Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

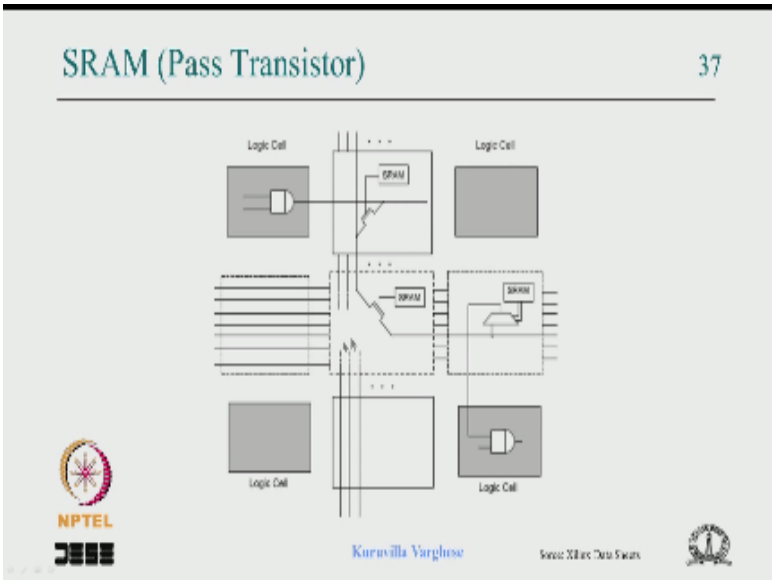
Lecture-37
Xilinx Virtex FPGA's CLB

Welcome to just lecture on field programmable gate array is in the course digital system design with PLDS and FPGAs. In the last lecture we have looked at the interconnection technologies, design methodology, what is a tool flow and what are the commercial tools available etc. and I just started with Virtex CLB architecture. So we will have a quick look at the slides and move on to today's portions.

(Refer Slide Time: 00:56)

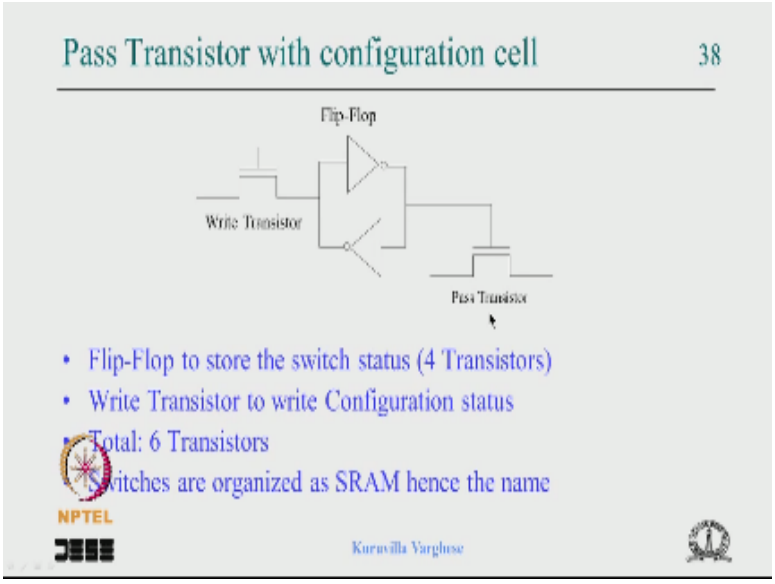
And so last time is at these are the 3 programmable interconnection technologies.

(Refer Slide Time: 01:02)



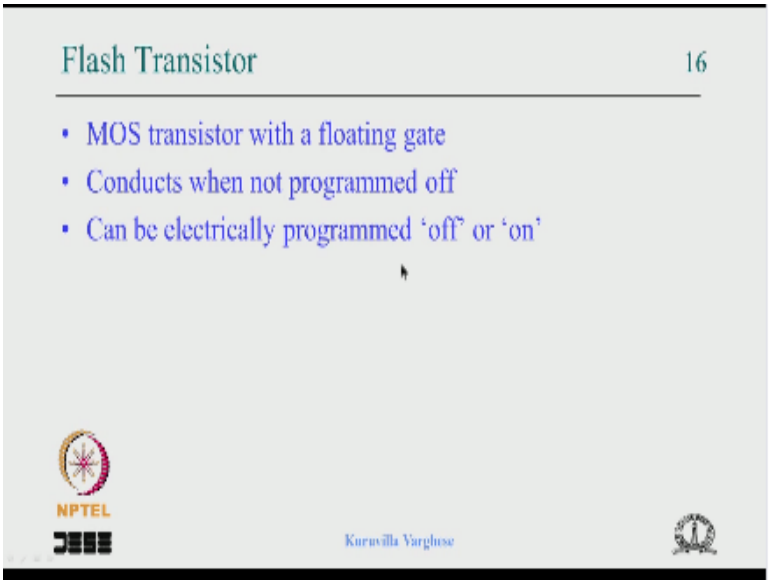
And the SRAM uses the pass transistor as a switch with the gate status stored in a flip flop which is arrange the SRAM, so that SRAM.

(Refer Slide Time: 01:14)



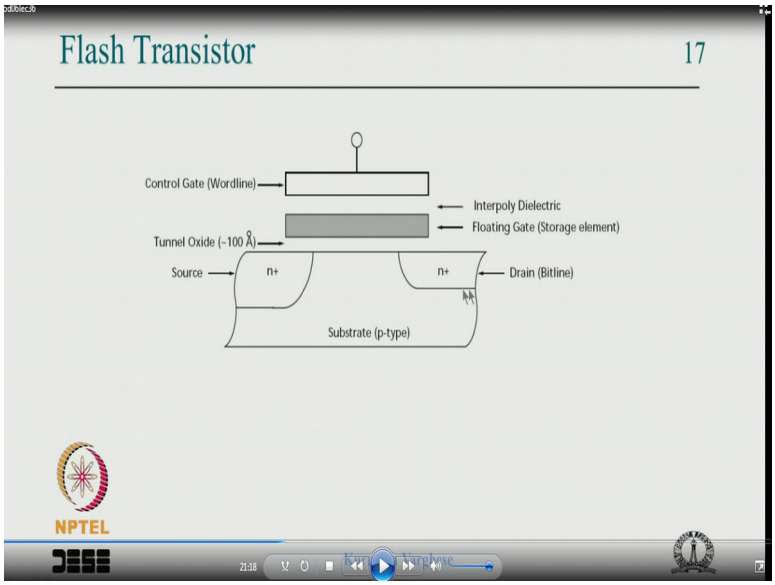
And this flip flop cannot stop cross coupled invertor which occupies 4 transistor, the right transistor and the real switch it consists of 6 terms is quite big and that is why the cells are made the logic blocks are big.

(Refer Slide Time: 01:33)



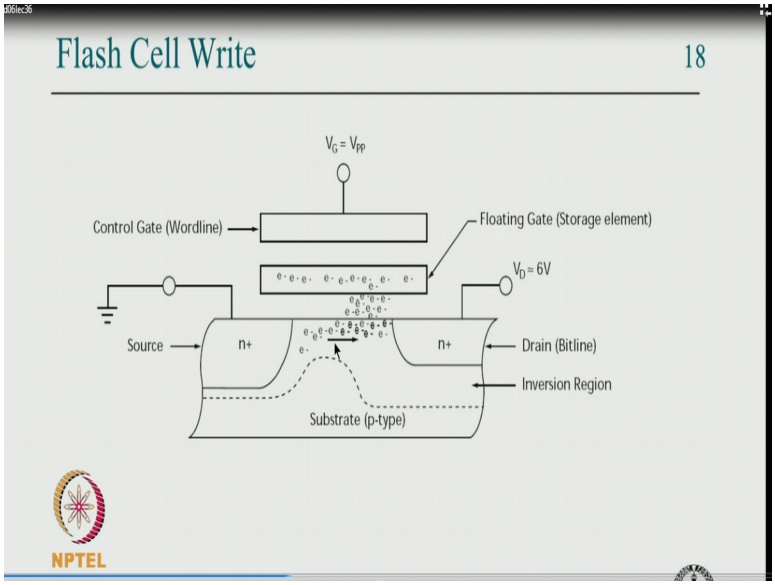
And flash transfer is a normal transfer with the floating gate which can be turned off by trapping electrons.

(Refer Slide Time: 01:41)



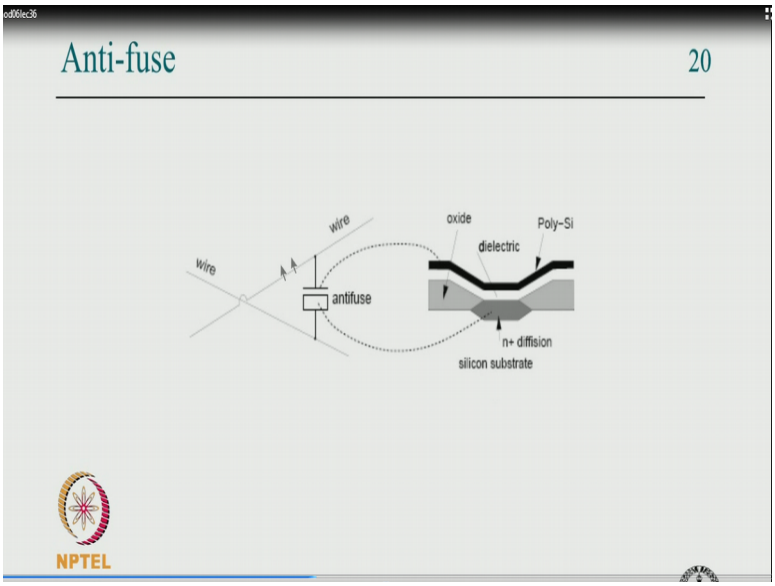
It can be raised electrically with the clarity opposite to that of the writing.

