Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 13
Combinational Design Decoders plus Encoders

Welcome to the third week of this course, I hope you have been joining the course so far. So, we are into the 3rd week and slowly we are getting into more sophisticated topics. So, in this week we will see lot of new things, so I hope that you have been able to spend some time, over the last 2 weeks on this course. So, there is a lot of activity on the forum asking questions and lots of things that is a good thing, so continue doing it and whenever you get struck on the forum, please email to the forum.

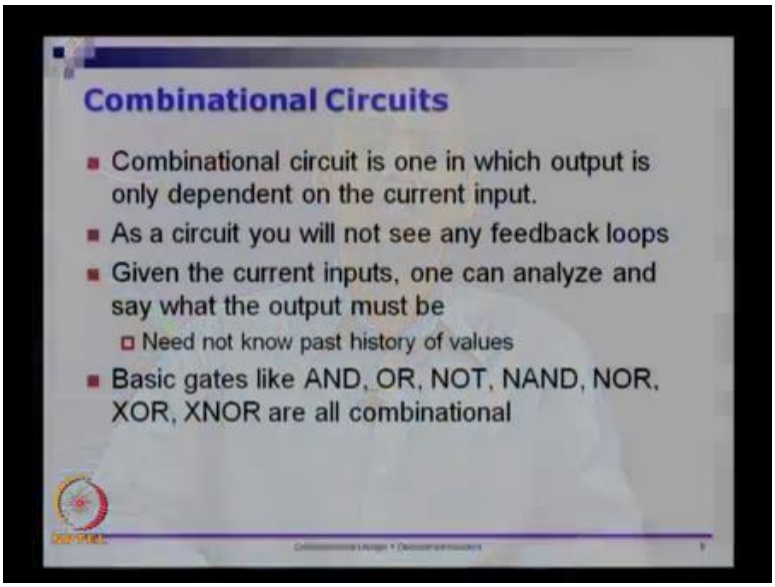
So, one small request that I have to all the students is that, if you are going to post anything new, just go and do a search on the forum first, if somebody as already posted on this topic. And if somebody is already posted, then just go and add to that specific email. Otherwise, what happens is we see too many emails and it is very hard to track whether all these things are replied back or not. So, I have been marking many of these things as duplicates over the last 2 weeks.

So, but this just a small request to all of you that you go and search for questions that you are asking on the forum first. If somebody is already ask that question, go and add to that discussion rather than opening a new forum new post. The other thing that I want is sometimes you are posting on the forum and the subject line has no relevance to the contents that you are posting. So, it is better that the contents that you have and the subject line of the emails should match. So, do not go and high jack another thread and go and ask question which is not relevant.

The third thing that I want to say is some of you have been answering some of the question form home works, write in the post itself, and those are deleted without you and informing you. So, this is not in the spirit of the course, so you can help to the level that people can understand the material, but giving away solutions is not acceptable on the forums.

So, I have been deleting some of these emails without our informing you about that. So, anyway given all the other things, I hope you are able to spend time most importantly and we are able to understand the material that is in the course. So, one another thing that you had noticed is, the videos are now available online. So, you can go and download the videos also now and that I have already posted in a forum. So, please take a look at that also. So, let us move on to this 3rd weeks lectures, in this week we are going to see combinational circuits. So, we will start with combinational design to begin with.

(Refer Slide Time: 02:58)



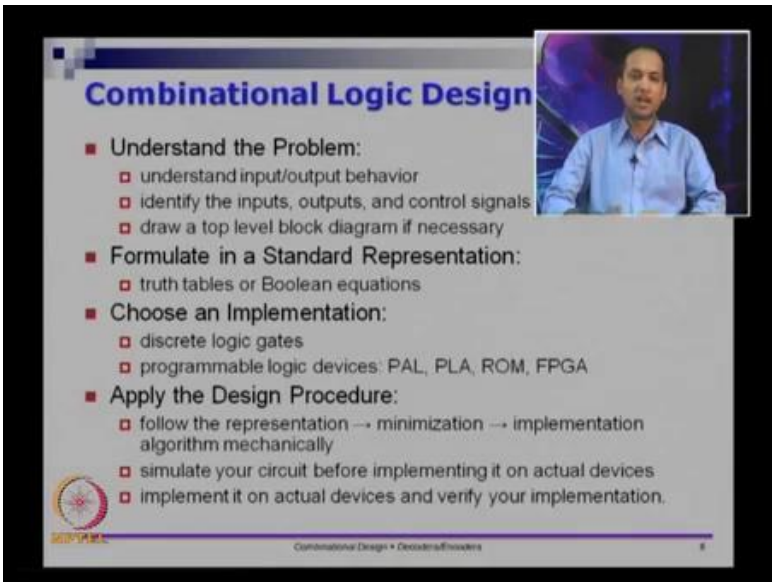
So, combinational circuits are essentially circuits in which output depends only on the current set of inputs. So, there is no notion of history that is associated with combinational circuits, the current output depends only on the current set of inputs, not what kind of inputs where given earlier and what is the state of the circuit earlier and so on. So, a combinational circuit is usually one in which there is no feedback path in this circuit itself.

You will never see an output of a gate connected to somewhere else which forms the input path. So, you will see inputs feeding a certain set of gates, from there you get another set of gates and so on all the way to the output. But, nothing that is going to feedback from output back to the input. Actually at any intermediate level, you will not see any wire that is connecting to somewhere in the previous steps.

