ECE 124 - Digital Circuits and Systems

Kevin Carruthers

Winter 2013

Boolean Logic

Truth Tables

Truth tables are a method to define binary functions. A truth table lists the values of a function for all possible input combinations.

There are three main operators: AND, OR, and NOT.

Table 1: AND (denoted by multiplication)

x	y	f
0	0	0
0	1	0
1	0	0
1	1	1

Table 2: OR (denoted by addition)

x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

Table 3: NOT (denoted by exclamation marks, negatives, and primes)

x	f
0	1
1	0

We can write logic functions as equations, using these denotations. For example

$$f = !x!y + xy$$

is

x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

Note the following transformation from the truth table to the equation form

$$\begin{aligned} f &= !(x + y - xy) \\ &= !(x + y) - !xy \\ &= !(x + y) + xy \\ &= !x!y + xy \end{aligned}$$

Thus we can derive

$$!(x + y) = !x!y$$

Symbols

- AND is a half circle with the inputs on the flat side
- OR is a crescent moon with the inputs on the concave side
- NOT is a triangle with a circle on the tip, on the output side.

Boolean Algebra

The means by which we can manipulate Boolean logic equations. It is an algebraic structure defined by a set of elements (0 and 1), as well as operators (+, *, and !), and satisfies some postulates.

Postulates

1. Closure - The system must be closed with respect to its elements for each operator
2. Identity - The operators must have identity elements $\big(x + 0 = x, 1x = x\big)$
3. Comutivity - The operators must perform the same function regardless of element order $\big(x + y = y + x, xy = yx\big)$

4. Distributivity - The elements must distribute within operators $\big(x(y+z) = xy+xz, x+yz = (x+y)(x+z)\big)$
5. Opposite - The elements must have opposites within their scope $\big(x+!x = 1, x!x = 0\big)$
6. Uniqueness - There are at least two non-equal elements. $\big(0 \neq 1\big)$

Theorems

1. $x + x = x,\ xx = x$
2. $x + 1 = 1,\ 0x = 0$
3. Involution: $!!x = x$
4. Associativity: - $x + (y + z) = (x + y) + z,\ x(yz) = (xy)z$
5. DeMorgan: $!(x + y) = !x!y,\ !(xy) = !x + !y$
6. Absorption: $x + xy = x,\ x(x + y) = x$
7. Cancellation: $x(x + y) = x + xy = x(1 + y) = x$
8. Reduction: $x + \bar{x}y = x + y$

Function Manipulation

We can use theorems and postulates to simplify and manipulate functions. This can give us smaller, cheaper, faster circuits. Aside: for any binary variable x, you always have a positive literal x and a negative literal $\bar{x}$.

Example:

$$\begin{aligned} f &= !c!ba + c!ba + !cba + cba + !cb!a \\ &= !c!ba + c!ba + !cba + !cba + cba + !cb!a \\ &= (!c + c)!ba + (!c + c)ba + !cb(a + !a) \\ &= !ba + ba + !cb \\ &= (!b + b)a + !cb \\ &= a + !cb \end{aligned}$$

Physical Cost of Circuits

Inverter gates at the input are free. Any AND or OR gate costs 1 + the number of inputs it has.

Minterms and Maxterms

Every n variable truth table has 2^n rows. For each row, we can write its minterm (an AND which evaluates to 1 when the associated input appears, otherwise 0) and maxterm (an OR which evaulates to 0 when the associated input appears, otherwise 1).

x	y	z	Minterm	Designation	Maxterm	Designation
0	0	0	$\bar{x}\bar{y}\bar{z}$	m_0	$x+y+z$	M_0
0	0	1	$\bar{x}\bar{y}z$	m_1	$x+y+\bar{z}$	M_1
0	1	0	$x\bar{y}z$	m_2	$x+\bar{y}+z$	M_2
0	1	1	$\bar{x}yz$	m_3	$x+\bar{y}+\bar{z}$	
1	0	0	$x\bar{y}\bar{z}$	m_4	$\bar{x}+y+z$	
1	0	1	$x\bar{y}z$	m_5	$\bar{x}+y+\bar{z}$	
1	1	0	$xy\bar{z}$	m_6	$\bar{x}+\bar{y}+z$	
1	1	1	xyz	m_7	$\bar{x}+\bar{y}+\bar{z}$	

Canonical Sum-of-Products (SOP)

A way to write down a logic expression from a truth table using minterms. Thus gives

Table 4: f(x)

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$f(x) = \bar{x}\bar{y}z + x\bar{y}\bar{z} + xyz$$

Canonical Product-of-Sums (POS)

A way to write down a logic expression from a truth table using maxterms. Thus $f(x)$ gives

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

To **convert** between the two, we have

$$!(m_0 + m_2) = !m_1 !m_3 = M_1 M_3$$

Standard SOP

The canonical SOP is fine, but is not minimal. Any AND of literals is called a **product term** (ie. a minterm is a product term, but not all product terms are minterms).

