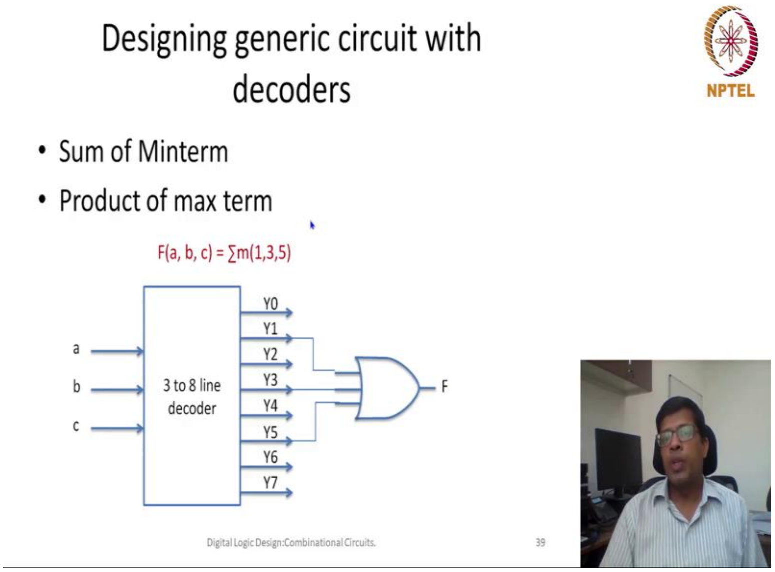
Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology, Ropar
Encoders

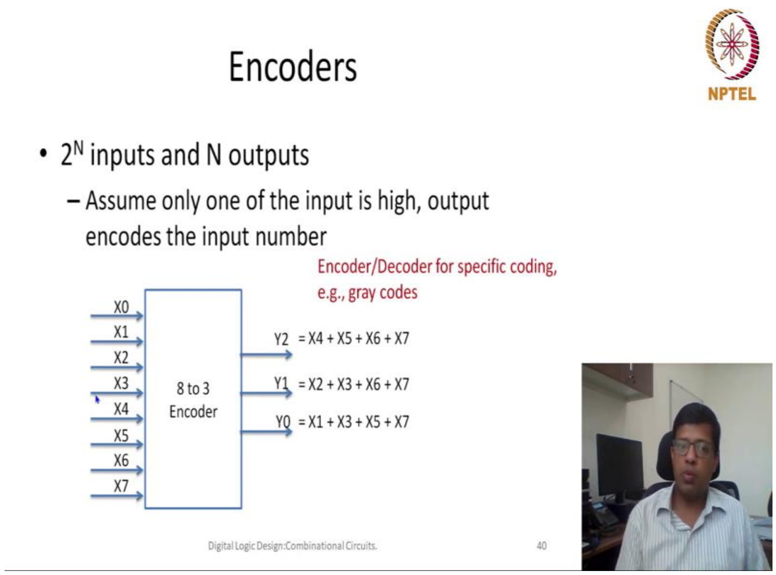
(Refer Slide Time: 00:14)



Now if you want to create a circuit which is kind of a inverse of encoder that means there are 2 to the power N inputs and N outputs. And we again like decoder; we again assume that only one of the input is 1 at one point of time. So if there is only one input is 1 at one point of time then what this encoder essentially means? This encoder will tell me which particular input is 1.

So essentially what we are doing is that we are saying that we have N inputs. All of them could be 0 or 1, only; of course only one of them is high at one time or active at one time. Now we want to see that out of those wires which one is 1. And that is essentially means that we are giving some number to all of them but they do not know these numbers are assigned to them. As a designer we would require it.

(Refer Slide Time: 1:21)



So I can write this like, I have these 8 inputs, and I am taking again example of this 8 to 3 encoder. Now there are eight inputs, and these eight inputs any one of them could be 1. Now we are giving these numbers, decimal numbers to these inputs for example. Decimal numbers means this input we are calling the 0, 1, 2, 3, 4, 5, 6, 7. The question is that which particular input is 1 I want to know. I have assigned some ID to it. I have assigned some number to it. I want to know which particular number input or which particular ID of the input is 1. This is a question I am solving.

