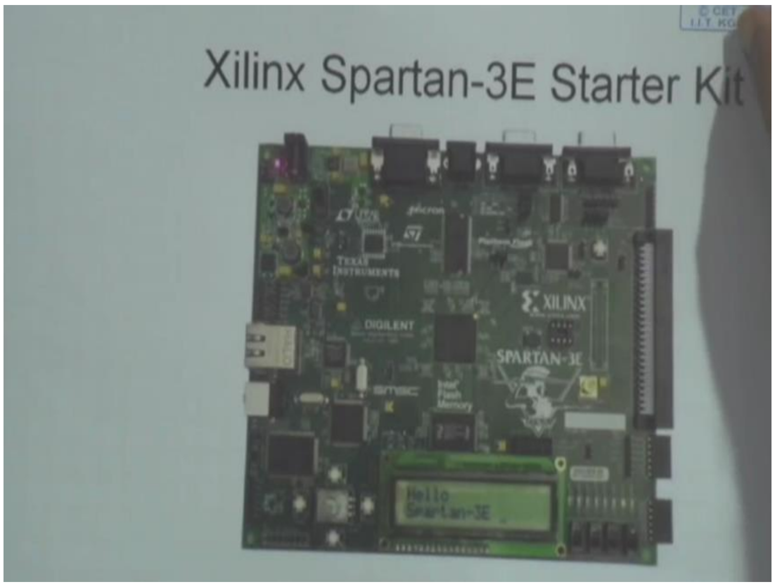
Embedded Systems Design

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

Lecture - 07
FPGA (Contd.)

In the last class we were discussing about this typical FPGA board which is Spartan of Xilinx.

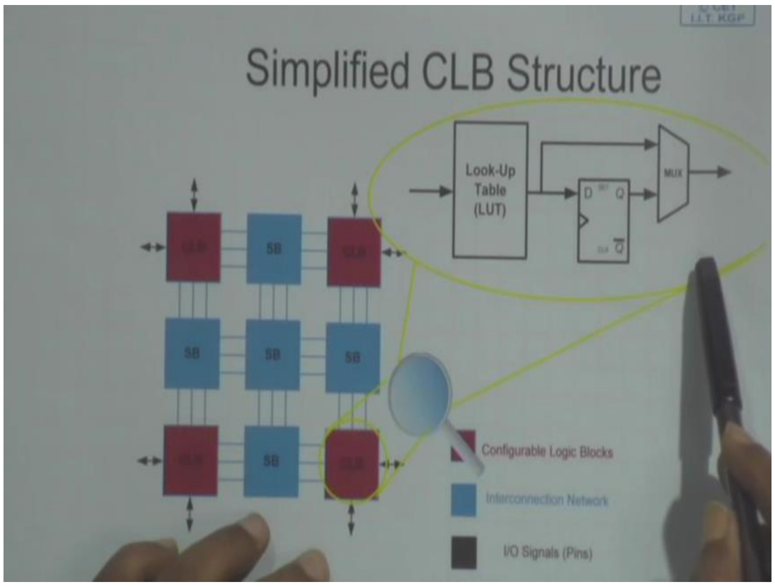
(Refer Slide Time: 00:25)


Xilinx Spartan-3E Starter Kit

So, where here is a FPGA chip and around that you have got the IO pads, the LED’S here and the number of switches here you got a display and you can have many other auxiliary chips also on the FPGA development kit. Spartan is one similarly there are other starter kits or other kits on which you can develop; now this kit has got a number of interfaces whose layout is not known to the FPGA itself. So, you have to explicitly connect establish a connection between the FPGA CLB’s and the IO ports; when we do the programming.

Last class we were talking about some programming, which is in the form of programming the interconnects. So, in that we will show how we design the configuration beds, but for complex circuits it will not be possible to do it in the way that were doing in the class.

(Refer Slide Time: 01:46)

