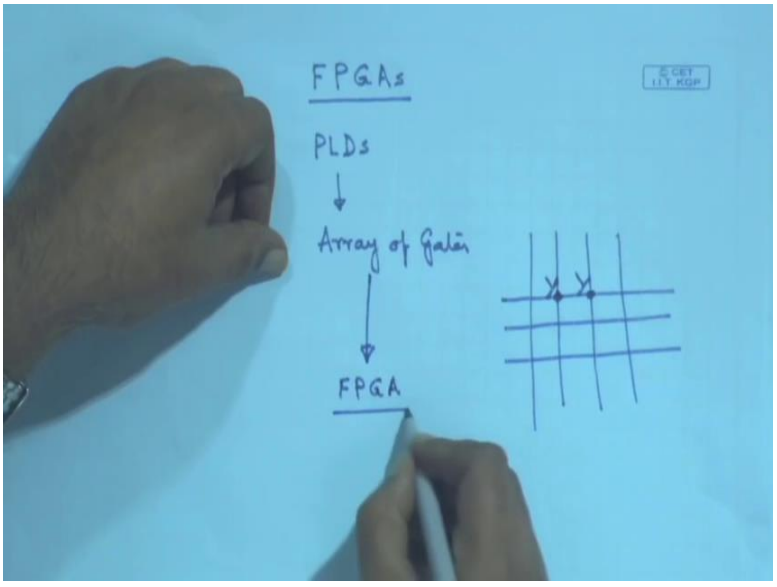
Embedded Systems Design
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 06
Introduction to FPGA

Good morning, we were discussing about the single purpose processors or single purpose chips, single purpose computers; now which are very tightly bound to a particular application. On the other extreme we have got the general purpose processors, where we have got immense flexibility.

(Refer Slide Time: 00:56)



So, in between there are application specific instruction processors, but also we have got another variety which we were discussing in the last class, that is FPGS or field programmable gate arrays.

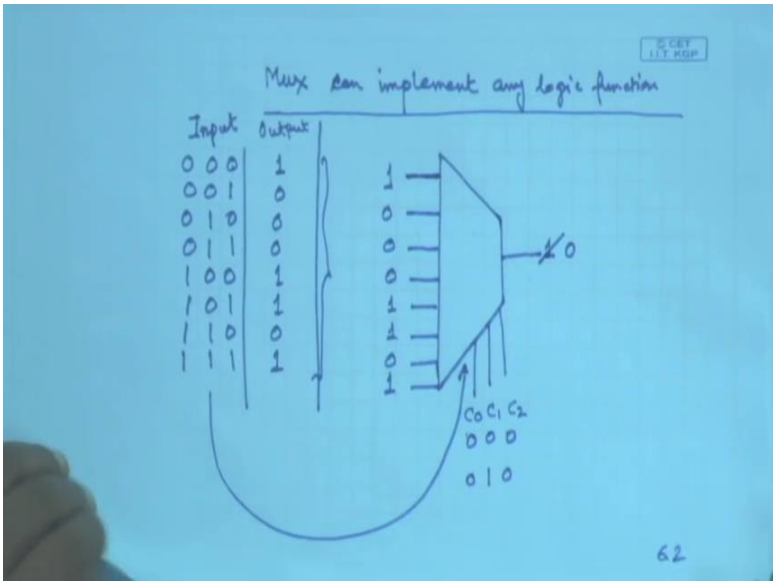
Now, the FPGA's are successors of programmable logic devices PLDs, which appeared initially as array of gates right. So, in the last class we had talked about and array alright, suppose each of these on this plane we have got and gates and on this plane we will have the or gates, and I can program this interconnections. So, this means the where one wire

is going like this, another wire is going above this. Now if I establish a link between these 2 they get connected. So, although it is showing that there intersecting there still now not intersecting, but I can make them connected by some switch right if I connect them then this point gets connected this gets connected.

Now we can when we talk about programmability, that programming is essentially in establishing such interconnections. As we had discussed that we can implement any logic function with and, or, and not right. So, therefore, I can have a canonical form of a Boolean expression and divide and write it down as a sum of products formed. So, the products are essentially the ands and the sums are the ors, therefore I can implement a Boolean function using that.

Now, that has been further extended to much more powerful FPGA's where we have got the possibility of configuring different logic circuitry on a single chip. Now FPGA's before going into the details of FPGA's, we will discuss about some basics of digital logic that we had all of you have done at some point of time. Now all of you know that nand gates are universal gates. So, you can implement any logic function using a nand gate, similarly with nor gates right. Now if I say multiplexer is a universal gate, would you agree?

