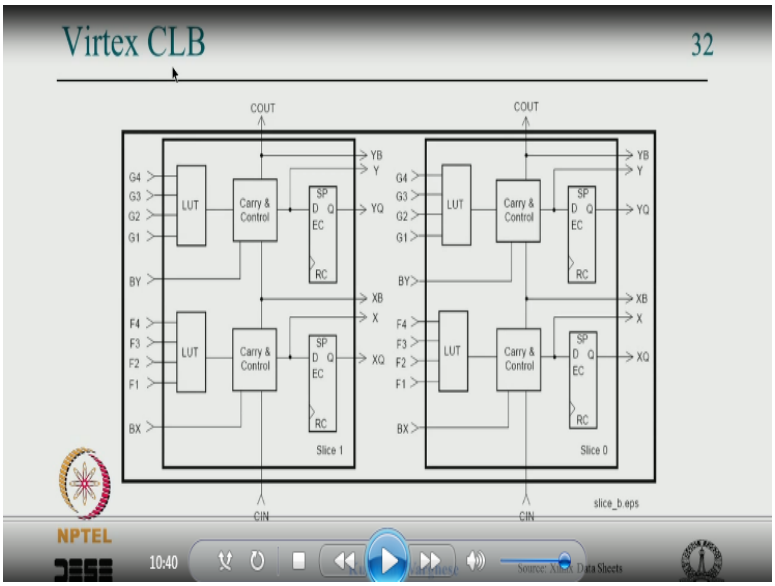
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-38
Xilinx Virtex Resource Mapping, IO Block

Welcome to this 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 Virtex, Xilinx Virtex configurable logic blocks and we have a look up tables, how that can be combined, then we have looked at the flip flops and we started with the carry chain or the carry logic. So we will continue with that.

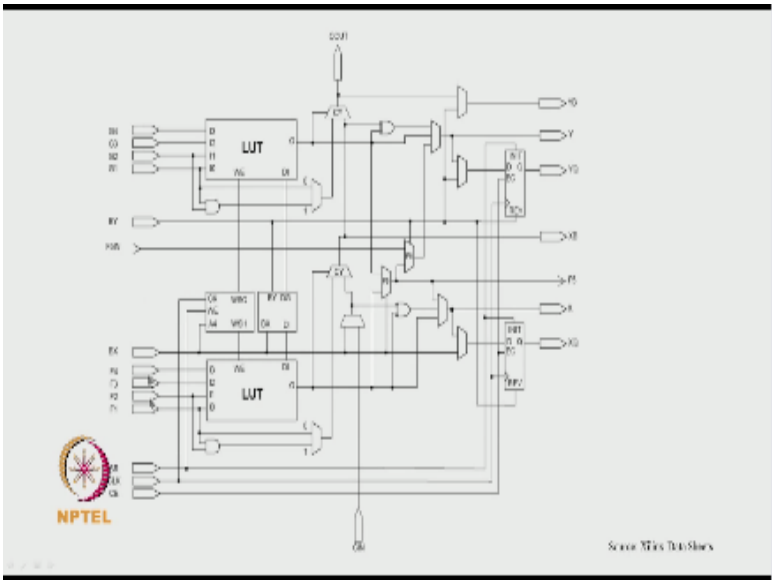
We will continue looking at the carry logic the flip flops and something with the clocking and all that. In today's lecture and so let us briefly look at the last lecture portion, so that we get some continuity.

(Refer Slide Time: 01:11)



See this is this is the CLB or the configurable logic block of the Virtex, FPGA, those two identical parts call slices and within a slice you have 2 again two identical part, 1 part consists of four input lookup table which can implement any function of any Boolean logic function of the four variables the output can go to a flip flop.

(Refer Slide Time: 01:41)

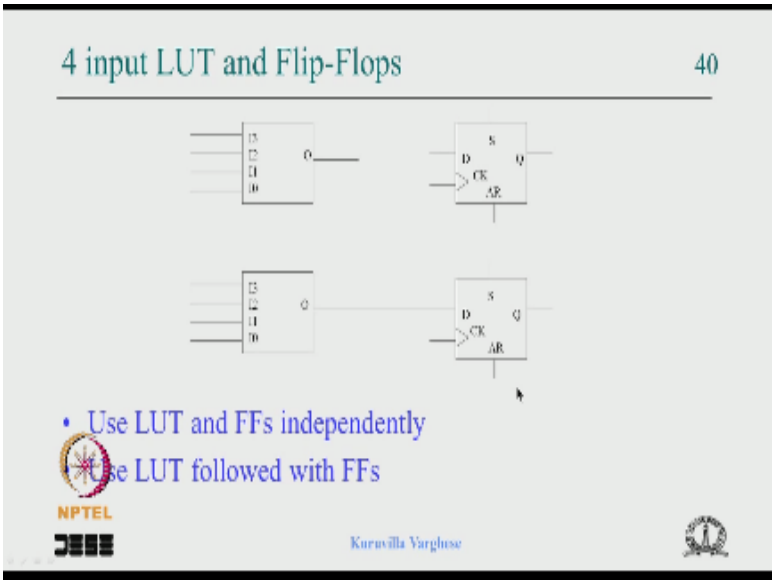


And the detailed diagram have seen here and we said that this is a four input this output that can be used as a combinational output in that case just flip flop can be used separately with this input if you want you can register the output which is the case of a data path or a sequential circuit and so on, you could combine the four 2 four input lookup table using a this Mux and the select line act as a variable and such 5 input 2 look up table can be combined with this.

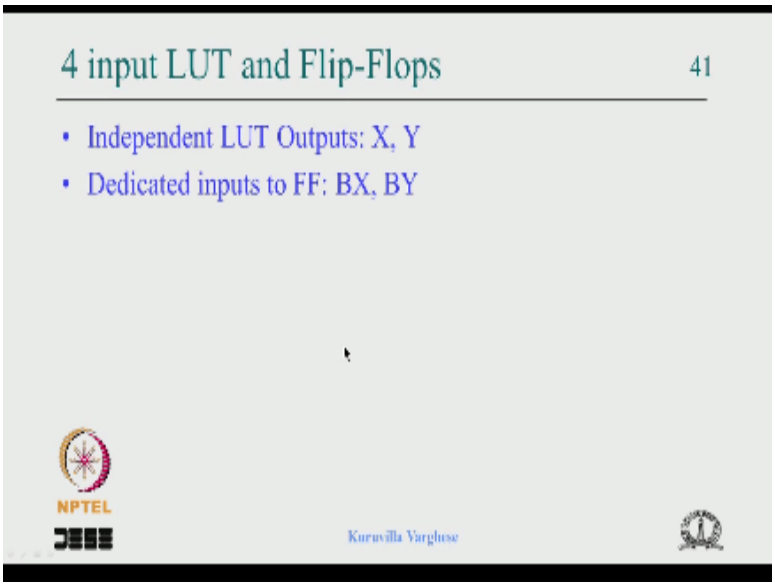
Because assume there is one here to look up table which is formed a five input lookup table the output of that if F5 Mux come here and F6 Mux can make the 6 input look up table and this is the 6 input and this can be taken directly out of the flip flop. In that case the like if you do not use a flip fop and if you do not register the output using a flip flop.

This flip flop cannot be used because this is used for the single input look up table, same is the case with the five input lookup table and this flip flop and we have seen that you know this flip flop as a data input output with 2 output set and reset ok and the clock can be connected to the option to connected to the different clocks or the user clocks ok that there are global clocks and the user clocks.

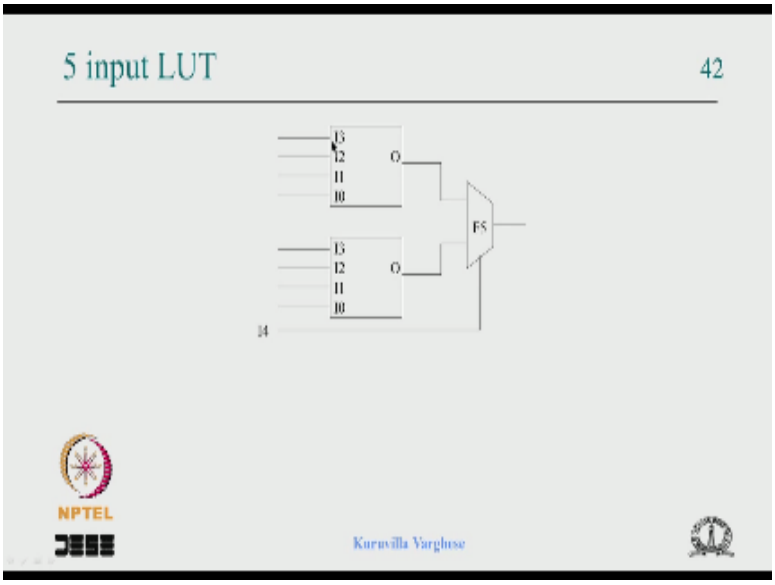
(Refer Slide Time: 03:36)



(Refer Slide Time: 03:37)

