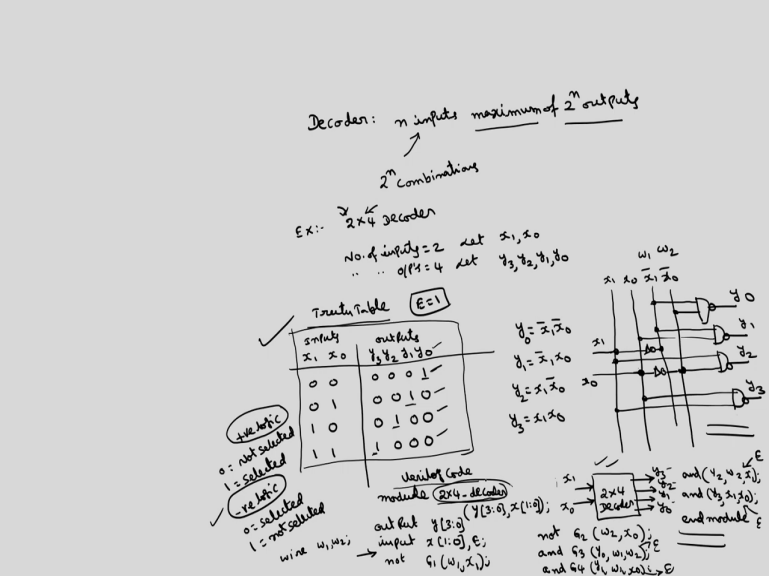
System Design through VERILOG
Prof. Shaik Rafi Ahmed
Department of Electrical and Electronics Engineering
Indian Institute of Technology, Guwahati

Gate level modeling - II
Lecture - 07
Decoder, encoder and multiplexer

Ok, in the last class we are discussing about the various combinational circuits and their Verilog codes. So, we have discussed about the adder, subtractor then multiplier, comparator. So, the next important combinational circuit which comes under the control logic of processor is a decoder.

(Refer Slide Time: 00:58)



Decoder is a combinational circuit with n inputs and a maximum of 2 raised to the power of n outputs. This is maximum not always it will be equal to 2 raised to the power of n ok. For each combination of this input if we have this n inputs. So, I can have 2 raised to the power of n combinations because each input can be 0 or 1.

So, corresponding to each combination one of the output is selected ok. See if I take a simple example of a 2 by 4 decoder this first number represents inputs the second number represents output. So, we know how to design a combinational circuit ok. So, we have to first I mean

find out the number of inputs available number of outputs required the number of inputs are 2 number of outputs are 4; you have to assign letter symbols ok.

So, let the inputs are x1 x0 where x 1 is msb, x0 is lsb. Let the outputs are y3, y2, y1 and y0. Then you have to form a truth table corresponding to all combinations of the input. x1, x0 inputs and outputs are y3, y2, y1, y0; corresponding to 0 0 one of the output is selected as I have told. So, y0 will be selected.

Selected means 1 not selected means 0 this is called positive logic and we can use the negative logic also; 0 1 means 0 0 1 0, y1 is selected, 1 0 means y2 is selected, 1 1 means y3 is selected. Then you have to write down the Boolean expressions and you have to simplify it, but here only one mean terms are there. So, we can directly write the expressions for this is y0 is equal to x1 bar x0 bar, y1 is equal to x1 bar x0, y2 is equal to x1 x0 bar, y3 is equal to x1 x0.

Simply the circuit diagram will be, you have two inputs x and y, x1 x0, this entire line is x1 line and this one is x0 line and you can generate two more signals complements of x1 and x0 through the inverters. So, this entire line is x1 bar line; this entire line is x0 bar line. This is x1 x0 x1 bar x0 bar.

Now, you have to give the connections this is the design of this is y0, this is y1, y2 is x1 x0 bar, y3 is x1 x0. This is the design of a 2 by 4 decoder ok. So, we can write Verilog code. It is very simple circuit. So, if I take this as a block 2 by 4 decoder with 2 inputs, x1 x0 and 4 outputs, y3 y2 y1 and y0. So, if I take Verilog code; we have to generate x1 bar x0 bar from x1 x0. So, we have to use 2 bytes also ok.

