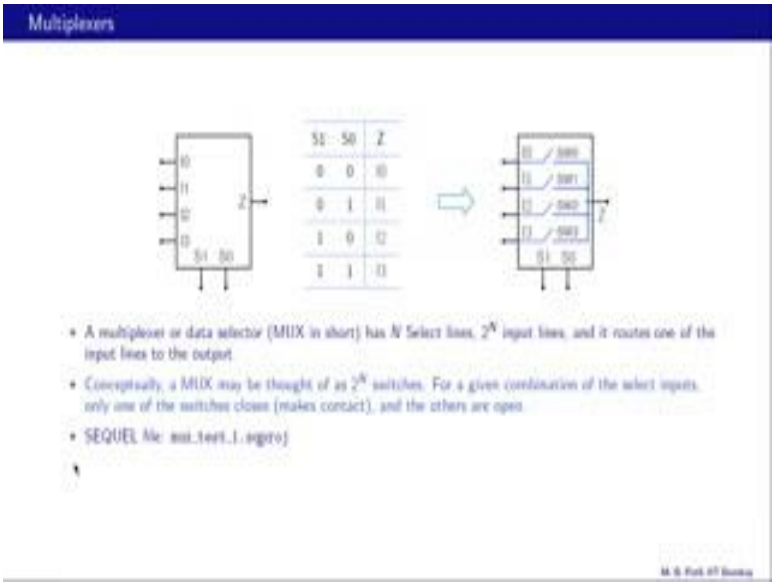
Basic Electronics
Prof. Mahesh Patil
Department of Electrical Engineering
Indian Institute Technology, Bombay

Lecture – 59
Combinatorial circuits (continued)

Welcome back to Basic Electronics we are looked at logic gates so far in this lecture we will look at a more complex block called the multiplexer or MUX in short. We will describe the basic functionality of a multiplexer with the help of a simulation example. Multiplexers can also be used to implement logic functions and we will see how that can be done. So let us start.

(Refer Slide Time: 00:46)

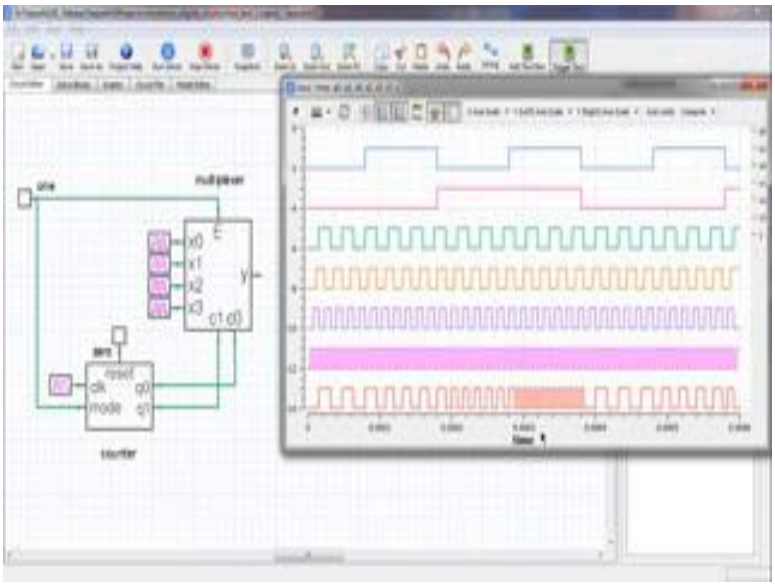


We have looked at basic digital gates, NOT, AND, OR XOR. And using these gates we can make up more complicated blocks. And let us start with this block called the multiplexer. Here is an example. In this case we have 4 inputs i stands for input to select lines S stands for select and 1 output and here is the truth table if S1 S0 0 0, what is 0 0? That is decimal 0. Then the output is I0 so; that means, whatever a data we have here we will appear at the output. If S1 S0 is 0 1 that is decimal 1 then the output is I1 so the data from here will now appear at Z. If S1 S0 is 1 0 that is decimal 2 then Z is I2 if S1 S0 is 1 1, that is decimal 3, then the output is I3 so this is the functionality of the multiplexer. And now let us make some more comments about it.

A multiplexer which is also called a data selector or MUX in short has N select lines. In this case N is 2 because we have 2 select lines, 2 raise to N input lines in this case we have 2 raise to 2 that is 4 input lines. And it routes one of the input lines to the output so if S1 S0 is 0 0, then this line is routed to the output. If it is 0 1 then this line is routed to the output and so on.