So then I can create an encoder. It is kind of a reverse functionality of whatever decoder was offering. So in encoder only one of them is 1 and so I will have three outputs because the number of outputs is going to be in log N. So if my inputs are N then Log N is the number of outputs. So now how to find out the output for all of them? So one, one of the thing is that we can, we have to create probably K-map where I have eight inputs. And, but I have not taught you. You do not know how to create K-map with 8 inputs. Similarly we have been taught, we have been told clearly that if you are going to create, you are going to solve K-map with 8 inputs it is going to take like take lifetime.

So how to do things? So here the good thing is that we know only one of them is 1 at one time. Because only one of them is 1 at one time so what I can say? So when this Y0 would be 1, or let us say when Y2 would be 1? Y2 would be 1 when X4, X5 ,X6, or X7, any of them is 1 then I can say that yes my Y2 would be 1. So I can keep this equation like this because here the assumption is that I know that all of them cannot be 1 at the same time. Only one of them would be 1.

Similarly for Y1 I know that it would 1 only if the condition is either X2 is 1 or X3 is 1 or, X6 or X7 is 1. So I can put this condition like this. Similarly Y0 means all the odd values 1, 3, 5, 7. So we can create these equations for this encoder. Now this was kind of a, where we have assigned decimal numbers to our input IDs. Now instead of decimal numbers if we assign some other numbers like Gray code, BCD code, XS3 code then also we can, we can design our encoder and the encoder like the output, for example here Y0 Y1 Y2 we calculated. We see that what was the corresponding ID and then we calculated. So basically it was drawn from the truth table and from the truth table we took an idea that what should be the value of the output.

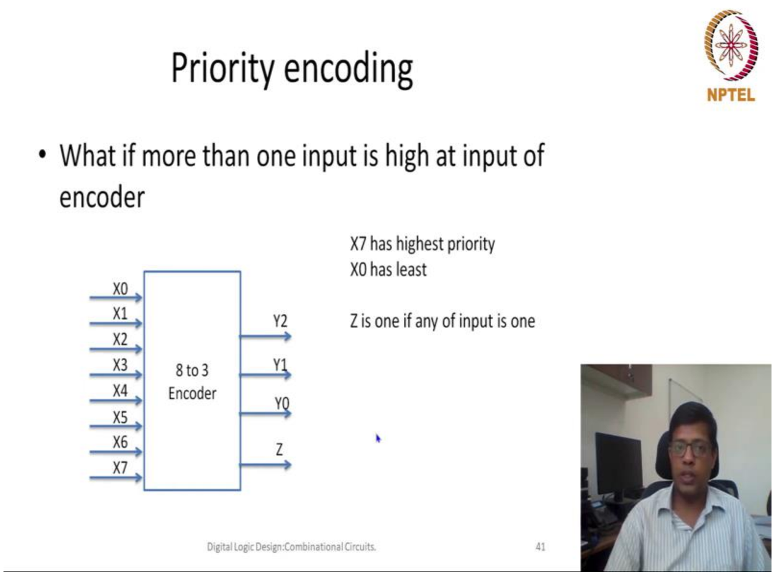
So if the code is something else rather than binary, rather than decimal system then it could be, let us say Gray code, XS3 code, BCD code. Then we can create these equations accordingly. And reverse is also true. For example, these decoders; do not assume that decoder is always going to be, always going to give us these decimal number 0, 1, 2, 3, 4, 5, 7, 8, 9, 10. It could be in terms of a Gray code or in terms of XS3 code, any other code, correct. So this is quite a generic thing.

Now these encoders, how and where do we use them? So one, one quick example could be a sort of interrupt-based system. So you can ask what is interrupt. So interrupt means, let us say you will see that whenever you plug in any particular USB device computer gets to know that this particular device is a Nokia phone, is a mouse or is a keyboard. So, and similarly like some of you have done Arduino-based design or Raspberry Pi based designs so there also there could be multiple devices which could be connected to your Arduino chip.