So, this is module 2 by 4 decoder outputs are y[3:0], inputs are x[1:0]. Then we have to define output y[3:0], input x[1:0] then we have to define two wires. Let us call this x1 bar as wire w1 this as w2, not we can give some gate number sometimes I am not giving the gate title ok output is w1 input is x1. Similarly, NOT G2 w2 is the wire x0 is the input. So, output is w2 is nothing, but x0 bar.

Now we have 4 AND gates and G3 y0 is the output, inputs are w1 w2 and G4 output is y1 inputs are x1 bar is nothing, but w1 and then x0 and y2 for y2 x1 x0 bar x0 will be w2 x1 and

y3 x1 x0 end module. This is a simple Verilog code corresponding to 2 by 4 decoder, but there are some several issues here.

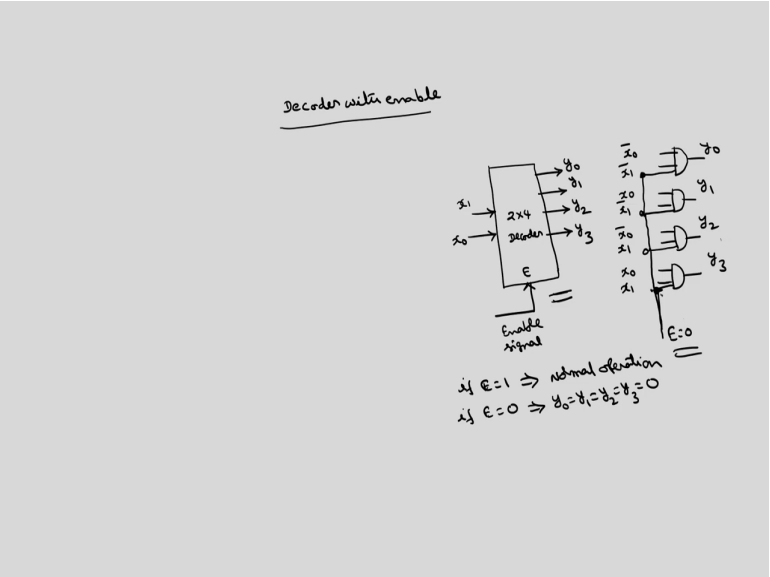
So, one is so normally we will use this NAND gates we will use negative logic this is called positive logic where 0 is not selected 1 is selected ok, but most of the IC's uses negative logic. Negative logic is reverse to this one 0 is selected, 1 is not selected. So, this will become you have to change all 0s to 1s, 1s to 0s here ok and this can be implemented by using NAND gates.

So, the reason for using this negative logic is NAND gate is a universal gate ok. So, because of that most of the IC's will be having negative logic type of thing ok. So, this is the negative logic using negative logic you will get, you have to replace this AND gates by NAND gates. So, you will just you mean consider only the positive logic only now.

So, another thing is see here if I take this for positive logic this is the truth table, so at any time only one of the output is selected. There is no case where none of the output is selected there may be possibility that I do not want to select any of this output devices I want to disable this decoder; that option is not possible here if I take this decoder block diagram.

So, we have x1 x0. So, for a four combinations one of this output is selected there is no combination of the input for which none of the output is selected ok. So, if you want to provide that. So, you have to use a decoder with enable line. So, you have to use a decoder with enable line.

(Refer Slide Time: 11:00)



So, I will take this same 2 to 4 decoder. I am considering only positive logic for the sake of simplicity; x1 x0 I will give one more signal called enable signal E; E is enable signal. Outputs are y 0, y 1, y 2, y 3. If E is equal to 1 then normal operation; in the sense the same this truth table will be enabled this truth table will be enabled only if E is equal to 1.

On the other hand if E is equal to 0 implies all the outputs will become 0, y0 is equal to y1 is equal to y2 is equal to y3 is equal to 0 the entire decoder will be disabled ok. So, in order to design this type of a circuitry we have the same NAND gate logic only or AND gate depends upon positive or negative logic as per the circuit diagram of a 2 to 4 decoder that we have drawn using positive logic this is y0 y1 y2 y3. So, these signals are this is x0 bar. So, I am not showing the inverter x1 bar.