So, all the basic gates that you have seen so far like AND gate, OR gate, NOT gate, XOR, XNOR all of those are actually combinational basic circuit blocks. There are all

combinational in the sense, you give inputs and you get current outputs, these from the basic gates for combinational designs.

(Refer Slide Time: 04:10)



So, let us see how to design a combinational logic circuit. The first thing we need to do is to be able to understand the problem. So, you are usually given problem in English description, so even though in the course I have been giving equations directly. In the real world, most of and you will be given a design problem which is given as English description, you first have to understand the input and output behavior of it and you identify what are the inputs, how many inputs are there and so on.

And draw a top level block diagram, if it will help you, so sometimes pictures are very useful to understand what we have supposed to do. Then, you go and formulate standard representation; you go and write either the truth tables or Boolean equations. So, whatever is comfortable for you do that and once you have done with it, then you can choose the implementation.

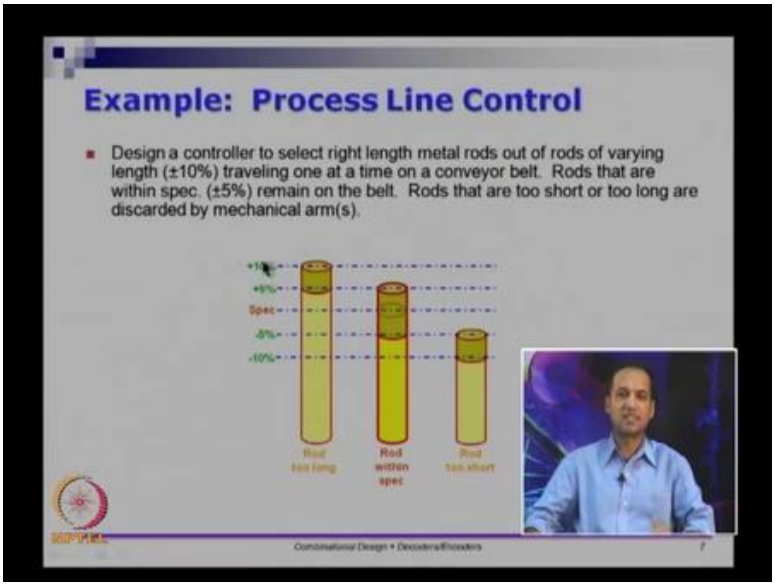
So, you can either choose discrete gates, so I am sure in a digital circuit labs**,** you would probably pickup 74 series chips and connect them together or if your labs have access to FBJ based devices, you are going to take the circuit that you are designed and put it down on the FBJ boards. So, either way you need to decide what kind of implementation you are going to do and finally, apply the following design procedure.

So, take the representation, minimize the level of hardware that you have and go and implement the algorithm, go and check the whether the circuit works correctly by doing

simulation and finally, implement it on the device. So, this course will teach you the first two bullets, namely how to take a problem, represent the problem, minimize it, implement it algorithmically. We will also see how to simulate and check whether the design is correct or not, for this we will use Verilog.

And the final implementation is not something that I am going to cover in the course. So, this is beyond the scope of this course. So, for that we actually need real hardware with which we can do things. So, let us take a small example problem, so this is a design problem, where it is a process line control problem.

(Refer Slide Time: 06:14)



So, imagine there are steel rods coming out from a steal plan and they are supposed to be of a certain length. Let us say 100 centimeter in length, that 1 meter length and what you want is the rods could be plus or minus 5 percent in length and that is acceptable for the engineering problem that you have. But, if it is beyond that you want to throw it away. So, the industrial process is good enough, it gives you rods which are not too big.

If you ask for 1 meter, it is not going to give you rod which is of 2 meters or let us say 0.5 meters and so on. The rods come in the range 110 centimeters to 90 centimeters. However, the acceptable range is let us says 95 centimeter to 105 centimeters. We want to be able to detect something like this and throw away the rods which are outside the range. So, this is the design problem. So, you want my... So, that is a margin of error plus or minus 5 percent is acceptable.

But, beyond 5 percent is not, either minus 5 percent or plus 5 percent beyond that is not

acceptable and we know that the rods are controlled and the rods are too smaller to big in length, they are within plus or minus 10 percent anywhere. So, what we need to do is, reject the plus 5 percent to plus 10 percent, also reject the minus 5 percent to minus 10 percent rods and we need to come up with the circuit for that.

So, this is the design problem and so imagine the set of rods like this, so let us represent the length of the rod along this vertical access and let us say this is the line from which you are measuring, imagine a line at the bottom from which you are measuring. And let us say this is the spec, whatever length that is some spec, so I said 100 centimeters, well let us keep it as 100 centimeters from here to here and so this plus or minus 5 percent are this band.