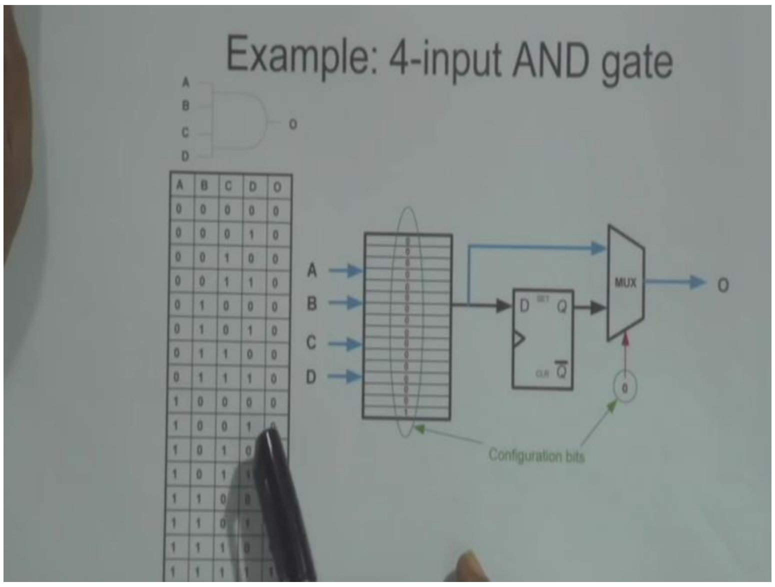


So, here is just as a recapitulation here is a typical FPGA structure, where you have got a number of configurable logic blocks and the interconnects can be programmed using the switch boxes and each of these switch boxes is like a crossbar switch with the array of the interconnects and I can establish the connections in between them using 2 types of possible technology for establishing the interconnects; one is the Antifuse another is the SRAM based. So, Antifuse based interconnections are permanent and SRAM based interconnects are programmable therefore, we can program that also in different ways this we did in the last class.

So, here is again what we did last time that if we take a look at a simple CLB configurable logic block, inside that it is a very it can be even more complex than this, but there is a simple one where we have got a lookup table and d flip flop and MUX. So, the input will come as the input of the lookup table and from the this look up table will be programmed with the configuration beds and based on the input one particular location of the lookup table will be activated and will take that output and that output will be fed at the MUX. Now depending on how I configure the MUX or whatever control signal I give to the MUX, either the input that is coming directly from the lookup table or the output that is coming from the flip flop this input is also coming at the input of the D, D flip flop. So, depending on my desire I can either select this input or this input to come to the output.

Therefore my programming will not only include programming this look up table, but will also include programming this the selection bit or bits of the MUX; here it is only one bit, that is required because I have got only 2 inputs and have to select from one of them. Now if I select through the d flip flop then; obviously, I am getting a state concept here, I can get it at the next clock I can come to this right there by because of the presence of such d flip flops, if you think a little bit you will be able to see that I can also combined the number of CLB's to make a register or a shift register for that matter. A shift register each bit of a shift register is nothing, but a flip flop right. So, I can create the interconnect of that.

(Refer Slide Time: 04:47)



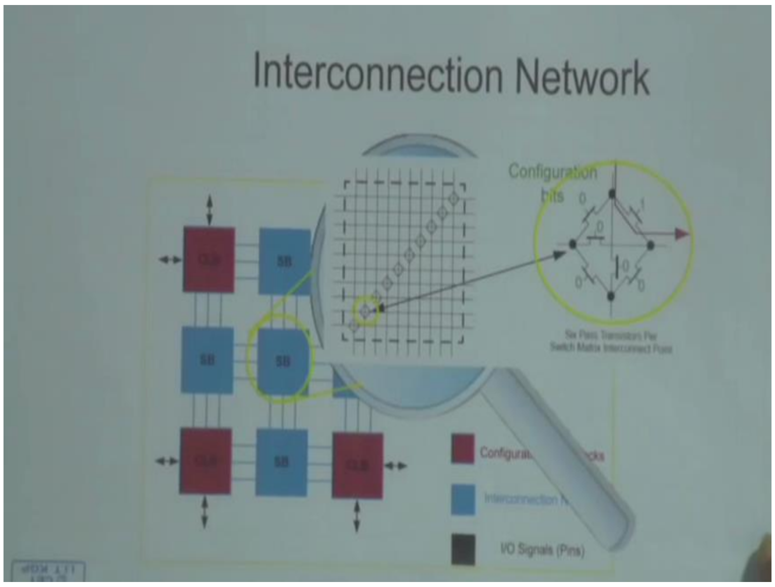
Example: 4-input AND gate

A	B	C	D	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	[illegible]
1	0	1	1	[illegible]
1	1	0	0	[illegible]
1	1	0	1	[illegible]
1	1	1	0	[illegible]
1	1	1	1	1

Now we were doing a very simple implementation, what was that that was a 4 input AND gate and the approach that we adopted is writing down the truth table of the AND gate the 4 inputs are coming at the input of the lookup table of course, through the address decoder alright. The outputs are being fed here the output is coming is being programmed into the lookup table and depending on whatever the input is any one of those bits will come out at output right.