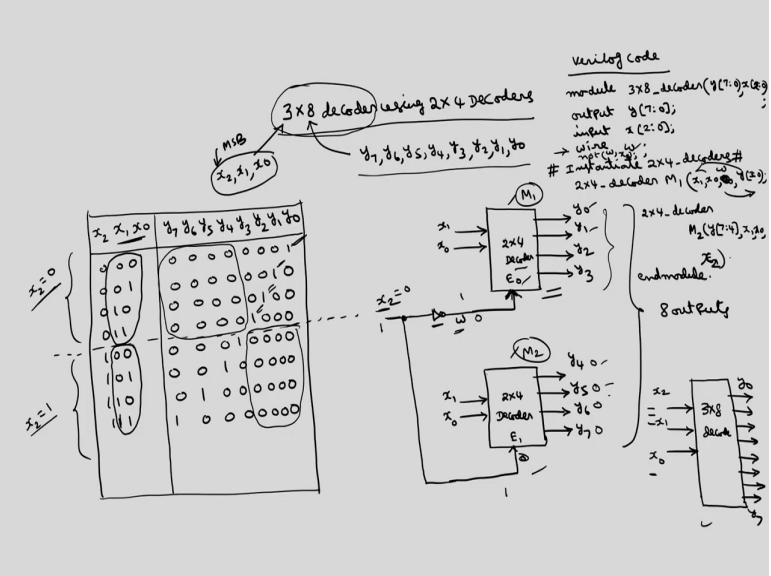
So, 1 means x0 x1 bar this is x0 bar x1 this is x0 x1 this is ordinary decoder without the enable signal. If we have enable signal you have to connect this enable as a third input to all the AND gates. This is enable. So, the operation is clear that if E is equal to 1 E is equal to 1. So, this output depends upon x1 and x0 if E is equal to 0 if one input to this AND gate is 0 regardless of the other inputs outputs are 0s ok.

So, you see the circuit diagram and the corresponding block diagram of this circuit is this you have two inputs four outputs enable signal is there ok. So, normally this type of enable the

decoder with enable signal is very much useful so in memories also. So, in order to select in order to construct the larger memories from this smaller memories we require this type of a decoders with enable signal.

Even if you want to construct the larger decoder using smaller decoders. So, you have to use this enable signal ok. Now, I will discuss how to construct a 4 by 16 decoder or 3 by 8 decoder using 2 by 4 decoders. Because if I want to write this Verilog code I can instantiate with this 2 by 4 decoder ok.

(Refer Slide Time: 14:35)



So, this 3 by 8 decoder using 2 by 4 decoders; how to construct this 3 by input 3 inputs let us call this one as x2 x1 x0. This is MSB, x0 is LSB and 8 outputs you call as y7 to y0. So, how to construct this? So, how many such 2 by 4 decoders are required? And what are the external circuitry that is required to construct a 3 by 8 decoder ok.

Basically we need two 2 by 4 decoders. This is one 2 by 4 decoder. So, this will give four outputs and two inputs and I am taking the enable signal with enable. I am taking one more 2 by 4 decoder this also will give four outputs total together is 8 outputs; this is fine. So, we have got the total 8 outputs.

I will take these first four outputs as y0, y1, y2, y3 this is y4, y5, y6, y7. Now this two decoder will be having two inputs and there will be one enable line. We call this enable line

as E0 this as E1. Now what about the inputs? Totally we have three inputs; how to connect these three inputs.

So, if I take these 8 combinations. So, what will be the truth table of this 3 to 8 decoder x2, x1, x0, y7 to y6, y5, y4, y3, y2, y1, y0. Corresponding to 0 0 0; y0 is selected. I am using positive logic. 0 0 1, y1 is selected; 0 1 0, y2; 0 1 1, y3; 1 0 0, y4; 1 0 1, y5; 1 1 0, y6; 1 1 1, y7.

So, what you can observe here is; in the first half of the location this is the half locations 4 is the half. So, in first half of the locations x2 is 0, in next half of the locations x2 is 1, but what happens to x1 x0? x1 x0 varies from 0 0 to 1 1 here also it varies from 0 0 to 1 1 ok.