(Refer Slide Time: 04:09)



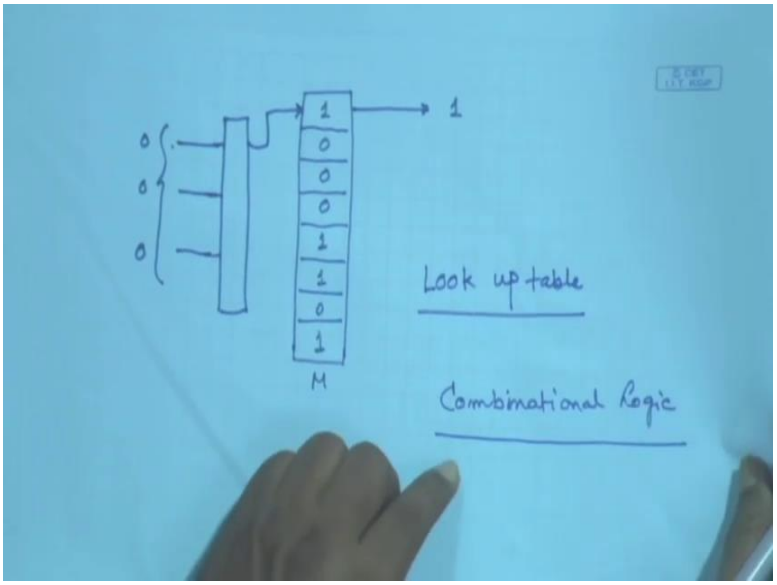
If I say multiplexer can act; I should not say is a universal gate, but max can implement any logic function, is it true or false? It is true because say I have got a 2 input or say 3 input Boolean functions. So, 0 0 0 and I will have some output. So, I will have 1, 2, 3, 4, 5, 6, 7, 8 combinations and corresponding to each of these combinations these are the input combinations, this is the input corresponding to this input combination, I will have different outputs, say my output is something arbitrary I am writing 1 0 0 0 1 1 0 1 say implementing some logic function.

So, I have got 3 inputs, and I have got ones and zeros 8 possible outputs. Now if I select the multiplexer with these input lines serving as 3 control lines. C 0, C 1, C 2 alright C 0, C 1, C 2 and all these 8 outputs coming as all these coming as 8 input lines; 4 5 6 7 8; 8 input lines. Now I can put in this as 1, this as 0, this as 0, this as 0, this as 1, 1, 0 1 and here is my output line. Now depending on whatever the input is, if the input is 0 0 0 that means, this input line will be propagated to the output. So, I will have the output to be 1, if this be 0 0 0. If this be 0 1 0, if I make it 0 1 0 the control lines; that means, the third input will be propagated to the output. So, then this will become 0, in that way any truth table I can implement using a multiplexer. This is one of the concepts that will be requiring while we work with FPGAs.

Student: (Refer Time: 07:50).

These of course, depending on if you want to minimize that if you want to; if you once you do the logical minimization, then you can certainly do that you can transform it. But it is the straight forward implementation; when here of course, I have not thought of any optimizations. The other possibility could be that suppose these inputs are essentially address lines, suppose I have got an 8 location memory; 8 bit memory.

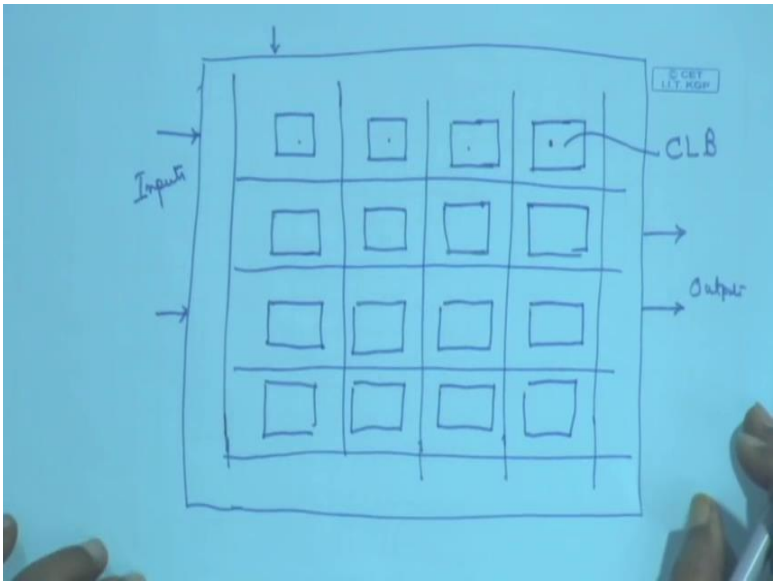
(Refer Slide Time: 08:39)



So, I have got a memory single bit memory, with 8 possible locations; it is the memory M, each location has got single bits 2 3 4 5 6 7 8. Now just like here I can have this ones and zeros stored in this memory location, I can have it 1 0 0 0 1 1 0 1 and these input lines for 8 locations of the memory I need 3 address lines. So, if I have these 3 address lines coming up here, say the address lines at 3, where the input is coming in and from here I establish a decoder and that is selecting any one of these locations depending on this combination and I will read from that location. So, if it be 0 0 0 then I will get 1; if it be 0 0 1 then I would be getting 0 like that.