Now when we want to connect multiple devices we should know which particular device is sending interrupt or which particular device is 1 at a time. Then I will assign IDs to all of these devices and we will want to know finally that which particular device is seeking attention. Then we have to use certain kind of, such kind of encoders. There would be certain more application, so based on the application, based on the statement, problem statement you can find, you can map that whether this problem looks like encoder problem or not, and then you can do it.

Now one more question here. So let us say, even for this interrupt-based scenario or even this scenario what would happen if multiple of these 1s are, multiples of these inputs are 1, correct? So you are seeing that all of these inputs are connected to different devices. And one device would be independent of the other device. If devices are independent to each other then how can we control that only one of the device has to send 1? There is the possibility that multiple devices are sending 1.

(Refer Slide Time: 07:50)



If multiple devices are sending 1 or essentially multiple inputs are high at one time what should we do? We have to see that we have to take 1. So out of them I have to encode one of them. That is clearly good solution. So let us say, out of these 8 inputs, 4, 5 and 7 all of them were 1. Which should I give as output, whether 4 or 5 or 7? Now this can be done using assigning some sort of a priority. I say that if 4, 5 and 7 all three of them are 1 at same time then whoever will have higher priority, I will make that as assignment.

So let us try to map it to real life? So you know you have a, let us say some sort of Arduino board or some kind of a computer where there are four devices which are seeking attention. There is a computer. Arduino board is let me say is a computer. Now you have a computer or you have a processor chip. Now there are multiple devices which are seeking attention.

Let me give you an example. One of the devices keyboard, the another one is, let us say printer. And the third one is a very high speed device. So which one should you give more preference?

The one because computer will work very, very fast; we are very, very slow, so whatever is slower probably could be given lower priority, and whatever is the fast, faster one could be given higher priority. That could be one criteria.

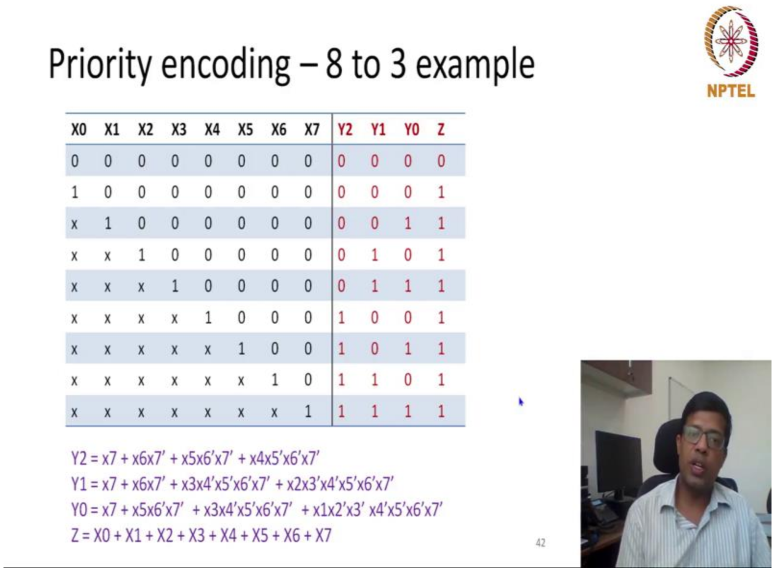
There could be other criterias also. That you know that this particular event or this particular device is the most important one. It cannot miss; it cannot wait for even a single cycle. In that case that priority would be the higher one or highest one. So similarly we can arrange those devices in that priority list and we can say, let us say, we say here that, with the same example, so we say that let us say, this, so let us say X7 is the highest priority and X0 is the least priority. So the priority is set in the order. And if I have 8 devices these 8 devices are connected in such a way that the highest priority device is connected to X7 input and the least priority device is connected to X0 input.

Now my Y0, Y1, Y2 can tell me that which particular device is 1. So here is one issue, one additional issue here that what would happen if none of them is, none of them is 1? If none of them is 1 then what would it encode into? In our 8 to 3 encoder previously our definition was that at least one of them has to be 1. Here we are making an exemption to both of these conditions. More than one input could be 1. The other is also true that none of them could be 1. All of them could be 0.