And conceptually we can think of a MUX as 2 raise to N switches has shown here. For a given combination of the select inputs only one of the switches closes that is makes contact and the others are open. So this is the conceptual data of a multiplexer. If S1 S0 is 0 0, then this switch called SW0 will close and this line will then get routed to Z if S1. S0 is 0 1 then this which SW1 will close and then I1 will get routed to Z and so on. So that is how we can think of multiplexer conceptually.

(Refer Slide Time: 03:38)

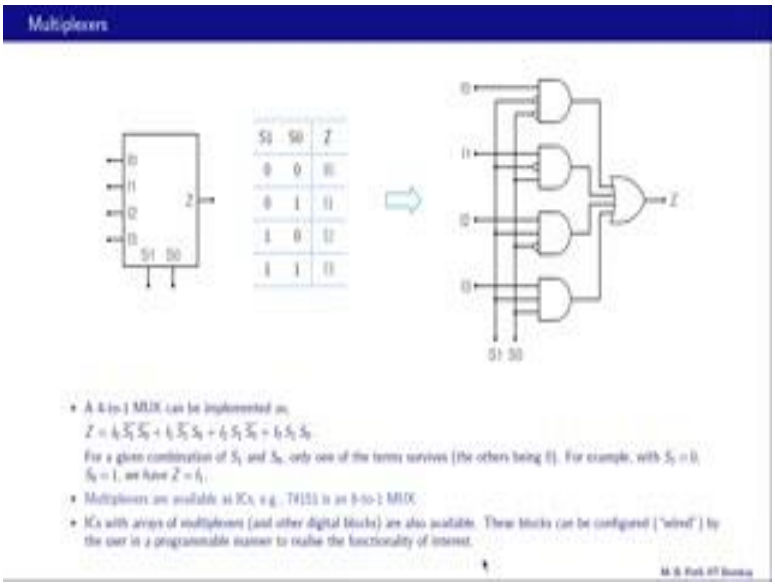


All right let us now look at an example using this circuit file. So here is an example. What we have done here is we have generated 4 different input signals X0 X1 X2 and X3. So these correspond to our input lines I0 I1 I2 I3 of the multiplexer. X0 is here it is a square wave with a certain frequency. X1 is also square wave with a higher frequency. X2 the frequency is still higher and X3 it is one higher. So these are the 4 input signals this c1 c0 or our select lines S1 S0. This q1 is connected to S1 and q0 is connected to S0. And this is a counter we will study this circuit in detail later. For now, it is only purpose

is to generate these select line signals. And those are shown over here. This is our select line 0. What is c0 here? And this is our select line one that is c1 here.

All right, now let us see what happens in each of these intervals from there to there. In these intervals our select lines are changing. Here we have S0 equal to 0, S1 also equal to 0. So what do we expect? We expect that the input line 0 would get connected to the output; that means, this signal should appear at the output, and if you look at the output that is exactly what we see. What about this interval? We have S0 equal to 1 S1 equal to 0. So the binary number is 0 1 that is decimal 1 and I1 will then get connected to the output. And where is I1 here? This is I1. So that will appear at the output. Similarly, when S1 S0 is 1 0 then I2 appears at the output. And when S1 S0 is 1 1 then I3 appears at the output. So that is how a multiplexer works.

(Refer Slide Time: 06:19)