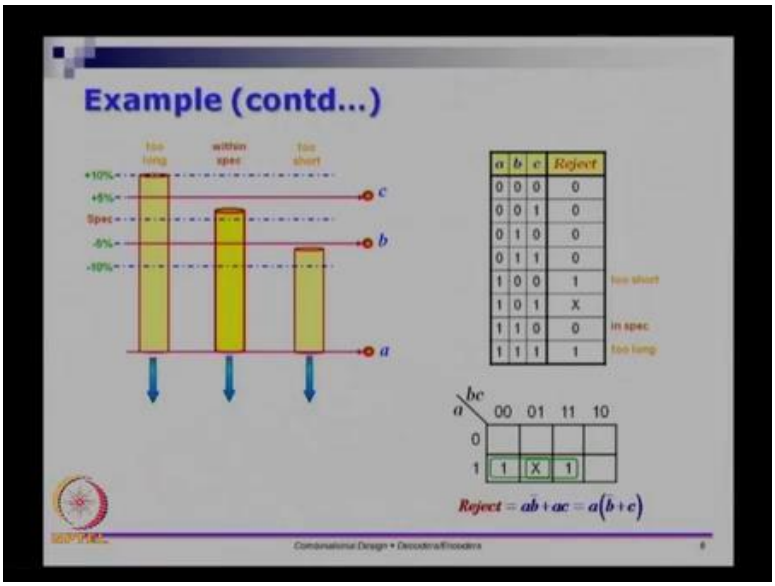
So, if it is plus 5 percent if you measure from here, this is 1 meter plus 5 percent, if you measure from here this is 1 meter minus 10 percent and so on and this one is 1 meter plus 10 percent. So, I know that no rod will be greater than this length, if I start measuring from here, it will not be more than 10 percent of the expected length. So, this rod is too long, because it is above 5 percent, this rod on the right most side is too short, because it is below 5 percent, whereas this rod here is within the spec.

So, you from this second line here, from the second line here to the fourth line here, if the rod is within that, it is within spec. So, I want to design a control process to do this. So, one way in which we can do this is, put senses here and put enough senses here to see whether this rod is of a specific length or not. So, imagine some sensor which is from this side to this side and let us say I shoot a beam from here to here, if there is a rod the beam is block, if there is no rod the beam is not block that is what I am going to do.

Similarly, here if the rod is sensed I am going to see something, if the rod is not sensed I am going to see something else and so on. So, I can put senses on two sides to check whether the rod is in this belt or not. So, I am going to imagine rods coming in belts like conveyor belts, like you would have see in movies and what not or if you been to factories you would have seen this. So, you see a belt in which you play something and this keeps moving along.

So, I am going to imagine that the conveyor belt keeps moving along, at some point there is a sensor which is going to sense the rods or not and we need to place enough senses. So, the first question is how many sensors should be placed to detective there is a rod or not.

(Refer Slide Time: 10:05)



So, let us imagine something like this, the rods are going to move in this direction and what we are going to do is, at some point we need to detect whether the rod is there or not. So, I am going to place in this example 3 sensors, the first sensor I am going to call it a, I am going to place it here and I am going to place two other sensors namely b and c, b I am going to place it at minus 5 percent line and c I am going to place it at plus 5 percent line, so I am going to place 3 sensors.

So, I could place sensors here, here, here and so on, but for this problem I know ahead of time that these three sensors are enough, we will see why these three enough in a little while. So, I am going to place three sensors and I am going to draw truth table, so from a problem description, we identify the set of inputs. So, now a, b and c are 3 sensors which are going to give us 1 or 0 and if a gives us 0, it means that there is no rod detected in the line of site of a.

Similarly, if b gives us 0, there is no rod detected in the line of site of b and c gives us 0 there is no rod detected in the line of site of c. So, let us see how to detect whether there is a rod first of all and if there is a rod, then how do you reject it, if it is beyond the range that we do not want. So, let us first see the case, so if a senses no rod, b senses no rod and c senses no rod, it means the rod I am imagining that it is coming from top to bottom, there is a belt running from the top to bottom.

If none of these sensors sense a rod, then it means that the rod is not even available, there is nothing you can reject there, there is no rod there is nothing to reject. So, if a senses, if

a does not sense any rod, b does not sense any rod and c is does not sense any rod, there is nothing to do, we are not rejecting. There is nothing to do, so it is 0. Then, if... So, this is the case where the second case, a senses no rod, b senses no rod and c senses a rod.

So, which means the rod is just enter from the top and it has cross this sensor c. If it cross the sensor c, at this point you still cannot decide whether the rod is supposed to be rejected or not. So, in fact to design whether rod should be rejected or not, it is not only enough that it has cross c, it should have cross b and it should have come to a. Only when it comes to a, you can really make a decision whether you want to reject the rod or not.

So, you can see that when c is on, we cannot do anything about it, when b is on, the rod has not reached a, even if both b and c are on, the rod has not reached a. So, when a turns on, you can see that, that is in the bottom 4 combinations, when a turns on we are going to make some decisions. So, let us see what these decisions are. So, the first case is in a, the rod is sensed, but in b it is not sensed and in c it is not sensed.

So, if it is, you are sensing it in a and if it is not in b and not in c, it means that the rod is shorter than 5 percent of the target length. So, my target length is here, so if it has cross this minus 5 percent, it should have been within 5 percent of that, at least it should have cross this line. But, this case it is not, so imagine this has 94 centimeter rod, so this is the 95 centimeter, this red line is 94 centimeter line. So, if this rod is only 94 centimeters, a would have sensed it here, because you are measuring it from a, but b would not sensed and c would not sensed.

So, in this case 1 0 0 means it is a short rod, so rejected. So, if you want to reject, we are saying reject means 1. Then, let us look at the second case, a senses a rod, b does not sense, but c senses a rod. So, if you imagine that there is only one rod in the belt at any point of time, we cannot have a single rod which is sensed in a, not sensed in b, but again sensed in c, this is not going to happen. So, this is a do not care condition, this will not happen, so it is a do not care condition.

So, remember this do not care makes a lot of sense here, it is not 0, it is not that we are saying we are not rejecting it, we are saying that this condition cannot happen. So, the difference between putting a 0 verses putting a x. Then finally, if I have 1 here and if I have 1 here what that means is, a and b are both sensing the rods, but c is not sensing. So, that is this row here 1 1 0, so this is correct is the rod that we want, it is in spec.