So, that is why I am going to connect this x1 x0 to the inputs of both the decoders same. So, we have connected together ok. Then x2 I am going to use to select the enable signal. So, when x2 is equal to 0. So, you have to select this if enable is equal to 1 the decoder will be enabled otherwise the outputs will become 0 as I have told in the last slide.

So, what you have to do is because E is equal to 0 for x2 is equal to 0, I have to select this because this is giving first 4 outputs for E is equal to 0 I want y 0, y 1, y 2, y 3, ok. So, we have to make this x2 complement this is the x2 input this we have to give as a complement.

So, if x2 is equal to 0 this becomes 1 if x2 is equal to 0 this is 1. So, this will be enabled so these four outputs will be enabled. And I will give this same x2 directly to the second decoder enable ok. So, that when x2 is equal to 0 this is equal to 0 this four will becomes always 0 0 0 0.

So, in the first four locations this, this, this, this. So, these are four outputs are 0 0 0. So, here the all will be 0 0 0 because enable is 0 for the first half of the locations. For the next half of the locations. So, this will be 0 E x2 is equal to 1. So, x2 is equal to 1 if it is x2 equal to 1. So, this becomes 0 this becomes 1 this will be enable this will be disabled. So, like that we can generate.

So, this 8 outputs using 3 inputs. So, this MSB input you have to apply as a selection signal for the both are 2 to 4 decoders. So, for the upper one you have to select through the inverter

for the lower one you have to give directly. So, this is how you can implement a 3 to 8 decoder using 2 to 4 decoders.

So, now, we can call these 2 by 4 decoders twice to write the Verilog code of a 3 by 8 decoder ok. So, what is the name I have given for this 2 to 4 decoder? I have given the name as 2 by 4 underscore decoder. So, if we want to write the Verilog code here module 3 to 8 decoder. So, for 3 to 8 decoder this is equivalent to a block diagram with 3 to 8 decoder.

How many inputs and how many outputs? This enable is internal ok. So, we do not have any enable input if want you can put this enable at the output also ok. So, this is x 2, x 1, x0 this is 3 to 8 decoder outputs are y0 to y7. So, overall for this 3 to 8 decoder we have 3 inputs and 8 outputs. Of course, you can place a enable line for this one also ok.

Now, here I am not considering enable line. So, these outputs are y7 comma 0 inputs are x2 comma 0. Now what is required one wire is required you call this as a wire ok. So, wire output y7 comma 0, input x2 comma 0. Then you have to define wire in the earlier code I think I have not defined the wire.

So, wherever is missed you can write this wire you have to define wire w. Here we have to define wire also sometimes I will forget to define this we have to define on that the wire that we have defined with w1, w2. Then we have to instantiate 2 by 4 decoders. The name that I have given is I can call this as module M1 this as module M2, 2 by 4 decoder M1. So, what are the inputs and outputs of this? Inputs are 3.

So, this was the code I have written for this without enable ok. So, if I include the enable also then what is the modification that you have to do? So you have to write one more input here ok this is without enable. So, with enable means you have to write one more input that is E if you want the code for this.

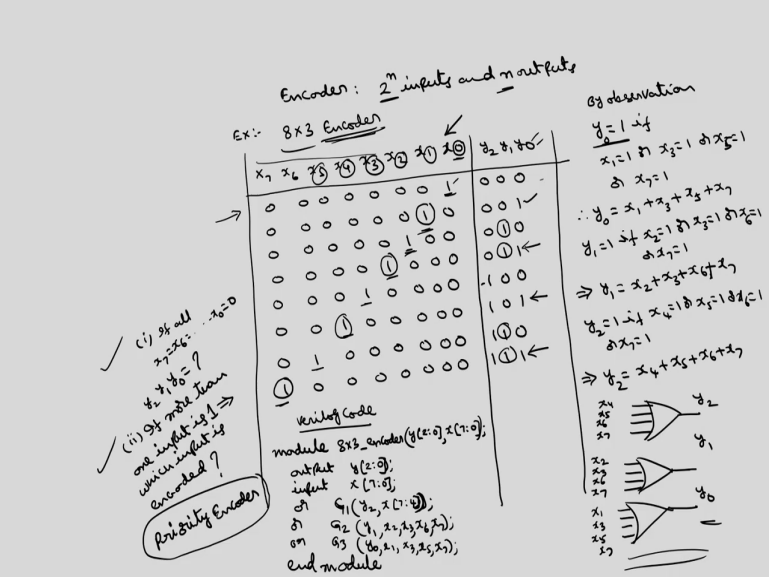
So, I am just I am changing this previous program. So, this is E and then you have to include this AND gates you have to include one more signal here E ok. Here you have to include E, here also for AND gates here, one more input E, one more input E. I am considering this as a 2 by 4 decoder this is along with enable signal.

