Review of Combinational Logic Design

This lecture reviews the design of combination logic.
After this lecture you should be able to convert an informal description of the behaviour of a combinational logic circuit into a truth table, a sum of products boolean equation and a schematic.

Logical Variables and Boolean Algebra

A logic variable can take on one of two values, typically called true and false (T and F). With *active-high* logic true values are represented by a high (H) voltage. With *active-low* logic true values are represented by a low (L) voltage. Variables using negative logic are usually denoted by placing a bar over the name ($\overline{B}$), or an asterisk after the variable name (B^*).

Warning: We will use or 1 and 0 to represent *truth values* rather than voltage levels. However, some people use 1 and 0 to represent voltage levels instead. This can be very confusing.

Exercise: A chip has an input labelled $\overline{OE}$ that is used to turn on ("enable") its output. Is this input an active-high or active-low signal? Will the output be enabled if the input is high? Will the output be enabled if the input is 1?

To add to potential sources of confusion, an overbar is sometimes used to indicate a logical complement operation rather than a negative-true signal. The only way to tell the difference is from the context.

Combinational Logic

A combinational logic circuit is one where the output is a function only of the current input – not of any past inputs. A combinational logic circuit can be represented as:

- *a truth table* that shows the output values for each possible combination of input values,
- *a boolean equation* that defines the value of each output variable as a function of the input variables, or
- *a schematic* that shows a connection of hardware logic gates (and possibly other devices) that implement the circuit.

Truth Tables

For example, the truth table for a circuit with an output that shows if its three inputs have an even number of 1's (even parity) would be:

a	b	c	p
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1

Exercise: Fill in the last two rows.

Sum of Products Form

From the truth table we can obtain an expression for each output as a function of the input variables.

The simplest method is to write a *sum of products* expression. Each term in the sum corresponds to one line of the truth table for which the desired output variable is true (1). The term is the product of the each input variable (if that variable is 1) or its complement (if that variable is 0). Each such term is called a *minterm* and such an equation is said to be in *canonical* form.

For example, the variable *p* above takes on a value of 1 in four lines (the first, fourth, sixth and seventh lines) so there would be four terms. The first term corresponds to the case where the input variables are $a = 0$, $b = 0$ and $c = 0$. So the term is $\overline{a}\,\overline{b}\,\overline{c}$. Note that this product will only be true when a, b and c have the desired values, that is, only for the specific combination of inputs on the third line.

If we form similar terms for the other lines where the desired output variable takes on the value one and then sum all these terms we will have an expression

that will evaluate to one only when required and will evaluate to zero in all other cases.

Exercise: Write out the sum-of-products equation for *p*. Evaluate the expression for the first two lines in the table.

Common Combinational Logic Functions

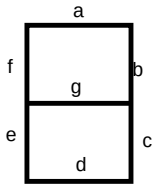
In addition to the standard logic functions (AND, OR, NOT, XOR, NAND, etc) some combinational logic functions are widely used:

- a *decoder* is a circuit with N inputs and 2^N outputs. The input is treated as a binary number and the output selected by the value of the input is set true. The other outputs are false. This circuit is often used for address decoding.
- a *priority encoder* does the inverse operation. The N output bits represent the number of the (highest-numbered) input line.
- a *multiplexer* copies the value of one of 2^N inputs to a single output. The input is selected by an N-bit input.
- a *demultiplexer* does the inverse operation and copies one input to one of 2^N outputs.
- *adders* and *comparators*, perform arithmetic operations on inputs interpreted as binary numbers
- *drivers* and *buffers* do not alter the logical value but provide higher drive current, tri-state or open drain or open collector (OC) outputs

Exercise: Write out the truth table and the canonical (unsimplified) sum-of-products expression for a 2-to-1 multiplexer.

Example: 7-segment display driver

LED numeric displays typically use seven segments labeled ‘a’ through ‘g’ to display a digit between 0 to 9:



This example shows the design of a circuit that converts a 2-bit number into seven outputs that turn the segments on and off to show numbers between 0 and 3. We use the variables A and B for the two input bits and a to g for the seven outputs. We can build up a truth table for this function as follows:

B	A	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	0	0	1

From the truth table we can then write out the sum of products expressions for each of the outputs:

a = $\overline{A}\overline{B} + \overline{A}B + AB$
b = 1
c = $\overline{A}\overline{B} + A\overline{B} + AB$
d = $\overline{A}\overline{B} + \overline{A}B + AB$
e = $\overline{A}\overline{B} + \overline{A}B$
f = $\overline{A}\overline{B}$
g =

Exercise: Fill in the last line of the table. Draw the schematic of a circuit that implements the logic function for the ‘g’ segment.