So, this rod is, from measure from a it has pass the b line, but it has not touch the c line, so it is acceptable. But, if it goes from a touches b and touches c, this is the rod which is more than 5 percent of the target length. You want to reject that, because it is too long, so now this is the truth table. So, starting from a problem we are come up with the truth table which actually gets what we want. If you think about this, now once I give this table in ((Refer Time: 15:31)) you go and think about whether you really need these sensors at minus 10 percent line, at this spec line at plus 10 percent and so on.

So, what we need is, we need to be able to measure something below 5 percent and something above 5 percent and all the other things are clear, whether you want to accept. So, in this range you accept, beyond that you reject, beyond this you reject, you are looking at ranges. So, there are 3 ranges, from this 1 to minus 5 percent, from minus 5 percent into 5 percent and plus 5 percent to anything above plus 5 percent.

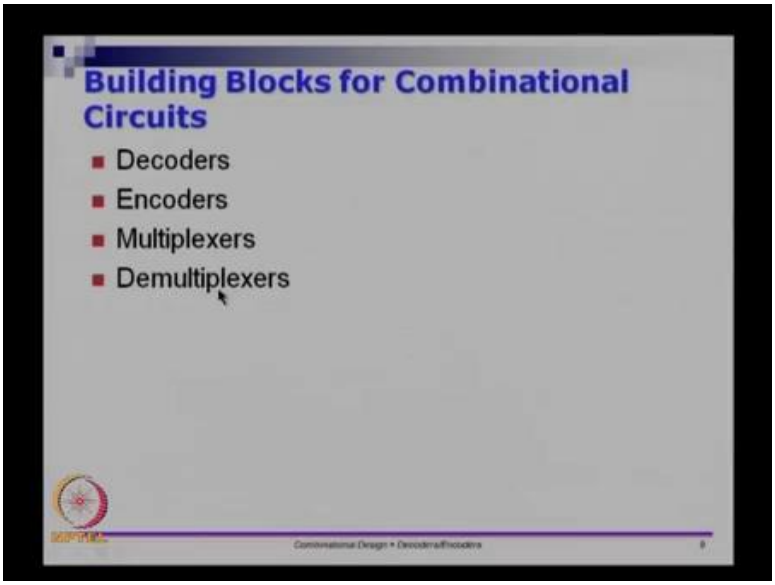
So, since there are only 3 ranges, 3 senses are enough to see whether they are in the ranges, that is why I decided 3 inputs a, b and c. So, you can go and think about this and justify the fact that 3 senses are enough and you can also go and think about, whether just two senses are enough or whether you need 4 at all. So, I knew this ahead, but you can go and think about, why two is not enough for this problem and why 4 senses would be 2 minute.

So, once you have the truth table you know how to do Boolean minimization, in this example it turns out that the min terms, so there is 4 and 7 which are on and 5 is a do not care. So, you combine them together, it is a into b bar plus c. So, an English way to represent that is, you should sense it in a and either it should be sensed in c or it should not be sensed in b, in which case you should reject this rod. So, that is the way I would interpret it in English, I should sense it in a, which means the belt has come this for, so that I can stop the making the measurement.

And either I sense it in c or I do not even sense it in b, if that happens then the rod is rejected. So, this a b bar gives the condition that the rod is too short and a c gives the condition that the rod is too long. So, if it is either too short or too long, you want to reject it, so that is the English way to say, what this whole thing is about. So, it is you could have actually done this even ahead of time without doing any of the tables, you might have you return this directly also. So, if you get the intuition for solving a problem like that, please go ahead and do that. But, this gives you a systematic way in which you

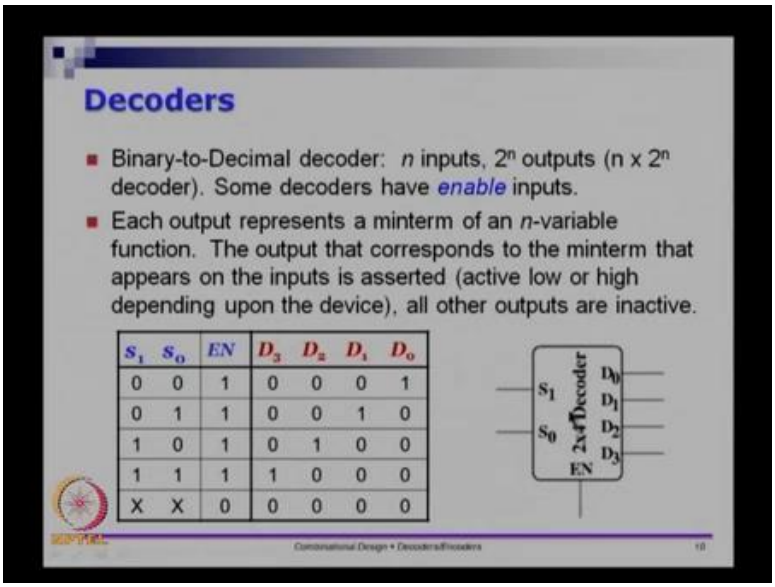
can approach the problem.

(Refer Slide Time: 17:55)



So, now let us go and look at building blocks for various combinational circuits. So, we going to learn about four different building blocks, namely decoders, encoders, multiplexers, Demultiplexers.