So, how many inputs will be there now? Three inputs x1 comma x0 comma E0 are the inputs and outputs are y3 to 0. Here also this actually this program I have written for the decoder without enable. So, with enable you have to input here also 1 enable I have to write enable. So, we can do that modification this is a simple modification only ok. And then I have to instantiate second 2 to 4 decoder M2.

So, for M2 outputs are, you can write the outputs first and inputs next; y7 comma 4 comma inputs are same x1 x0 comma E1 or you can give x2 itself E1 is nothing, but x2 and this E0 is nothing, but w. Because I have defined this as a wire and I have to define NOT also here. Before this NOT output is W comma input is x2; then end module.

So, this is the Verilog code corresponding to 3 by 8 decoder using 2 to 4 decoders. So, you have instantiated this two 2 to 4 decoders to construct a 3 by 8 decoder. In a similar manner we can go for the higher order decoders 4 to 16 also you can construct using two 3 to 8 decoders and so on ok. So, this is about the decoder circuit.

(Refer Slide Time: 27:28)



So, the next combinational circuit is the opposite of this decoder is encoder. Encoder is a combinational circuit with 2 to the power of n inputs and n outputs. This is reverse to the decoder. So, for each combination of the input, the output code will be generated which is the

binary equivalent of the input. The binary equivalent of the output which is generated correspond to each input.

If I take the same example of 8 to 3 encoder. You have eight inputs: x7, x6, x5, x4, x3, x2, x1, x0. Then we have only three outputs y2, y1, y0. So, I will assume that initially a simple encoder where at any time only one of the input is 1. So, I am not considering the other cases ok.

So, here I will assume that initially x0 is 1. So, the input which is 1 is encoded. So, this is 0, 3-bit binary equivalent of 0 is 0 0 0. So, we will get this binary equivalent of that input; 0 0 0 0 0 0 1 0 this is 1, the 3-bit binary equivalent of 1 is 0 0 1. So, the input which is 1 is encoded using 3-bit binary number.

So, 0 0 0 0 0 1 0 0 so 2; what is the 3-bit binary equivalent of 2; 0 1 0 0 0 0 0 1 0 0 0. 3-bit binary equivalent of 3 is 0 1 1. 0 0 0 1 0 0 0 0 this is 4; 4 is 1 0 0. 0 0 1 0 0 0 0 0 this is 5; 5 is 1 0 1; 0 1 0 0 0 0 0 0 this is 6 which is 1 1 0; 1 0 0 0 0 0 0 0 is 7; 1 1 1. So, this is a truth table of 8 to 3 encoder this is just simple encoder.

So, how to write the expression for the Boolean? How to write the Boolean expressions for this outputs y0 y1 y2? So, y0 is 1 when x1 is equal to 1 or y0 is equal to 1 if x0 x1 is equal to 1 or this one this is x3 is equal to 1 or x5 is equal to 1 or x7 is equal to 1.

In fact, if you want to solve this using a Boolean expression we have to write a 8 variable K-map. This has inputs of 8, but without that. So, for this type of truth tables we can easily write the expressions for the output by observation. So, I am writing this by observation. This is by observation y0 is equal to 1 if under four conditions we will get this.

So, what is the Boolean expression for y0 this statement you can convert into Boolean expression or means or gate just simply x1 or with x3 or with x5 or with x7. Similarly you can write y1 is equal to 1 if this, this, this, this, this. So, what are the inputs corresponding inputs? This is x2 is equal to 1 or this is x3 is equal to 1 or x6 is equal to 1 or x7 is equal to 1. So, implies y1 is x2 plus x3 plus x6 plus x7.