(Refer Slide Time: 01:46)



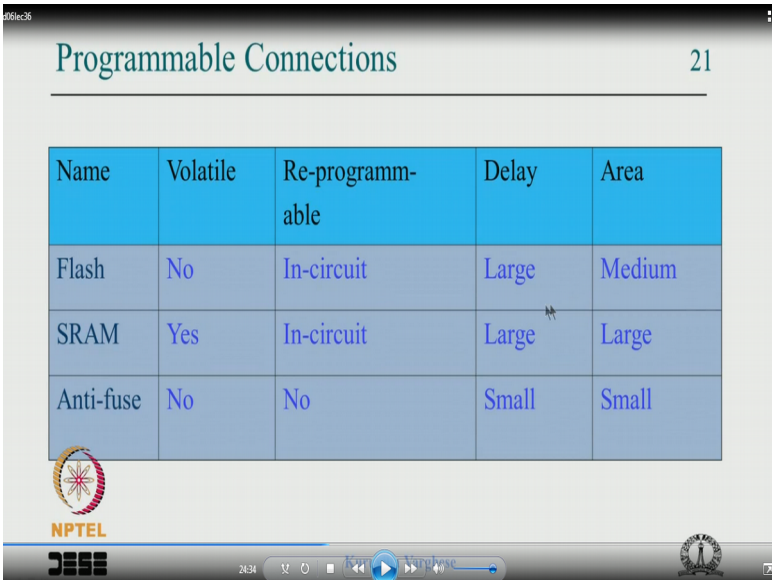
And anti-fuse is where between the wires in the layers within the wire something is deposited.

(Refer Slide Time: 01:55)



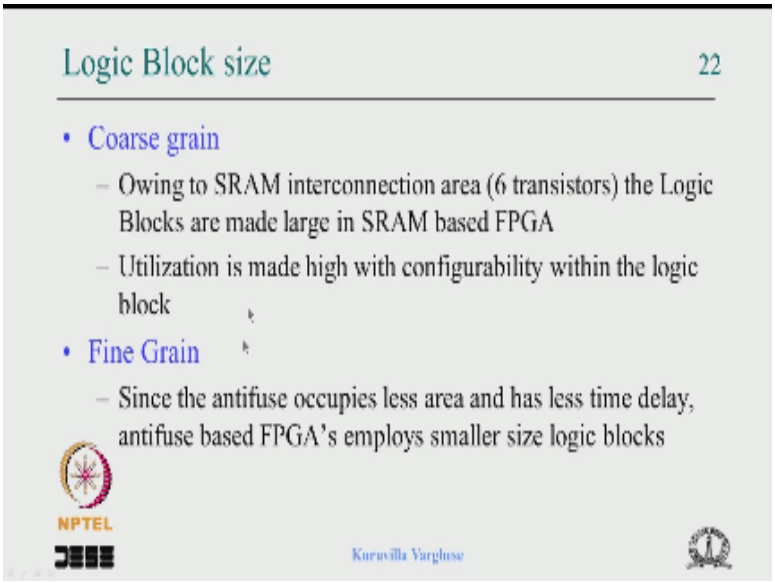
And normally it is not connecting in when you apply voltage it conducts it is not an active switch it is a passive connection which is one time program is a very small area because it is all the wires you need to interconnect the layers of wires. And in between layers, so in that wire are the deposit is made to do, though the transfer is required to isolate the voltage applied to fuse it, but it is very small delay in the small area.

(Refer Slide Time: 02:33)



And that is kind of summary we have seen, the SRAM is easiest need to be program and each time it is powered on but it is non volatile but it is largely and large area and if used as very small area small delay but it is non-volatile one time programmable flash comes in between is non-volatile reprogrammable large delay in the media. So most of the FPGA when does used SRAM. It is limit on the re-programmability and technology fabrication process in OS ok etc.

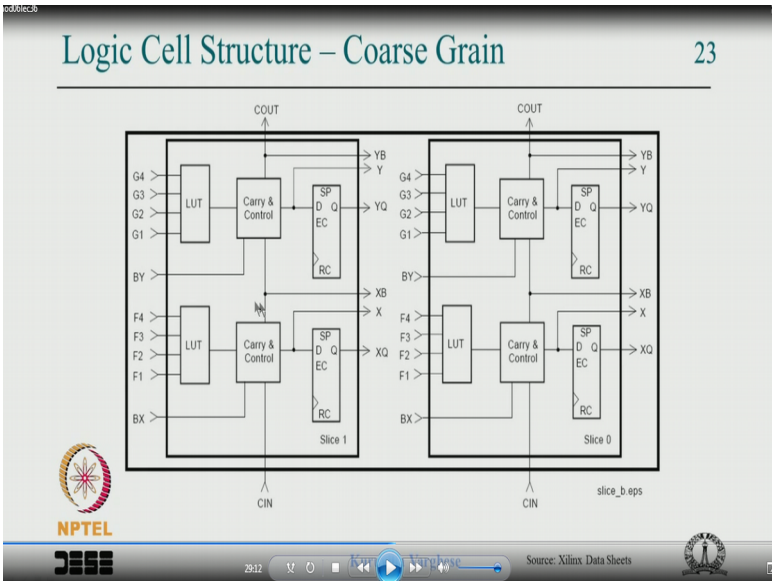
(Refer Slide Time: 03:19)



And the SRAM base technologies you make the logic block bricks because the connections are costly if you make it small and then the connection the switches and associated circuit will be huge in comparison with logic the anti-fuse when this will make the logic block small, so that less is wasted, but here as I said they will make sure that nothing is wasted.

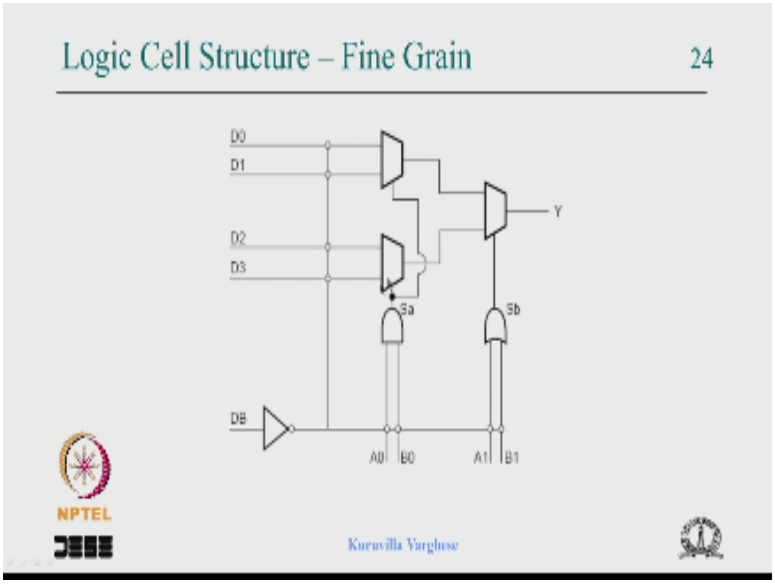
Then be lot of kind of configurability that things are can be used partially split into two parts and things like that or you can use a combinational and flip flop separately put it together all kinds of programmability within the logical block is given.

(Refer Slide Time: 04:06)



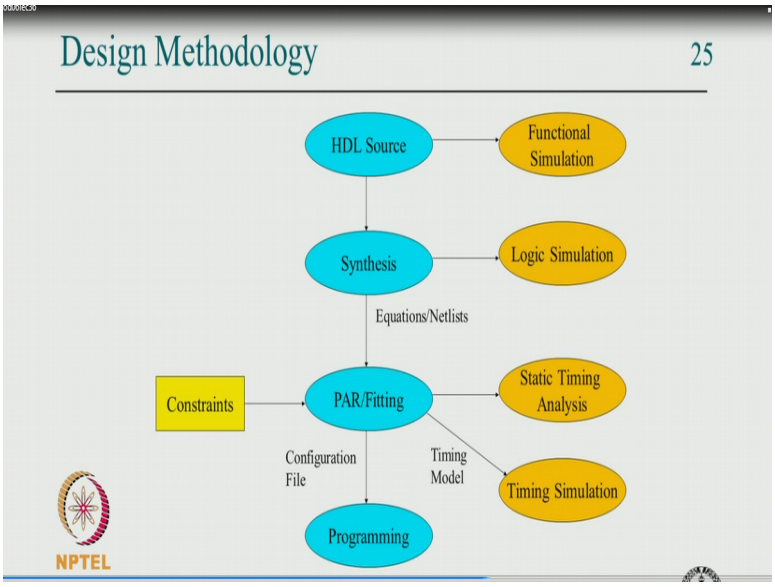
And this shows a coarse grain architecture with 4 lookup table and 2 slice.

(Refer Slide Time: 04:12)



And this just a fine grain logic block of Actel which is a kind of 4 to 1 multiplexer essentially.

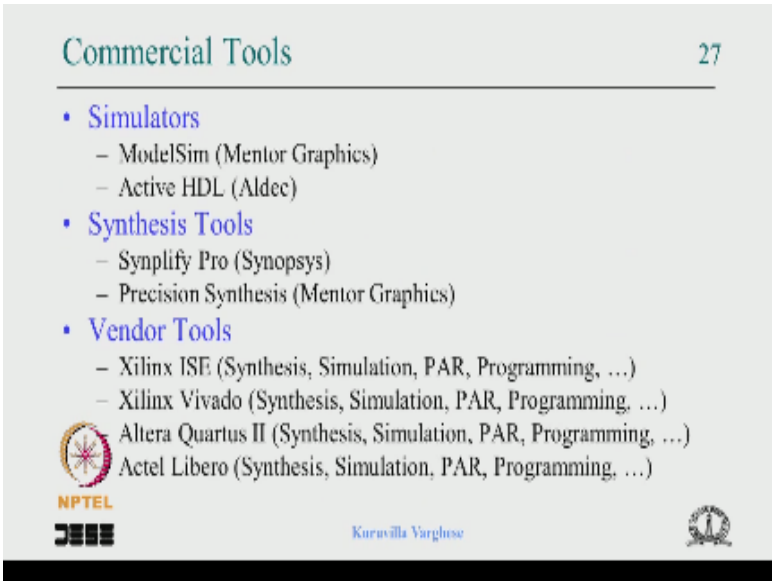
(Refer Slide Time: 04:19)



We are looked at the design methodology we start with the HDL source to functional stimulation, synthesis generate the netlist then you do logic simulation then you go for the place and route you can specify the confines and timing constraints you can do a Timing Analysis which is quicker than timing simulation ultimately can do timing simulation and everything is fine you can program.