(Refer Slide Time: 18:09)



So, let us start with decoders, so the decoder is general term I am going to use this specific binary to decimal decoder, a binary to decimal decoder will have n inputs and it will have 2 power n outputs, so many times we will call such decoders n cross 2 power n decoders. So, if you have n inputs and 2 power n outputs will call them n cross 2 power n

decoder and each input represents a min term.

So, in this case we have 2 inputs, so we have 1, 2, 3, 4 combinations. So, there are 4 combinations for this two variable thing s 1, s naught and the output corresponds to the min term that appears in the input is asserted. So, you go and look at this, so a decoder this has 2 inputs and 4 outputs, so this is a 2 to 4 decoder and 2 to 4 decoder there are 4 combinations of inputs, there are 2 inputs, but 4 combinations of the inputs can take and the output tells you which of the combinations is in the input.

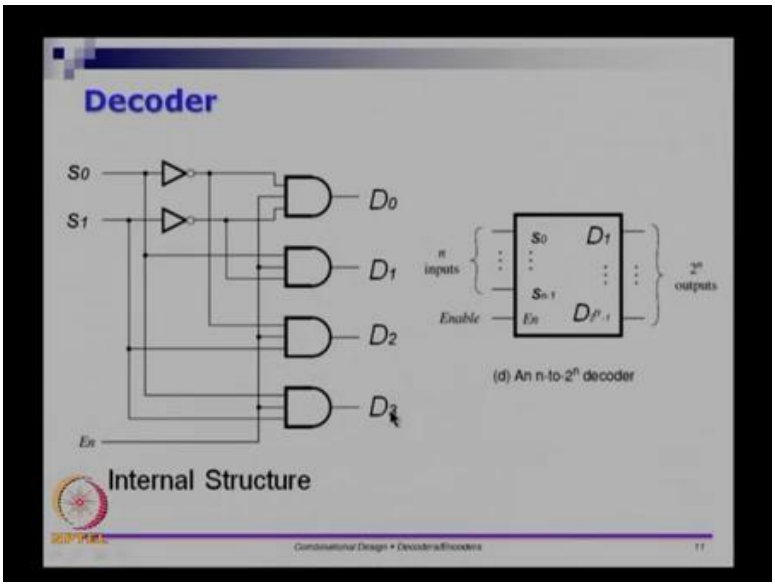
So, if D naught is on, then the in input that is given is 0 0 if D 1 is on then the input given is 0 1, then if D 2 is on the input given is 1 0 if D 3 is on then the input given is 1 1. So, you can think of this as somebody gave an input code and you try to detect which of the inputs was given. So, that is why it is called decoding, the input is a code it is a smaller than size, the output is more number of outputs then the inputs, so this is called decoding.

So, if you place 0 0 here this one will turn on, if you place 0 1 here then D 1 will turn on, if you place 1 0 here then D 2 will turn on and so on, this is called a decoder, let see how to build a decoder like this. So, there are decoders which also have enable inputs, so some decoders have enable inputs, in this case enable is an extra line when enable is on it works like the decoder that I talk about, if enable is off no matter what the inputs are then the output is kept a 0.

So, if you have all 4 of them to be 0, so when the enable is off all 4 of them will be 0, so we cannot track the, what inputs are. So, this is the decoder with the enable line, so if you see the picture that I should earlier, it did not have the enable line coming in. So, there are 2 inputs and 4 outputs, but generally what you have is you have something with the enable. However, you do not come this as the number of inputs, you still call this 2 to 4 decoder.

So, it is 2 inputs bits decoded into 4 separate lines, so you do not say it is a 3 to 4 decoder, enable is implicit you do not count the enable towards the number of inputs, now let us go and see the internal structure of decoder.

(Refer Slide Time: 21:19)



So, internal structure is something like this, you want all the possible min terms essentially ((Refer Time: 21:26)). So, we look at the table you want 0 0 is a min term, 0 1 is a min term, 1 0 is a min term and 1 1 is a min term, the first thing we need to do is to be able to generate all the min terms. So, this is the way to do it, so you have s naught and s 1 and let us go and look at this gate initially. So, this take s naught compliment and s 1 compliment and ANDs it.

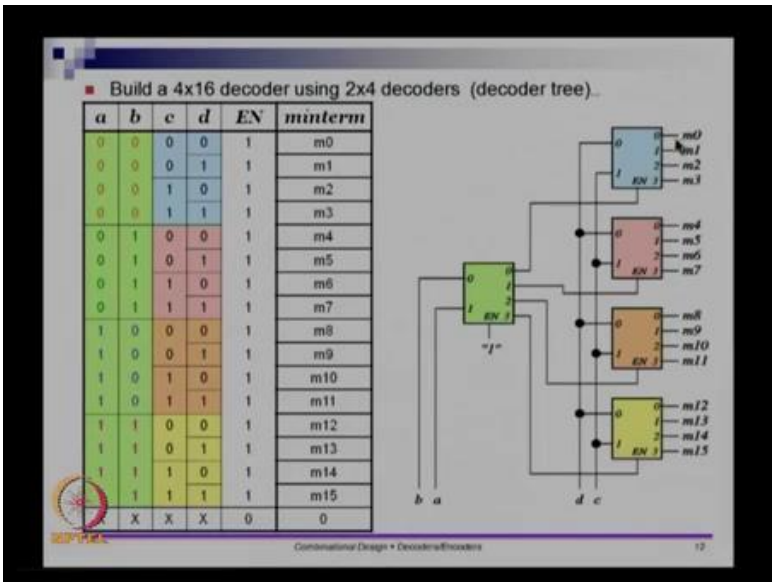
So, this is s naught compliment, s 1 compliment and there is line that is coming from here which is enable. So, D naught is s naught bar s 1 bar enable that is what this line is and this one you can notice is talking you can see that this is s 1 which is given in the complimented form and you also have... So, if you notice this line that is enable and this one is s naught. So, if s naught is on, s 1 is off then D 1 terms on that corresponds to min term 1, so then s 1 is on, s naught is off that corresponds to min terms to 2. So, line 2 terms on and s if both of them are on line 3 terms on.