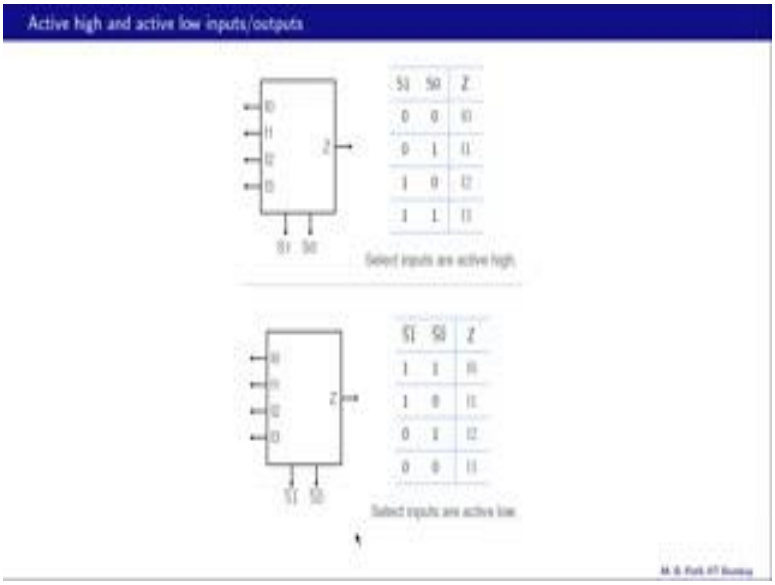


Let us now look at the implementation of multiplexer. And we will take the same example again. Which is a 4 to 1 MUX? Why is it called 4 to 1 because it has 4 input lines and 1 output line all right? So here is the implementation. These are and gates and these circles indicate the not operation. So for example, let us take this gate it has got I1 as one of the inputs, then it has got S1 bar. S1 bar because we have the circle here and then it has got S0. So the output of this gate is I1, S1 bar S0. And the outputs from these and gates are connected to this or gate and that gives us the output.

And if we write a logical expression for Z using this implementation, then we arrive at this expression I0 S1 bar S0 bar plus I1 S1 bar S0. The first term of course, is coming from this gate. The second is coming from this gate and so on. Plus, I2 S1 S0 bar plus I3 S1 S0. And for a given combination of S1 and S0 only one of these terms will survive. The others will be 0. For example, with S1 equal to 0 and S0 equal to 1 that is S1 bar equal to 1 and S0 equal to 1 this term would be one and all others would be 0. So in that case we will have Z equal to I1 and it with 1 that is Z equal to I1. So that is how the functionality of the multiplexer is implemented.

Multiplexers are available as ICs. For example, the 74151 chip is an 8-to-1 multiplexer; that means, it will have 8 input lines and therefore, 3 select lines because 2 raise to 3 is 8. Also ICs with errors of multiplexers and other digital blocks are available. These blocks can be configured or wired by the user in a programmable manner to realize the functionality of interest. And this style of design is now increasingly popular increasingly common. Because the cost has been coming down and this programmable functionality is a very big advantage.

(Refer Slide Time: 09:14)

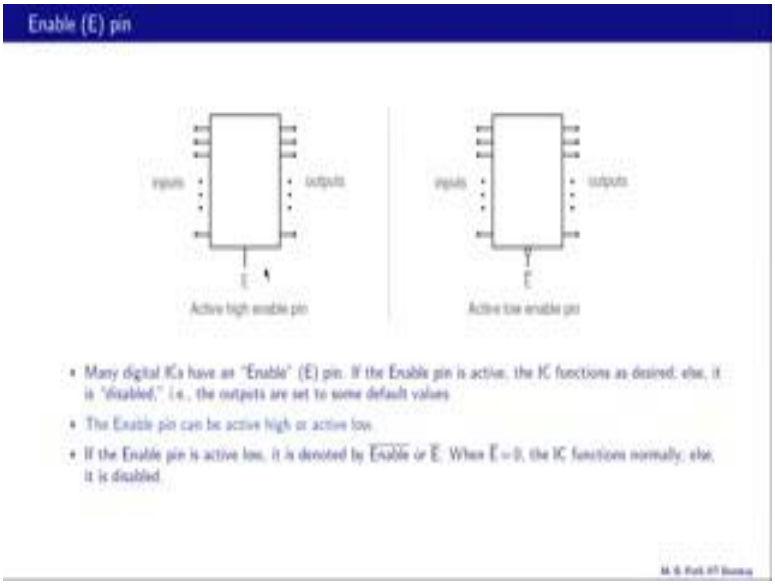


Let us now discuss a very important topic and that is active high and active low inputs or outputs. And we will discuss this topic in the context of multiplexers, but we should remember that this is very general terminology and in fact, we will see this in several other digital circuits as well all right. So here is a multiplexer with this truth table here.

In this case the select inputs are active high; that means, if this S0 is a high voltage, and then it is considered to be 1. If it is low such as 0 volts then it is considered to be 0 and so on.

And the functionality is then described by this table. Let us consider this second example in which the select inputs are active low and they are denoted by S1 bar and S0 bar rather than S1 and S0 here. Now if S0 bar is 0 or low then it is considered to be active. So for example, if S0 bar is 0 here that corresponds to S0 equal to 1 in this table and similarly for S1 bar. If S0 bar is 1 then that corresponds to S0 equal to 0 in this table. So in other words this second table is obtained from the first table by simply inverting all of these entries; so that is how this active high and active low terminology words.