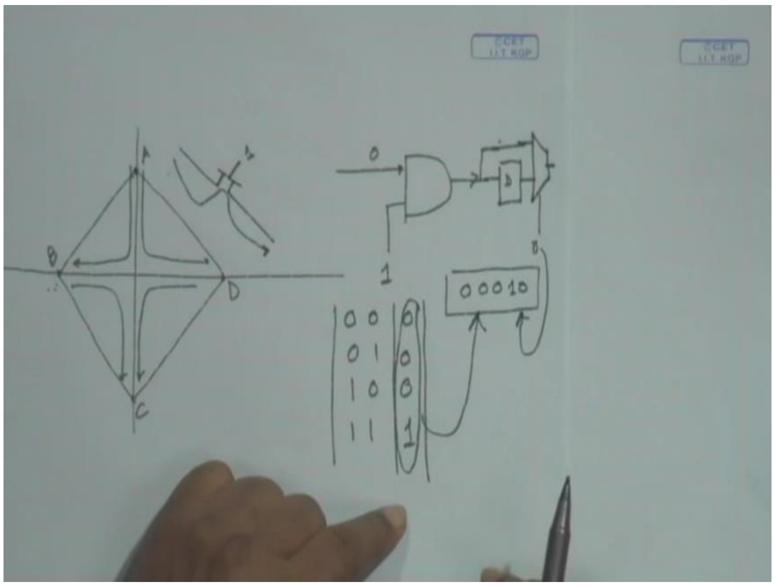
So, if the input is 0 1 1 0. So, this will select the particular location from here, which is programmed to be 0 that will come here and I also program the MUX to select the first input the top input therefore, this is also 0 and hence whatever is coming out from here will come out as the output, thereby I can implement a 4 input AND gate using this that was a very simple design by programming 1 particular CLB; here only one CLB has been used and that CLB has been programmed as an for input AND gate.

(Refer Slide Time: 06:19)



Next we also discussed about the interconnection network and how we establish the inter connect. In the programmable scenario the interconnects are being established using pass transistors, alright; these are the pass transistors. So, I have got my papers have gone there. So, I can draw it again. So, I have got 4 points, 2 wires are coming I can either establish the signal to flow in this direction or in this direction or in this direction or in this direction etcetera. So, there are 6 possibilities right this. So, with something like this once again I repeat.

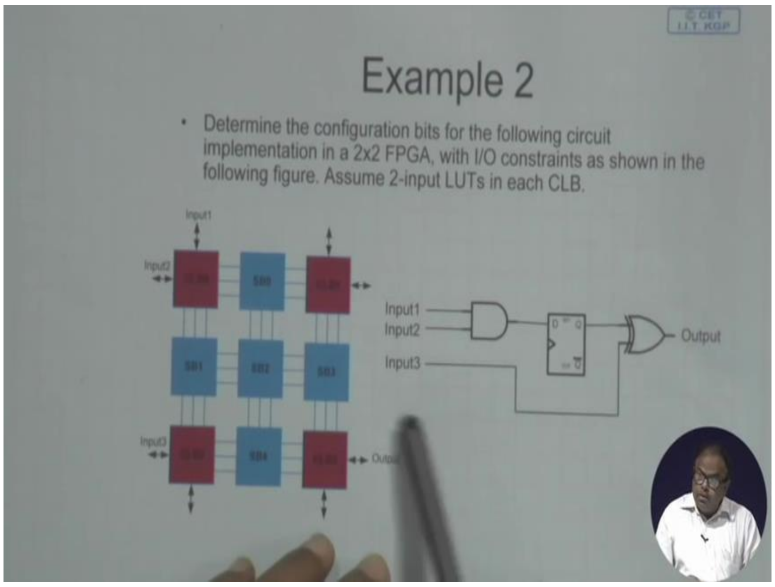
(Refer Slide Time: 07:02)



I have got 2 wires. So, I can make the signal flow in this direction or in this direction maybe in this direction, this direction or it can flow diagonally from here to here like that. So, I have got 4 points A B C D, I can either come it from A to B, B to C, C to D, A to D or whatever or A to C or B to D. Therefore, any one of the 6 lines can need to be shorted. So, for that for each of these lines are connected to a pass transistor, where if I put a 1 then this pass transistor will be conducting therefore, it will flow like this and if I put a 0 this will not be conducting.

Therefore, if I take this example now you can see that I have written a 1 here; that means, this pass transistor will be on, others will be off. Therefore, any signal coming to through this path will flow to this part right. So, in that way any switch box I can program by using this pass transistor gate bits, this is a transistor and this is a signal that we are giving at the gate of the transistor. So, depending on the programming of the gate bit I will get I can program the interconnect also, because of this programmability FPGA actually is called a programmable gate array.

