Episode 4.02 – Truth Tables

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, using rule-based algorithmic descriptions, we described the operation of four different logic gates: the AND gate, the OR gate, the exclusive-OR gate, and the inverter or NOT gate. Describing a gate's operation this way is not efficient and is prone to misinterpretation. What we need is a method to show the output of a logic circuit based on each of the possible input patterns of ones and zeros.

A truth table serves this purpose by making a column for each of the inputs to a logic circuit and a column for the resulting output. The input columns, typically labeled with capital letters, A, B, C, and so on, are on the left side of the table while the output, typically labeled X, is the rightmost column. Beneath the heading labels, a row is added for each possible pattern of ones and zeros that could be input to the circuit. For example, a circuit with three inputs, A, B, and C, would have two to the third power or eight rows beneath the heading, one for each possible pattern of ones and zeros. We identify these rows with the different values for A, B, and C of 0-0-0, 0-0-1, 0-1-0, 0-1-1, 1-0-0, 1-0-1, 1-1-0, and 1-1-1.

For some learners, the hardest part of creating a truth table is being able to enumerate the possible patterns of ones and zeros for the inputs. It helps first to recognize that if we use n to represent the number of inputs, there must be two raised to the n-th power different patterns of input values. Therefore, we know a row is missing if the truth table has less than 2^n rows or that we have duplicate patterns if there are more than 2^n rows.

There is also a trick to deriving the combinations. Assume we need to build a truth table with four inputs, A, B, C, and D. Since 2^4 equals sixteen, we know that there will be sixteen rows representing the possible combinations of ones and zeros. For half of those combinations, A will equal zero, and for the other half, A will equal one. When A equals zero, the remaining three inputs, B, C, and D, will themselves go through every possible combination of ones and zeros for the three inputs. Three inputs have 2^3 or eight patterns, which coincidentally, is half of 16. For half of the eight combinations, B will equal zero, and for the other half, B will equal one. For the top four rows where both A and B equal zero, C and D will go through each of their possible combinations of ones and zeros, half of which will have C equal to zero and half of which will have C equal to one. The same will be true for the next four rows where A equals zero and B equals one.

By repeating this process through all sixteen rows, we begin to see a method whereby we can create a truth table for four inputs. Beginning with the A column, list eight zeros followed by eight ones. Half of eight is four, so in the B column write four zeros followed by four ones in the rows where A equals zero, and then write four zeros followed by four ones in the rows where A equals one. Half of four equals two,

so the C column will have two zeros followed by two ones followed by two zeros then two ones and so on until we get to the last row. The last column, the column representing D, should have alternating ones and zeros. If done properly, the first row should have all zeros and the last row should have all ones.

Now let's use truth tables to describe the functions of the four basic logic gates beginning with the inverter. The inverter has a single input. Therefore, there is one column for an input, A, and one column for an output, X. For the single input, A, there are exactly two possible values: logic one and logic zero. Therefore, there will be two rows of values for the inverter truth table. In the first row, where A equals zero, the output X column will have a one. In the second row, where A equals one, the output X will be zero.

Now let's move onto the AND gate. Remember that an AND gate outputs a logic one only if all of its inputs are logic one. When making the truth table for the AND gate, the output for every row except the last one is zero. The output for the last row, where all of the inputs are one, is the only row with a one in the X column.

Now for the OR gate. The output of an OR gate is set to logic one if any of its inputs equal one. Another way of saying this is that the output of an OR gate is a zero for exactly one row of the truth table, the row where all of the inputs are zero. If written as described above, the truth table for the OR gate will have a zero for X on the top row and ones for X everywhere else.

The exclusive-OR gate's output is set to logic one if there are an odd number of ones at the input to the circuit. Remember that for a two-input exclusive-OR gate, that means that if the inputs are equal, the output is a zero. If the inputs are different, the output is a one. That means the two-input truth table for an exclusive-OR gate will have zeros for X on the top and bottom rows where A and B are the same, and ones for X on the middle two rows where A and B are different.

Trying to describe the truth table for the exclusive-OR gate with three or more inputs is a little difficult as the pattern becomes a little less predictable. For three inputs, the top half of the table, where A equals zero, looks exactly like the two-input exclusive-OR table: 0-1-1-0. The bottom half, where A equals one, is the inverse of the top half: 1-0-0-1. That means that as you go down the rows of the three-input truth table from the top where A=0, B=0, and C=0 to the bottom row where A=1, B=1, and C=1, the pattern going down the X column is 0-1-1-0-1-0-0-1.

For four inputs, there are sixteen rows. The top eight rows, where A equals zero, looks exactly like the three-input exclusive-OR table: 0-1-1-0-1-0-0-1. The bottom half, where A equals one, is the inverse of the top half: 1-0-0-1-0-1-1-0. This process repeats for any number of inputs to an exclusive-OR gate. It is interesting to note the effect caused by adding an input to the exclusive-OR gate. When the additional input is equal to zero, the exclusive-OR gate has the same output as if the new input didn't exist. When the additional input is equal to one, the exclusive-OR gate's output is inverted when compared to the output for the gate with one fewer inputs. In the two input exclusive-OR, for example, the output, X, in the top two rows where A equals zero, simply follows the value of B. In the bottom two rows, where A equals one, X is the inverse of B.

In our next episode, we're going to start putting some of these gates together in order to create circuits that are more complex. The results may be absurd when we try to do this with an audio podcast, but who knows, we may hit it out of the park. For transcripts, links, or other podcast notes, please check us

out 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.