(Refer Slide Time: 11:11)

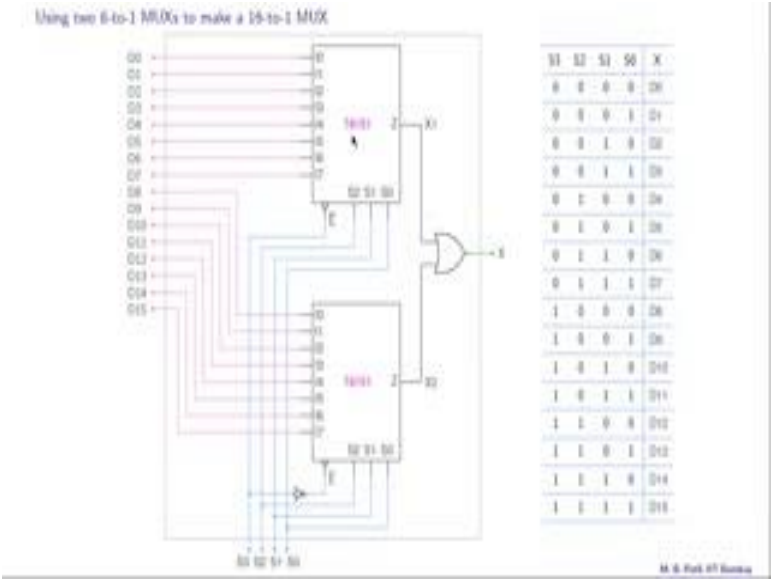


Let us now discuss something called the enable pin which is very common in digital circuits. Here is an example we have an IC with some inputs and some outputs and then there is this pin called the enable pin or E. In this case it is an active high enable pin. Let us see how it works. If the enable pin is active the IC functions as desired. For example, if this is a multiplexer and this enable pin is 1 then this IC will actually function as multiplexer. If the enable pin is low then the IC is disabled; that means the outputs are set to some default values.

So in order to make the IC work as desired we need to make the enable pin active. The enable pin can be active high or active low in this case it is active high and here is an

example where the enable pin is active low, and that is why it is denoted by E bar. If the enable pin is active low, it is denoted by enable bar or E bar. And in that case when E bar is 0 that is active the IC functions normally, else it is disabled; that means, the outputs will be set to some default values.

(Refer Slide Time: 12:45)



Let us now take up this problem in which we have 2 8-to-1 multiplexers and we want to make up a 16-to-1 multiplexer using these 2. So here is the 8-to-1 MUX 74151. It has got 8 input lines I0 to I7. 3 select lines S2 S1 S0 and it also has an enable pin. And this enable pin is active low that is why it is denoted by E bar. So we have 2 of these and using these 2 we want to make up a 16-to-1 multiplexer. So let us see how to go about doing that.

So, here is the truth table that we expect from our 16-to-1 multiplexer. These are the select lines and that is the output. So this entire box is supposed to function as the 16-to-1 multiplexer. These are the input lines from D0 to D15. These are the select lines S3 S2 S1 S0 and that is the output X. When S3 S2 S1 S0 is 0 0 0 0? We would like X to be D0. When these are 0 0 0 1 then we want X to be D1 and so on. All the way up to 1 1 1 1 in that case we want X to be D15. What we can do is split this functionality into 2 parts. We can implement this first part with one MUX, one 8-to-1 MUX; and the second part with a second 8-to-1 MUX. And we note that this select line S3 is 0 for these first 8 entries and it is 1 for these second 8 entries. And that is something that we can use to enable either

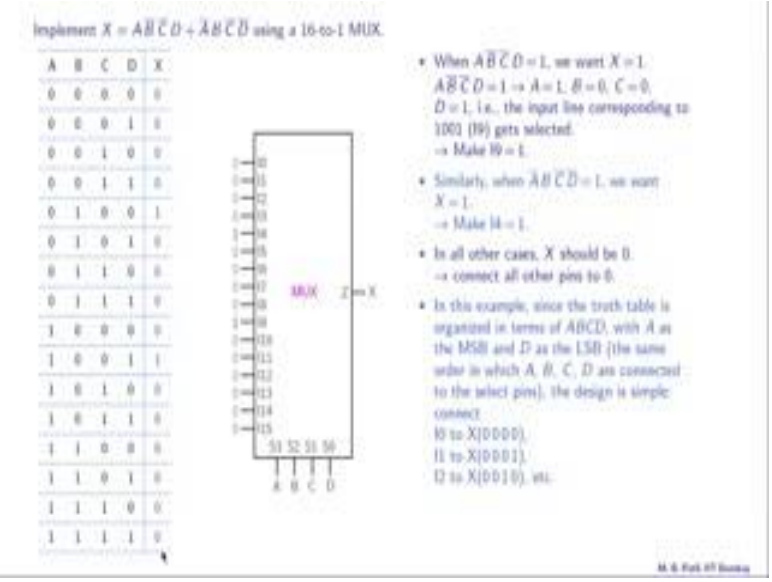
the first multiplexer this one or the second multiplexer this one. And that is what we have done in this implementation. S3 here is connected as E bar of the first MUX and S3 bar is connected as E bar of the second MUX.

