Episode 5.02 – NAND Logic

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. Our topics involve a bit of figuring, so it might help to keep a pencil and paper handy.

In our last episode, we covered three attributes of the sum-of-product expression including its general format and the benefits derived from it, a method to generate a truth table from a given sum-of-products expression, and lastly, a simple procedure we can use to create the proper sum-of-products expression from a given truth table. In this episode, we're going to look at a fourth attribute. While this last trait may appear to be no more than an interesting exercise in Boolean algebra, it turns out that it offers several benefits when we implement our SOP expressions with digital circuitry.

In Episode 4.08 – DeMorgan's Theorem, we showed how an inverter can be distributed across the inputs of an AND or an OR gate by flipping the operation. In other words, we can distribute an inverter from the output of an AND gate across all the gate's inputs as long as we change the gate operation to an OR. Similarly, we can distribute an inverter from the output of an OR gate across the gate's inputs as long as we change the gate operation to an AND.

We're going to use this property to replace the OR operation at the output of an SOP expression, and then use that replacement to show how any SOP expression can be realized using nothing but NAND gates…and the occasional inverter. Why do this? Well, recall from Episode 4.04 – NAND, NOR, and Exclusive-NOR Logic, with current technology, the NAND gate is the quickest, cheapest, and smallest logic device we have.

Start by picturing an OR gate. The truth table for this operation shows that a logic one is output for any row with a one in any of its input columns. Another way of saying this is that the output is a logic zero only when all the inputs equal zero. If at the output of this OR gate we connect two inverters in series, we keep the same truth table. Invert an inverse, and you go back to the original value, right?

Next, take the inverter that is closest to the output of the OR gate, and push it through the OR gate to distribute it to the inputs of the gate. By DeMorgan's Theorem, doing this will change the operation from OR to AND, and what we have is an AND gate with inverters at all of its inputs and at its output. What does the truth table look like for an AND gate with all its inputs inverted and its output inverted? First, remember that an AND gate with its output inverted is a NAND gate. What does the truth table for the NAND gate look like? Well, the NAND gate's output is the inverse of the AND gate's output, which means that the only time a NAND gate outputs a logic zero is when all its inputs are a logic one. Since all the inputs to our circuit are inverted, however, the only time this NAND gate circuit is going to output a logic zero is when all of the circuit inputs equal zero. That sounds an awful lot like the operation of an OR gate. Actually, it sounds exactly like an OR gate.

Now comes the fun part. Let's replace the OR gate of an SOP expression, that single gate that combines the outputs from all the products to create the final output, with this NAND gate circuit with inverted inputs. The circuit operates the same, it just looks a bit different. The input signals enter the AND gates that create the products. The output of these products each pass through an inverter before entering the NAND gate that generates the final output. Well, if the output of each of these AND gates is sent through an inverter, isn't that just another NAND gate?

By replacing the inverted AND gates that generate the products of an SOP expression with NAND gates and replacing the OR gate that sums the products together with a NAND gate, we have a circuit that's nothing but NAND gates! In fact, an inverter really is just a NAND gate with a single input, so even the inverters used to invert individual signals being input to the products are tiny NAND gates.
Since an SOP expression can be created for any truth table, and since any SOP expression can be implemented entirely with NAND gates, then we really only need one type of gate: a NAND gate! And how cool is it that that one gate is the quickest, cheapest, and smallest logic device we have? This trait of a NAND gate, where any Boolean operation can be implemented using only NAND gates, is referred to as functional completeness.

Our next step towards making digital circuits faster is to reduce the number of gates in the circuit and reduce the number of inputs to those gates. Later in this series, we will learn about a graphical method that generates the most simplified sum-of-products expression from a truth table without all of that messing about with the properties of Boolean simplification.

In our next episode, we're going to take a break from the sum-of-products to discuss a second standard form of Boolean expressions: the product-of-sums. For this type of expression, we're going to be looking for the zeros in the truth table, not the ones. 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.