(Refer Slide Time: 09:17)

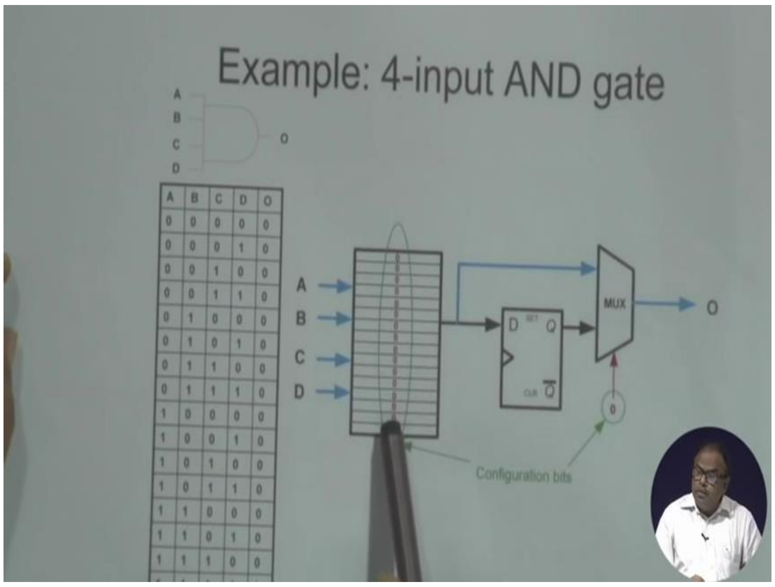


Example 2

- Determine the configuration bits for the following circuit implementation in a 2x2 FPGA, with I/O constraints as shown in the following figure. Assume 2-input LUTs in each CLB.

Now, let us take another example, I think we did up to this in the last class. So, let us take another example. So, we have to establish a program we have to establish a program, we have to write a program to establish to implement a particular logic circuit or particular logic. So, here is an example a second example, determine the configuration bits for the following circuit implementation will be done in the 2 by 2 FPGA, with IO constrains as shown in the following figure. We assume that they are 2 inputs in LUT’s in each CLB, which circuit this is what is there in this is the circuit that I want to do alright.

(Refer Slide Time: 10:08)

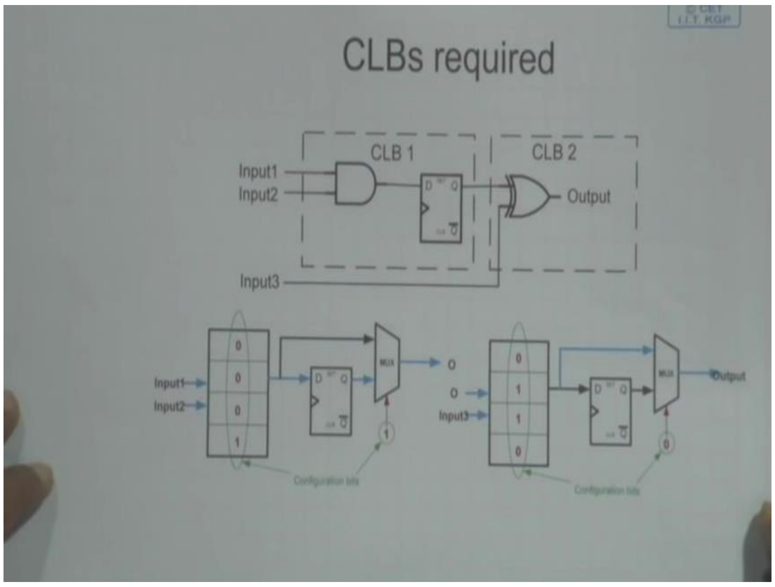


Example: 4-input AND gate

A	B	C	D	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0

Now, we have already seen what is there in one CLB; in one CLB we have got a lookup table at a d flip flop and the MUX. So, that is my basic unit using that I have to implement an AND gate which is coming to the d flip flop and another is the output or another input is coming at the next or gate and I will get the output. So, this is what I want to achieve using the 2 input look up tables. Now look up tables the look up table that I had shown earlier did not have any restriction on this, here I am and saying that there are 2 input variant look up tables. So, how can I implement this if I break this up into 2?

(Refer Slide Time: 11:05)