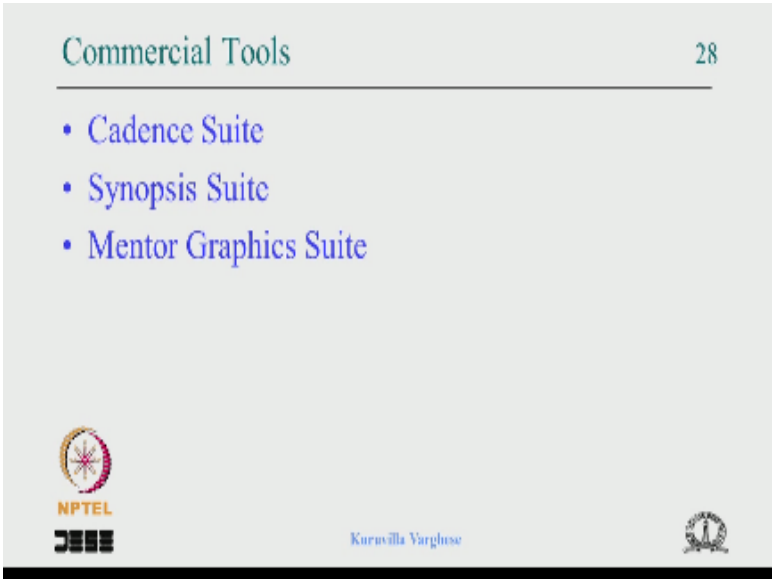
It was here like you eat right here then you come down its weight when you come down through Static Timing Analysis you can it right then ultimately you can it right through timing simulation then ultimately program it.

(Refer Slide Time: 05:00)



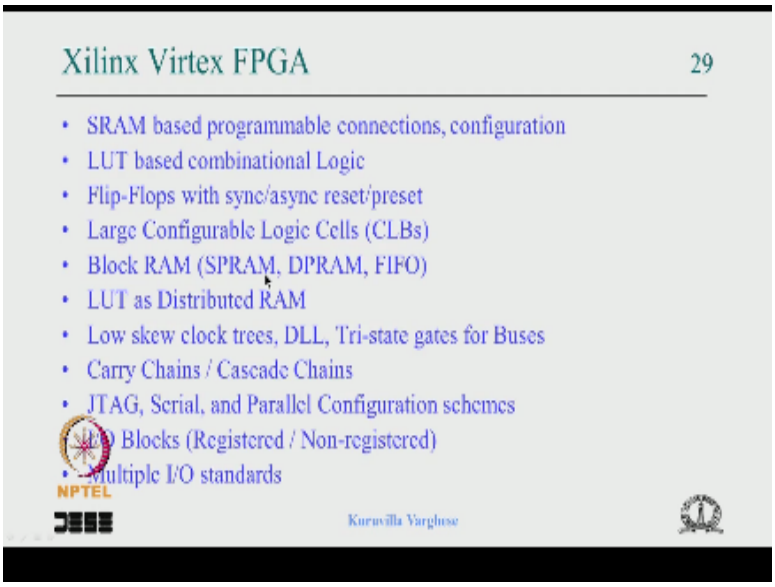
If you seen the commercial simulators, commercial synthesis tools and when the tools which built which has everything in a quite good most of the vendor tools are good.

(Refer Slide Time: 05:16)



And these are the commercial other kind of very you know complex EDA tool suite and it consists of lot of packages within you can do synthesis and simulation with that but the place and route have to go to the vendor tool.

(Refer Slide Time: 05:34)



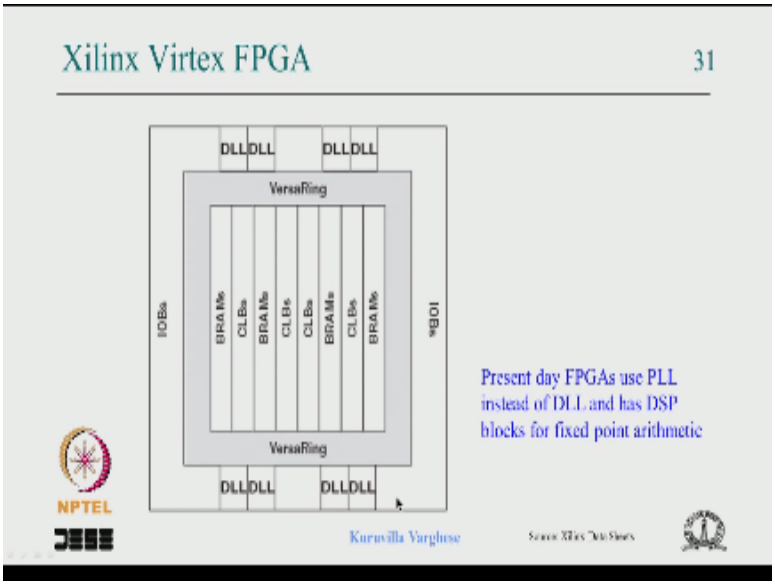
So let us come to today's portion we will look at the this particular FPGA, Xilinx, Virtex FPGA, it is not commercially kind of manufacture now it is probably only supported but this was a kind of I choose this is a kind of on which you can build the other kind of architectures, you take the latest Spartan 6 of Virtex 7. If you understand this well and you can you know understand other FPGAs.

The FPGAs from Xilinx itself and from other vendors, so this is a summary basically it is a SRAM based programmable connection and configuration. The lookup table is used as a combinational logic, you have flip flops within the logic block with synchronous and asynchronous set or reset, large configurable logic clock is quite big, then you have dedicated memory call block up ok, which can be configured as single port dual-port FIFO and all that.

Look up table can be used as memory, we will see that the there are clock distribution network which is clock trees, which has low skew once again we will see why it need to be low skew, there are DLL and currently the FPGAs use DLL, but this is a dedicated component you have instantiated or from the code generator and you can you it, then you have tri state gate for forming the buses, may not be available in the current FPGAs.

You have carry chains, we will see what is carry chain or carry logic, cascade logic, then the various configuration schemes, the serial parallel JTAG and all that and there are I/O blocks and which support multiple I/O standards ok.

(Refer Slide Time: 07:48)



So this is the kind of a top level view of the Virtex FPGA, I will not go into specific of this, first thing is not that all around is I/O block and you will have the PLLs ok now in this DLL, but there will be PLL ok and inside will be the block Ram CLB that is a block RAM is a dedicated memory, CLB is a configuration logic block and so on ok. This sandwich one of one of the other.

And the recent FPGAs you will you will even find the DSP blocks which allows basically fixed point arithmetic you can have a fixed point addition or multiplication and things like that so.

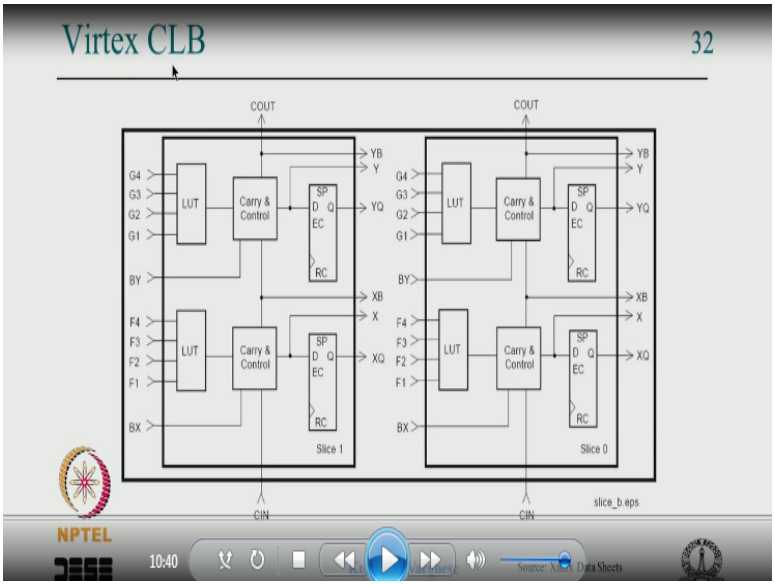
Those blocks will be built in to the FPGA, then there could be some reason FPGA might have the ARM Core, ARM processor core built in some might have some memory controller in an ok. So that is in a nutshell the top view of the what is FPGA and there are many things to look at the FPGA, but the logic connection these logic blocks and all that but like if you take the memories.

Then it is kind of it comes in kind of say you know built-in and you have a tool to kind of insanity this in various width and depth and so on. So there is and there is nothing much too kind of worry about it and you have it used it okay, though there are my network switch I will mention as we go along and but one thing which we will go little is a configurable logic block.

But that is where are the main functionality get built in ok, so we will look at this little more detail because they are you can do lot of optimisation you can understand you can be debug, you can you can find why certain things are happening and things like that ok or why to occupy so much area, so much more area, so much less area and so on.

So you will like when you have done a design you look at the resource utilisation we are wondering why it has taken so many lookup table. So you will be we will given figure it out at the end of the lecture. So we will look deeply into this CLB catch up on the other standard things the basis.

(Refer Slide Time: 10:39)



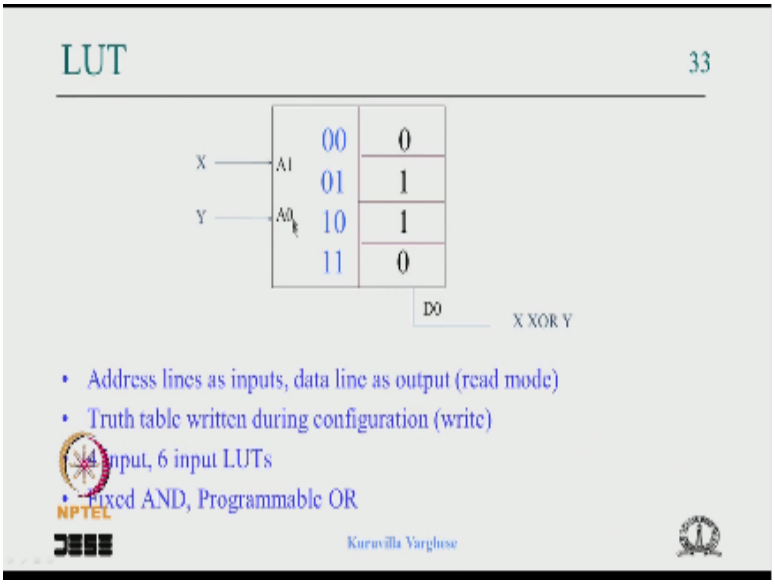
So let us come to this Virtex configurable logic block ok. So a configurable logic block is this, this is the top level schematic what manufacturers taken from the Xilinx data sheet and it has two identical slices Xilinx call slice, so this is a slice and this is another slice and within one slice you have two identical blocks ok. So one CLB is 2 slice, 1 slices two parts which is identical okay.