Similarly y2 is equal to 1 if this 4, 5, 6, 7 implies y2 is equal to x4 plus x5 plus x6 plus x 7. So, simple circuit diagram of a 3 by 8 decoder is simply we require 3, 4 input OR gates.

These outputs are y2, y1, y0. So, y2 is x4, x5, x6, x7, and this one is x2, x3, x6, x7, this one is x1, x3, x5, x7, this is the simple circuit diagram of a 8 to 3 encoder.

So, what is the Verilog code corresponding to this? If I take this as a block module 8 to 3 encoder outputs are y[2:0] inputs are x[7:0]. Then output no wire is required here; input then we have or some gate G1 y2 comma x[7:4] or G2 y1 comma x2, x3, x6, x7, or G3 y0 comma x1, x3, x5, x7, end module. So, in all this gate level modelling so we need to derive the internal circuit of the given system.

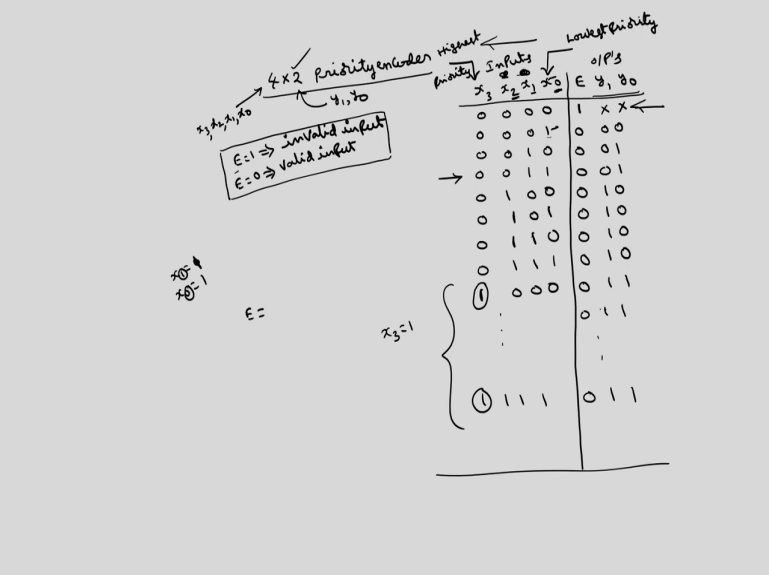
So, whereas, in the higher level modelling we need only the block diagram ok; without any internal details. So, that is why I am concentrating more on this gate level because gate level is more optimized we can optimize at the gate level. So, whereas, in block diagram behavioural model and the remaining models. So, we cannot optimize the problem to a great instant.

Now, here the problem is again this is also a simple encoder. Here I am considering at any input only one of the input is 1; here this is 1, this is 1, this is 1, this is 1, this is 1, this is 1. Two questions will arise here, one is if none of the inputs are 1, what should be the outputs; that we have not considered here. That is one case one point that we have not considered is if all the inputs are x7 to x0 all are 0s.

Then what is the output? What is y2, y1, y0? This we have not taken care in this table. The 2nd one is if more than one inputs are 1; then which input will be encoded? These two questions we have not answered in this, a simple 8 to 3 encoder. So, if you want to answer these two you have to go for an encoder called priority encoder.

So, you have to assign some priority to the inputs. So, that if more than one input is 1 the input with highest priority will be encoded ok. So, I will just simply give this truth table of this priority encoder. So, the Verilog code and all you can do in a similar manner.

(Refer Slide Time: 37:58)



I will take a simple case of 4 to 2 priority encoder; priority encoder. A simple encoder will be having 4 inputs and 2 outputs. Here I am going to place an additional output also. So, let the inputs are x3 x2 x1 x0 with x3 is MSB x0 is LSB, outputs are y1 y0, but in addition to this y1 y0, I will include one more output.

That is if I take the truth table x3 x2 x1 x0 the inputs, output I will make as E, y1, y0 these are the inputs and outputs. So, this E is basically if all are 0's then this is error. So, I will make this E is error output. So, error output I will make as 1, this y1 y0 are don't cares.