In all these cases if enable is off, if enable is 0 you can see that all the AND gates are getting this line directly. So, the output will all be 0 if enable is off, for this to work add to decoder enable should first of all we termed on and you will give one of these combinations and one of the lights will or one of the outputs will turn on. So, if you put an LED here exactly one of the lights will glow if enable is turned on, if enable is off none of the LED is will glow, if you put LED is in the outputs.

So, there is no specific symbol for a decoder, a many times we use this symbol here

where we I have n inputs and enable line. And so these are mark from s naught and s n minus 1, the number of output should be 2 power n not 2 power n plus 1 do not count the enable 2 power n and they go from D naught to D power D 2 power n minus 1. So, this should have been a D naught, this should have be D naught. So, D naught to D 2 power n minus 1. So, there are so many of them here, so this is a n to 2 power n decoder.

(Refer Slide Time: 23:53)



So, let us see how to build a 4 to 16 decoder using 2 to 4 decoder, this is the logical problem. So, I am giving you and infinite supply of 2 to 4 decoders and I want to you do build 4 to 16 decoders, let us say how to do that. So, I am first going to write the truth table required for the 4 to 16 decoder. So, what should the 4 to 16 decoder do if it is 4 to 16 is one which you have 4 inputs and it should have 16 output lines that is what a 4 to 16 decoder is and on top of that you will have another extra line called enable.

So, if enable is on then it work like a decoder, if enable is off the output should be 0, let see what this means. So, if I have a, b, c, d. so all the 4 inputs for the 4 to 16 decoder on 0 and if enable is on, then this corresponds to min term 0. Then, similarly if I have a and b as 0 0 and c and d is 0 or 1 this corresponds to m 1 and so on. So, we know that if I enumerate this since we are figuring out which min term is on will have it for m naught to m 15.

So, this is the min term combination, so I am not showing the outputs here, because the output should be 16 in number, there is no space for that inside I draw the circuit directly. So, all the says is if 0 0 0 0 comes on we want line D naught to be turn on, if 1 1 1 1

comes on we know we need that line D 15 to be turn on and so on. So, one thing you can notice is there are a and b here which are given in green and there are 4 combination, this blue, this pink is 1, the orange 1, the yellow 1 these are 4 different combinations for difference choices of a and b.

So, in the blue 1 a and b 0 0 when it is pink it is a and b or 0 1 and so on, let see how to design some circuit like this. So, you know how to design a decoder for this path. So, if I give you 2 to 4 decoder, then if it is 0 0 0 1 1 0 1 1 if I put a 2 to 4 decoder like this, I will get 4 outputs. So, I get 4 outputs here and for each of these I put a 2 to 4 decoder. So, you can see that c and d are going as inputs to all 4 of them, so it is a shade input that is going to all 4 2 to 4 decoders.

So, at the stage what we have is there are decoder, so all these decoders will essentially behave the same now, if you give c and d is input and if let say enable was on for each one of those, all of them will... So, if m naught turns on you will see the m 4 will also turn on, m 8 will turn on, m 12 turn on and so on. So, forget this for a while now, if I have connected if you look at only this part, they are getting the same input they are all the same circuit, if enable is on then you will have this issue that based on c and d there will be 4 lines of the 16 lines will turn on.

What we are now going to do is, we are now going to take 1 step for them, so we are going to put another decoder here, it is another simple 2 to 4 decoder to which we are going to a and b as input, when we give this we know exactly one of these lines will turn on. So, if a and b are both 0 0, then line 0 will turn on if a 0 b is 1 line 1 will turn on and so on. So, the problem we had was if it is all c and d connected and all of them a turned on, all of them are enable at the same time then you will have 4 of these things turned on, one from here, one from here, one from here and one from here for each combination of c and d.

Now, we are going to control the enable, so when we put this green color decoder here, only one of these 4 outputs is going to turn on and the output is connected to the enable line. So, let see what happens if the input combination is 0 0, so I am in 0 0 0 0, so that is the input to circuit 0 0 0 0. So, a b is 0 and c d are 0, so because a and b are both 0, this D naught line of this decoder will be on, which means this blue decoder is enable.

However, these three lines will be off D 1, D 2, D 3 this decoder is would be off, since they are going as enable lines for the pink, the orange and the yellow one these three