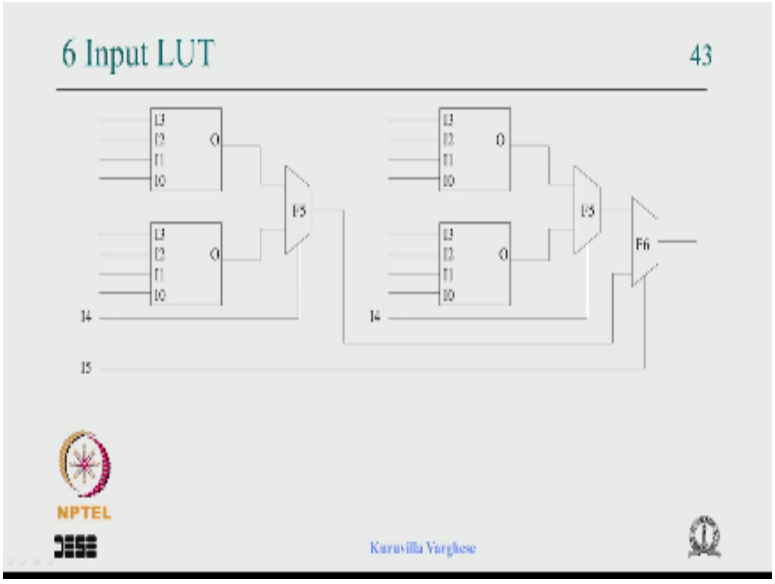


(Refer Slide Time: 03:43)



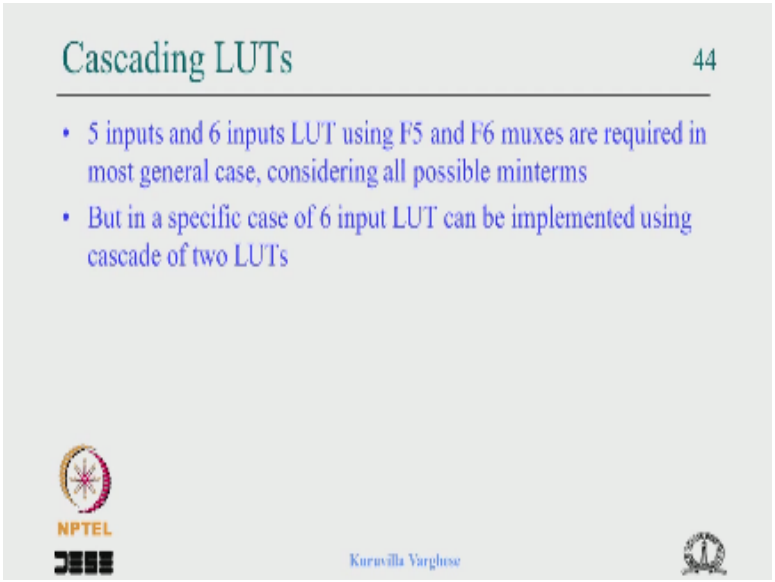
And we will come to this after I like we will what a CC will talk about as we go along so I have shown all that you know using the lookup table.

(Refer Slide Time: 03:48)



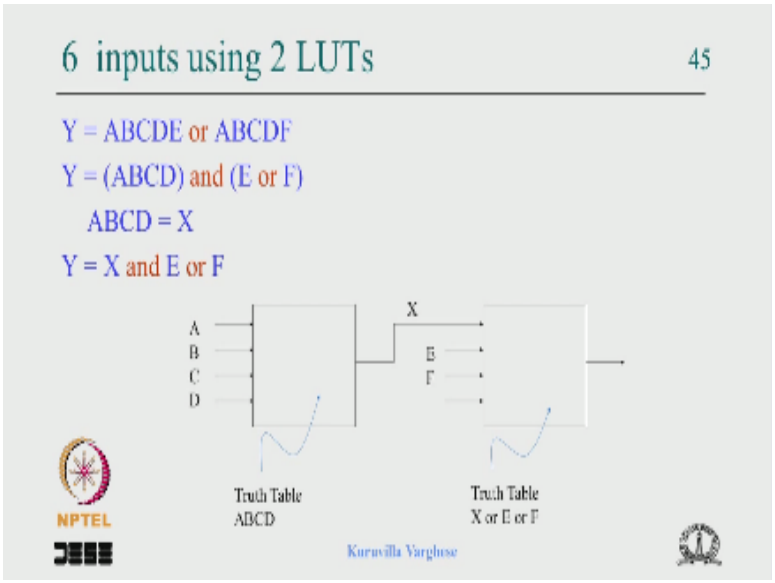
And flip flop separately together and forming the five input lookup table from two 4 input look up table and forming the 6 input lookup table with two 5 input look up table. So which consumes all the four 4 input look up tables ok.

(Refer Slide Time: 03:58)



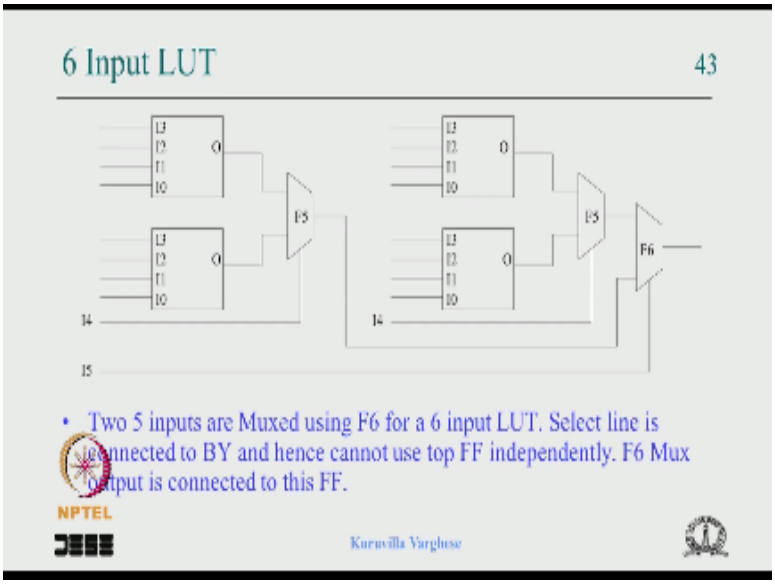
And we have also seen that.

(Refer Slide Time: 04:03)



It is not necessary that you have a 6 input function.

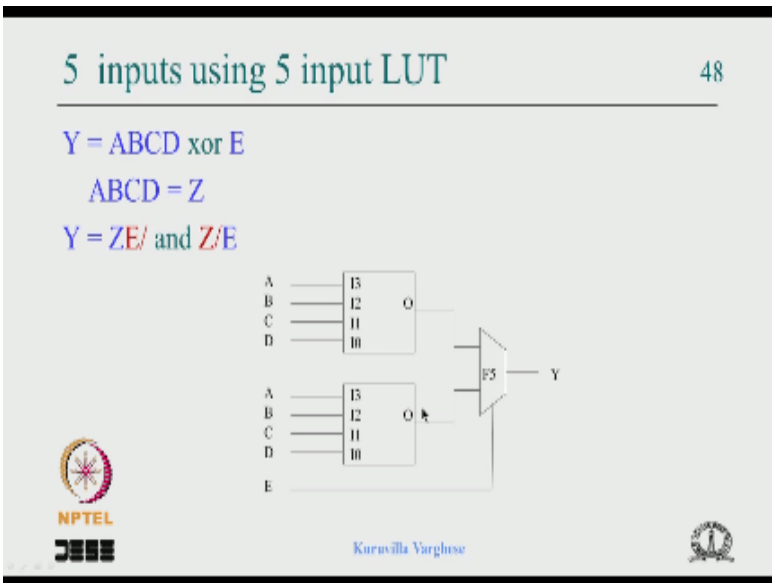
(Refer Slide Time: 04:08)



Always you need use the you know 4 look up table okay four 4 input lookup table depending on what function implementing you could cascade a lookup table and implement that we have seen an example of 6 input ABCD and ABCDEF, so there is a common product term and we take it out and implement that in a lookup table that output is cascaded.

This is that output is X, X is cascaded with E and F to form the real output the Y ok. So you know that whenever you have 6 input you need to go for a 6 input lookup table okay, you could cascade 4 input look up table.

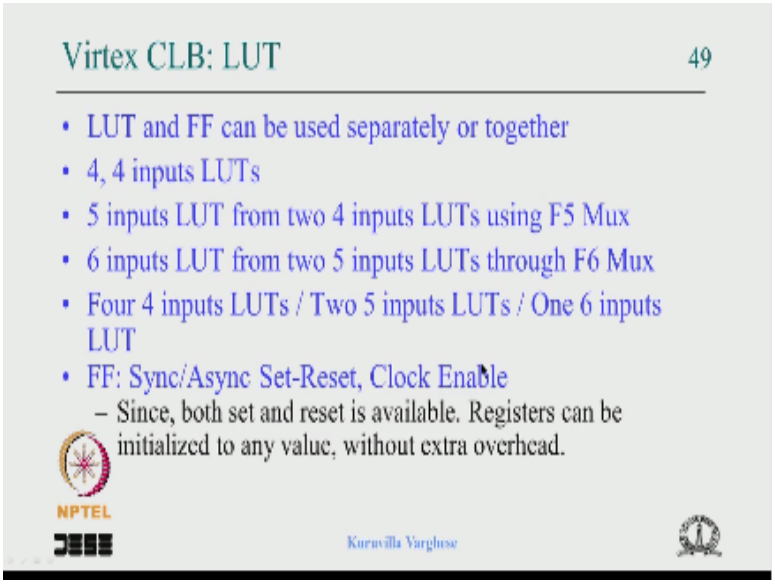
(Refer Slide Time: 04:57)



This shows a five input function using 2 cascade look up tables than an F5 Mux and combining them, but there are cases you know you take this function Y is A B C D xor E you