So, this E is equal to 1 represents invalid input because for the encoder at least one of the input should be 1. Of course, more than one input also can be 1, but all the inputs cannot be 0. So, if all the inputs are 0 this is invalid. So, I will make this E, is equal to 1 when all the inputs are 0's.

Now for the remaining 15 combinations at least one of the input is 1; 0 0 0 1 only one input is 1. So, this 0 is encoded this I will make this error as 0; E is equal to 0 means valid output or valid input. So, to represent whether the input is valid or not I am providing one extra output signal E. So, for the remaining all combinations E is 0.

So, 0 0 0 1 only one input is 1, 0 the 2-bit binary equivalent of 0 is 0 0. So, 0 0 0 0 0 1 0 this is E is equal to 0 because this is valid. So, 1 the 2-bit binary equivalent of 1 is 0 1; 0 0 1 1

now more than one inputs are 1. Here x1 is also 1, x1 is also 1, x0 is also 1. So, whether we are going to encode 0 or 1. This depends upon the priority you have to assign some priority.

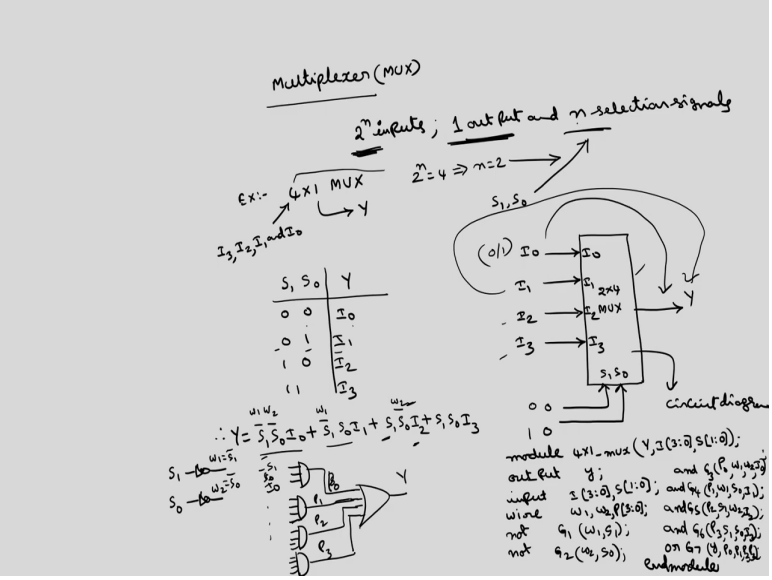
So, the normal procedure is the input with lower suffix I mean assign lowest priority and this is the highest priority. So, this is lowest after this, this priority after this, this priority after this, this. So, you see the order in which we are going to assign the priorities this is the order of increasing the priority. So, now, between x1 and x0, x1 is having larger priority. So, I will define this one also, this is also valid input, so 0 1 only.

So, 0 1 0 0 only 1 input is 1. So, E0 and x2 is 1 0. The 2-bit binary equivalent of 2 is 1 0. So, 0 1 0 1 now x0 x2 are 1, but x2 is having larger priority compared to x0, so x2 will be encoded. So, this is 0 same as 1 0. So, 0 1 1 0 this is also as long as x2 is 1, x1 x0 will not be encoded. So, this is also 1 0; 0 1 1 1 also the same 0 1 0 then 1 0 0 0. So, this is x3, x3 is 1 1.

So, for the remaining all this up to the last 1 1 1 1 1 this is same repeated. So, 0 1 1 1 is repeated 0 1 1 because this will be for the remaining half conditions for eight locations this x3 is equal to 1 if x3 is equal to 1; 3 is having highest priority. So, always 3 will be encoded ok. So, this is the truth table of a 4 to 2 priority encoder ok.

So, the expressions for E; the output E represents whether the input is valid or not. If input is valid, E is equal to 1 then only I will consider y1 y0 otherwise y1 y0 are don't cares. So, you can easily draw the logic diagram and you can write the so Verilog code corresponding to this priority encoder.