So because we are taking two conditions so one could be solved by priority, we are prioritizing if more than one are, more than one input is high then we can solve it using priority. But if all of them are 0 then we have to have additional output. This output will say that at least one of them is 1. If all of them are 0 then this Z will also be 0.

So I would require an additional output in this priority encoder case. This additional output would also be required if there is a possible case in my 8 to 3 encoder that all of the input could be 0. Then I would require this additional output to confirm that, yes this is the case when all of them are 0, which has a side meaning also. So as side meaning, this means that this Y0, Y1, Y2 is available or are the output only if Z is 1. If Z is 0 then they does not have any meaning. So sort of, this would work as enable for any of the input device for this 8 to 3 encoder.

(Refer Slide Time: 12:43)



So now I want to design this circuit. How will I go about it? So then I need to write a, I need to write a truth table for this. And truth table would look something like this, which means that if X7 has the highest priority is 1 and my any other input is, whatever it is, all of them could be 1 it does not matter, if X7 is 1 the output would be 1 1 1. Similarly if X6 is 1, if X6 is 1 and X7 is also 1 then this will come into this case.

So that means that when X6 will take the output only if X7 is 0. And all of others could be Do Not Care that means they could be 1, they could be 0. It does not matter. Then the output is going to be 1 1 0. Similarly for X5 to be the output all other could be Do Not Care but X6 and X7 has to be 0. Similarly all other cases. So if all of them are 0 then my Z is also 0 and Y0, Y1, Y2 is also 0. Actually they are Do Not Care but yeah to make it simple we have said so.

Now the output Z would be 1 in all other cases because at least one of the input is 1. Now if this is the scenario then how will I write the function or expression for Y0, Y1, Y2? In previous case when I was designing 8 to 3 encoder where only one of the output, one of the input was 1 then I make a OR statement and it was easy. But here many of these output, inputs are 1. I have to probably write a K-map sort of thing.

Now there could be other method of solving it. But one of the straight method is that we can create a Boolean expression directly from looking at the truth table. So for example I want to write what is the Boolean expression for Y2. So I can see when this Y2 is 1. Y2 is 1 in these four rows. So I can say that when this condition is 1 or when this condition is 1 or when this condition is 1, when this condition is 1 then Y2 is going to be 1. So I can look at all the input in these last 4 rows and then create an Boolean expression for Y2.

So it would come something like this. That Y2 would be 1. So this particular 1 would come if X7 is 1 and this particular 1 will come when X7 is 0. That means X7 is X7 bar and X6. Similarly this one will come when X7 is 0, X6 is 0 and X5 is 1 equivalent to this expression. So this condition would be achieved when X7 is 0, X6 is 0, X5 is 0, X4 is 1. So I can write this expression for Y2.

So wherever 1 is there corresponding to that, whatever are the input, based on that I have created these expressions. All these Do Not Care because they are not going to impact my output so I have simply left it. Similarly I can create Boolean expression of Y1 and Y0 also. So Y1 would depend on wherever this output Y1 is 1. So wherever is 1 corresponding to those inputs I can create my product term. Similarly corresponding to this one I can create my product term and, yeah.

So it is not that it is only sum of products is possible. Product of sums is also possible. Wherever I have 0, corresponding to those 0s I can create my Boolean function. What about Z? Z is, if I can take a simple OR of all these; X0 OR X1 OR X2 OR X3 OR X4 OR X5 OR X6 OR X7. So all these ORs I can take to tell me that this is the expression for (())(16:47).