When S3 is 0 then this MUX is enabled. It works as a multiplexer. This MUX is not enabled. So we get X2 equal to 0. And the output is the or of X1 and 0, that is X is equal to X1 in that case. Now when S3 is equal to 1 then these ICs enabled it works as a multiplexer. This one is not enabled, so X1 is made equal to 0 and the output is the or of X2 and 0 that is X becomes equal to X2.

What we will do now is to consider 2 cases: one corresponding to this first part of the table and second to this second part of the table. And in each case we will verify whether this output X is what we would expect from this truth table all right. Let us take this case first in which S3 is 0, S2 is 0, S1 is 1 and S0 is 1. So if those are the inputs applied here we expect D3 at the output and let us see if that happens. Since S3 is 0 this MUX is enabled this MUX is not enabled. So this X2 is made equal to 0. And our X is equal to X1 as we have seen before. Now what are the select lines for this first MUX? S2 S1 S0 that is 0 1 1; so we have 0 1 1 here that is decimal 3; that means, this line I3 gets connected to Z here. And what is line I3? Line I3 is the same as D3. So that D3 gets connected to z. So our X1 is equal to D3 and therefore, X which is equal to X1 is also equal to D3. So we do have then the expected output in this case.

Let us now pick one case from the second part of the table. Let us say this 1. So what is S3? S3 is 1. So these ICs enabled. This one is not enabled and X is equal to X2 all right. And what are S2, S1, and S0? We have 1 1 0. So this IC gets 1 1 0, 1 1 0 is decimal 6. And therefore, this line I6 will get connected to Z. What is I6? Let us look at that. I6 is the same as D14. So therefore, X2 and also X will be equal to D14. And that is what we would expect. So this whole implementation works as a 16-to-1 multiplexer.

(Refer Slide Time: 18:38)



Let us now look at implementation of a logical function using a multiplexer. So here is a 16-to-1 multiplexer. It has input lines from I0 to I15. That is 16 input lines. And 4 select lines S3 S2 S1 S0. And of course, one output. Now using this multiplexer, we want to implement this function X equal to A B bar C bar D plus A bar B C bar D bar. Let us see how to do that. Let us first prepare the truth table for X, and step number one would be to enumerate all the input combinations and that is what we have done here.

And a systematic way of doing that once again is to allow D to change every 4 entries, C to change every 2 entries, 0 0 1 1 0 0 etcetera, B to change every 4 entries 0 0 0 0 1 1 1 1 0 0 0 0 etcetera, and A to change every 8 entries, so we have 8 zeros followed by 8 1s all right. Now this X has got 2 minterms and therefore, we will have 2 1s in this truth table. Where is this 1 located it corresponds to A equal to 1, B equal to 0, C equal to 0, and D equal to 1, so it is 1 0 0 1, so that is right here, so that is the first minterm. This second minterm corresponds to 0 1 0 0. So that is right here and of course, all other entries are 0.