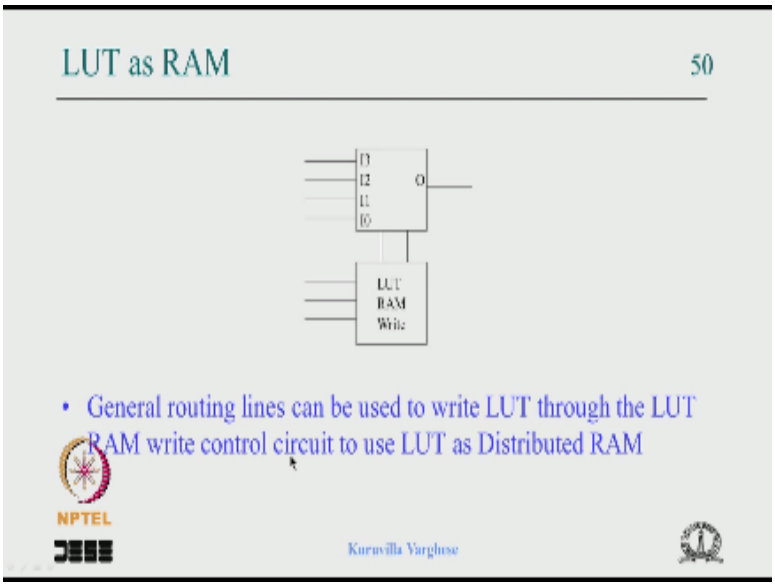
are forced to use the tool definitely asked to use 2 look up table with F5 Mux that know the way.

(Refer Slide Time: 05:20)



So this a summary.

(Refer Slide Time: 05:23)



And we also said that the look up table is nothing but a memory 1 bit memory and when it is 4 input look up table it is 16*1 RAM and the RAM right usually is controlled by the configuration circuit because the look up table is written during the FPGA configuration. So that the case is that if by chance there a lot of lookup tables not been used in your design this memory goes waste ok.

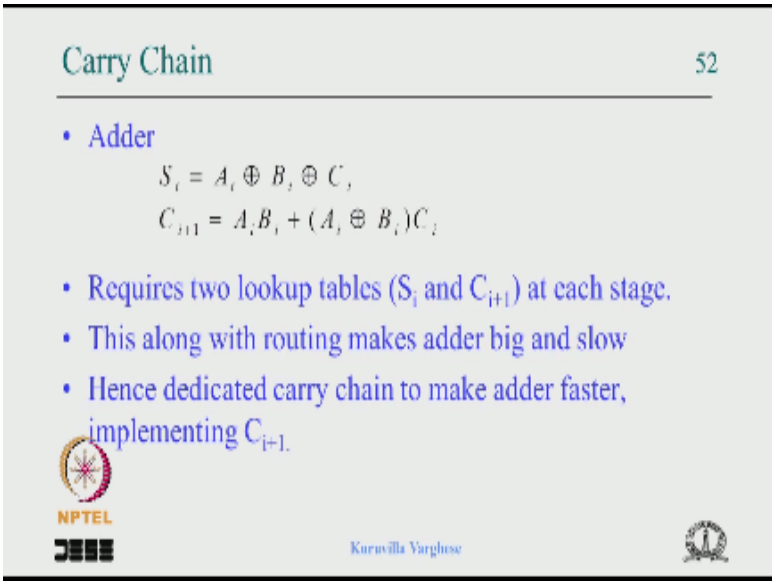
So the right control is also available to the to the general is routing line principal you could use this look up table as a Ram and use the tool for generator to instantiated the only thing is that then it cannot be used for logic, but if you have enough if your design where in lot of lookup tables are you know remaining free and we have used up all the block memories the hard code memories.

Then you could use this as a memory and since they look up tables are distributed all over, it is called distributor Ram and I mentioned that there could be unavailable latency because it can be distributed one need to take care because if the output is not registered and then there you will experience in the read different delays that could you know a kind of mess up your design supposed.

Example is that suppose you are generating trying to generate waveform by lookup table okay. Then you will find that you know in a waveform generation you will store the sample and output it at you know the regular intervals, and if the read access of the memory is variable then there will be and the distortion will be in the waveform ok you're my not think about it but this could be the result ok.

So if use the distributor Ram waveform generation huge waveform with lot of samples store then you will find that the waveform is kind of distorted depending on the frequency of the waveform ok. So it is a low frequency wave you are trying to generate may not create problem but if you try to read the memory very fast where in this variation in the in the in the latency of fact that you have a problem, so you should take care of that ok.

(Refer Slide Time: 08: 15)



So let us come to the carry logic so this is the full adder we are talking about ripple adder, so this is a ripple adder equation, the sum of I stage is AI exclusive OBi exclusive OCi and carry of that stage out output carry I+1 because this is that carry input to the next stage Si+1 ok and that is Ai Bi or Ai exclusive O Bi and Ci that means that the carry generated is both inputs are 1,1, then you definitely know that they carry generator.

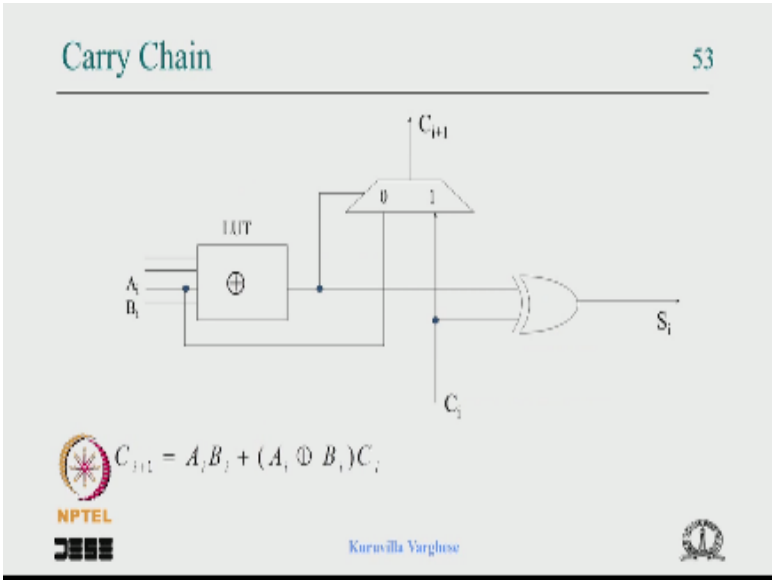
Does not matter the input carry is 1 or 0 because 1+1 is 0 and the carry goes out and even if this carry input carry is one, it just kind of you know just get it becomes one, so the sum becomes one it does not and if not we both are not want if either of the money is 1 then the carry out will depend on the carry input is the carry it is one day, so that is what equation say. Now if you implement this in FTP using lookup table you would need one lookup table for Ci+1.

Because lookup table has only one output, so for a 4 bit adder you need 8 lookup table much more than that you know that the Ci+1 should be kind of using the wire should be connected to the input of the next stage ok, so all of wiring is required and this will slow down the adder ok. Now if the adder is slow the multiplier will be slow and so on. All the arithmetic operation will slow down if the address low.

So this happens because there is interconnection between the adder changes. So the one thing we can do is at the carry equation can be implemented in hardware we will see how this implemented ok, that then can propagate like Ci there is a dedicated hardware along with this

Si stage and the Ci+1 go to the next stage without not through the programmable interconnection metrics or programmable switches it directly goes by the wire.

(Refer Slide Time: 11:00)



So that is how it is speed up, so I think we are just started describing in the last class, so this equation is implemented here and this look up table and this Mux ok that is what it does. So we will see that in the in the real CLB but before going there to this is a lookup table, a 4 input look up table, what is done for Ai Bi is Z and A1 and Bi is disconnected here mind you Ci is not connected here.

It is not that you do you implement the sum by Ai exclusive Bi exclusive or Ci was here in a lookup table is not done because you see the trick is that you see Ai exclusive Bi in common do the sum and carry ok. So the idea is to implement this common thing in a lookup table. So that is one done here, so you connect Ai Bi implement xor of Ai Bi and now this Mux will form the carry.

This xor will form the sum ok. So these are kind of dedicated circuit, so this will form Ai exclusive or Bi on the exclusive O BI and you can see that is coming here combining with the CI from the which is coming out of the previous stage, so this xor form Ai exclusive or Bi exclusive or Ci and some comes there ok and that can go out directly or can go to a flip flop. Now you see this the Ai exclusive or Bi is a select line of this Mux ok.

So if it is 1 assume it is 1 then you see the Ci, so that this is the part. So Ci+1=Ci, that is what is this equation saying this is 1 then Ci+1 is Ci ok, if either of the bit is 1 that carry depends

on the input carry, so of the carry input 1 then the carry output is 0, carry input is 1 carry output 0, it will not like this is 0 then the carry I+1 is Ai Bi, so you can see that it is 0 this path is selected, now there are 2 possibility, if this is not you know either of them is not one then there are only two possibilities both can be zero or both can be one ok.

So it is enough that you do not have to do an and just pick up Ai and connect it here if Ai is 0 then the carry out is 0 Ai is 1 then both are one because 0,1 1,0 this is a pass selected 00111 this path selection, both are one then definitely will be the one that is enough and this A+1 go to the second street exactly like that. So let us go to the CLB the slice diagram and you can see that say look at here.