So now the one part if you look at it there is a four in lookup table and there something called carry on control. We will see what it is and followed with the flip flop okay, so lookup table is used as a combination circuit, this is used for combination cute implementation and this is for registering. So you have as data path with register then combination cute and register that is implemented like that ok things like that ok.

So or you have next state logic with input then the present state some of it back through the wire backer and the next state is loaded to the present state ok, so you get my point. So that is what it offers so we have seen what is a lookup table in the case of like when we have discuss the simple PLDs ok. We have seen that the simple PLDs evolution was starting with the PROM.

So at that point we have looked at how the memory can be used for Boolean function implementation and I hope you have seen that lecture if you are not seen I will briefly explain, so that it is you do not have to go back in search of the lecture particular module.

(Refer Slide Time: 12:41)



So let us look at the lookup table as I like look up tables nothing but a memory. So suppose we have this particular Boolean function x, x or y to be implemented ok, so how that is implemented is that you take it 2 address line a memory that means 4 location memory because there are two variables and we have to implement all min terms so xy as x bar, y bar x bar yx, xy bar and xy.

So you take a 2 address line or 4x1 memory and what you do is that are you write the truth table of this function into the memory ok. So the truth table is that other 00001 and 10111 is right that is that particular location. Now that writing like that writing is you can say configuring the lookup table or programming the lookup table. Then so when you right to the memory that is in the right mode that is during the configuration it happens.

Now you want to like as a you say you want to use this as X XOR Y and put the memory in the read mode that means was reading the data line is taken as output okay X XOR Y output or Y is equal to Z equal to x axis Y. Now what you do is hurt you give 00 then it goes to 0^{th} location and the you get 0,1 goes to first location 1,1,0 second location 1, the data line and 1, 1 third location 0.

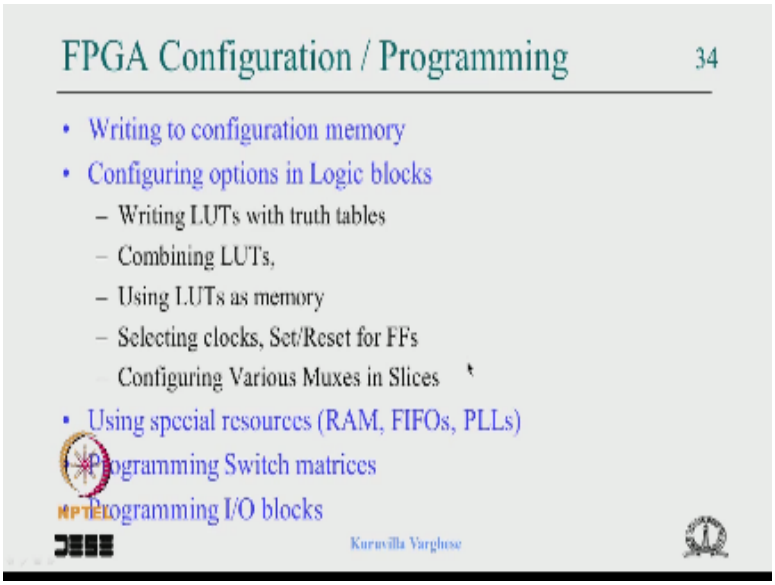
So you get the picture, so you program you take if you have and variable implementation you take 2 raise to N in 1 bit memory then program the truth table of general function in the in the location okay in the memory location put the memory in read mode connect the variable to the address line ok like A1 and A0, the way you program you have connect and use put the memory in read mode.

You use the data line as the output ok that simple as it is one advantage with this is that you can program any function of two variables like it is so flexible that if you take you want implement and or xor NOR everything is possible. So that is the kind of flexibility of the lookup table when we have discussed the PROM have said that overkill because there are the PROM was containing lot of location.

It was very wide kind of implementation and then it means lot of kind of the location and for that particular application it was using it basically address decoding need only one product and we were implementing quite a number of them, so that was a waste but here it is only few lines and all the functions can be implemented. So the thing to remember is that right into the memory is while configuring.

And that right control is controlled by the configuration circuitry of the FPGA but as a user you use a memory in the read mode with address line connected to the input variables and data line connected to the used as output and as I said in the PROM case it is a fixed and which country of all the minterms and there is nothing but a Programmable law, because this is a AND or we are we are implementing an OR of this ok.

(Refer Slide Time: 16:49)



So I want to tell you before going further when we say we are programming an FPGA or it configuration FPGA, it means essentially writing truth configuration memory ok and so all the interconnection patterns are configured here very specifically when your configuring the

logic blocks and writing the configuration memory configure everything possible now all interconnection, all logic, all special resources everything.

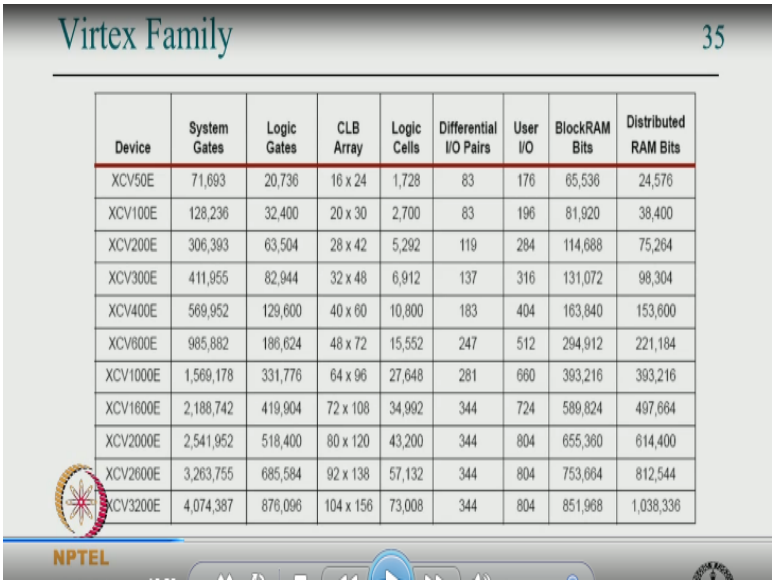
But when within a logic there is lot of things to configure one is writing the lookup table with truth table like you are using lookup table as a Boolean logic implementation you write the truth table sometime a lookup table you need more in boards then you can come by in the lookup table and that need some kind of additional circuit to be configured that is also done through the configuration.

Then look up table is nothing but a memory, so we are using the memory as a combinational logic implementation but you could use it as a memory also if it is not being used to implement the logic you could use it as a memory we will discuss that to the flip flop you can select number of clocks maybe more than other CPLDs you can do the set and reset of flip flops and their lot of multiplexers within the logic block to make you know various programmability waste to make it flexible.

And all that is program this much parts of program by writing the configuration memory. In addition you can configure the memory FIFO, PLL, all that like there could be memory blocks which is combined to make a large of width or large of depth, all that combining is done through various extra circuit which is configured using writing to the configuration memory and all programming the interconnection.

And the I/O block there is programming available for using it as output input, I/os and so on so all the it is done so when we say FPGA configuration of program is not merely the interconnection of the block there is lot within the blocks okay, lot to do with special resources and quite a bit in the I/O blocks and we will see all that one by one.

(Refer Slide Time: 19:34)



Device	System Gates	Logic Gates	CLB Array	Logic Cells	Differential I/O Pairs	User I/O	BlockRAM Bits	Distributed RAM Bits
XCV50E	71,693	20,736	16 x 24	1,728	83	176	65,536	24,576
XCV100E	128,236	32,400	20 x 30	2,700	83	196	81,920	38,400
XCV200E	306,393	63,504	28 x 42	5,292	119	284	114,688	75,264
XCV300E	411,955	82,944	32 x 48	6,912	137	316	131,072	98,304
XCV400E	569,952	129,600	40 x 60	10,800	183	404	163,840	153,600
XCV600E	985,882	186,624	48 x 72	15,552	247	512	294,912	221,184
XCV1000E	1,569,178	331,776	64 x 96	27,648	281	660	393,216	393,216
XCV1600E	2,188,742	419,904	72 x 108	34,992	344	724	589,824	497,664
XCV2000E	2,541,952	518,400	80 x 120	43,200	344	804	655,360	614,400
XCV2600E	3,263,755	685,584	92 x 138	57,132	344	804	753,664	812,544
XCV3200E	4,074,387	876,096	104 x 156	73,008	344	804	851,968	1,038,336

And when you start looking at the data sheet you will be looking at a family you will have a family of devices which goes from low complexity to high complexity ok and very important specification as far as we know as far as FPGA concern is this one, the array of logic pro, how many logic prox are there ok. So you take this particular 100 device XCV100E, there is an array of 20x3 6000 logic blocks ok.

So when it comes to the largest advice it has around 1 or 4 into 1,5,6 logic and in the recent FPGAS and this is just you know see it is a 100000 if you multiply but if you look at the current FPGA it may be having two million logic blocks and things like that, so 1 important and we are going to see what is inside the logic block, so 1 importance that you look at is this the number of logic block ok.

Now again you can disregard this logic cell because you see the 20 to 3600 and we have seen is that other 4 identical kind of logical and 604 is 2400 but you know they are shown some little bit of a you know kind of scaling up to accommodate the extra overheard so quickly which itself can be used on time because there are some overhead to use to combine the logical sources.

But at times that itself can be used for some kind of logic implementation to include that also instead of 4 x 4 x 4.5 but anyway the understanding that important is the number of CLD definitely the number of I/O pins that could be single ended IO,. differential IO like you want more no sensitivity you can use I want to do with that the common mode noise when you can use a differential IS.