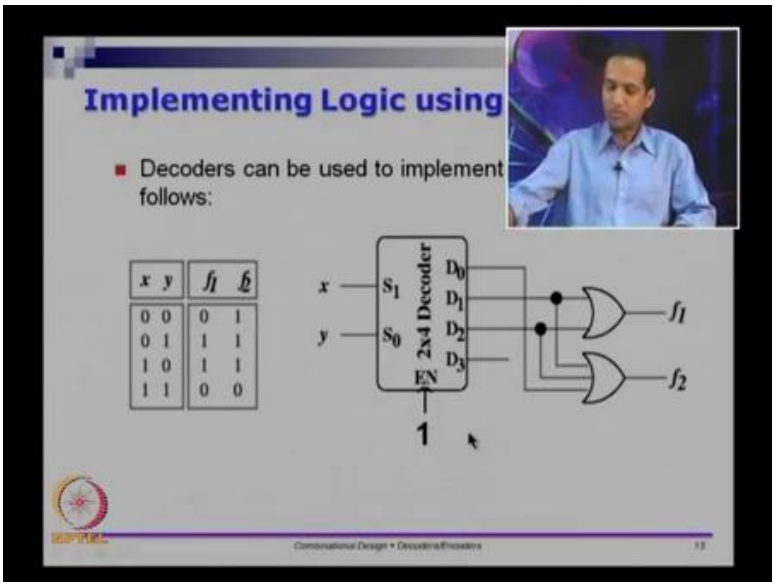
decoder will be turned off. So, a and b being to a 0 will enable this blue decoder, now if I put c and d as 0 also, then this is already enabled c and d are both 0. So, m naught will turn on, so now let us go and reason the condition for m 10 to be turned on will go the other where on.

If you want m 10 to be turn on, first of all this one should be enable and the notion of a decoder is if m 10 is on, everything else must be off. So, for m 10 to be turned on this must be enable all the others must be disable, which means if we go back here the D 2 must be turned on D naught, D1, D 3 must be turned off. Now, if you go and imagine when d 2 will turn on, it will happen for a equal to 1 and b equal to 0. So, let us keep this in mind. So, to for m 10 to turn on then a must be 1 and b must be 0, we have that for.

Then, with in this decoder if you want m 10 to be turned on, you go and look the inputs c and d what combination must be there for c and d, c must be 1 and d must be 0 for this 1 0 line to turn on. So, if c is 1 and d is 0, it is d 2 will turn on, so that is the d 2 for this decoder. So, if you have 1 here and 0 here, this decoder will turn on this line, so we already had a equal to 1 and b equal to 0 and because of the conditions that we want m 10 to be turned on c equal to 1 and D equal to 0 that is the other thing.

You go and look at the row 1 0 1 0, then a b is 1 0 and c d is 1 0 go and look at 1 0 1 0 that corresponds to the term in m 10. So, this is just a way to cross check, what you done is correct or not. So, these are the simple example of building a complex decoder using a smaller decoder. So, you will see that this decoder this very useful later when we talk about what is called memory. So, in your computers you have the notion of RAM, when we discuss RAM it we will see that this notion of decoder is very handy.

(Refer Slide Time: 30:55)



We can also use decoders do some few interesting things, s in this example let say we are given two functions f 1 and f 2 and there are 2 inputs x and y and we want realize this using a decoder. So, you have f 1, f 2 here and you know that if you want these 4 input combinations, we can give it to a decoder keep it enable and exactly 1 of the 4 lines will turn on. So, if 0 0 is there this line will turn on, if 1 0 is there this line will turn on and so on.

Now, what we are going to do is, you going to put two OR gates, one for the output f 1 and one for the output f 2. So, what are the conditions and which f 1 is turned on, if you notice here the conditional and which effort is turned on is if input is 0 1 or if input is 1 0. So, that can be capture by checking whether D 1 is on or D 2 is on. So, when will D 1 be on, 1 will be on when this is 0 and this is 1 and when will D 2 be on the x is 1 and y is 0.

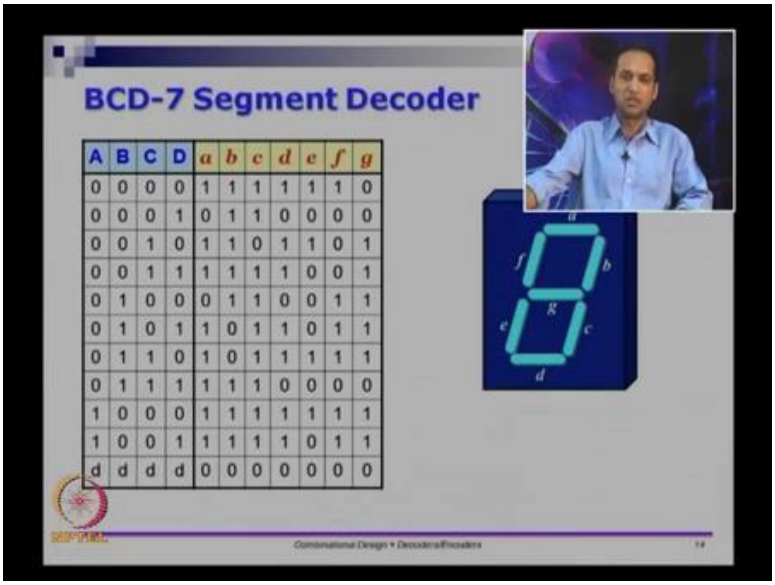
So, under these two conditions if you top those wires and connect to the OR gate f 1 will be on, when these two conditions are satisfied. Similarly, if you want to check when f 2 will be on, either it should be for min term 0 or min term 1 or min term 2. So, we top those 3 lines, namely D naught D 1 and D 2 and connect then into 3 input OR gate that gives me a function for f 2. So, in both these things D 3, D 3 does not matter, we can notice that D 3 is not connected to any of the OR gates and if you go back to the table the D 3 row is 0.

So, this row has 0 in both the outputs, so it is not connected, so these a way in which if I

given a decoder, you can use the implement other logic circuits also. So, you could have done 2 inputs circuits directly also, but if I given a decoder and if I ask to designer circuit with a decoder that keeping to remember is give inputs, the outputs correspond to the min terms and exactly one of the min terms will turn on corresponding to the inputs. Then, you top all the corresponding min terms, you put them into to OR gates and you are done.