So, same thing can be used using this sort of memory which we will call as lookup table. So, as if we have got a table, where all the outputs possible outputs are stored in the proper combination of the input, then as that input arrives I will get that output clear. So, that is also another way by which we can implement the combinational logic. So, by this or by the marks or by this lookup table, I can implement the combinational part of the logic.

(Refer Slide Time: 11:09)

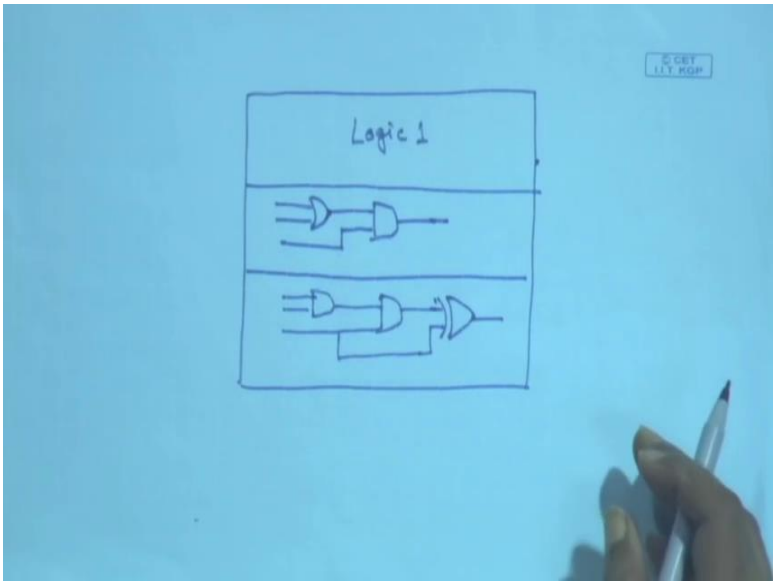


Now in the last class we had seen that an FPGA is nothing but I had drawn a diagram where I had shown that there are a number of configurable logic blocks. Suppose we have got such an array of logic blocks each of them is known as the CLB or configurable logic blocks, and they are surrounded by a number of lines. I am just showing 1 line, but it is actually much more through which I can establish interconnections.

Besides that now I will establish interconnections along this lines, and around this entire chip we will have the inputs, if I just draw a boundary of this then I will have the inputs coming here the input ports, the output pins will go come out from here; there will be clocks and all those things will be coming in around this we will show that in a second.

Now, one of the interesting features of such FPGA's is that; that by establishing the interconnections of this configurable logic blocks, I can establish different functions.

(Refer Slide Time: 11:29)

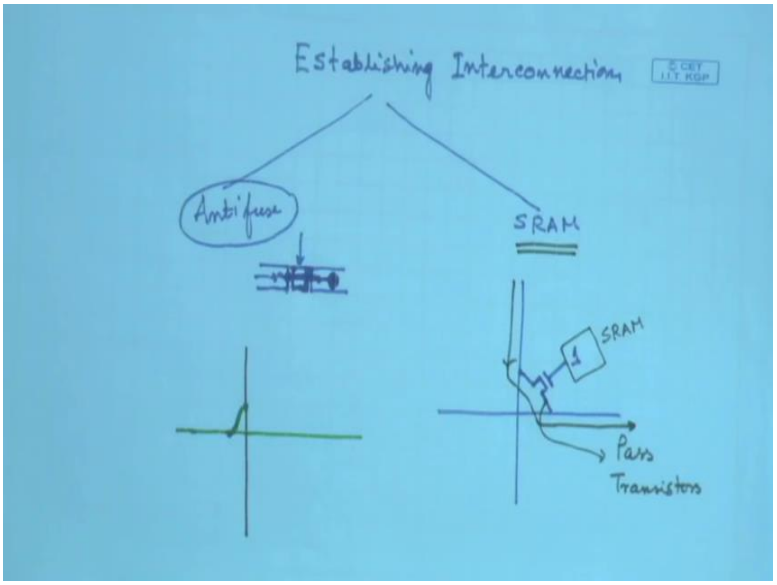


Consequently what can happened is that in us I can divide 1 FPGA into a number of slices, maybe this slice is implementing some logic 1, this slice may implement another logic for example, say its establishing a logic like this, alright and this slice is implementing another logic, some logic is coming here maybe I am implementing a XOR. Now the point to note is since the same chip different parts of the chip different parts of the chip can be assigned different functions, I can implement them; they can all act in parallel, therefore I can get some speed up.