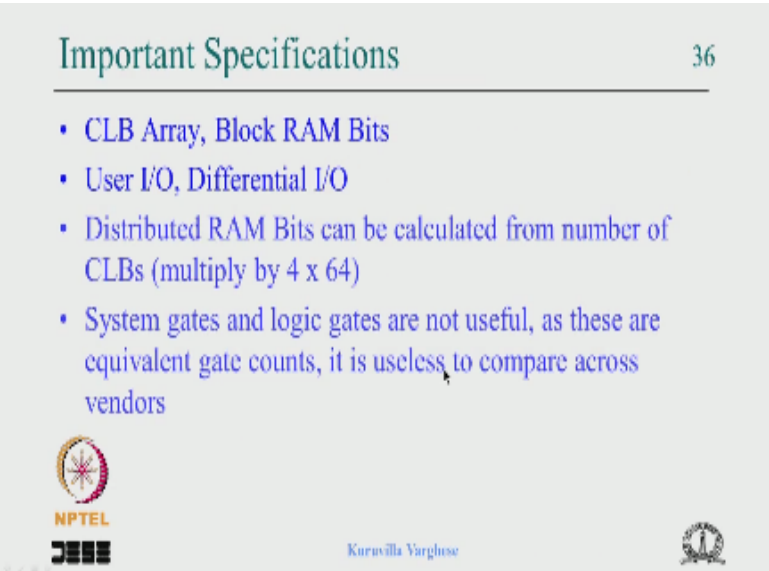
And another important thing is a block Ram which I have how many dedicated memory is available so you are around 81 kilobytes are available when it comes to the last point near it is ok and this distributors is again something to do the CLB array, so this can be calculated from this CLB array spec not a big kind of spec, so I would say that the important things look is a number of CLB.

The number of pins, number of block ok and when you are encountered with the latest if yes definitely have to look at the number of PLL else available maybe number of the DSP blocks available all that ok. So I think you can extend this but there are things which is not useful is that the number of like system gate, logic gate, these are equivalent gate count and you know very clear how that is calculated.

And it is maybe you can use it to you know kind of decide between the kind a within a family ok, but there is no point in comparing the system gates are one family with other family from one vendor with another chip from another vendor this will make much sense and I am just saying coating big numbers in that 1 million or 100000, 200000 and one should not choose FPGAs depending on these system gate and logic gate.

Though this indication of the complexity, if you understand well then you can kind of judiciously you know make your decision objectively knowing the architecture and knowing the essential spec.

(Refer Slide Time: 23:48)



So basically I would say and in this case the CLB array block RAM bits, use I/O, differential I/O, all that the importance spec in the reason in the current FPGAs and you can add DSP block with the number of DLL any other dedicated resource which is available. So let us look at the CLB itself. So as I said one CPLB as two identical slice and 1 identical slices two parts. So we will concentrate essentially on this.

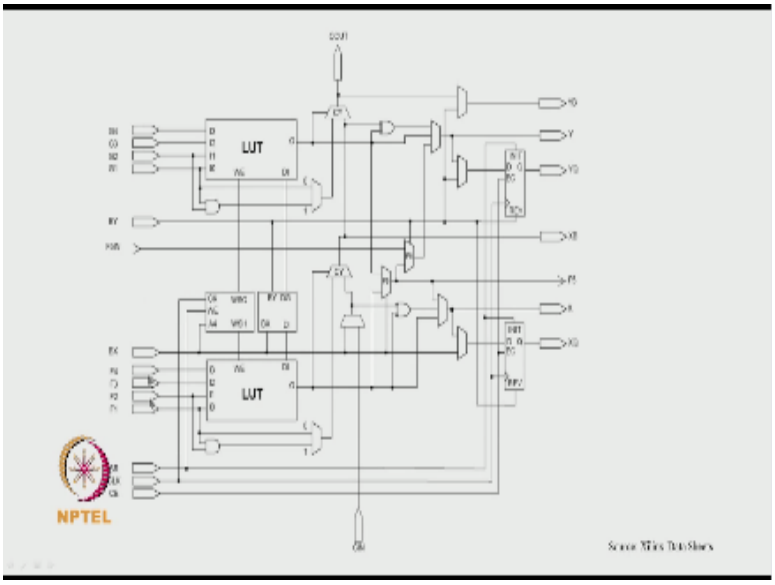
So it has a 4 input lookup table, so that means you can connect any four variables like ABCD implement any function of the ABCD and there is no optimisation more okay there is no when the number of variables are 4 or less absolutely no optimisation it just write the truth table we have the lookup table ok, then it works ok but if you have more than 4 there may be different ways of handling it.

I will give you some possibilities and you can see that the output is going to the flip flop and flip flop DQ the clock set synchronous reset and you know the reset ok and there is something enable clock, we will see what it is ok and one thing to know what is that it is not necessary that the look of table output should directly go to the flip flop, sometime we want to use this as a combinational logic which is combined with other combinational logic to implement complex functionality or like you know you would like to take it out directly to a pin ok whatever else.

So that is possible like you can see that that is directly going out in that case the flip flop can get wasted ok. This picture is just of like top level schematic. We will go into detail like just I want illustrate not like the connections are not that correct and this changes input which can go in principle to the input of the Flip Flop, so like you could use other look up tables separately and the flip flop separately or you can do it used together ok.

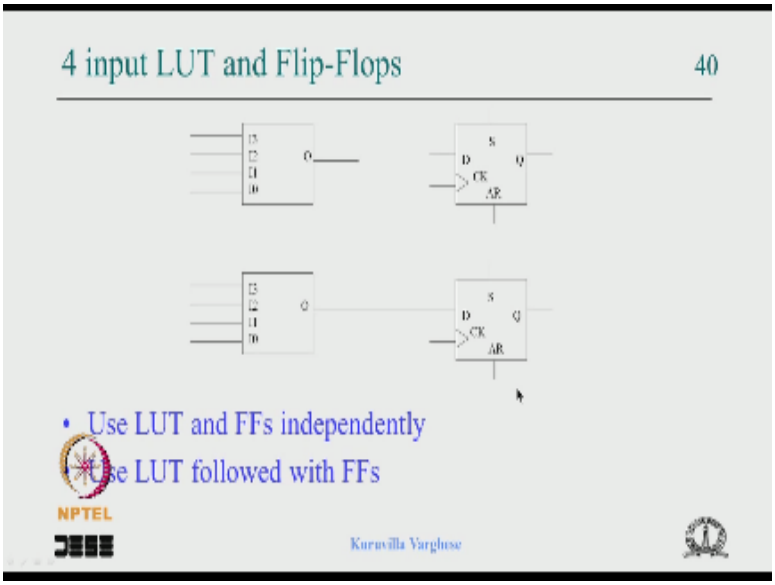
So that that is one kind of level programming, so what we are going to do is that we are going to look at the blow up of this one slice, so that we can understand it little more deeply ok.

(Refer Slide Time: 26:46)



Now that is one slice at the beginning it looks little intimidating but I do not worry, so let us look at say these are the identical part that is 1 part and this is a second part and you see there is something shown here which is there on top of this also but we will this is common for both look up table. So we will see what it is and there is something in between here we will see again what it is all these ok and there are 2 flip flops ok.

(Refer Slide Time: 27:16)



So the first thing I said that is possible that you can use the lookup table output you can implement any function of 4 variables ABCD and output can go to a flip flop to implement data path or sequential circuits whatever. But if it is it that you want use it separately like you want to give ABCD is that a sum function of ABCD and we are taking it out and we are giving from the wires interconnection wires.

Some input directly to the flip flop and take the output it is possible. So you can be used together can be used separately ok. So let us go back to the slide are the main diagram and you see this look up table there are four input 1 output ok. Now this one output goes here you can see it can go through this 2-1 Mux to the flip flop. So this is combined with flip flop or if you are not kind of you do not want that you can take outside.

When I say you can definitely you are going to write some VHDL code which is going through a synthesis tool will be synthesized ultimately will be place in Rockford within this particular second block. So that the tool does it but then that is how you know the things happen but suppose you use this look up table independently of the flip flop, then this flip flop can be used separately.

You can see that this pin by, this signal by and that directly can be taken to the input of flip flop and flip flop can be used safely ok. So this you can combine with the flip flop or you can use it separate in that case flip flop can be used separately ok, same thing here you can see here independent or to the flip flop and the BX is used in to the flip flop if need be used some kind of separately ok.

So that is what it is about now mind you should not lose the global picture this is only one slice, there are 2 slice in a logic block and you should know that there are a lot of vertical wires here, horizontal wires here, horizontal wires there are switches here, so and their input connection from the wires coming into this, this part all these parts, you can connect this wires yeah that meaning of it.

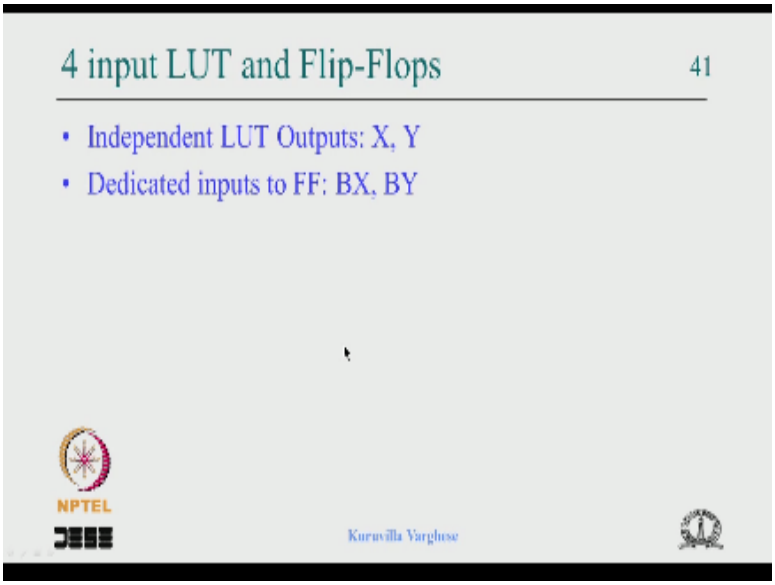
Similarly these output can be taken to the to the wires again and can be taken to other logic from that not you should have in your mind now the question to ask is that suppose we need 5 input lookup table okay say we have a five variable Boolean function how do we get a five input lookup table, the questions that can we get implement 5 input lookup table using foreign but look up table ok.