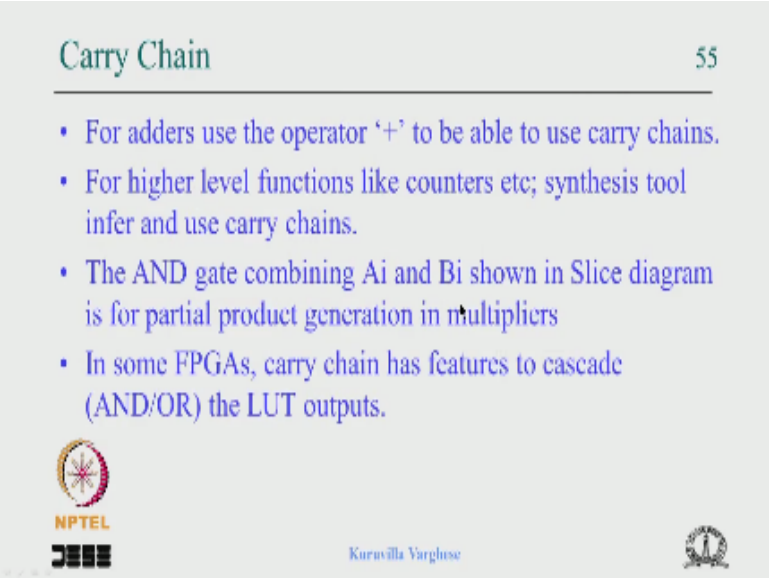
Here is where Ai and Bi are connected to form the A1 exclusive or Bi in this look up table and you see that that is coming here to this exclusive OR gate and the carry in from the previous page is coming through this Mux and is combined with A exclusive O Bi and you get a sum, so now you can take it out directly or you can register to from depending on the function of trying to implement if maybe you are trying to implement counter then it goes here.

If you are not registering it for whatever reason then you can it goes out directly ok now you can see the other function Ai exclusive or Bi is going to the select line of the Ci Mux and you see that Ci+1 is nothing but Ci then it is 1, it is 0 you see the the Ai or Bi is coming as input ok. So this is the carry logic which is implemented, it works definitely along with A exclusive or Bi here and it goes there.

And it it forms the carry logic for the next I+1 index and this is the sum, so that the sum is form using this look up table and xor gate. In the I+1 this look up table and XOR gate and carry circuit is here for the I state I+1 stage is this and so on it goes like that and you can see there is a Mux in Mux in this state this allows you can see that if you are starting the 0 state of the adder is this state.

Then you can have a carry input to the state okay like a the first state can have C0 or C in 0, so that is what is this for which is not available here, it will be available only in the next slice, so that is how it goes.

(Refer Slide Time: 16:23)



I hope that is clear to you, so the suggestion is that when you implement an adder in FPGA you do not write the equation of the adder using a loop or generate as we shown in the VHDL because when VHDL we were in discussing probably the FPGA implementation part we were discussing the VHDL synthesis ok. Now when you come to FPGA if you write the equation for a ripple adder.

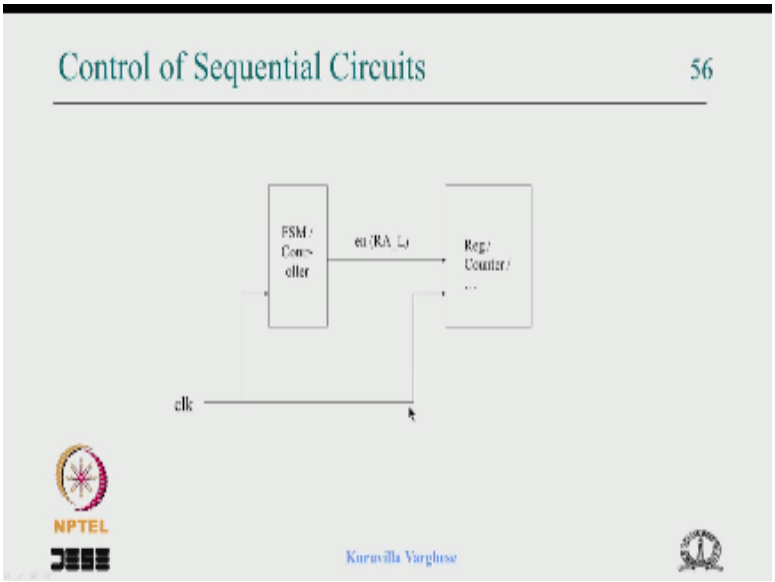
Then you will get implement everything will get implemented in lookup table and everything will be slow. So to be able to carry chain that there is one direct method there is very so cute as a method that mother is that use the operator plus ok. So you want to add a and b is same as she gets A and B ok straight forward otherwise you can go very low level you can start playing with the lookup table carry logic instantiate them interconnect them and all that.

But not a very good idea not required. So advice for that period designs that whenever you use try to use + operator is used a carry logic, when you use a counter and higher level function somehow the + will appear you say count gets count+1 and we have seen at in our example then automatically the carry logic will be used and there is an AND gate which shown earlier which is used for the partial product and limitation in the keys of multiplier.

And in some cases the carry chain can be used as for cascading that means that you implement a say suppose the inputs are four ABCD, then you implement A and B and C and D suppose you want to and it should say if say EF EH there is a way that you know it comes here and it in this kind of circuit and that anded here and we can take the output ok. So now all the FPGAs at least not the Virtex but in there are FPGAs where this AND cascading or OR

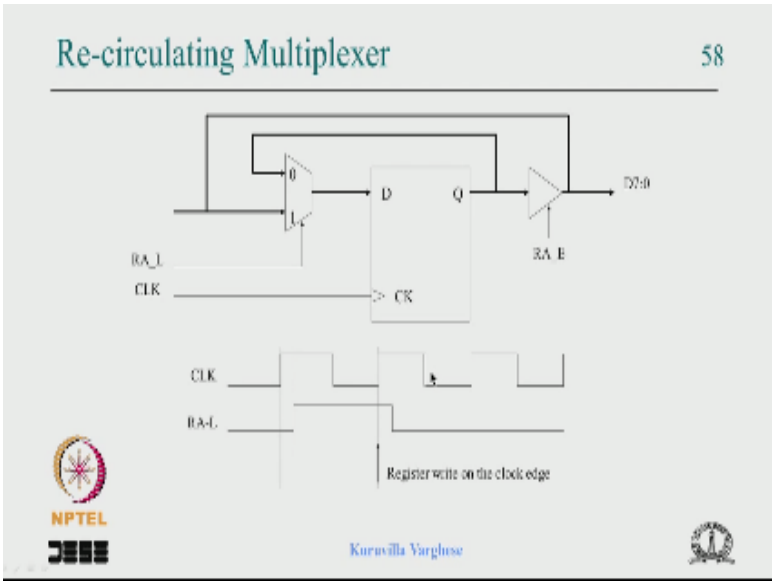
cascading can be done. And you can think about it is very easy with a multiplexer kind of scheme it is very easy okay. So that is this then in that case is call cascade logic or cascade chain.

(Refer Slide Time: 19:07)



So let us come to the flip flop now this part I want to talk about this E C or call enable clock and when we discuss our initial case study of the CPU we said the controller will control the data path many a time it control a enable register or a counter or things like that in such cases.

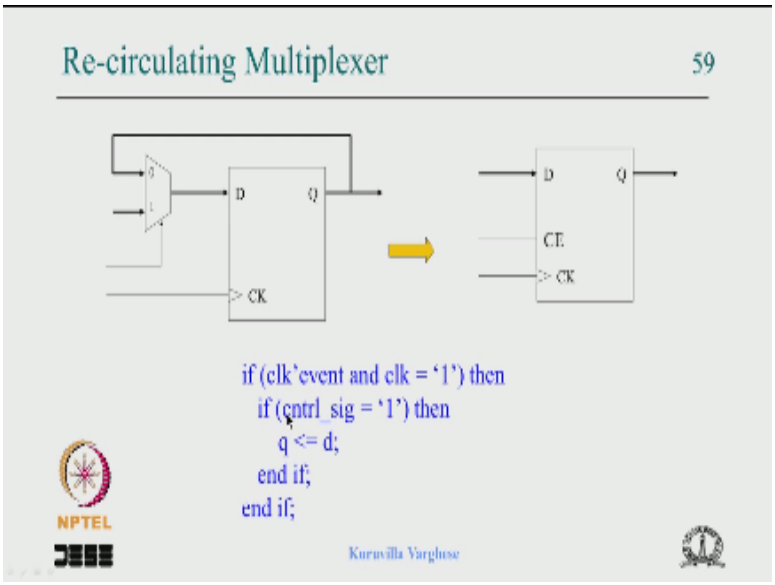
(Refer Slide Time: 19:37)



And we have seen one full tool way of implementing that is using it re-circulating 2 to 1 mux that suppose this is a latch signal from the controller and this is the register then what happens is that when the latch is one the input gets registered, the latch is 0 that output gets

re-circulated and there is no timing issue, if you do clock getting there is timing issue since the latch signal is coming from the state machine.

(Refer Slide Time: 20:14)



They will be a delay with respect to the to the clock the tcq+tcom delay and you can see that this enable signal will come outside to the clock period and in the next clock edge when the comes here to the register this edge the data, data is coming in from here it gets latched and the data has lot of set up time you know you have all the time to set it up ok because it gets enable much before the latching clock edge.

So that is what is ah re-circulating Mux does. Now this re-circulating Mux is built into the flip flop of FPGA, so the real FPGA flip flop is a kind of what is 2 to 1 mux plus if the flip flop come back ok. So this get into the to the flop and this is not ok stock enable. So whenever you see that will assume there is a 2 to 1 Mux with this clock enable as a select line which allows you to kind of implement an enable signal.