On the other hand if I looked at a general purpose processor, typical sequential general purpose processor where I am executing a C code that C code will be sequential and any hardware for that matter can be executed in parallel if I have got parallel data paths activated through the controller; and the same can be achieved with in an FPGA. So, quickly collecting up the things that we have discussed till now, we will have established this interconnection of the programmability will come from this interconnection right and we will take help of the basic phenomenon of this basic power of multiplexers as well as the look up tables or the memories.

And will establish interconnections through this now. So, these are the now interconnections, how do you carry out the interconnections between this?

(Refer Slide Time: 15:49)



Establishing interconnections; there are different techniques for establishing interconnections: one is Antifuse, another is known as SRAM based now antifuse. So, what do I want to do? I want to I have got 1 wire like this, another wire like this, there are 2 different colors and one is they are not connected; I want to establish a interconnectivity. So, I can have a connection established here right, how can I establish it here? In antifuse what is done is there are there is a conductor, one part of the conductor is just like a switch; if I close the switch then these are connected; if I open this switch that is not connected. So, I have got 2 pieces of conductor and in between there is an insulator therefore, this part and this part are disconnected. Now if I apply voltage this insulator, actually breaks down and this becomes conducting therefore, by applying suitable voltage I can make it conduct and I can establish the connection between them alright. A fuse wire what happens, when high current goes the fuse wire snaps, here when I apply this it actually join this two, that is why it is called an antifuse; this one way it can be done.

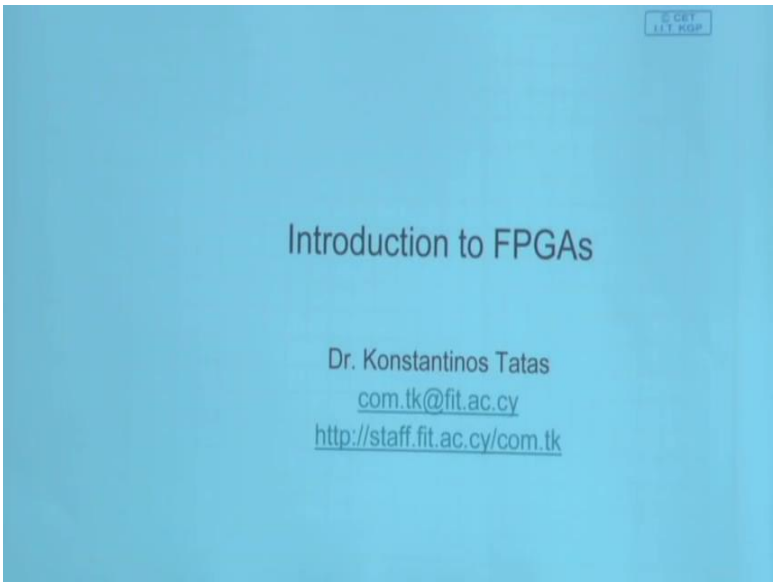
Now, antifuse technique since I am just like a fuse wire if its snaps, you have to replace another wire. Similarly in antifuse if I fuse antifuse it if I establish the connection that is permanent. So, I cannot at will change it remove the connection or establish the connection. On the other hand the SRAM plays thing is something like this, that here

between these 2 I have to establish a connection and I have got a transistor here alright. Now if I apply a 1 here, then this transistor will conduct. So, this transistor are known as pass transistors and whether I will give 1 or 0 that this pattern I can keep in some sort of a 1 bit RAM right.

So, these programmable I can write a 1 here lot of 0 here, if I write a 1 here then this pass conduct sorry pass transistor becomes conducting, there for any signal that is coming through this path will go through this path and if I make it 0, then obviously, this connection is snap therefore, such interconnection is programmable. At will I can establish the connection or can withdraw the connection. So, SRAM is very popular and used and fuse is of course, not programmable, but that is another way we can establish the interconnection.