So now the thing is that 5 input lookup table need 32 locations ok, because 2 raise to 5 is 32, 4 input needs 16 you know 16 2 raise to 16 location, so it looks possible 16 location we have 16 location here, so how to combine that one idea is that when the fifth because you can

connect ABCD maybe our fifth variable. So one idea is that when the variable the fifth variable is 0 program the lookup table here.

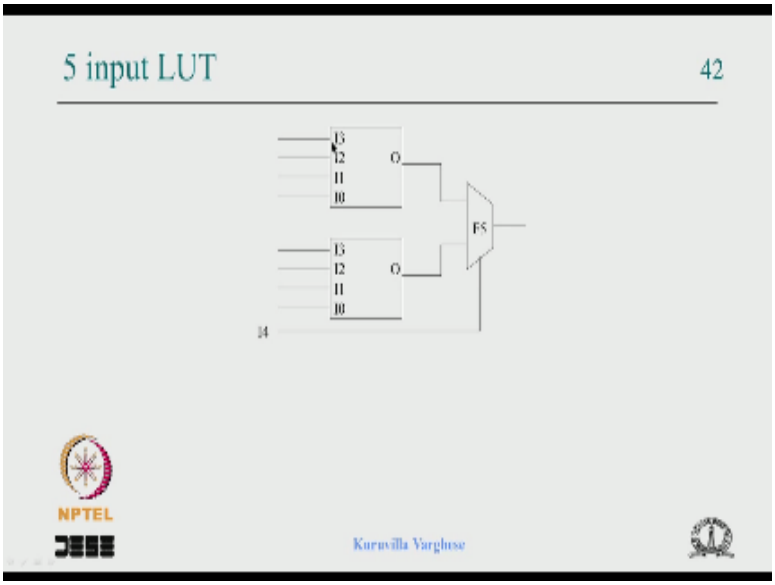
And choose the output of this first look up table as output, when the third shift variable is one that look up table can be here and that should be used there. So that suggest a Mux 2 to 1 Mux in between the select line of the mux can be used as variable.

(Refer Slide Time: 31:37)



So that is how that you like 5 input lookup table is built.

(Refer Slide Time: 31:42)



So you have a 4 input look up table, you write ABCD and here other 4 input look up table, 2 outputs are mux and 2 mux and in the case of what FPGA is called F5 mux and the select line of that forms the fifth variable I0, I1, I2, I3, and 4 variable, this shows variable, when it is 0

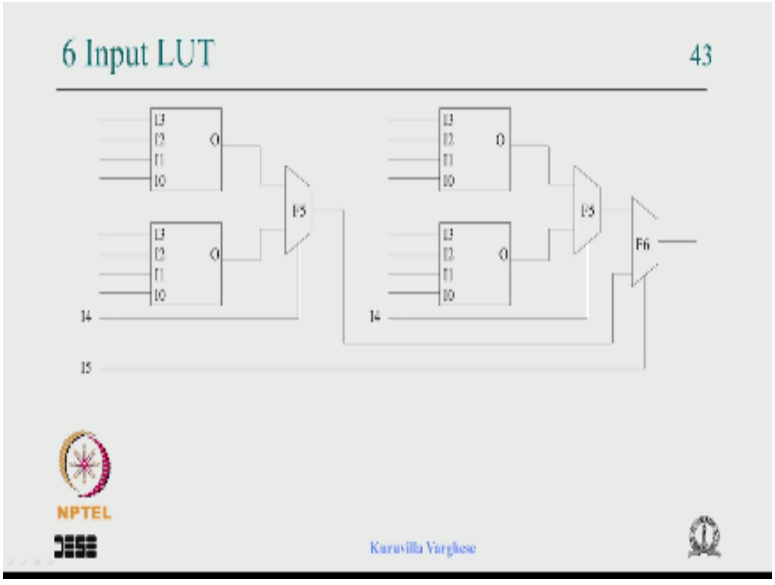
that part of the lookup table is here, when it is one that part is program here and ABCD is common here E is connected you ok.

And this is the output of the 5 input look up table, so we will go back to the to the diagram of the size and you see this is the output of the lookup table ok and first top look up table and this is output of the bottom look up table, you see that there is a 2 to 1 mux which is called F5, and the select line here and F5 output goes there and the select line is here. So you have BX and this is the fifth variable.

So if you have a b c d e and it is a b c d here, abcd here and you connect e and you can see that the you get a five input lookup table output that can be registered using this particular flip flop ok or it can be taken out directly out ok. So that is how the 5 input look up table is formed ok. And you can or the next question can be formed a 6 input lookup table because you have already using a slice so this is now 1 slice ok.

You should remember that, within the slice 2 look up table one lookup table is 4 input, but you can combine to 4 input look up table to make a 5 input look up table that occupies a slice, so the next question is that, suppose we have another slice here which has like another 5 input lookup table, can we combine them like 2 five input lookup table to make a 6 input look up table okay.

(Refer Slide Time: 34:06)



The answer is yes we have yet another mark so we have like we can do this is a v lookup table this is a five input lookup table, now use 2 to 1 mux between the output of 2 5 mux we

call F6 mux the select line of that is the 6 variable. So you have abcd, abcd here, is same for both because like when E0 a program here is one program here is E0 program here is one program here.

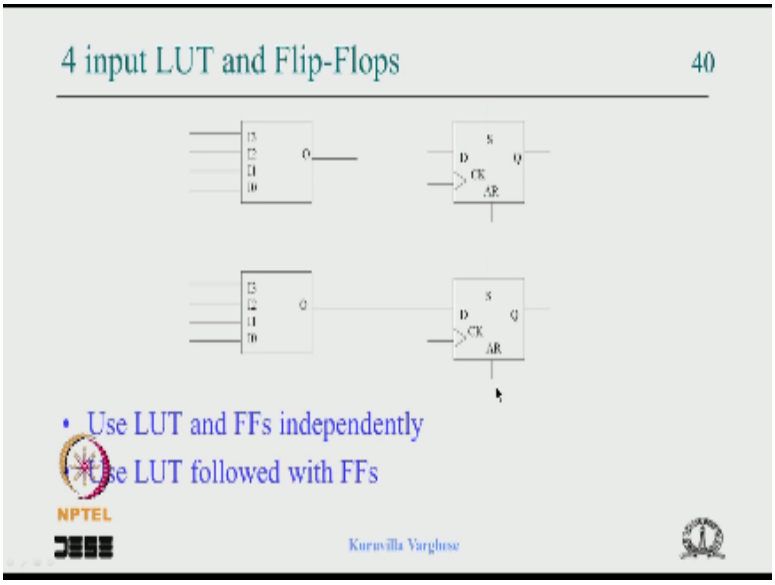
And F like when it is you know 0 e program this section F1 it is this is section, this goes like that in that F0 is program here is one program there and when you know E0 is program here, E1 program is there and F is 1 E0 is program here ok so let us go back to the slide diagram and you look at this assume that we have combined this look up to look up table to five input using an F5.

And assume that there is a slice before and behind that or before that that could be coming from the same logic block on a different logic block does not matter and now you can see that that assume that F5 output is coming here ok, in the case of single CLB this F5 output is connected to the F5 input of the next slide. So that is coming here and you see this F5 output and F5 output of the previous slice is combine in a F6 mux.

The select lines can be used the by can be used to use for the 6 variable ok and that F6 mux can be directly taken out or can be registered in this particular flip flop ok. Only disadvantages that if you come by if you make a 5 input look up table then you cannot and if you using technique out directly you cannot use this flip flop input supply early because the BX is already used this kind of fifth variable.

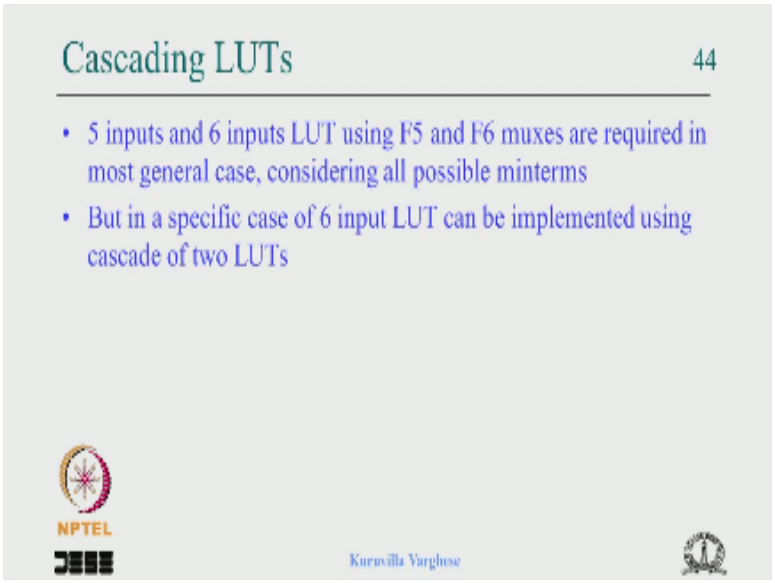
And similarly if we are implementing a 6 input lookup table this select line is nothing but the by. So then this flip flop cannot be used separately, so that should be kept in mind. So in a slice you can have 2 four input look up table or a five input lookup table in a CLB because it has two slices like one 5 input here and one 5 input here which can be combine to make a totally S6 input lookup table ok.

(Refer Slide Time: 37:14)



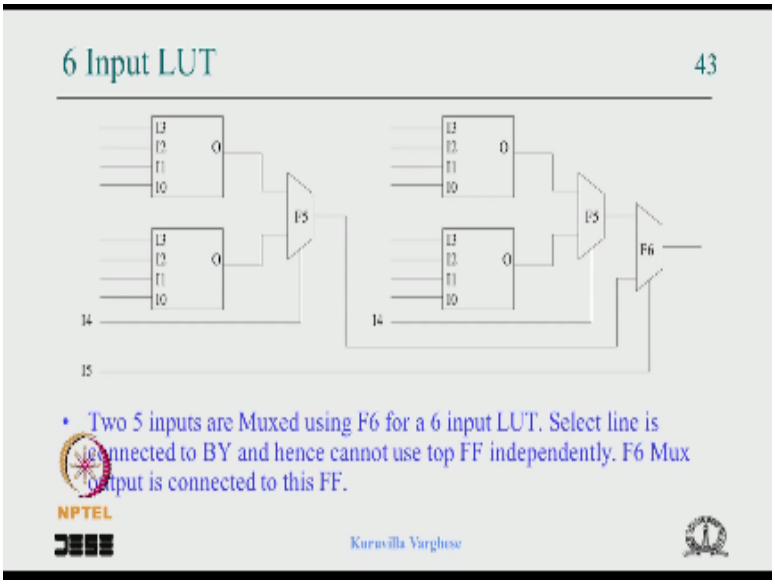
So that is what we have discussed ok.