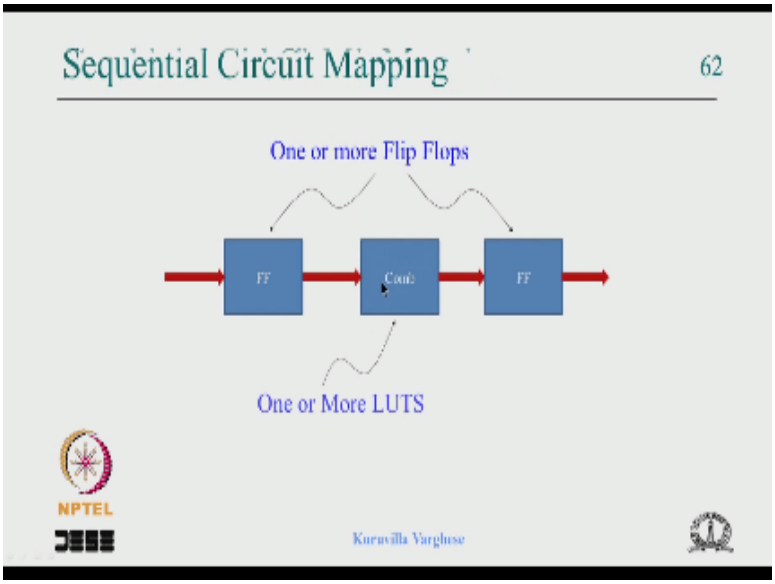
So you look at this VHDL code which say that if clock is 1 and clock is equal to 1 then give some control signal is 1 then Q get d ok. So normally some 2 to 1 marks required to implement that but in a FPGA since it is built in whatever is a signal is connected with the clock is connected here and this is the d this is it you are you do not have to do any one level of 2 to 1 mux is available.

But then you need for the control then you need to have access outside this would help you but in most cases on an average you need kind of 1 level of control enabling counter or

anybody register and so on so that is he is implemented using this clock enable and some vendor call it enable clock ok. So some people clock call CE clock enable some people call EC enable clock kind of to make a difference probably between a the device is ok or vendors.

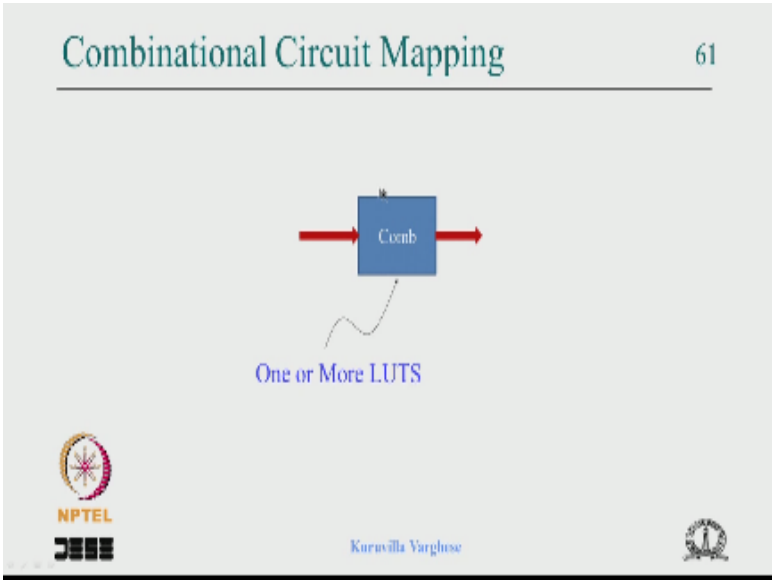
So now let us see how the circuit get map to this scheme of this kind of architecture, so one thing is that suppose you have combinational circuit that you know that look up table is used to implement, it can get combined if there are more input like two 4 input, 5 input, two 5 input, 6 input and so on ok and can be for the combined or it can be cascaded ok.

(Refer Slide Time: 23:28)



So basically when you have your have a kind of combinational circuit.

(Refer Slide Time: 23:30)



It implement one or more look up table, so it could be cascaded it could be combined using mux whatever it is, but if you have a you know datapath ok it can have sequential circuit but then assume as a datapath or part of sequence that does not matter you have a flip flop to combinational circuit or the flip flop ok. So you know there are in the slice of the CLB there are lots of flip flops.

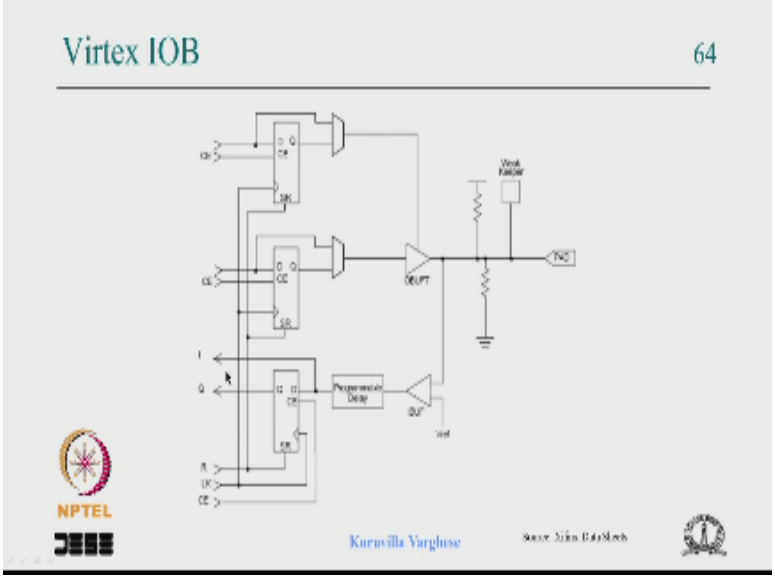
All these get implemented in that the flip flop of the CLB and the flip flop output goes to the combination circuit, the combination circuit goes to the flip flop and this get implemented in one or more. We have seen the diagram the combination circuit output can be taken to the flip flop ok and this can come from the flip flop before ok. So that is how it gets implemented.

And if you take an FSM finite state machine you have a logic output goes to flip flop, so flip flop is fed back to the logic and some input comes there. So again same thing you know you can implement the next state logic using lookup table with gets fed back flip flop through some is and input is coming with the wire and output of next state logic is going to the flip flop and a flip flop output again goes to the output logic.

So that if u look you have the lookup table which is acting next state logic the output goes to the flip flop, this output is fed back here to form the feedback and this can go to the next state lookup table to implement output logic. So that is how things get implemented in FPGA and let us look at the I/O block, ok so we have seen that all around the FPGA are I/O pin and attach I/O pin is an I/O block ok.

So the star of I/O block of the main element which implement this I/O functionality is this tri state gate ok. So when that there is an enable when there is enabled it act an output when it is cut of it act as an input. So there are essentially 3 part, 1 is so this is an I/O pin and it can assume the FPGA wires are here to which all this signals can be connected, so there are three part, 1 is from these wires internal wires to the output pin.

(Refer Slide Time: 26:22)



From the output pin to the wires inside their FPGA like this and there is an enable signal which is also control by the wire ok. So 3 part output input and the enable path, now all this can be combinational can come directly, can go directly and this is enabled path also can be combinational, but see there are two things we might get an in input which is not synchronise with the clock here.

So it can be input synchronize ok we have seen that the problem metastability we told about the synchronizer, let say synchronizer, so you have the option of taking the input directly inside to connect the wires or through a synchronising flip flop and that flip flop as you can have the clock selection multiple clocks, the clock enable the reset all that is there in the all the flip flop ok.

Now there is a programmable delay which you can add input delay to make the whole time 0, because that to me the whole time is quite a tough thing and we have seen that whole time for the whole time violation not to happened tcq+tcom mean should be kind of greater than the whole time, so many times to whole time is largely difficult me to this can be adjusted to make the whole time 0, but you increase the setup time it does not matter.

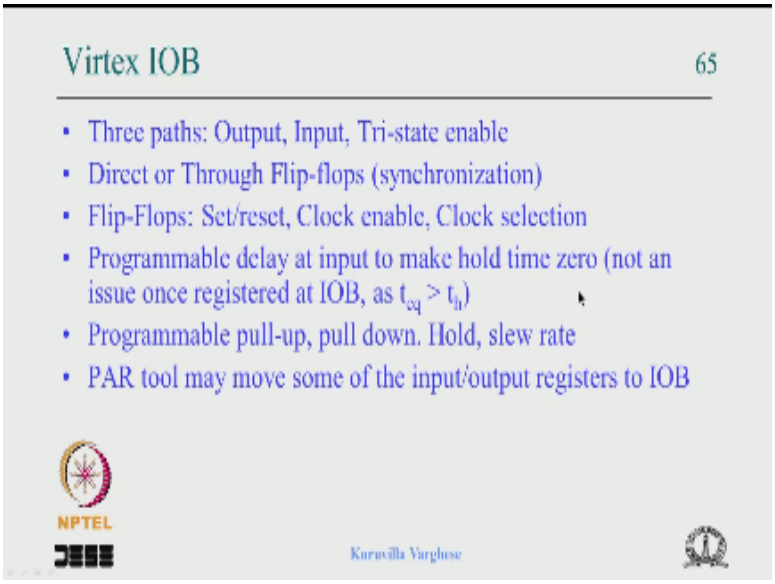
Similarly you have the enable path and that can j kind of come directly to a combination path or can come register ok. So that is a that is basic I/O block and some time some voltage standard you need to give reference voltage and that is this particular pins so these pins support various I/O standards which can be program in the constraint edit ok. We will see the

constraint editor when we discuss the tool, but this kind it support various voltages maybe 3.3, 2.5 other PCI standard.

