FREQUENTLY ASKED QUESTIONS

April 13, 2017

Content Questions

How do you know what the minimum number of logic gates you need is?

This is not always easy: here's an answer from Stack Exchange.

How do look-ahead operators work?

More sophisticated digital circuits than our simple "ripple" adder have a "look-ahead" function: here's a Wikipedia link describing it.

What are multiplexers used for?

Multiplexers have many uses. They route signals according to an input address. They can be used to pass on only signals of interest. This can be useful to cut down on the number of channels you need. For example, suppose you have lots of signals you want to digitize or process. But most of the time the lines are quiet. It's expensive to build processing electronics for every single channel. Let's say you have 100 signal lines, but at any given time only one is active. You can use a multiplexer to select only the active line to pass on for processing, and thereby save money by building only one channel of electronics for processing instead of 100.

Multiplexers can also be used to turn parallel data into serial data, i.e., to change a set of bits presented in parallel to a set of bits sent one at a time. Eggleston discusses this application in Section 8.15.

What are demultiplexers used for?

Demultiplexers do effectively the opposite of multiplexers. For a multiplexer, one of several input lines is transferred to an output line according to an address (so can work for parallel to serial data transfer). For a demultiplexer, it's the opposite —information on one input line can be transferred to any of several output lines according to an address. This could be used to convert serial to parallel data, for example. Or you might want to route data to one of multiple outputs (say, to one of several users).

So you can put a voltage into a logic gate and the output is the input voltage, not "1" or "0", as in the multiplexer?

No, not really... in today's multiplexer example, the "D" input is supposed to be a logic input, i.e., its value is 0 or 1. In general you don't put non-logic levels into logic gates, or you won't get sensible results.

However the term "multiplexer" can refer to a more general circuit: it can mean any circuit for which multiple input lines are selected by an address. You can have a circuit with multiple *analog* input lines, such that the signal on only one line is sent through to an output (or processed in some way) if that line's address is set on some digital input line. However in this case the analog signals probably wouldn't be going directly into logic gates. They would be going into some "front-end" circuit that would handle them appropriately.

How do you know if the input your address bits correspond to have something interesting?

Well, what is interesting and how to decide what is interesting depends entirely on your particular application... usually it will have to do with which lines are active (i.e., have changing logic levels, or voltages exceeding some threshold, or have certain patterns or logic conditions characterizing them).

As an example from my research, you might have a set of detectors which only produce signals some of the time, when a particle goes through them. If the voltages of the signals on some set of detectors exceed some threshold within some time window (the condition might be simple or complex), then a "trigger" is generated. The trigger signal can be used to tell a multiplexer to send data through for the channels that are active.

Or, say you want to send a phone signal through to a particular phone number. Then you would address the lines corresponding to that number in order to pass on the information. (Of course in an actual phone the circuits are more complicated that our humble little MUX box. But the general concept applies.)

What do you do when you want to select more than one address?

I'm not sure what you mean exactly, but you can devise logic to do more complicated things than just the basic multiplexing we saw today. You can create a device that takes two separate addresses as input and then puts the

state on one of the input lines through to the output for whatever address condition you want. You could design this using the truth table method.