(Refer Slide Time: 37:21)



Now like when we say that this should be kind of understood.

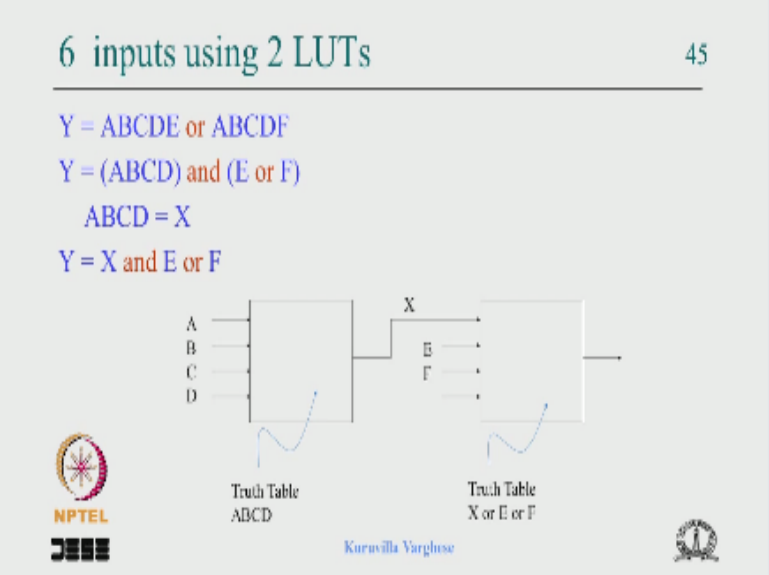
(Refer Slide Time: 37:29)



When you are when I say 5 variables like we mention that you have like a b c d e like we have 5 variable then you implement that it using a 5 input lookup table using an 5 input look up table using a 5 mux, many times did not required ok or you say you have a 6 input 6 variables then necessary not necessary that you have to go for a six input look up table for implementation.

You could in some specific is as you can cascade just have ok depends like this when we say it consider all possible min terms ok but in specific cases of 6 input look up table it can be implemented using cascade of 2 look up tables okay, if you built a 6 input of look up table from 4 input you need 4 of it, those two 4 input combines 5 input look up table and four 4 input combine 6 input look up table. So like and specification case take an example here okay.

(Refer Slide Time: 38:42)



Let us take an example say this is a function we want implement why is a b c d e or a b c d e f ok so you we have now 6 variable ABCDE, ABCDF, ok now one way of doing is that going to disconnect a F here, connect e here abcd you connect everywhere of all the four and you write a big truth table for you know whenever there is 1,0 and come back and program it ok but there is another way why you see abcd is common, so we take abcd out.

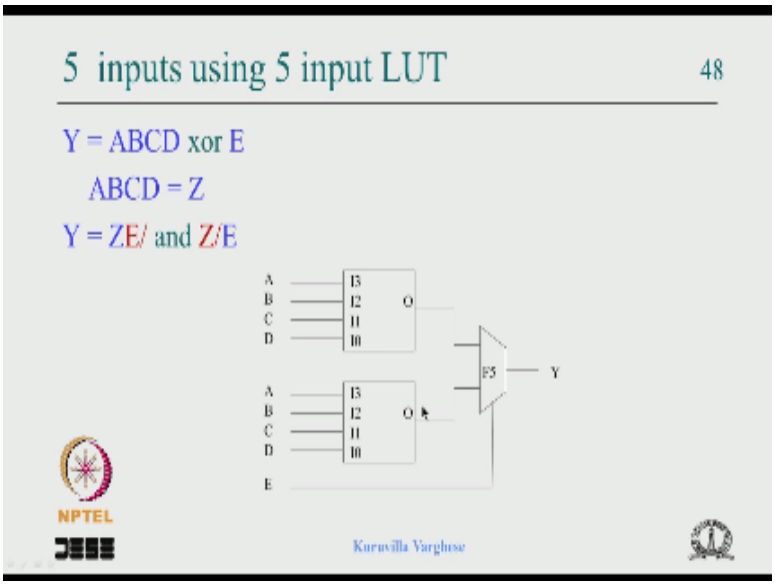
So it becomes A B C D and E or F E O ok, so what we can say A B C D is x and y becomes x and E or F, now what we can do as that one lookup table is used to implement ABCD which is X ok and output of the lookup table is connected as input to the next lookup table and then you connect E and F and have and you implement is x and E or F. It does not matter because we have seen that any Boolean function of up to four variables can be programmed in the lookup table.

So this can be implement, so this is 6 input but it can be used it can be built using 2 look up table and also 4 look up table that should be kept in mind and of course disadvantages that there are 2 lookup table delay you encounter when you implement like that, but if you may be like if you implement like this then it is nothing but one lookup table delay as a 5 mux delay which will be less than this look up table delay.

Not much of a problem because many times in FPG what dominates is a interconnect delay and in any case give you have say 2 look up table which is quite precious ok. So that is how it can be done. Similarly you can use say 5 input instead of using 2 look up tables using 5 mux you can kind of a you know combine cascade 2 look up tables and ABCDE can be

implemented as A B C D and E and we implement a ABCD X in a location and X is cascaded with E using another look up table, then you get it okay, you do not need an 5 mux or you can use a cascade.

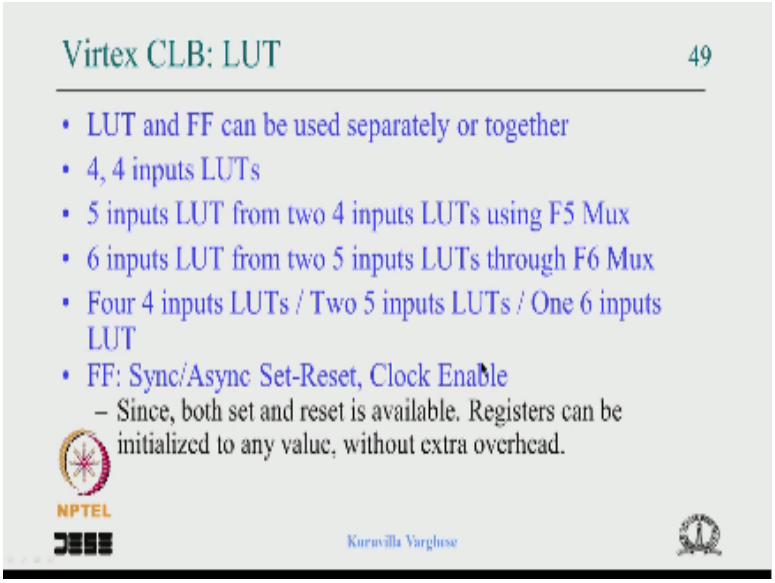
(Refer Slide Time: 41:32)



But there are cases like you know you show this particular function A B C D xor E ok now like we say ABCD Z that means we have expand it becomes ZX xor E that is Z E bar and Z bar ok then nothing come, like when it is E bar it is said Z and when it is E it is Z bar ok. So you cannot have a common look up table to cascade you need real 2 five input lookup table.

So your first use this architecture, so this can be expanded to 6 input cases, so there are 6 input function comes out which definitely need the 4, 4 input look up table or 1 real two 6 input table and then cascade.

(Refer Slide Time: 42:27)



So these are various kind of these all are done by the tool what I suggest that you know the VHDL I will demonstrate that kind of the tool to at the end of the course and during the case study and that time you can write all these and see how it is fitted and you can even see how things are routed within going to the floor planner tool to see what is happening inside the look table and all that ok.

I will you say that maybe I will show you are not take this code and show you but then I will show you in general how you can view the floor planning. So this all can be tried know that is how you learn you try something and then you see what is happening within the device is there is one way of those people kind of take the 2 and you 2 large lot of features you are right about the new click some button you choose some options.

And some magic happens hourglass come then lot of control messages come are you enable some optimisation and you get some advantage you get some time and the tool gives you lot of features some are used and some are not used, but whatever maybe that whatever maybe the technology, if you if you try to understand little bit detail by playing with fire, by playing with the quote, by playing with the design by playing with the tool option.

Analysing the various output trying to figure out what is happening right then you will get a deeper understanding all that the report turned out by the tools make sense to you and much more than that you get ideas how to optimise ok maybe that that inside okay many times the tool right in corporate all that back to the to write any special inside which is kind of you get

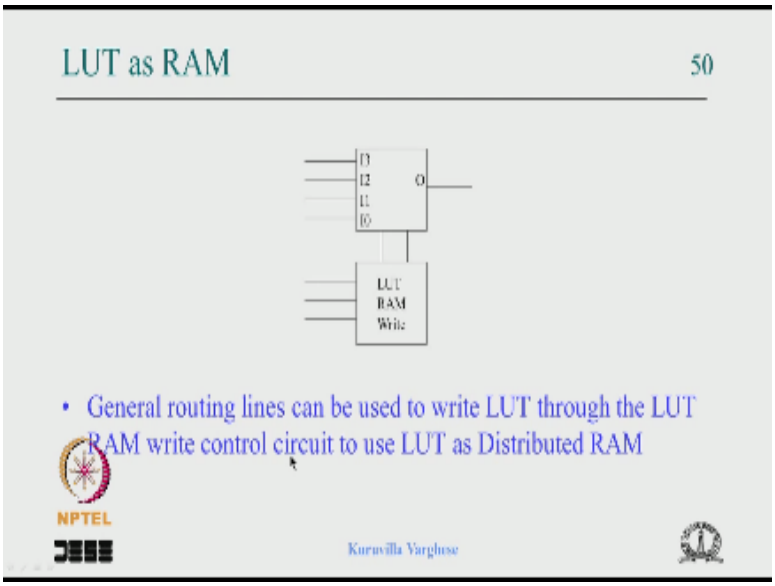
it from you know working out with various examples they try to incorporate within the tool that make you smarter.