Because where different things can vary and the slew rate in the program at the rate of change of output can be program, you can program a pull up resistor or a pull down resistor you discuss that if it is tri state that this will be high impedance and will pick up noise and if there is an output whatever is connected at the output those inputs can stop switching because it can be maybe near to the threshold voltage.

Then a slight noise make it go up and down to the actual logic level and the input circuit can switch and disappeared power, so this avoid that you can pull up pull down and there is a old circuit or a weak keeper which remember the last value. So if you can choose either pull up, pull down on this ok. This is a latch which latches the previously driven value very weakly.

(Refer Slide Time: 29:56)



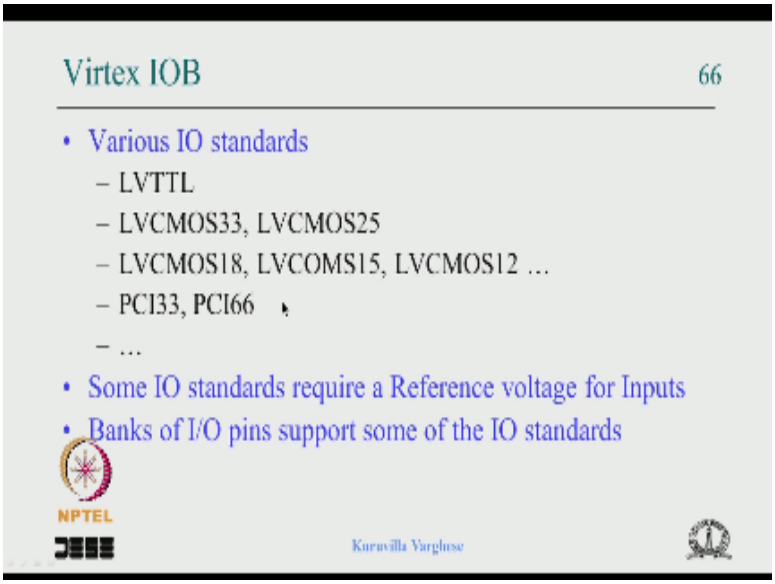
So that also can be a program. So essentially this is what that slide say there are three paths, output, input and tri state can be taken directly as a combinational path or through flip flops and flip flops as set/reset clock enable, clock selection and all that, programmable delay can make the setup time 0, so the whole time 0, you have the Programmable pull up, pull down, hold and slew rate all that is programmable ok.

You can program it sometime you know you have implemented something in a CLB ok and that the CLB output is taken to a pin or a pad, so the what they tool does is that, it move that flip flop ok to the to the I/O block because it is very very nice to have the last you know

latching or registering at this point, because it is summer inside CLB then again you have to take the wires that will had lot of wire delay.

And which may create timing problem with the following you know the circuit following logic time, so many a times it is a wise idea to register the output before leave the pin because that goes with the minimum delay after the clock at there is it easy to delay so there will be lees timing problem for the input ok, which was using this output as an input will have less problem.

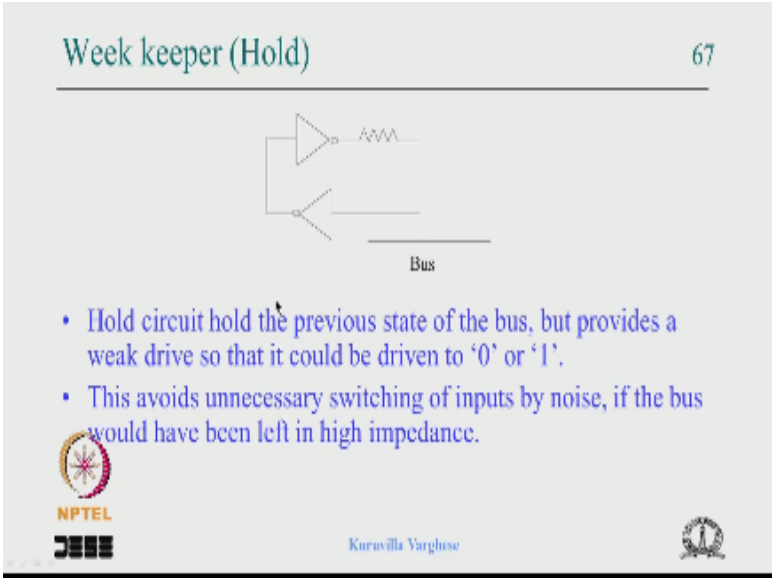
(Refer Slide Time: 31:54)



Similarly like we have the in which was asking we can synchronise everything is synchronised this tri state it enabled also can be synchronised and it support wireless standards 3.3, 2.5, 1.8 and all that and all the time there will be kind of course supply which supply to the internal and the I/O block supplier, because the core can work at a low voltage resulting in low power dissipation.

And when it comes to the I/O pins this can be scaled up because you if the screen is more than be more participation and so many times the most recent if not earlier FPGA are the core will work at a low voltage in 1 volt and when is the pin it is scaled up to 2.5 or 3.3 volt, so that is done in FPGA tool.

(Refer Slide Time: 32:38)

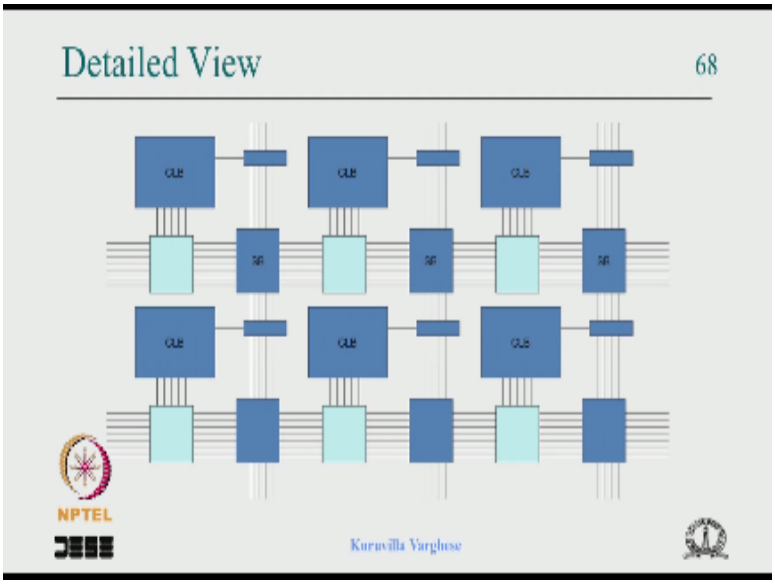


And this is the hold circuit suppose you have a bus and if you program the whole circuit to the bus then what happens is that suppose the bus and what happen is that suppose the bus is given one then what happens this 1 is latch here and 1 comes here, this becomes 0, this 0 will become 1 and it gets latched, but if you put a normal latch then this invertor will drive that bus was very strongly.

And if somebody some other output is driving at 0 and if this is driving at 1 then be a short circuit, so that is why high resistance is kept here which will be clear drive the bus and hold on to the previously driven value very weekly, very weak team is that since it is true or some other output can pull it okay make in 0 and since there is a high impedance nothing happens.

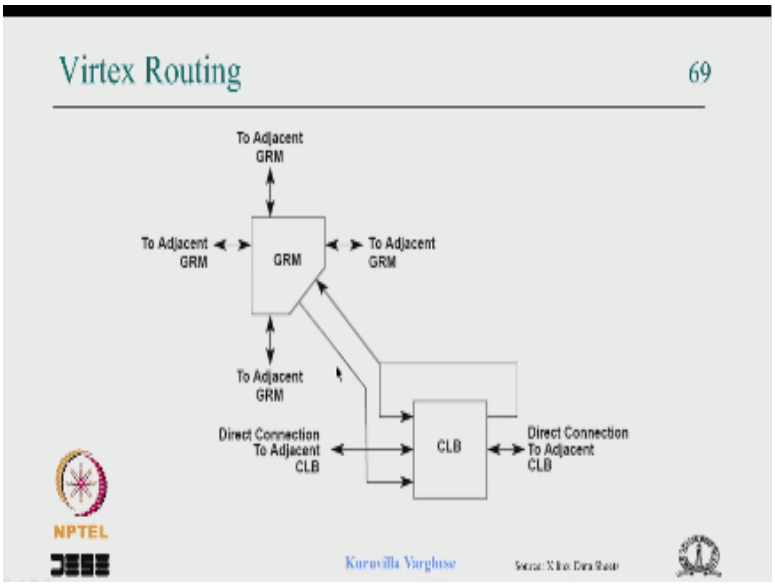
So that is what the purpose of the hold circuit which remembers the last value with week keeps and hold on to the last value of the bus and thus avoid the glitches and switching under section which is the spitting power and create noise and so on okay.

(Refer Slide Time: 33:55)



And this is the view of the internal wiring structure you have lot of wires, there is a huge switch here which connects all the vertical horizontal wires together and then input to the CLB from here, but in Virtex CLB there are connection from this switch directly to the CLB ok.

(Refer Slide Time: 34:20)



So that is what is shown here, this is that switch which connects to the adjacent switches. So this is that switch here and this is the logic block so this is the logic block hear what it shows is that that is which has direct connection to the CLB ok and the CLB output is fed back to the CLB input. So you need not go through the general wires ok, it is not CLB output is directly taken here.