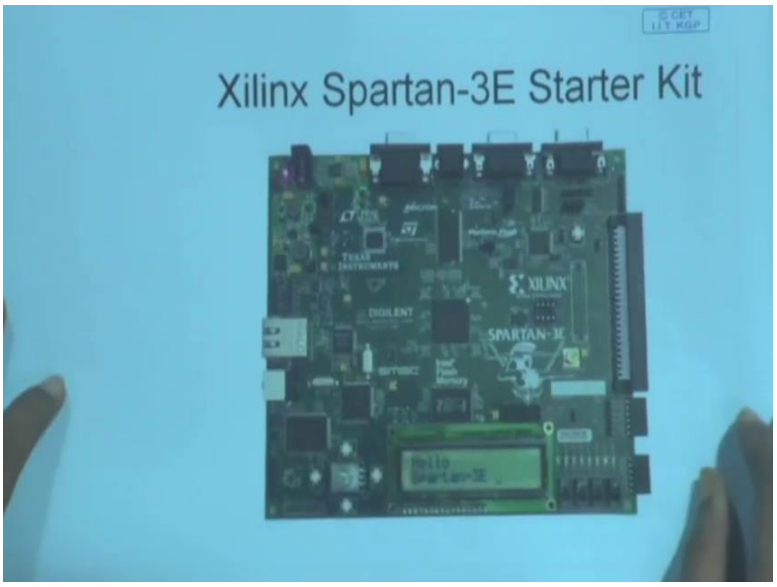
Given this background will now move 2 discussions on FPGA's and this part of the lecture I will borrow heavily the slides from Dr Konstantinos Tatas taken from the internet of course his acknowledgement, you can also find the slides in this site.

(Refer Slide Time: 20:55)



So, we will briefly look at I told you in the last class that there are 2 major manufacturers of FPGA's: one is Xilins another is Actern alright there are others also and Xilinx them itself makes different varieties of FPGAs.

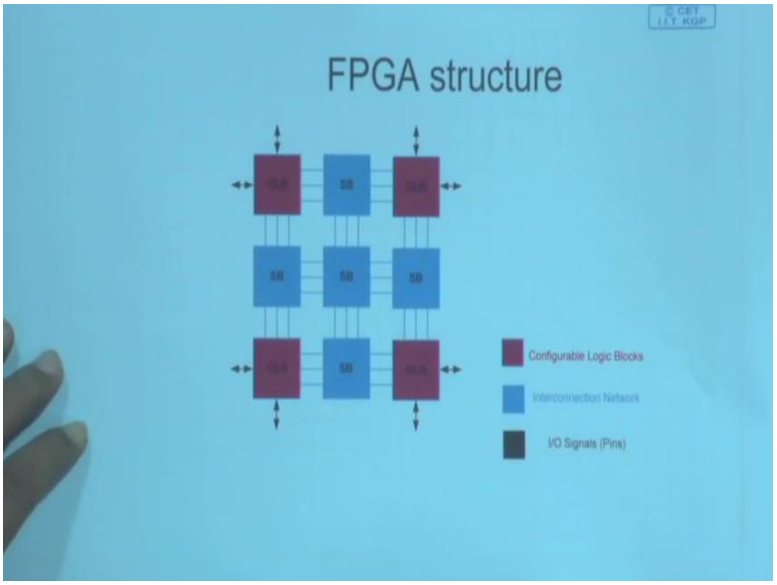
(Refer Slide Time: 21:35)



So, one of the FPGA's FPGA board is, it just like a board here you can see you can see this right. It is a board, it is a Spartan 3-E board, where you can see the FPGA here it is a chip, this is the FPGA alright and this is what we will decompose it will break it up soon, but right now this is a chip and here you can see we have got so many things these are some LED array; there are some LEDs, here these are some switchers these are some switchers we can say here you can see the input output pads, these are connected with which I can connect to the other outer world; and in the present we have got much more complex FPGA boards being made now, which have got some signal processing activity, some memory all those things come up over here alright.

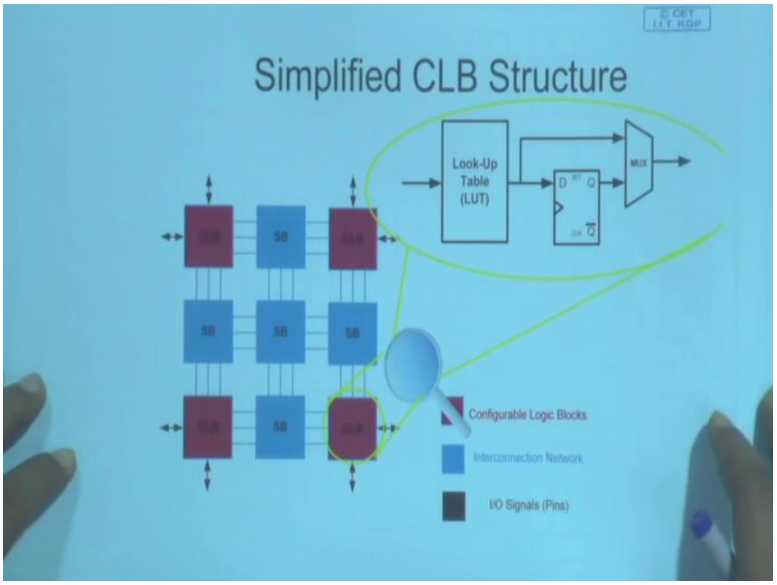
So, this is one typical board and I would like if you do some development on such FPGA's now, let us start our journey. Here is the FPGA, now what is there inside? Now depending on the complexity of the FPGA's that if inside will vary.

