Episode 4.01 – Intro to Logic Gates

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.

Representing numbers using transistors is one thing, but getting the computer to do something useful with those numbers is an entirely different matter. The computer uses digital circuitry to perform operations such as addition or multiplication, manage data, or execute programs. This episode presents the fundamental components used to create this circuitry and so much more.

Unless you are an electrical engineer, an understanding of the operation of transistors is unnecessary. One level above the transistors, however, is a set of basic building blocks for digital circuitry. These building blocks are called logic gates, and it is at this level where we will begin our discussion.

The basic logic gate takes one or more binary signals as inputs, and using a well-defined algorithm, generates a single binary output. Each time the inputs change, the output bit changes in a predictable fashion. For example, the algorithm for a specific gate may cause a one to be output if an odd number of ones are present at the gate's input and a zero to be output if an even number of ones is present.

There are several standard gates, each of which has a unique symbol that can be used to create diagrams representing the interconnections between many gates. If you're listening to the audio version of this episode, well, let's just say we're going to do our best to describe each of these symbols. As we describe the symbols, please note that the inputs will be coming into the gate symbol from the left side with the single output exiting the symbol from the right side.

Let's begin with the NOT gate. This logic gate, sometimes referred to as an inverter, is the only gate described here that has a single input. The symbol is a triangle pointing toward the right with a small circle placed at the right tip of the triangle. The single input goes into the flat vertical edge on the left side of the triangle symbol while its output exits the gate through the small circle at the opposite tip. Note that it is the small circle that represents the operation of this gate, so it should not be left out. In fact, later in this series we will see how the triangle is optional, and the circle can be placed at the inputs or outputs of other gates to indicate that an inverter has been placed there.

The inverter is used to flip the value of a binary signal. In other words, it changes a logic 1 input to a logic 0 while it changes a logic 0 input to a logic 1. An automatic light sensor used to turn on a porch light when the sun goes down might serve as an analogy here. During the daylight hours, sunshine enters the light sensor acting as a logic 1 input. Since it is daytime, the porch light needs to be turned off, a logic 0. When the sun goes down and no light enters the light sensor, this logic 0 input results in a logic 1 output and the porch light is turned on.

The next gate we're presenting here is the AND gate. In linguistics, the word "and" is a conjunction, and is used to join words or phrases together. In digital circuitry, the AND operation joins binary signals together and outputs a logic one only if all its inputs are logic one. If any input is a logic zero, the output is a logic zero. For example, if an AND gate has three inputs, A, B, and C, then the output is a one only if A is a one and B is a one and C is a one. See the conjunction?

Let's see if we can come up with an analogy for the AND gate. Picture a lamp that is connected to a plug in the wall that is controlled by the light switch that is protected with a circuit breaker. For the lamp to be on (logic 1), the switch at the lamp must be on and the wall switch must be on and the circuit breaker must be on. If any of the switches turns to off (logic 0), then the lamp will turn off.

We can see from this analogy that the AND operation is really like a series circuit. For a connection to be made across a series of connected switches, all the switches must be closed. Some mathematicians have another way of looking at it. The AND operation behaves like the minimum function. The output of the AND gate is equal to the minimum of its inputs. In the case of binary, the only way the output can be one is if none of the inputs are zero.

What does an AND gate look like? Well, it sort of looks like a capital 'D' in Arial font. Two or more inputs will enter the gate on the flat left side of the D while the single output will exit from the rightmost tip of the curved portion.

Now for the third gate, the OR gate. Like the word "and", "or" is a conjunction joining words or phrases in a sentence. In digital circuitry, the OR operation joins binary signals together such that the output is a logic one if any of its inputs are logic one. The only way the OR gate outputs a logic zero is if all its inputs are logic zero. For example, if an OR gate has three inputs, A, B, and C, then the output is a one if A is a one or if B is a one or if C is a one. Once again, we have a conjunction.

A security system can serve as an analogy for an OR gate. Assume that a room is protected by a system that watches three inputs: a door open sensor, a glass break sensor, and a motion sensor. If none of these sensors detects a break-in condition, in other words, they all send a logic zero to the OR gate, the alarm is off (logic zero). If any of the sensors is tripped, it will send a logic one to the OR gate which in turn will output a logic one to indicate an alarm. It doesn't matter what the other sensors are reading, if any sensor sends a logic one to the OR gate, the alarm should be going off. Another way to describe the operation of this circuit might be to say, "The alarm goes off if the door opens or the glass breaks or motion is detected." Once again, the use of the conjunction "or" suggests that this circuit should be implemented with an OR gate.

The operation of an OR is like a parallel circuit. Picture a set of switches oriented horizontally where all of the left side contacts are connected and all of the right-side contacts are connected. In a circuit like this, a signal can get from the left side to the right side through any of the switches. This is a parallel circuit. If any of the switches is closed, a connection is made from one side to the other. Mathematically, the OR operation can be described as behaving like the maximum function. The output of the OR gate is equal to the maximum of its inputs. In the case of binary, any input equal to logic one will set the output to a logic one. The only way the output can be a zero is if all the inputs are zero.

For our listening audience, describing how an OR gate is drawn can be a bit tricky. Picture the flat left edge of the AND gate where the inputs enter. On an OR gate, this edge is concave curving inward

towards the center of the gate. The right side of the gate comes to a point instead of a curve where the output exits the gate.

We're almost done with our gates. The last gate we want to present here is the exclusive-OR gate. The Exclusive-OR gate, sometimes referred to as the XOR gate, counts the number of ones at its inputs and outputs a logic one if the count is odd and a logic zero if the count is even.

For two inputs, the XOR gate behaves like a compare circuit. If two binary signals are compared, and they are equal, the XOR gate will output a logic zero. In other words, if both bits are zero, then there are zero ones at the inputs. Since zero is an even number, the XOR gate outputs a zero. If both bits are one, then there are two ones at the inputs. This means the XOR gate will also output a zero. If the bits are different, then only one one exists at the inputs, and the XOR gate outputs a one. If we connect an inverter to the output of this two-input XOR gate, then the comparator will output a logic one for binary inputs that are the same and a logic zero for binary outputs that are different.

If we have two or more inputs to our XOR gate, then the gate becomes an even parity generator. An even parity generator is a circuit that receives multiple input bits and generates an additional bit that when combined with the input bits results in a total number of ones that is always even. This additional bit is referred to as an even parity bit. If we connect an inverter to the output of the XOR gate, our circuit becomes an odd parity generator. In this case, the combination of all the inputs along with the odd parity bit will result in a number that has an odd number of ones. These parity bits are often used in circuits designed to defect errors.

Now for the shape of the exclusive OR gate. The shape is almost identical to the shape of the OR gate except that the concave curve at the inputs is a pair of side-by-side concave lines with the inputs going into the outer line.

Later, as we expand our use of logic gates by combining them into complex circuits, trying to describe their operation using their algorithmic definition will become tedious. In our next episode, we will introduce an easier method for representing the operation of any digital circuit incorporating the NOT, AND, OR, and XOR gates. 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.