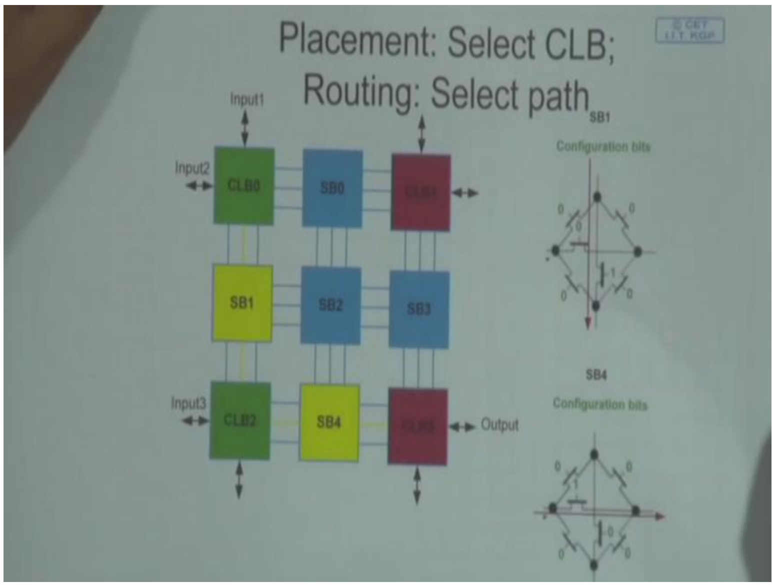

CLBs required

Let us. So, work it out. So, there are 2 components of the circuit right this and this and I need 2 CLB’s, I am achieving 2 CLB’s of course, you could have said that well I will make a you could have said that well, I will make a truth table of the whole thing and I could do it is one look up table right, but here I am showing I am intentionally doing it with 2 CLB’s to show you how the programming is done. So, suppose this part input one and input 2 goes to the d flip flop and I will take the output from the d flip flop, that will be my that will be implemented using CLB 1 and the CLB 2 will simply implement the XOR gate where one input is coming from the output of the D flip flop of the previous stage and the other one will come from the third input.

Now here is the programming that we have done just look at this simple, what I am trying to do is an AND gate here nothing else, AND gate will be 2 input AND gates everything 0 except for 1 1 inputs; when the input is 1 1 so that will come out through this, but my desire in this example is unlike the previous one I will take the output through the D flip flop. So, one state later that is my specification therefore, unlike the earlier case of 4 input AND gate, here I have to program the multiplexer with a one. So, that this is not selected instead this one is selected right. So, this parts programming the configuration bits will be 0 0 0 1 and multiplexer will be 1. So, this programming will yield this circuit; again coming to this circuit, here I have to implement just the XOR gate and the XOR gate can be implemented using the truth table 0 1 1 0 because it will be 1 only when not a and b or sorry not a or b and a not alright. So, either one of them will be true both of them cannot be one or both of them cannot be 0. So, I take out this, but here the output is directly coming out therefore, from this look up table again I program the lookup table according to the logic that I want to have here you, and I program the multiplexer to have a 0, because I want this output to be propagated to the actual output, this is actual output clear.

Is it ok? So, and what will be the input of this lookup table? The input of this lookup table will be the output of this and the third input this input should come here. So, once I decide on this design that I will select 2 CLB's, and I will implement this part with this CLB and this part with this CLB, the next part that will be required is selecting the CLB's on the FPGA available. So, you see this is the programming I have done some allocation here, allocation of the tasks to 2 different CLB's, but where those CLB's should be placed?

(Refer Slide Time: 15:39)

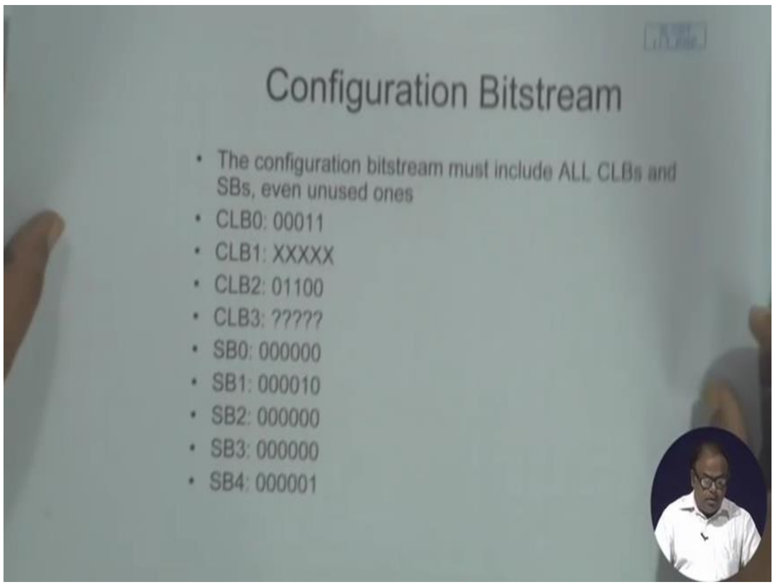

Placement: Select CLB;
Routing: Select path

Now, if I come here and say I decide that my 2 inputs will be coming here; input there are 3 inputs in my desired circuit, input 1 and input 2 will be coming here, input 3 will come to this CLB and I can take the output directly from here or suppose I want to take the output from here from this CLB just for the sake of it, I could have taken it also from here; then my interconnections. So, this is the first of all placements, this is not so trivial the way I say, if be a complicated circuit with a number of logic elements then your placement is also important because there is some delay involved in each of these propagations, this switch boxes, going through switch boxes and all those and also the area is very important.

I cannot arbitrary go on putting in the logic functions on the CLB’s, we need some optimization for that. Just for the sake of example if I decide that there are 2 functions; 2 CLB’s we decided to work with and this was 1 and this was another, this function and this function I say decide that this 1 will go to CLB 0 and the other 1 this 1 will come to CLB 2, then I have to establish an interconnection between them. Now how do I establish an connection between them? That will be through this switch box 1, and let us have a quick look at the switch box 1, I want the current to flow from in this direction, right? Not through this, not in this direction, not in any other direction, but in this direction therefore, this pass transistor, this pass transistor which is connecting node A and B say A B C D A and C that should be 1 all others will be 0 agreed?

