PROFESSOR: Propositional operators play a basic role in the design of digital circuitry, and we're going to illustrate that in the section by designing a little binary addition circuit.

So let's begin with a review of binary notation and addition in binary.

So the way binary works is like decimal. Except instead of using powers of 10, you're using powers of 2. So here is the binary representation of the number 39. The way to understand that is this is the ones place. That's the twos place. That's the fours place.

So 1 plus 2 plus 4 is 7. Then this is the eighth place with nothing, this is the 16th place with something, and this is the 32 place with 1. So we get 32 to 7, and get 39.

Likewise, the binary representation of 28 is 011100. I'll let you check how that works with contributing 1, 2, 4, 8, 16, and 32.

And finally, let's add these two numbers in binary. Now, binary addition works just like decimal addition except that the only numbers are ones and zeros so that when you get 1 plus 1, you have to carry 1. Let's do that.

So 1 plus 0 is 1. That fills in the first column. Now we have another 1 plus 0 is 1. That's fine.

Now we have a 1 plus 1. And that's going do a 0 here and contribute a carry of 1 to the next column. Now, the next column has two ones. So it becomes a 0 and contributes to another carrying. Now we have two ones. We get a 0 and contribute another carry. And now we have two ones, and we finally get a 1 0.

So this is the binary representation of the sum, and you can check that this is 1 plus 2 is 3 plus 64. So the answer should be 67. And you can check that it is. So that's how binary addition works.

So now let's try to design a bit of circuitry using digital logic signals of 0 and 1, which will do addition. And so we're going to try to design a little six bit binary addition circuit. So I'm going to have as inputs, the six digits of the first binary number-- a 5 down through a 0 and then the second binary number. Let's call it b 0 through b 5.

So these are two binary numbers that are six digits long, and I'm going to add them up by

thinking of a 1 as a 0 or 1 signal. a 0 is a 0 1 signal. b 0 is a 0 1 signal.

And these can be transmitted down wires into some boxes that contain digital operators that will cause the right signals to come out. And what we want to come out of here is the possibly seven digit representation of their binary sum.

So d 0 is the sum of a 0 and b 0-- the lower digit possibly with carry and so on.

And then c 5 if the sum of two six digit numbers runs to seven digits, which it might as we saw in the previous example, then c 5 would become 1, otherwise 0.

So this is the specification. I want a and b to come in, and I want their binary sum to come out as d's with a high order c if need be.

Now, the way I'm going to do that is it's clear that the behavior of the inputs for a and b, which produced the lower digit, might produce a carry, and that carry has to be transmitted to the next column if it exists. So I'm going to need a wire that sends a 0 1 signal from this box over to that one that carries 0 or 1 and likewise for all of the others.

So this is the kind of basic structure of my binary addition circuit. This is called a ripple carry organization. It's mimicking exactly the way that we added up to two numbers column by column, possibly propagating carry of 0 or 1-- or really a carry of just 1 to the next column. And I've got all the wires in place that I need. What we need to do is design the digital circuitry that's in those boxes.

Well, this box is different from the others because it's only got two inputs. All the others have three inputs. So the three input boxes, we'll call full adders and the two input boxes a half an adder.

And the specification of a half an adder, again, is that the output is the binary representation of a 0 plus b 0. So it's a two digit binary representation-- never be bigger than 2 because there's only two numbers.

The output of a full adder is it gets three inputs in this case-- b 1, a 1. And the carry, c 0. And it produces the binary representation of the sum of those three numbers, which is a two digit binary representation that might be anything from 0 to 3. OK.

Well, let's start with the easy case. What's a half adder? Well, a half adder, again, has inputs b

and a, and it's supposed to produce as output the binary representation of b plus a. So d is the lower digit in the zeros place, and c is the high order digit, namely the twos place.

Well, what does that look like? Well, here's the circuit. This is the digital designer symbol for an exclusive [INAUDIBLE] gate that returns.

So d is going to be the exclusive or of a and b according to this pictorial diagram. Notice I'm using this colon colon equal symbol which is convenient as a reminder that I'm defining the thing on the left. You could replace it by equal, but it's informative to realize that it's not an equality that you've proved or that some derivation of the two interesting things are proven to be equal, but rather that I'm just defining what the d is.

This output d is defined to be a XOR b.

And likewise, this is an AND gate. So the output c a AND b.

And let's check that. The low order digit is definitely the [INAUDIBLE] sum, XOR of a and b.

And when is there a carry? Well, the only way there's a carry is when the value is 2, in which case the output c would be 1 and d would be 0. And that's exactly when both a and b are 1, that is c is a and b.

So that's a half adder. That was easy.

Well, a full adder looks like this. It's a little bit more complicated, and I'm going to write out the equations without trying to justify them completely. But I need a name in order to describe this with propositional operators. I need a name for that important signal. Call it s, which is what we were not calling it in the previous one.

But now this is a half adder with inputs a and b and outputs s, which is a XOR b and another output here, which we know is just going to be a and b. OK.

How do I express this set of connections as formulas? Well, first of all, s is the output of this first half adder, which is a XOR b. OK.

The output d I get by taking s, and it's the first output of the second half adder, which means it's c in XOR or s. That's easy.

And what about c out? Well, c out is getting-- this is an OR gate by the way. So c out is going

to be an OR of what comes out of this half adder, which is c in and c s and OR with the output of this half adder, which is just a and b.

So there are a bunch of equations that completely characterize the structure of this little bit of digital logic and how it is wired up and fits together.

Now, let's go back to describing our ripple carry circuit of what was going on here. Now that we have the equations that characterize the behavior of these full adders and half adders, I can explain to you what the formulas are for all of these outputs-- the c's and the d's. And that goes as follows.

So the first one, looking at this half adder with a 0, b 0 coming in and c 0, d 0 coming out, I know that d 0 is a 0 XOR b 0 and c 0 is a 0 AND b0.

That's just that the formulas that we have for the half adder when the inputs are a 0 and b 0 and I call the outputs b 0 and c 0.

Now, the more general case of the full adder-- what's coming in here is an a and a b with the same subscript-- a i and b i. And what's coming out is the ith digit of the binary sum, d i and the carry c i.

And I could describe those just by using the formulas for the full adder. So what it means is that I'm going to introduce a new convening variable, s i, which I'm going to define to be a i or XOR b i.

D i is then going to be c i minus 1-- the carry from the previous place-- XOR with s i. And the new carry, c i, is going to be the output of the first half adder, which is c i minus 1 AND s i or the output of the first half adder, which is a i and b i.

So the point is that I've just taken the wiring and translated it into equations like this, and you can see how these equations might be better to use than the particular way that you drew the picture with all the wires connected because the logical behavior of the circuit doesn't depend on how it's laid out. It just depends on these logical connectives between the values of these different variables.