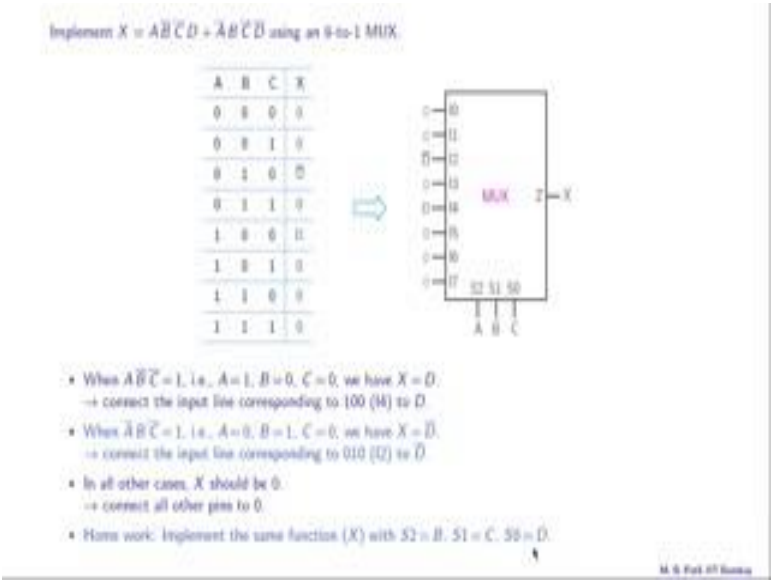
So let us now talk about our implementation. The MUX output is the same as our logical function X. And these select lines are connected like that S3 is A S2 is B S1 is C and S0 is D, and notice that this order is the same as this order here. And that makes it very convenient to make connections to these input pins as we will see. So what is left now is to make connections to these input pins such that we get this fortune at the output. Let us look at the min terms in our function. The first min term is A B bar C bar D. And when

that is equal to 1; we want X to be equal to 1; that means, when A is 1, B is 0, C is 0, and D is 1, we want the output to be equal to 1, that entry is right here. So when that happens when we have 1 0 0 1 here what happens in this multiplexer 1 0 0 1 is decimal 9, so therefore, this I9 gets connected to the output. And we want that to be equal to 1. So therefore, what we can do is to connect 1 over here to this I9 pin.

Similarly, we can treat the second minterm and that minterm is here so A B C D is 0 1 0 0 in that case. Now 0 1 0 0 is decimal 4, so pin I4 gets connected to the output, and therefore, we should make I4 also equal to 1, like that. And these are the only 2 minterms so therefore, in all other cases that is for all other combinations of A B C D the output must be 0. And therefore, we can connect all other pins all other input pins to 0 like that. So that is our implementation of this function.

Now, in this example, since the truth table is organized in terms of A B C D with A as the MSB and D as LSB and that is the same order in which A B C D are connected to the select pins. The design is particularly simple. This order and this order is the same and therefore, what we could do is simply reproduce this column over here and that would work.

(Refer Slide Time: 23:43)



Let us now implement X equal to A B bar C bar D plus A bar B C bar D bar, which is the same as the function that we saw in the last slide using an 8-to-1 multiplexer now.

And here is one possible design. The output of the MUX is the same as the logical function that we want. And the select lines are connected to A B and C. Now this choice is not unique. We can have some other choice, but suppose we go ahead with this one. Let us now figure out what we should connect at these input lines. So as to get the desired function at the output; to begin with let us prepare a truth table in terms of A B C since those are the variables which appear over here. And we will create A as the MSB, since that is connected to S2 and C as the LSB. Since that is connected to S0. And we write these input combinations in the same manner that we described earlier. And with A as MSB and C as the LSB this number corresponds to decimal 0, this one to decimal 1 decimal 2 3 4 5 6 and 7.

Now, to complete this column let us look at our minterms. This first minterm has A B bar C bar. And this part is 1 when A is 1, b is 0 and c is 0. That is 1 0 0. Here and when that happens our X becomes equal to D and this part of course, is then 0. So that is why we have X equal to D over here. Similarly consider this mintern. This A bar B C bar is 1. When A 0 B is 1, C is 0; so 0 1 0 that is right here, and when that happens X becomes equal to D bar 1 handed with D bar that is D bar. So that is why we have D bar here. And in all other cases X is equal to 0.

Since we have only 2 minterms; so that is how we have prepared this truth table. The rest is straightforward. We can now consider these minterms one by one, and figure out what we should connect over here. Let us take this minterm that is right here. So we have A B C equal to 1 0 0. 1 0 0 is decimal 4. So this line I4 should then be connected to D like that. The second minterm is here and in that case our A B C is 0 1 0 that is decimal 2. So line I2 should then be connected to D bar like that. And in all other cases X should be 0 and therefore, we connect all other input pins to 0 like that. So that is our design and once again since this order is the same as this order, we could have actually just reproduced this column over here, rather than looking at each of these minterms once again all right.

Now, there is some homework for you. Implement the same function X with S2 equal to B, S1 equal to C and S0 equal to D. In summary we have looked at a new digital block in this lecture namely the multiplexer. We have also seen how a multiplexer can be used to implement a logic function. That is all for now, see you later you.