Similarly, if I want to propagate the output that is being generated from here, to this I have to select the s b 4 switch box 4, and program it so that this path is activated. Now if I had taken the output from this CLB of course, this switch box should not be needed at all, that is also very well done I mean I can do it in this way. First of all therefore, I want to program this 2 inputs and the output coming through this CLB. So, in order to achieve that what are the 2 things that I have to program, just for the first one the CLB 0 has to be programmed and SB 1 has to be programmed; what should be the program of CLB 0? What should be the program of CLB 0 we have already seen, CLB 0 will be implementing this AND gate therefore, it be 0 0 0 1 right and the multiplexer should be 1.

(Refer Slide Time: 19:36)



Configuration Bitstream

- The configuration bitstream must include ALL CLBs and SBs, even unused ones
- CLB0: 00011
- CLB1: XXXXX
- CLB2: 01100
- CLB3: ?????
- SB0: 000000
- SB1: 000010
- SB2: 000000
- SB3: 000000
- SB4: 000001

So, 0 0 0 1 1 that will be the programming of CLB 0, and that is also known as a the configuration bits of this for that I have to program for CLB 0. 0 0 0 1 this much was coming from the lookup table and 1 for the multiplexer, right? That is the CLB 0 and what do I need in which switch box am I needing here? I am needing switch box one. So, switch box 1 will required only this to be 1 and all others to be 0 therefore, the switch box 1 will be all zeros except this particular bit to be 1, that is the programming of the switch box one.

On the other hand these switch boxes are not active. So, they are all zeros you can see, switch box 0, switch box 2, switch box 3 are all zeros.

Student: (Refer Time: 20:26).

Yes of course, that there is some convention, I mean at the end I will show how you can establish those connections, but each of these bits each of the bits in this configure that this configuration bits are referring to one particular pass transistor position in the case of switch box. In the case of CLB’s it is of course, the look up tables and the inputs right now so far so fine. So, if I want to have the CLB 2 that is this logic circuit implemented this logic circuit implemented in CLB 2, then what will be my program in bits? 0 1 1 0 and 0 0 1 1 0 and 0 for the multiplexers; so that should go for the CLB programming here, so CLB 2 will be programmed as 0 1 1 0. Actually for CLB 1, CLB 1 which I am not using CLB 1, CLB 3 I am not using I really do not care what I am giving here, what is there in the look up table that is not of concern to me for this particular purpose.

Now so for that CLB 2; for that CLB 2 I wanted to have the multiplexer control to be 0, and 0 1 1 0 to be the lookup table inputs, accordingly CLB 2 you have got 0 1 1 0 and 0 for the multiplexer; fine? Now to make the things complicated, I have decided that well I will take the output not from this CLB, I will take output from this CLB; that puts me in a spot because now this output that was coming out from the CLB has to be propagated to this and will come out from here. So, how can I do that?

Now this switch box will have to be programmed; so that the connection is established in this direction right and for that I have to make this bit to be 1 and all the other bits to be 0. That is quite visible here, only 1 this bit to be 1, all other bits will have to be 0. So, accordingly again if I go to the configuration bits, switch box 4 it is switch box 4 right? So, switch box 4 will have all these zeros and only one to be 1. Now what will happen to this CLB? Data will come from here and we come out from here, without any transformation. So, what should it be how should I program this CLB?

Student: (Refer Time: 23:54).

Absolutely, I can do that with a simple AND gate with one to be 1one and whatever is coming as the output will go out here right. So, my truth table will be that of an AND gate 2 input AND gate 0 0 0 1 1 0 1 1 alright and the outputs will be all zeros, and this to be 1. So, that is what I will put in the CLB and this will come to the D flip flop as well as directly and it will come to our MUX, right? So, what would be the control for the MUX here? It should be 0. So, my programming for this will be 0 0 0 1 0 these AND gate information coming from here and this MUX information coming from here right that will be the configuration for the CLB 3.

Student: (Refer Time: 25:00).

Just a minute; so here I have shown that the CLB 3 what it would be. So, you have already found out. Yes for CLB 2 the MUX was 0 because CLB 2 was just giving the output from here and I am just taking this output at the input of CLB 3. So, this is coming through the switch box. So, this is an example and I just had added this part so that you work out what the logic will be. So, based on this we can implement the circuit that was given this or this circuit can be implemented using this would be my program, this will be the configuration bits of course, this has to be filled up with the patterns of zeros and ones as you have found out. So, this is something that we load it on that FPGA board, so that the logic is implemented. Now one thing is that is input 1, input 2.