So, this essentially captures the AND part of the circuit and this captures the OR part of the circuit. So, you have the AND OR logic, but this AND logic in the decoder can actually produce all the min terms, so this a very useful circuit.

(Refer Slide Time: 33:28)



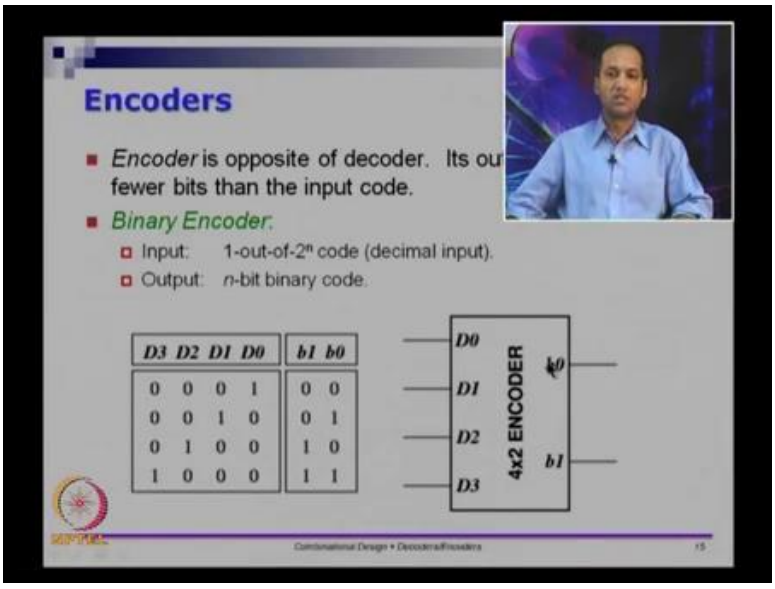
So, this is useful in several places, so for example you might of see in this notion of some segment LED displays in your in your labs. And so for example, if you want let say I want to display number 7, if you want number 7. So, let us go and look at 0 1 1 1 that is number 7. For number 7 to be turned on I would like this segment a, segment b and the segment c to be turned on, if all the other segments are off that would mean number 7.

Similarly, if I wants number 6 to be displayed except b I want everything to be turned on, this is the kind of circuit, you want you can go and notice this here. So, to capture if you want to display 6 which is 0 1 1 0, turn on everything except b, this is b turn on everything accept b. Similarly, if you want e, if you want let say 4, then you want f, g, b and c to be turned on let us go and look at 4 b, c, f and g are all turned on other things are off.

So, there is a multi output a 4 input circuit, so there are 7 outputs and 4 inputs and if you want to design the circuit for this, then is much more easier to put a single decoder for all the combinations here, you take all these things and take a combinations here for each of the outputs, you put the corresponding circuit. So, if min term 0 or min term 2 or 3 or 5 or 6 or 7 or 8 or 9, then turn on a. This segment will turn on only under these conditions.

If you want to look at the middle segment g, you can think about it we will needed for number 2, we will needed for number 3, we will needed for number 4 and 5, we will needed for 6 needed, we needed for 8, we do not needed for number 0, 1 and 7. So, you can that the column here, so we do not needed for number 0, 1 and 7 for all the other input bits input combination you want them on. So, that is what is captured here, you take the corresponding min terms put 1 single OR gate connecting other min terms and you are done. So, these a very nice and simple example for a decoding.

(Refer Slide Time: 35:51)



Then, there is this corresponding circuit called the encoders, encoders of the opposite of decoders, it is a output code has fewer bits then the number of input code. So, think about it as generating a code, you have more inputs and you creating lesser outputs it somehow encoding the what input you gave. So, binary encoder is 1 in which you get 1 out of 2 power n code. So, it is the input will be 1 out of 2 power n values that are possible and the output will be a n bit binary code.

So, let see an example here, so this is just like the table that we are earlier except there inputs and outputs are flip. So, D 3, D 2, D 1, D naught there are 4 inputs given to a 4 to

2 decoder and there are 2 outputs. So, what we have is if D 3 is turned on, then we will indicate that by turning on both b naught and b 1. If D 2 is turned on and if the others are off, then will indicate that we saying b 1 is 1 and b naught is 0.

Similarly, if just D naught is turned on will indicate that by saying b 1 is 0 and b naught is 0. So, as long as exactly one of the inputs is turned on, the output form a encoder will be one of the 4 combinations. So, here we have 4 inputs if exactly one of them is turned on, then one of the 4 2 bit combinations will come here. So, encode has typically do not have enable lines and you should ensure that exactly one of them will be turned on.

So, if you notice even though there are 4 inputs we do not have 16 combinations, we have listed only 4 combinations. So, at no point and time you can have 0 none of them turned on or more then one of them turned on, exactly one of these lines should be turned on for this encoded to give a correct output. So, this in some sense the opposite of what we saw on the decoder.

So, this brings me to the end of this module, so what we saw or two really important circuits, decoders are very, very useful we see them in more detail later and encoder is just the opposite of the decoder, except that encode is do not typically come with enable lines and you have to take care of the condition, if you going to use encoder in a circuit somehow you should take care of the condition that exactly one of the inputs is on, the other inputs are not turned on. You cannot have 0 inputs turned on, you cannot have more than one input turned on also for an encoder. So, in the next 2 modules we will see what are called multiplexers, so see you soon.