plan also calls for learning how to implement Boolean equations using only NAND or NOR gates. This topic is actually covered decently in the Hambley textbook. **Read Hambley pp 363.** The only thing the textbook leaves out is that many real logic circuits (like in your calculators and computers) are built using NAND gates for the most part. The reason for this is that a NAND gate is the simplest and smallest of all logic gates. It is fastest and cheapest to build entire chips or logic circuits by mass producing/replicating one single design.

Now back to the main part of the lesson.

<u>Simplifying Boolean Expressions</u>

Just like the algebra we're used to, Boolean algebra has certain identities and properties that enable teachers to come up with challenging homework problems for students. Seriously, the identities and properties can make your life easier by enabling you to simplify Boolean expressions (reduce them in size and number of variables). Here's a laundry list of identities and properties for you to help complete. **Note that the symbols A, B, and C can be a truth value (0 or 1) or *any logical expression* (a sub-equation that evaluates to 0 or 1).**

A·1 =	A·0 =	A+0 =	A+1 =
~AA =	~A+A =	AA =	A+A =
AB = BA	AB+A~B =	A+B = B+A	A(B+C) = AB+AC
A(BC) = (AB)C = ABC		A+(B+C) = (A+B)+C = A+B+C	

One of the more useful, yet confusing identities, is the so-called **DeMorgan's Law**. DeMorgan's Law for two-variable equation and a three-variable equation:

Two variables: A·B = ~(~A+~B)
A+B = ~(~A·~B)

Three variables: A·B·C = ~(~A+~B+~C)
A+B+C = ~(~A·~B·~C)

Here are two important things to keep in mind when working with DeMorgan's Law.

a. The symbols A, B, and C represent logical variables or sub-expressions. The variables or logical expressions can be NOT'd already.
b. The Law can be generalized to any equation of two or more variables.

1. Practice simplifying Boolean expressions using the identities and properties from the previous page.

   a. Show that X + (XY) = X

   b. Simplify A~BC + A(B+C) as much as possible.

2. Transform the following Boolean expressions using DeMorgan’s Law.

   a. F = ~C + A

   b. F = AB + (~C + A)~D Replace each AND with an OR. (Hambley P7.33a)

The other day you learned how to turn Boolean expressions into truth tables. Now you will learn how to turn truth tables into Boolean equations.

We'll create the expression by creating sub-expressions for each row that might evaluate to 1 (True) and then OR'ing those sub-expressions together. Thus, when any row becomes True, the entire expression will then be True. A simple algorithm for doing the transformation is:

a. Find a row that evaluates to 1 (True).
b. AND together the variables of the row.
c. In the AND expression for the row, negate a variable if it was equal to 0. E.g., in a row with two variables A and B, if A = 1 and B = 0, then the AND clause would be "A~B"
d. Repeat steps (a) through (c) for each row that evaluates to 1, creating a separate AND clause for each row.
e. When all rows are complete, OR the clauses together.

Logical expressions consisting of AND clauses that are OR'd together are called **Sum of Product (SOP)** expressions. This is probably because AND is somewhat like multiplication and OR is a bit like addition.

3. Shown below is a truth table using just two variables. Create an SOP expression based on the truth table.

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Z =

4. The expression for Z in problem #3 is equivalent to what well-known logic function?

5. Convert the following truth table into an SOP expression.

F	G	H	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Out =

6. Simplify the expression for Out from problem #5.

   Out =

Another useful skill, a pretty easy one at that, is to turn a Boolean equation into a circuit diagram as well as the reverse. Try your hand at these problems.

7. Create circuit diagrams using logic gate symbols for the following equations.

   a. PQ + R

   b. ~S+~T+U

   c. (D~EF) + (GH~I) + E

8. Write down the logic expression performed by the following circuit diagrams.

   a.

   b.