(Refer Slide Time: 43:30)



So, the next combinational circuit is multiplexer. So, this is having several applications in all the fields of electrons and communication engineering. So, this multiplexer in general will be having 2 raised to the power of n inputs 1 output and n selection lines. So, the number of inputs is normally a power of 2 the relation between the number of input and selection lines is this if n selection lines are there 2 raised to the power of n inputs, but always output is 1.

So, if I consider an example of a simple multiplexer 4 by 1 multiplexer. So, short form of this one I will call as MUX. How to design a 4 by 1 multiplexer? So, we have to assign some letter symbols, we know the number of inputs, number of outputs and you have to assign some letter symbols then we have to draw the truth table.

Let the inputs are I3, I3 is MSB, I2, I1 and I0. Output is only one output let us call as y. And selection signals will be having two selection signals now you call as S1 comma S0 because 2 raised to the power of n is 4 here the inputs say implies n is equal to 2 so two selection lines. So, what is the truth table of this is; so this S1 S0 is going to decide the output.

If I take the block diagram of this one, this is 2 by 4 multiplexer. Here we have to say define the selection lines S1 S0 here four inputs I0 I1 I2 and I3. So, this is the output Y, this is I0 I1 I2 I3. So, if this is 0 0 whatever the information on I0 will be transferred to the Y ok. What is

that information this I0 can be either 0 or 1. If this is 0 1, one information will be transferred; 1 0, I2 will be transferred; 1 1, I3 will be transferred ok.

So, S1 S0, 0 0. So, what is output Y? Simply I0; 0 1, I 1; 1 0, I 2; 1 1, I 3; this is the truth table. This is a different type of truth table than the conventional truth tables of the combinational circuits. Now, what is the Boolean expression Y? What is the internal circuitry of this one? What is the circuit diagram?

So, for that Y is equal to so this can be written as Y is equal to 1. If these two are 0, I0, output should be 0. So, this is S1 bar S0 bar I0. So, if these two are 0's, 0 bar 0 bar becomes 1, output will be I0 plus S1. S1 is 0, so S1 bar; S0 is 1, so S1 bar S0 I1 plus this is S1 S0 bar I2 plus S1 S0 I3.

This is the Boolean expression for a 4 by 1 multiplexer ok. Circuit diagram will have two inputs we can generate from this S1, S0 are the inputs control signals and four inputs I0 I1 I2 I3 S1 bar S0 bar can be generated through the NOT gates. We can call this as a wire W1 W2 this is S1 bar this is S0 bar then this circuit diagram is simply.

So, we have 4 AND gates 3 input AND gates and one 4 input OR gate this is Y here we have this S1 bar S0 bar I0 and so on. So, what will be this Verilog code corresponding to this one. So, overall this is the multiplexer module 4 by 1 mux. So, what are the overall outputs and inputs, output is only 1, inputs are I[3:0], S[1:0] and output y input I[3:0]. S[1:0] then you have to define two wires W1 comma W2.

So, what are these wires? NOT G1 W1 comma S1, NOT G2 W2 comma S0. Then we have four AND operations and G3. So, the inputs are this S1 bar S0 bar. S1 bar is nothing, but W1 this is W2 this is W1 this is W2 ok. So, simply we have G3 the inputs are W1 W2 I0 output also you have to define here also some wires ok.

You call this wires as S we have written some P you call this as some P0 P1 P2 P3 ok. So, you have to define wire here also P. w2 comma P[3:0]. So, output is P0 inputs are W1 W2 then I0. Similarly, and G4, output is P1, inputs are w1, S0, I1 and G5 P2 S1, W2, I2.

This is S1 and G6 P3 is the output. Inputs are S1, S0 and I3 and then Y is equal to or of this, or G7 output Y final output is P0 comma P1 comma P2 comma P3 end module. This is the

design of a 4 by 1 multiplexer and corresponding Verilog code ok. Here also if you want to I mean construct the larger multiplexers; we can use this smaller multiplexers.

Using this 4 by 1 multiplexer as you can construct 8 by 1 multiplexers or if you have 2 by 1 multiplexers we can construct 4 by 1 multiplexers ok. So, how to construct the larger multiplexers using smaller multiplexers? How to instantiate this smaller multiplexers? We will discuss in the next lecture.

Thank you.