For example, for f we have

$$\begin{aligned} f &= m_3 + m_5 + m_6 + m_7 \\ &= \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz \\ &= yz + xz + xy \end{aligned}$$

These are **two-level** circuits: if you ignore inversions, you have two levels of logic (a single set of each AND and OR gates, for each possible circuit).

Standard POS

Similarly, canonical POS are not minimal. Any OR of literals is called a **sum term** (ie. a maxterm is a sum term, but not all sum terms are maxterms).

For example

$$\begin{aligned} f &= (x + y + z)(x + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + y + \bar{z})(\bar{x} + \bar{y} + z) \\ &= (x + z)(\bar{y} + z)(x + \bar{y})(\bar{x} + y + \bar{z}) \end{aligned}$$

Other Logic Gates

For actual implementation, some other types of logic gates are useful.

NAND

Performs the AND then NOT function. It looks like an AND gate with a NOT circle immediately after.

Table 5: NAND, $f = x\bar{y}$

x	y	f
0	0	1
0	1	1
1	0	1
1	1	0

NOR

Performs the OR then NOT function. It looks like an OR gates with a NOT circle immediately after. f = ab + b!c = b(a + !c)

Table 6: NOR, $f = x \bar{+} y$

x	y	f
0	0	1
0	1	0
1	0	0
1	1	0

Note that NAND and NOR gates are **universal** (they can implement any function).

On CMOS, NAND and NOR gates use less transistors (four, instead of the normal six).

XOR/NXOR

The exclusive OR is denoted as an OR gate with a second arc.

Table 7: XOR, $f = xPLUSWITHCIRCLEy$

x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

The NXOR also has a NOT circle at its tip, and gives us

Table 8: NXOR, $f = xPLUSIW\bar{T}HCIRCLEy$

x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

For more than two inputs, XOR will be true if the number of true inputs is odd.

Buffers

Denoted as a triangle, a buffer does nothing to the function. It amplifies the signal, which is especially useful in long wires.

Table 9: Buffer

x	f
0	0
1	1

Karnough Maps

A different graphical representation of a logic function equivalent to a truth table (ie. holds the same value). It's not tabular, but more like a matrix. Note that it only works for up to five inputs.

It is useful because we can do logic optimization / simplification graphically.

For a two-input function

x	y	f
0	0	1
0	1	1
1	0	0
1	1	1

we have

	0	1
0	1	0
1	1	1

By "drawing" rectangles on this chart (over top of the true values), we can find an equation for the function. Note that these rectangles can "wrap around" the chart or overlap each other, and that we can also go backwards to generate the table based on an equation (especially simplified equations).

Each rectangle gives an AND'd function, and the rectangles can then be OR'd. Longer rectangles cover more possibilities, and are rectangles always have dimensions which are equal to 2^n.

We can also draw rectangles on the false responses. This will give us a product of sums, instead of a sum of products. This can be easier that placing a rectangle around the true results in some cases.

In essence, the best option is to pick whichever one gives you fewer, larger rectangles. Though you will be able to simplify your answer either way, this will give you the simplest possible starting equation (note that some equations may still require some basic simplifications to be made the most simple possible).

Note that we can also do this in three or more dimensions with

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

and get

	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Four-variable maps have dimensions four by four. Five-variable ones can be represented as two four-variable maps.

"Don't Cares"

Sometimes the value of f doesn't matter, and we can exploit this face when minimizing: these cases are "don't cares". You can chose to set these values as either 0 or 1, whatever makes the logic simpler.

Multiple Output Functions

When dealing with two-output functions, ie circuits with two different functions, optimizing each one seperately is not always the best option. If we can design a circuit with shared circuitry, this will often decrease our total cost.