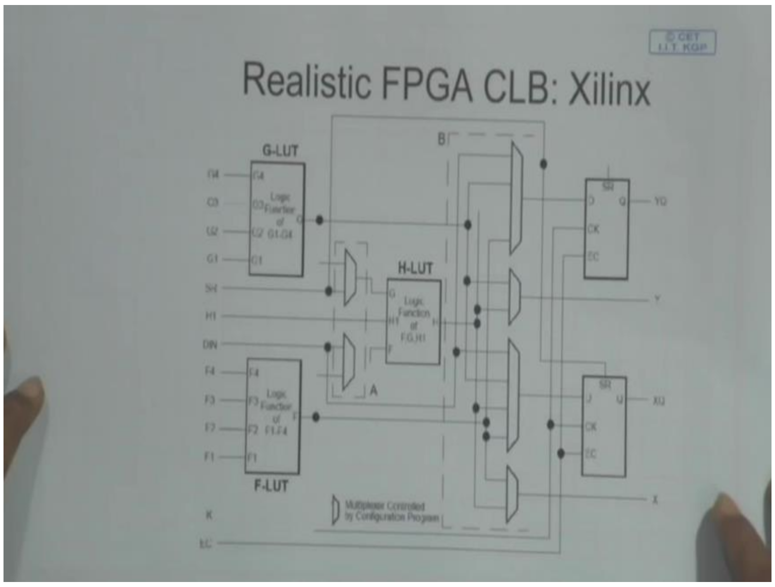
Now I would recommend all of you to try your hands on different FPGA boards whatever is available in your hardware lab, or in any other lab if you get access to different Xilinx or Actel FPGA's, now each of them; now for Xilinx or Actel again there are different varieties of the kits right. So, for each of the kits there is a different configuration the configuration that I had shown where they will have Xilinx FPGA, but what are there around it that is one important issue, you will have to adapt you have to know that and accordingly you have to do that.

Now, this arriving at this programming is achieved by using some programming tools, every s FPGA platform comes with development systems, where isc is alright; where you can put in very Verilog or VHDL code and they will convert it in the form of FPGA program and this FPGA program is nothing, but an array of set configuration bits and those configuration bits are uploaded into the FPGA memory and accordingly the interconnects are established and the whole thing will start working.

Now along with that programming you need 2 things, one is the pro one is the high level code that you give and you compile, and that will be uploaded on the FPGA also there is a user constraint file UCF; that file I will just show it to you later that that file has to be created so that the these inputs which this input 1 is connected to which input pin, this output 1 is connected to which output pin, you are to decide that and you can establish. So, how are you getting this placement? One thing is that if the entire thing is automated everything will be placed and you do not have to do anything and often that is suboptimal. So, sometimes we would like to do it ourselves. So, thereby you can establish the interconnection using writing that particular file ok.

Now, this FPGA that thought CLB that we were showing was rather simplistic so, but it is easier to understand we need not unnecessarily complicate it.

(Refer Slide Time: 29:36)

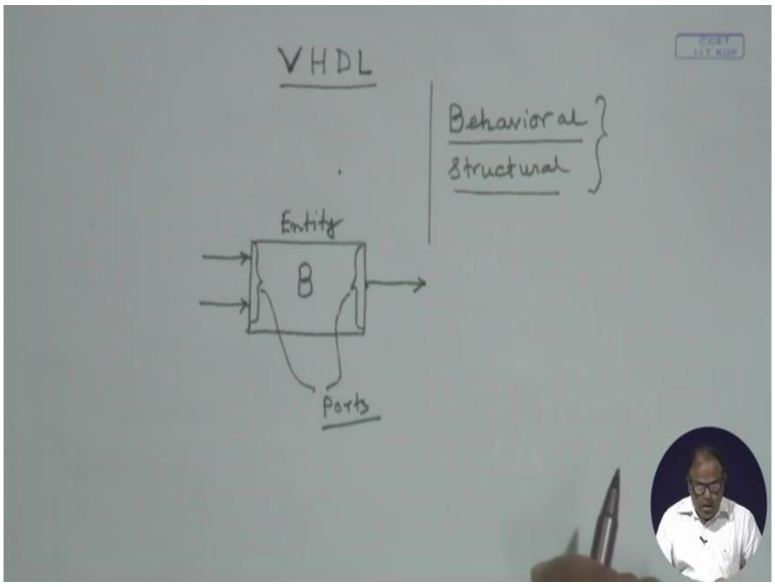


Realistic FPGA CLB: Xilinx

But here you can see this relatively realistic FPGA. FPGA CLP where is again from Xilinx you have got 3 logic units alright and a bunch of multiplexers and 2 D flip flops right and each of them are 4 inputs, you can see you can also connect. So, here from the experience they have decided that I can I will have the facility of implementing logic functions in a much more efficient way, if I have got 3 look up tables, each 4 inputs I can use them or I may not like to use them and a bunch of this multiplexers and D flip flops; this is a relative realistic situation of a FPGA of a CLB in a FPGA right?