So maybe like if your fat inside then that should be it can be communicated you cannot known it can be published and the vendors will take note of that and they will try to prove the device tools and so on, so that should be kept in mind so let us come back to the slide and this is what we have seen. So in a summary what Virtex CLB the lookup table and flip flop can be used separately or it has 4, 4 in look up table.

Five input lookup table can be built from two 4 input lookup table using a 5 Mux, 6 input look up table can be built from two 5 input using F6, so essential it is a 4, 4 input, two 5 input or one 6 input look up table of course we can do cascading and flip flop you have set reset clock enable and have you can one advantage of certain reset is that you can any sliced the registers to any value.

You do not have to say at the power on everything need to be 0 ok you can have a 8 bit register you can say that the power on it is be loaded with X value say three A like that. So that as a lot of flexibility like when you will complex function you might have some default registry values at the power on that can be easily kind of implemented using that.

(Refer Slide Time: 46:18)



So can we look at the slice diagram one thing you should not is that the when we use lookup table as a logic the read part of the circuit is available to us ok, the address lines are we connect the address line to the various signals input signals data line is used as output and this

memory is in the reboot ok now this memory is written during the configuration that means that the right control of the lookup table list to the configuration circuitry.

It is not available to the user ok at least at that like if you say it is dedicated right circuit when is not available to the user, but it may happen that this is nothing but a 4 address line memory, so it is 16x1 bit memory okay. Suppose you have an FPGA 50% of the lookup table is not used as a logical and you have used all the memory in the block RAM for some kind of has a memory.

And you want little memory but if you look at a lot of lookup tables are lying around and they are memories but you can use it because a right circuit is not available to the use ok. So what the Xilinx or other FPGA when do is at that light circuit is made available to the user ok. Now and you can use this you can see that this the right enable and data input ok. This is like internally you can assume flip flops.

The flip flop output is mux to the you know the O and the select line of the Mux is the address line okay the reach circuit but this is a input to the flip, d is the flip flop ok, again I disconnected in bus and this address can be used to decode the input flip flop ok. So that circuit is made available, so that if this look up table is not used as for logic, then it can be used as a memory ok.

So normally what you do is that you go to call generator which allows makes available the standard course, so you can instantiate the distributor Ram and sometime what you do is that you're right in VHDL say case statement like you say case on ABCD and you say 000 output is something 0,0,1 output is something else, for all the 16 possible values then this is an ideal candidate use a distributor Ram and it is put it in the look up table.

You might ask what is the different it can be treated as logic party has any it can go to block Ram also that it is in need to come to the distributed Ram but then we are discussing that look up table is not used as a kind of logic for logic implementation it can be used as a memory and since it is distributed it is called is RAM because it is all over in a block Ram it is in a one place.

But in a distributor Ram it is implemented you know it is distributed ok now you should know that this slide issue with and a thing is that when you have address line when you have lot of separate line particular you combine suppose you want a 16/16 memory that means 16 location 16 bit then you have two parallel 16 of the lookup table to get kind of 16 bit of the memory.

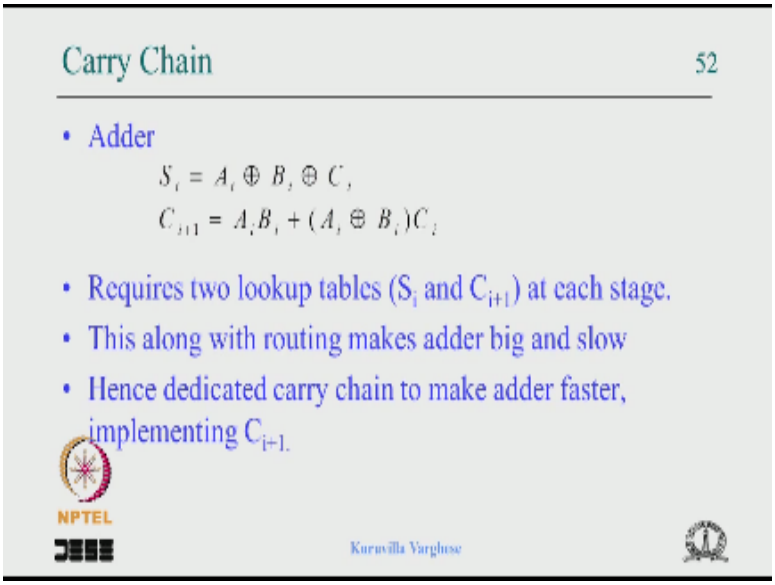
It may happen that you might assign the pin some of here some look up tables are close to the pin and some are away from the pin. So the read latency may be different for the different look up table and you will see if the read is not synchronise which is usually the case you do not synchronize the read you could synchronizer the read using the flip flop.

If you do not then you will see the variable latency and if you use it for some precision say like you are trying to generate a waveform using at back when you will find a zeta ok. So you should keep that that may not happen with the block Ram ok which is many a times output can be synchronised but you keep this in this in mind you know you used it a distributed RAM as memory which is spread across.

And we do not synchronised then you might get variable read latency and in some cases we can affect your implementation ok, that you keep in mind, so that is what is shown in slide so you basically you have a look up table. Now there is a block which is a lookup table right which is normally control by the time the configuration logic but now it is it is those wires are made available to the outside.

So that user can connect a the signals used as a memory and that is in the logic block in the slice diagram this is what is shown here ok. You have the address line at the data line the clock write everything is here. So you can I use undisputed memory, so let us look at the this issue.

(Refer Slide Time: 52:16)



Now maybe I will introduce that, so there is a carry chain within the FPGAs wise, now you know the equation of an adder, the sum is for any particular index a SI is AI exclusive OBI exclusive OC of I ok. That means if it is S7 is A7 xor B7, so this is a full adder ok and you can start a ripple adder by cascading the full adder and if you look at the carry I+1 is nothing but AiBi.

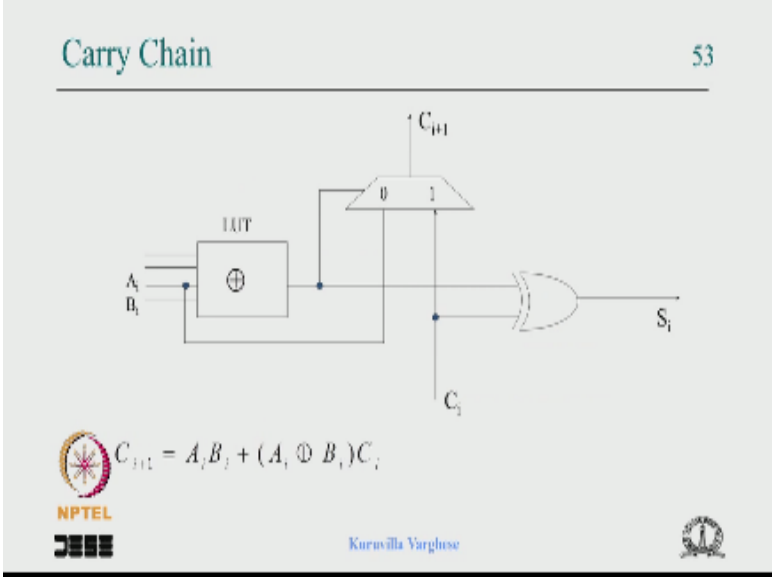
That means both are 1 and carry one or A exclusive Bi and neither of them is 1 and then if the carry input is 1 then the carry output is 1 or exclusive Ai Bi or A exclusive O Bi and Ci and many times we simplifying it by an inclusive O okay we say that Ai or Bi ok, it is bit of redundancy because when you say Ai or Bi it includes 1,1 ok not 0,1 1,0 1,1 but it does not do any harm.

Because 1,1 anyway generate a carry, so that is a bit of redundancy will look at this particular the correct optimise equation because it enable us to implement a carry circuit, now if you try to implement a ripple adder using full adder in an FPGA and suppose we are trying to implement this using the lookup table. Then like you need one lookup table you might give a AO Bo here.

You will implement S0 here and you will again give A0 B0 here and you will have to get you know you give C0 here and you will have to get C1 and take it out ok. Now combine the C1 and this S0, sorry C1 with A1 D1 to get the someone and another look up table for kind of C2 ok. So essentially what is happening is that you need 2 look up table per the full adder and all the interconnection is going to make everything slow.

So if the addition is slow at the multiplication you bill using the addition will be slow ok. So that is one issue, so if a separate lookup table is used for the carry logic then the interconnection will slow down the adder and slow down all arithmetic operation. So what is normally done is at this carry is kind of built in the career logic is built in to the slice.

(Refer Slide Time: 55:30)



So I will describe you the idea how is it and then we will go to the size so there is a lookup table, what is done as that and there is some additional carry logic which is part of the lookup table okay. So every look up table you take will have a 2 to 1 mux like this is connected like the select line is connected to the output of lookup table and one of the two input is going to the input of the mux.

And this output of the mux coming from the previous section is connected to the input of the mux and output of the marks goes to the input of the top mux ok because this comes from the bottom and this goes to the top mux ok there is a mux is here with the lookup table and this is an xor additional xor gate which is used for full adder implementation, it can be used for other purposes ok.

If permits that kind of location permit then you can use it and look up table output is going to the extra gate input and this is available as output ok. Now this implements the carry equation we have seen, so this Ci+1 Ai Bi+R Ai exclusive of Bi and Ci, that is implemented sorry that is implemented here ok. So how it happens is that you give Ai Bi here ok and that is Ai exclusive O Bi that is going to the select line of this.

And this is part and Ai Bi is this par ok, so that is that carry chain basically, so we are coming to the kind of the you know end of this part of the lecture, we will continue with the carry chain, so what essential we have seen today is a detailed inside a slice between the logic block. We have seen the look up table flip flop combining the lookup tables for various you now higher level complexity, cascading them.

And using the lookup table as a distributed memory and we are looking at the carry logic which kind of speed up the arithmetic operation. So please have a look at this the lecture notes you can look at the data sheets you can even look at the data sheets of other FPGAs, I suggest you look at the Spartan 3 FPGA data sheet little logic diagram which is identical and once you master it you can go to Spartan 6 Virtex 7 and so on. So please revise we will continue with this in the next lecture, I wish you all the best and thank you.