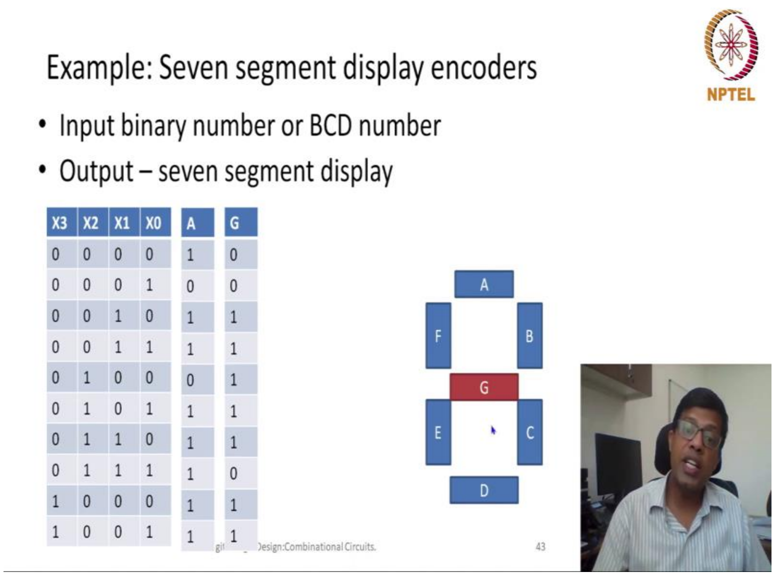
So let me repeat. The way we are calculating this expression, because we do not know what, we know that K-maps cannot be solved for 8 variables; Q-M is also going to be difficult. Now from the truth table itself we have created, from the truth table itself we have created the Boolean expressions. From the, wherever the output is 1, wherever Y2 is 1 so corresponding to that 1 what are the input combination, that form one product term.

So here it says, for example this 1 is corresponding to this input expression. That means X4 has to be 1. X5 has to be 0. X6 has to be 0. X7 has to be 0. So that means this particular product term. And because all of these terms will not impact so they will all become Do Not Care. When inputs are at Do Not Care the implication is different, when output is at Do Not Care the implication is different. So when input is at Do Not Care which means that it will finally get optimized away.

The expression, so for example this, this particular expression, this fourth one, so this expression, even this X3 value is 1 or 0 it is true that output is going to this. So that means it will get optimized away. Finally when we write a K-map or Q-M sort of method then, because X3 as well as X3 dash both are part of this input pattern so it is going to be optimized away.

So that is why these Xs has a different meaning than if we have X or Do Not Care condition at the output. Now similar to this, if there is any other priority given we have to create a circuit for this. We have to optimize. We have to find out what would be the Boolean equations and based on those Boolean equations we will try to design our expressions and from there we will get to know what is the output.

(Refer Slide Time: 19:00)



So let us take one more example of this encoding. So this is also quite a popular thing. In hardware you will see that sometime we use this seven segment encoder, seven segment displays. So this seven segment display will have, will take some number as input and it will produce output which is a seven segment. So seven segment looks like something like this. It has 7 LEDs. In standard circuits mostly there are 8 LEDs. The other, 8th LED represent a dot here so that we can represent any kind of fractional numbers also.

So let us still see it as a seven segment. So in seven segment we have these 7 LEDs. All of them we mark as some number, name, A, B, C, D, E, F, G. Using these seven segments you can see we can write all our decimal numbers, basically 1, we can say 2, we can say 3, we can say 4, 5, 6, 7, 8, 9, 0. And even we can write A, B, C, D, E, F also. A, we can write where D would be 0. D would be lit off and all of them would be lit on. B is very similar to 8. So that is why instead of writing B in capital we can use b in small so that it is, it can be distinguished easily. So in b we will lit only F, E, G, C and D while whenever we are writing 6 this b could also look very similar to 6. But in 6 we will have A also lit or enabled.

So similarly we can write C also, we can write D. In D also, D would again look like 0 so d also we can write in small where B, C, G, E and D all these would be lit. E also we can write, F also we can write, correct? So G is one of the letter which cannot be written but many other letters could be written. H is there. I, I and 1 again we will have some confusion. So, yes. So we can, i we can write either using only C or only E. So many such letters could be written using this seven segment display.