(Refer Slide Time: 23:20)



But a typical FPGA's structure is as shown here; the red ones are the configurable logic blocks. So, here I am showing 4 configurable logic blocks and as was shown in the earlier diagram that there are wires all around, going all around like this the wires are going all around and I have to establish the wires; in order to establish the connection, we are given some switch boxes we call them switch boxes, which are here alright. These are interconnection networks and there are IO signal pins, there is the overall layout with 4 CLB systems. Now what is there inside a CLB this clear? So, we have got the switch boxes and the CLBs.

(Refer Slide Time: 24:46)



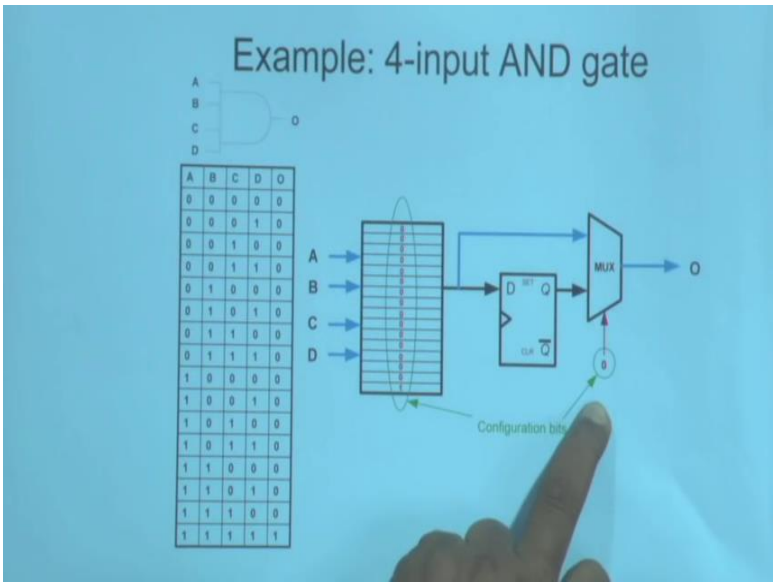
Now, what is there within this CLB? If I take 1 CLB and blow it up, will find that this is a very simplified CLB, very overly simplified, actual the CLBs will be a little more complicated now. Let us try to understand what is there, we have got a lookup table which we call LUT, look up table and we have got inside a CLB say (Refer Time: 25:24) flip flop. Now and there is a multiplexer.

So, whatever logic I implement will soon implement a logic, whatever logic we implement we can implement it to the lookup table we have explained a couple of minutes back and the output of that lookup table I can directly take out or I can take it at the next state therefore, I can put it in the D flip flop, and it can be remembered in the D flip flop alright and all of you know that the D flip flop will have whatever it is the input; now will be made available at the output after 1 clock delay. So I can establish a delay by this. So, and I can activate the control of this multiplexer to either select this input or select this out input any one of them can come alright is it clear.

Let me hold it for a while so that those of you want to draw it, can draw this. This is a lookup table and we have seen what a look up table is, a lookup table can contain the ones and zeros as we were showing there alright we were showing a couple of minutes back, what the lookup table contains. So, it was like this, the look up table where I can

put the inputs and output of that table will come out right. So, given this the same diagram I hold onto this diagram and we will try to implement one simple very simple logic function that is a 4 input and gate using this.