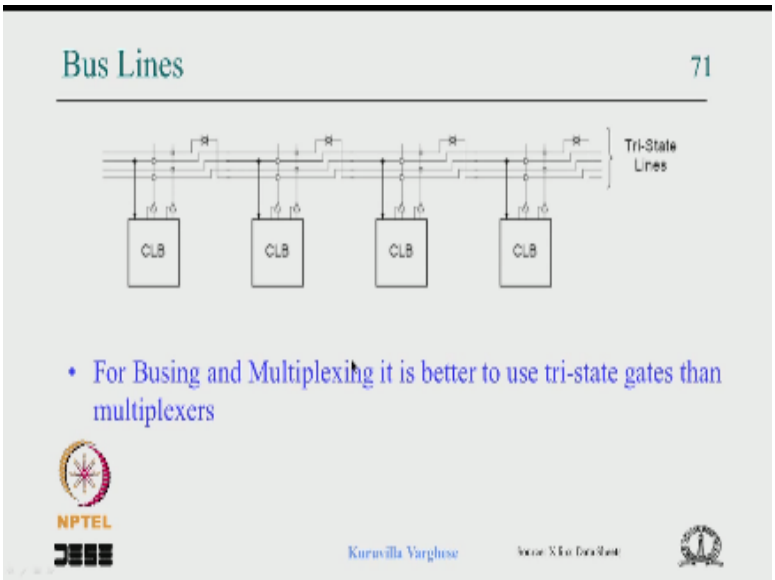
Suppose it has to be taken to the input need not go like this come all the way back and from here okay, it can directly go to the input and it also shows that if address the CLB directly connected, so there is a CLB here, that there are some output going to the input of the CLB of similarly some output from CLB is coming to the input of CLB. So there are no programmable interconnections.

So it is very fast you know the interface between the adjacent CLB is very fast ok, that makes if things are close together the delay between them is very minimal ok. So that these are kind of some statistics of the vertex routing wires and the adjacent CLBs are directly connected, there are 24 single landline, so it means that you take a switch like this between the switches here there are 24 wires okay.

Between these switch there are 24 wires, between the switch there are 24 wire and you have 72 buffered hex line that is from one CLB to the 6 CLB, so from here to the 6 CLB there are single wire running ok. So that makes the lost very large you know lengthy connection less costly, otherwise I would single end wire it needs lot of interconnection to reach the destination and all had the delay.

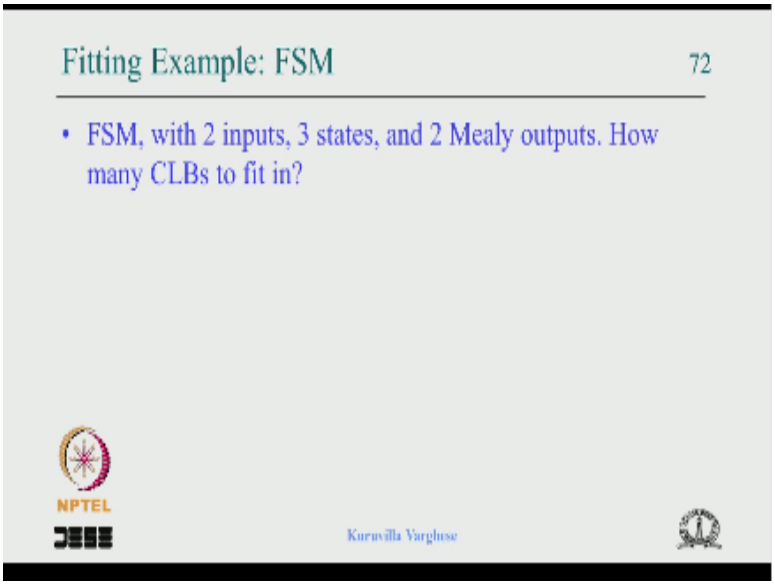
But this will reduce the delay from the first stage to the sixth stage okay, and there are 12 buffer long line that means there are lines running from one end of the chip to the other end of the chip ok and not only was only but vertically also ok. So per channel will be 4 tri state line both horizontal and vertical.

(Refer Slide Time: 36:51)



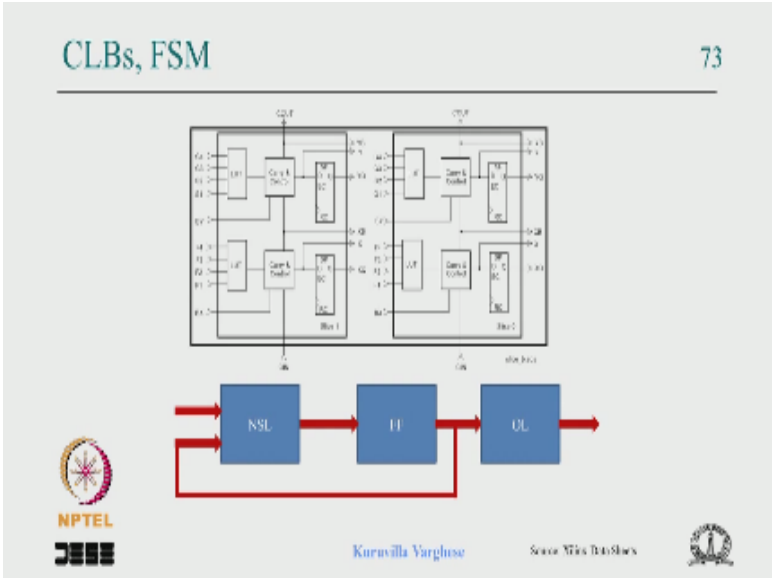
And goes there are separate line which can from tri state line or busing, so every CLB has 2 output, 2 tri state gate which is connected to some 4 common wires in some fashion, so essentially and 2 output is connected to 2 wires, so you can form a bus using this tri state gate as the output of the CLB, that is possible. So let us now ask having learn all the architecture.

(Refer Slide Time: 38:04)



We have looked at basically the architecture of the CLB we have understood how that is interconnected and so on. So let us just try to estimates on resource requirement, so my question is that say we have a finance it has 2 input or external input thus three states and 2 Mealy outputs, how many CLBs you require in a Virtex FPGA to fit this finite state machine that is a question ok.

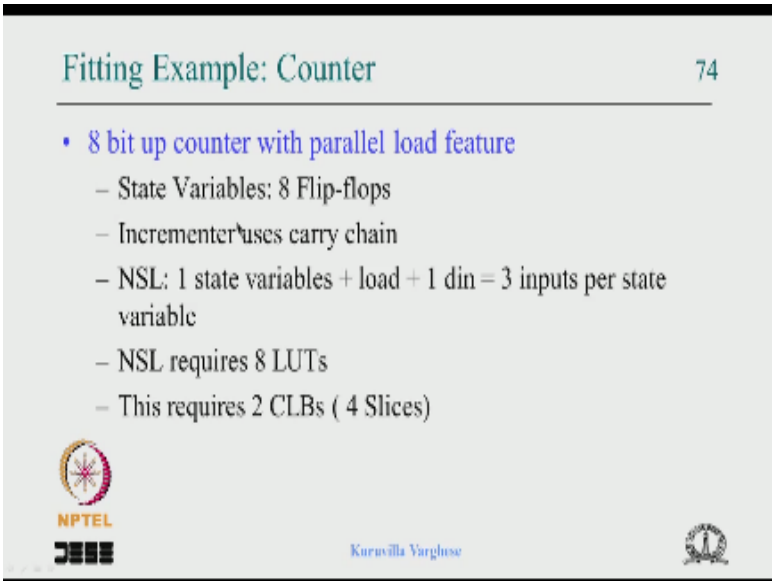
(Refer Slide Time: 38:07)



So if you look at the finite state machine you have next state logic which decode the present state and input ok. So we have to look at to decide how many look up tables this will combine this require we need to have we need to know how many flip flops are there and how many inputs are there okay similarly you have to determine the number of flip flops required and you should remember that next state logic is required for each flip flops.

There are 4 flip flops here then there is a 2q3, Q2, Q1, Q0, so there will be D3, D2, D1, D0 and the next state logic should give 4 output to connect to D3, D2, D1, D0 ok, so let us work out other this requirement and this requirement for this case and we know that the Virtex CLB has 4 input look up table and flip flop and set four of them ok.

(Refer Slide Time: 39:19)



So we have a this was example we have a finite state machine with 3 states, so it required 2 flip flop implement in a binary encoding. So the next state logic has 2 inputs and the present state which is two 2Qs from the current flip flop, so the next state logic will have input 2 state variable, the present state and 2 inputs so that is all, so 4 input. So and you know that there are two states like to Q1 and Q0.

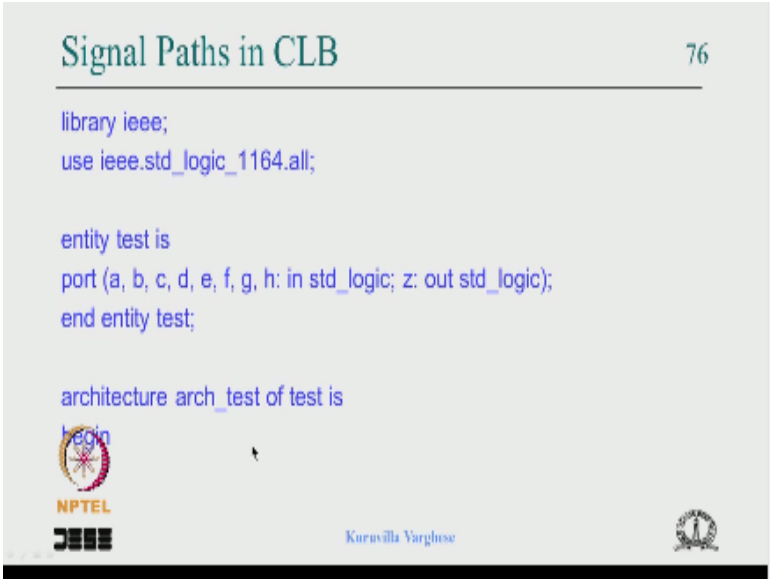
So we need to decode D1 and D0, so there are two 4 input look up tables are quite ok because one lookup table is required for 1, 1 bit and so you have required two 4 input looking table for the next state logic and when it comes to output logic we have Mealy output and that decode the present state and that not shown but input was ok. So there are two flip flop, 2 input.

