Episode 6.09 – Multiplexers

Welcome to the Geek Author series on Computer Organization and Design Fundamentals. I'm David Tarnoff, and in this series, we are working our way through the topics of Computer Organization, Computer Architecture, Digital Design, and Embedded System Design. If you're interested in the inner workings of a computer, then you're in the right place. The only background you'll need for this series is an understanding of integer math, and if possible, a little experience with a programming language such as Java. And one more thing. Due to the computational nature of this episode, you might want to visit the transcript page found at intermation.com to download the episode worksheet.

Back in the old days, televisions had large mechanical knobs that turned with a heavy "chunk-chunk-chunk" allowing you to switch from channel to channel to channel. It wasn't smooth, but heck, there were only three channels, so you didn't have to do it that often. It may seem odd to describe a mechanical rotary switch in an episode about digital electronics, but the requirements are the same: route one data stream from multiple available channels to a single output.

A multiplexer, sometimes referred to as a MUX or a data selector, is a device that replaces the mechanical knob with a set of binary control inputs. The pattern of ones and zeros at these control inputs selects which of several binary data inputs is to be connected to a single data output. If there are n binary control or "select" lines, then one of 2 to the n data inputs can be routed to the output. For example, if we have three select lines, labeled S2, S1, and S0, then the binary value on these select lines will determine which of eight data lines, D0 through D7, is to be connected to the output, Y. We refer to this configuration as an eight-to-one multiplexer since it routes one of eight inputs to the one output. You might also see it referred to as an eight-channel multiplexer.

As with any digital circuit, we can use a truth table to describe the operation of the eight-to-one multiplexer. There will be eight rows in our truth table, one for each possible pattern of ones and zeros for S2, S1, and S0. The output column is going to look a little different from what we're used to. It doesn't contain ones or zeros or even don't cares. Instead, it shows which input is being directed to the output. For the row where all three select lines equal zero, Y is going to equal whatever the data input D0 equals. Therefore, the output column will simply have D0 for the top row. When S2, S1, and S0 are equal to 0-0-1, the binary representation of a decimal one, Y will equal D1, so D1 is placed in the output column for the second row. When S2, S1, and S0 are equal to 0-1-0, the binary representation of a decimal two, Y will equal D2. When S2, S1, and S0 are equal to 0-1-1, the binary representation of a decimal three, Y will equal D3. This pattern continues until we get to the last row where S2, S1, and S0 all equal one thereby routing D7 to the output Y.

Creating the truth table in this way is much easier than trying to create the truth table using all of the inputs to the eight channel multiplexer. This device has eleven inputs: the selector bits, S2, S1, and S0, and the data bits, D0 through D7. This gives us two to the eleventh or 2,048 rows in our truth table. Describing a truth table this large could take hours. There is another advantage, though. It also allows us leave the type of data at the data inputs unspecified, meaning we could use the same truth table to specify an eight channel multiplexer used to route analog data.

This brings us to the digital circuit. In the case of a multiplexer that routes digital data, we can create this circuit as one complex sum-of-products expression. A Karnaugh map may fall short of our needs this

time, however. If we were to design a simple four-channel multiplexer, we would need to allow for the two selector bits, S1 and S0, and the four data channels, D0, D1, D2, and D3. Six inputs requires a three dimensional Karnaugh map. And if we wanted to design an eight channel multiplexer, we would need a six dimensional Karnaugh map to allow for its eleven inputs.

Just as we did in Episode 4.03 when we introduced combinational logic, we are going to talk our way through the design of a four channel multiplexer instead of using a Karnaugh map.

The fundamental operation of our multiplexer is that if the selector bit values identify a channel, and that channel equals a one, we output a one. Otherwise, we output a zero. That sounds simple. In fact, it sounds like an AND gate. Let's start with data channel 0 of our four channel multiplexer.
Channel zero is selected when both S1 and S0 equal zero. Therefore, we want to output a one when S1 equals zero AND S0 equals zero AND when data channel D0 equals a one. That gives us the product S1-bar AND S0-bar AND D0. Otherwise, if both S1 and S0 equal zero and D0 equals zero, we output a zero, which is exactly what this circuit does.

Let's move on to data channel one. Channel one is selected when S1 equals zero and S0 equals one. Combining this with the value being output from data channel D1 gives us the product S1-bar AND S0 AND D1. If S1 equals zero and S0 equals one and D1 equals zero, we output a zero.

Data channel two is selected when S1 equals one and S0 equals zero. The product S1 AND S0-bar AND D2 will pass D2 along to the output. Lastly, data channel three is selected when both S1 and S0 equal one. The product to route the data from D3 along to the output is S1 AND S0 AND D3. The final circuit for the four channel multiplexer is simply the sum of these four products: S1·S0·D0 + S1·S0·D1 + S1·S0·D2 + S1·S0·D3 (read as S1-bar AND S0-bar AND D0 OR-ed with S1-bar AND S0 AND D1 OR-ed with S1 AND S0-bar AND D2 OR-ed with S1 AND S0 AND D3).

In our next episode, we will flip the multiplexer in order to create a demultiplexer. Where the multiplexer routes one of multiple input data streams to a single output, the demultiplexer routes one input to one of multiple output data streams. For episode transcripts, worksheets, links, or other podcast notes, please visit us at intermation.com where you will also find links to our Instagram, Twitter, Facebook, and Pinterest pages. Until the next episode, remember that while the scope of what makes a computer is immense, it's all just ones and zeros.