(Refer Slide Time: 27:53)



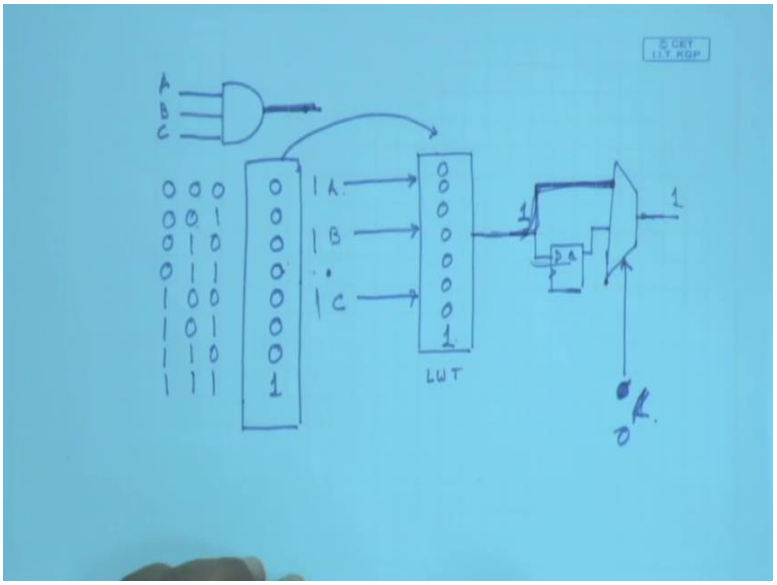
Simplest possible 4 input and gate. How can I implement the 4 input and gate? Simplest possible logic function A and B and C and D; so I have got these 4 inputs A B C D and here is a truth table for that. For all 0 0 0 0 0 0 I will have 0 0 0 0 1 will be 0, only when everything is 1 I will have 1, that is the logic function of 4 input and gate. Now I have to get this through my CLB, suppose this I want to do through only one CLB, I will just need one CLB to do this and what do I have within the CLB? I have got a look up table and a D flips flop, for this simple CLB.

So, now if I feed A B C D as the address of this look up table and this lookup table is programmed to have only this bit to be one it is probably not visible, all these are 0 0 0 0 only this one is 1, because here only this one is 1, this is output combination right. I program this with the desired output, with respect to this given inputs and I take that output here that output will come out through this and the output will also be available at the input of the D flip flop. Now as an and gate it is a combinational circuit therefore, the output will immediately follow the input. So, whenever these inputs come, this look up

table will feed that particular 1, 0 over here and then I have to select this path from the multiplexer therefore, I have to feed this multiplexer with a 0 clear not clear.

So, let me once again do this good that you said it is not clear.

(Refer Slide Time: 30:17)



So, I want to implement gate 4. Let me say I will even make a simplified I will make it 3 input gate, 3 input and gate and the output will come. So, what is the desired what is the combination? 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0 ,1 1 1; 8 and I will have the outputs to be its is an and gate. So, only 1 will be in all the inputs are 1. So, all these will be zeros, this will be 1 right. Now I just take this and feed it program it in the lookup table look up table. So, the lookup table will now have 0 0 0 0 0 0 0 1 right and at my input will be whatever the logic input is at A B C. So, they are coming in here; obviously, I am showing it directly coming here, but they are just coming as an address.

So, it will coming (Refer Time: 31:50) so A B C. So, what will be the output of this some bit coming from here? Now this one I want as an output, I want an output and what does an and gate do? And gate is combinational. So, whenever these patterns say 0 0 1 appears here, immediately the output should be 0. If this changes to 1 1 1 it should immediately change to 1, no delay, no delay at all therefore, this is coming to the multiplexer giving

me a choice of either selecting this or these also coming through a D flip flop with the clock input and this is also coming here. Now I have got the choice of either selecting this now, or I want to have it 1 clock later what would I like to do?

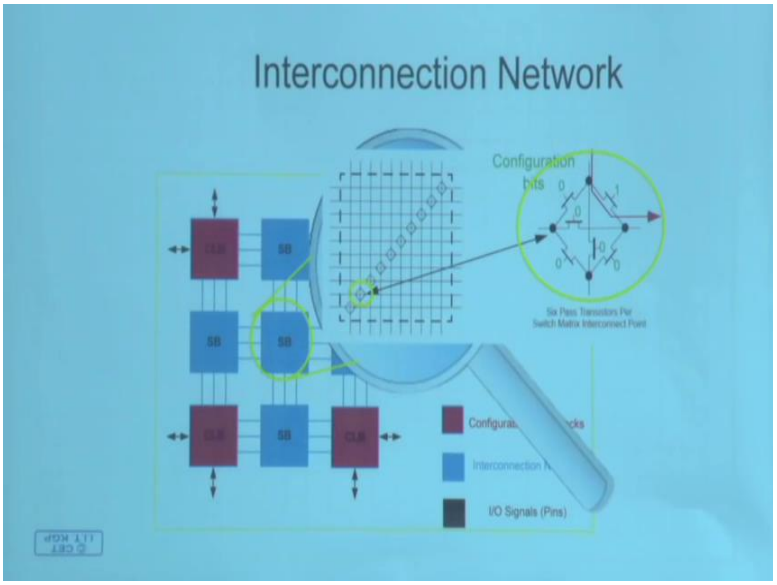
Now I want it now; therefore, which input would I select this or this, I select this directly in order to select this, what should I feed at the input of the multiplexer? I should feed a 0. Had I given a 1 here that would have meant that this is going, this is being selected and as soon as whenever I give say for example, I gave here 1 1 1 then my output is 1, it has come here one, but if I make this 1, what will happen? I made this 1, it will be 0 till the next clock right, but I want it immediately. So, this 1 I want here immediately therefore, I make it 0 clear.