Next although we will not discuss the languages VHDL or Verilog in detail in the class and I will ask all of you there are lot of reference materials for VHDL and Verilog and you can very well there even on the internet there are several tutorials, you can, you cannot you can you should go through those tutorials and learn it. Now today I will just simply talk about some parts some salient parts of VHDL and utilize it to show you how we can have a simple toy problems solved within a FPGA using VHDL.

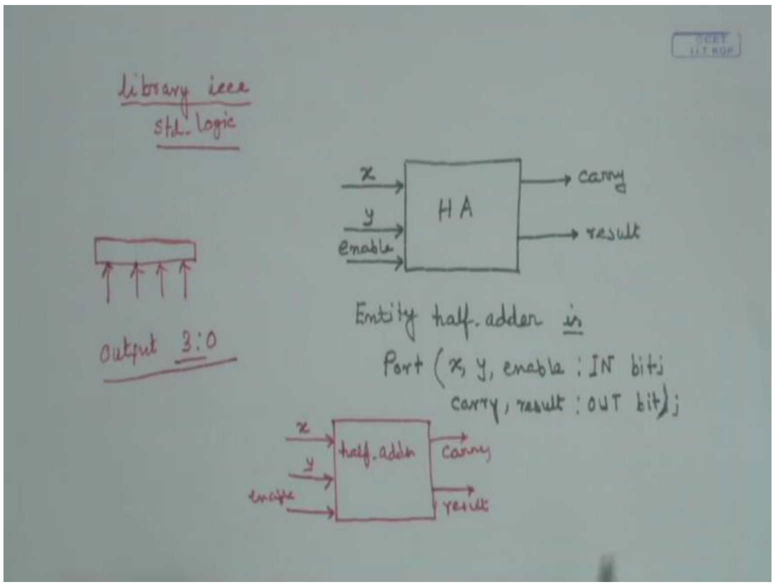
(Refer Slide Time: 31:32)



So, you know that VHDL is hardware description language and VHDL has been developed for VLSI hardware description language and it can be used for synthesis as well as for simulation. Now it can be at different levels I can describe a circuit, we will talk about that later when we talk about modeling, but usually we can again if you remember that Gyskis y chart is you will remember that we can have a behavioral description what it does, structural description.

So, I can describe a circuit or design intent, what I want a circuit to do using either in the behavioral level or at the structural level. Today let us start with some descriptions of the behavioral level say I have got some, some entity whatever that is some behaviors B, which will have say some inputs and some output. So, this is an entity we describe that as an entity; an entity is associated with the inputs and output which we call ports alright. So, these are the ports this and these are the ports of this entity.

(Refer Slide Time: 34:02)



Now, let us say I want to design a half adder, what will be the inputs of the half adder? Say I say x 1 bit, y is another bit and maybe I will put in an enable signal and what will be the outputs of the half adder? All of you know that it will be carry and then sum or the result right. Now this is an entity half adder is an entity. So, in VHDL I have I can describe this as entity, I can give a name to this half adder is; now these are keywords right entity half adder is then inside this I will describe this half adder. How can it be described with ports?

So, I have got port, what are the ports here? X, y, and enable these are ports which are input ports and of type bit whereas, carry and result are output ports and this is also bit I am sorry I have to complete this bracket that so that is my port prescription. Now if I just look at I do not look at this, I just look at this description and I want to redraw something.

So, I can redraw this suppose this is not shown to me I can just look at it and redraw it that there is an entity and it is name is half adder, I just got x, y and enable ports are ports and each of them are input ports and these are bits and carry and result output ports and this are also of type bits. Now as soon as by looking at this I take it out and you can see that these 2 are absolutely equivalent, this one and this one right? Now here before we conclude today we will continue with this VHDL just like in C language we have got some standard library functions, VHDL always includes library which is i triple e library.

So, we include library i triple e for before the entity declaration. Now this library uses some and also another library which is known as STD logic standard logic, there is another library. Now these libraries are included the standard logic is required in order to establish the interconnections, now you can see that we had taken the liberty of defining these things bit in out, what are these? Now bit is a standard logic type bit 1 bit that is known a priori we need not define those.

Similarly, if there be suppose I want to define a say 4 bits, I just want to say 4 bits coming here say that is support some output, I can say output to be 4 bits 3 2 0; that means, 0 1 2 3 4 bits this output is 4 bits out, we will see those in the next class. So, therefore, I can either have bit or I can have an array of bits. So, there are different data types and all those which will see with the next class, we will not do the VHDL in its entirety, but we will just do some parts and workout and exercise of an FPGA programming face to face here.