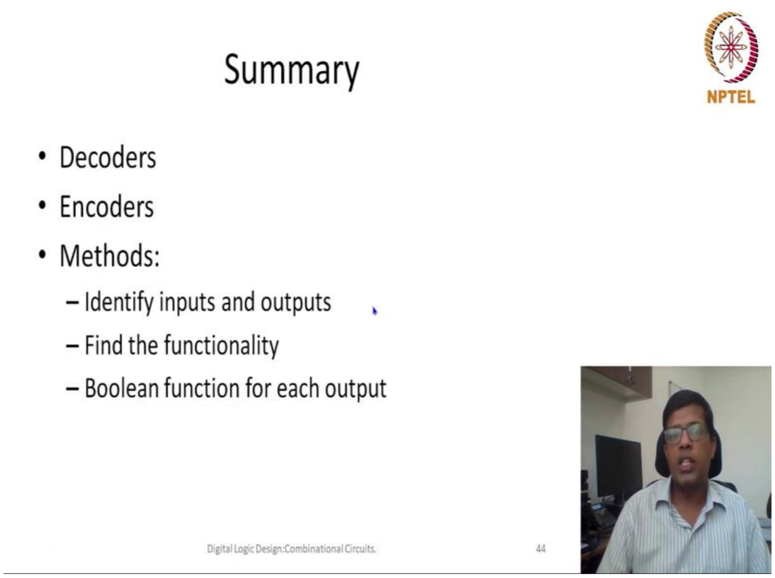
Let us say that this seven segment display is used to display eight, ten digits we have, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Then this is also sort of encoder because all of these input would finally, would lit any of these LEDs. So the input is, let us say X which is a four bit, X0 X1 X2 X3 and I, while writing the truth table I have written the number from 0 to 9. After 9; 10, 11, 12, 13, 14, 15 I would say that because we are considering only BCD numbers then we can simply ignore them. Or we can use them as Do Not Care conditions to make our output efficient.

So if I want to, what this encoder would do, it will try to know what are the values of A, B, C, D, E, F, G. So basically these seven values corresponding to my four inputs. Now all of these seven inputs have to be decided one by one. So for example, I need to know what would be the value of A. So for A I need to see from 0 to 9 where A will be lit as, will be on.

So then I will write the truth table for A. I know that it would be true for 1, true for 0 but for 1 it is not true. 2 also it is true. 3 also it is true. 4 it is not because 4 would be something like this. A is not there. 5 also it is there. 6 also it is there. 7 also it is there. 8 also it is there. 9 also it is there. So if I know this I can create a four input, four variable K-map and then I can optimize for this A and I would know how many, how to convert into a sum of product expression or product of sum expression.

Similarly for other inputs also, let us say for G also I want to make. Then I will create similar truth table for G also. G is 1 for; it is not 1 for 0. For 1 also it is not there. For 2 it is there. For 3 it is there. For 4 it is there. For 5 it is there. For 6 also it is there. For 7 it is not there. 8 it is there. And 9 also it is there. So this way for all the seven segments I need to generate what is the truth table and for each of these segments I need to optimize, write a truth table, then sorry, do K-map optimization and then finally solve the circuit so that I can have a optimized circuit for all of these A to G. So this is for BCD, this seven segment as I said can be used for other numbers also like hex numbers and even some of the A, B, C or alphabetical numbers this seven segment can be used.

(Refer Slide Time: 24:47)



So in summary what we can say is that today we have seen how, we have seen some variations about how decoders are designed, how decoders are used for hierarchical design or how this enable input can make this decoder like a demultiplexer. We have also seen that how decoder can be used to design any generic circuit with help of an AND gate or OR gate. And reverse functionality of decoder is encoder. We have seen this, how to design encoders and we have also observed that this decoder and encoder, although they are, we have seen in this particular lecture as a binary numbers but they could be some other encodings which are possible.

So for different encoding my circuit or my internal implementation of decoder and encoder would change. And if I am making any kind of a generic or any kind of known standard encoder or decoder, what I would do is I would try to identify what are my inputs, what are my output. What is the functionality? Or I will try to write a truth table. And if there are several outputs then for each output I will see what are the input in my truth table and corresponding to each one I can write sum of product expression or for each 0 I can write product of sum expression.

So this SOP or POS can be finely, if we want to optimize it further then we can optimize it using K-maps or Quine-McCluskey method and which will give us an efficient result. That is all for today. We will see you in the next lecture. Thank you very much.