Therefore, in order to get the logic circuit logic function implemented, what I needed is 2 parts of the programming, what are those? One is configuring the lookup table, and configuring the multiplexer control; so these are known as the configuration bits alright one part of the configuration beats, how I configure the CLB?

Student: (Refer Time: 34:38).

Because sometimes I may like to have it I can also try to implement some sequential circuits using. So, that I can 1 I may like to put delays, I would like to memorize this here, all those things I have I can do using this D flip flop, but right now for this simple and gate I do not need this path at all. So, I am by passing it, how do I bypass it making it safe.

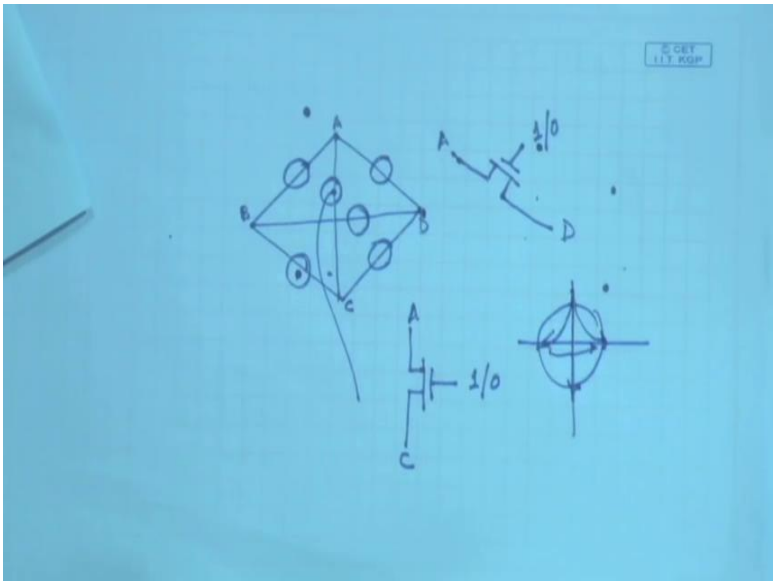
(Refer Slide Time: 25:12)



So, now after having a brief look at the configurable logic blocks, now we will have a look at the interconnection networks. Here we are trying to amplify one particular switch box, what is being shown here is a number of switches are, number of lines are coming in this way and there are several possible intersections. There many other intersections here also; at each intersection, at each intersection how many possible routes are here, how many possible routes are here? You can connect this to this, you can connect this to this like that or you can connect diagonally.

For example; so I have got.

(Refer Slide Time: 36:16)



So, I will just make it a little larger, I have got the way of connecting these to A B or B C A D, C D, B D got this therefore, at each of these junctions if at each of these points 1 2 3 4 5 6 each of these points if I put a pass transistor with a 1 or 0. So, any 2 points say this is A this is D; if I make it 1 that means, A and D will be connected. Similarly for this between A and C, I will have a pass transistor if I make it 1 and words 1 or 0 accordingly A or C will be connected. Similarly for this here I will have another pass transistor clear.

So, I have got this switch matrix here, this switch matrix here, where here what has been done? I have made this to be 1 and all others are 0s; that means, which 2 points are connected? This point and this point is connected others are not connected right. There by I can establish the interconnection between this. So, for this switch box what is my program? My program will be 0 0 0 0 0 1 right 6 bits, because of this 0 0 0 0 0 1 if I make it then this and this are connected. If I make it 1 0 0 0 0 0; that means, this and this are connected right am I clear? Thus by again by programming I can establish interconnections at my will.

Student: (Refer Time: 39:08).

Whenever that the line the; if you look at the array, then here I have got the lines coming. Now I may like to that the connection is in this direction or the connection is in this direction or the connection is in this direction or the connection is in this direction. So, for each junction in this case I am reading 6 (Refer Time: 39:36) because of this sort of possible interconnections. I am requiring 6 bits here just for this sort of you can have, you can vary the type of interconnections.

So, tomorrow you will take up another example that will be programming with FPGA and proceed with further details.