So there are four input output logic and that occupies 4 logical require for 4 input lookup table, so that is shown here, that is you have output logic as 2 input because we say to input and two state variables because there are 3 states and Q1 and Q0, so you require a 4 input look up table for output, but we have 2 outputs, so we need to two 4 input look up table, so we need two look table for next state logic, 2 look up, 2 flip flop for state variable and 2 look up table for output logic.

So if you look at a four 4 input lookup table and 2 flip flops and in CLB we know that in a CLB we have 4 look up table and 4 flip flops, so this require 4 look up tables and 2 flip flop. So basically this require one CLB-2 flip flop. So you can see the kind of the power of these architecture a small state machine required 1 CLB to implement okay. Similarly let us take an example of an 8 bit counter with parallel load feature.

That means a normal 8 bit counter with the reset and also it has a synchronous load ok we have seen the circuit of this counter. So if you look at it you have the 8bit, 8 Q7 to Q20 flip flop and that is incremented and given here. There are for parallel load they asked have a kind of load signal and data signal ok that is that is required for Each flip flop will be take there will be kind of 8 things fed back and you have the load and the data in and all that ok.

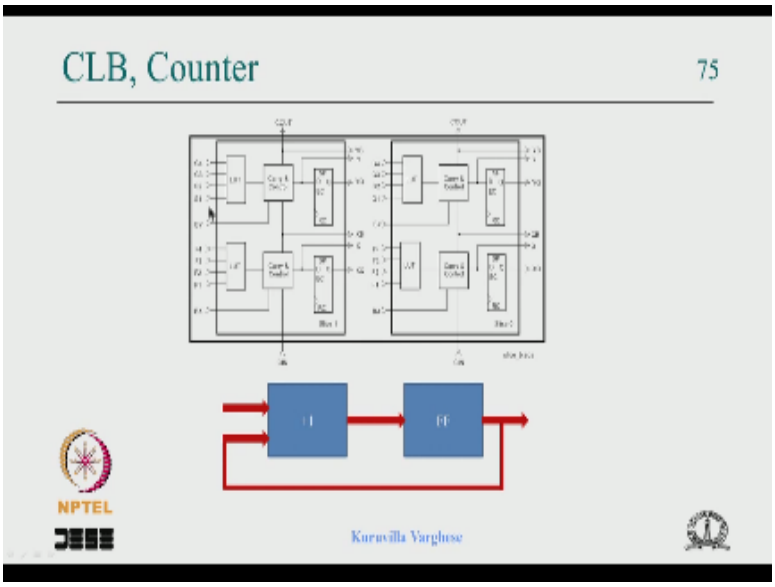
(Refer Slide Time: 42:46)



So we come to this 8 bit counter, 8 bit flip flop now the incremental can use carry chain, so essentially we need only 8 look up table and the carry chain will implement the +1 okay because you know that you can have an input adder with the carry input 1 then will

implement the +1 and that can go to the to the flip flops here, but so you have the input suppose whatever Q will come here.

(Refer Slide Time: 43:17)



And the carry chain will use at incremental to the +1 this BX will be 1 and that will add 1, 2 everything, you do not need, I need to give a +1 and the input of the lookup table but this can be used for plus one so that that is all the counter get implemented not only that need a lot feature and a data in input. So that you know that now the low input given here and the data in for that corresponding variable can be even here.

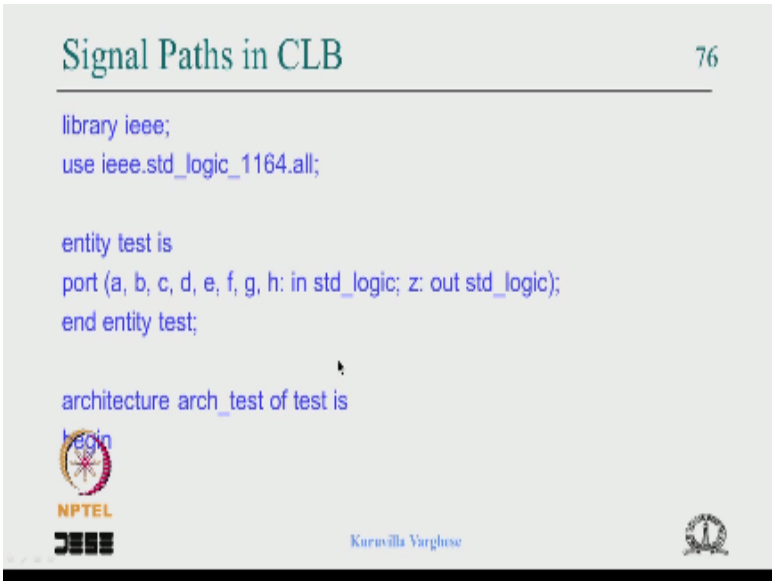
Still we need only at 3 input look up table, but there is 4 input look up table. So that is enough to implement increment plus the multiplex of multiplexing the input value ok. So essentially we require 8 flip flops, 8 lookup table with the carry chain, ok so that is what is required we have 8 flip flops in reminder use carry chain next state logic will use one state variable for +1 and one load signal and one data input signal for loading.

So 3 input plus state, so NSL next state logic would look up 3 input table, so we have we need search for 8 flip flop 8 look up tables, 8 flip flop. So there are you will end up 2 CLB or 4 slices ok. So you can if you have a circuit if you know what is inside a circuit you can estimate how many how much resource is required to implement the circuit okay knowing the logic lookup table.

So as I said we are discussing the Virtex architecture other what Virtex chip is not available now, the higher the reason version of the Virtex 5, Virtex 6, Virtex 7 different versions now or

you can you can look at the and you have the Spartan 3, Spartan 3 has a similar architecture as a Virtex, but Spartan 6 which is quite different and so all these you know now you can go back and look at the architecture of the CLB and you can understand the functionality you can be given a code given a spec, you can estimate how much how much lookup table how much flip flops are required to implement this functionality ok. So that is that can be easily done.

(Refer Slide Time: 45:56)



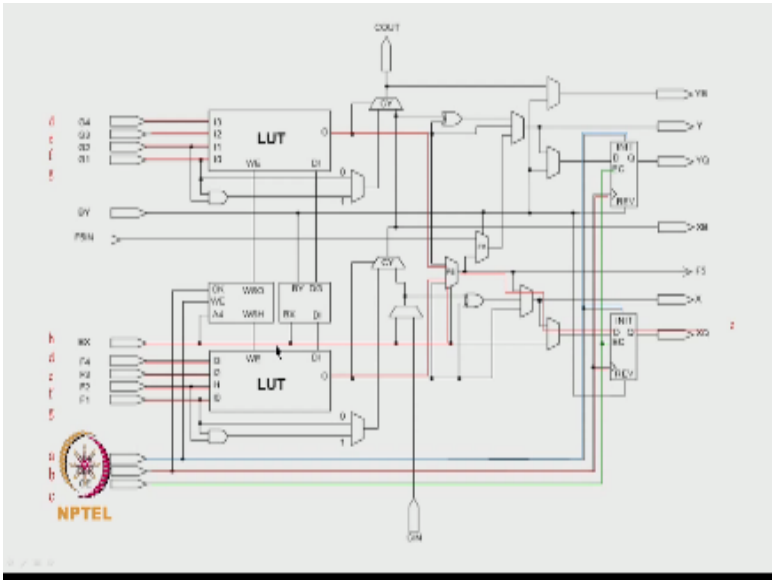
So we will take one more kind of example what will do is I will give a VHDL code and let us see how that VHDL code gets implemented within the CLB the logic block using slicers ok. So this is the library and the package declaration, this is the entity declaration very simple entity, a b c d e f g h, so you have 8 input and 1 output the combination logic the architecture you write a process which sensitive to A and B.

Begin if A is 1 then Z is 0, so it looks like a reset, a is a reset, reset is an output else if B ticj event and b=1, so we know that b is clock is reset Z is output underneath we say if C is 1, then Z is D, and E and F and G xor h ok. So we again or that when you say under the c=1 it is a control signal ok. So this can go to the clock enable that enable reset and Z is and for variable xor the flip variable and we have seen we have seen we have discussed this in the last lecture.

You cannot cascade to look up table you need combined to look up table with an F5 Mux to implement this because this is a bar and this or this as it is and H bar ok, so that is how the equitation comes then you need 5 input lookup table. So that is the essence of the circuit has

have not shown the synthesis occur because is easy to imagine is the A is a reset, B is a clock, c is a enabled clock=, z is output, this gets implemented in a 5 input lookup table.

(Refer Slide Time: 47:58)



So let us come to the slice diagram, so we need to combine two 4 input look up table using the F5 Mux ok, So d and e and f and g are common, so you see d e f g is connected to the wire to the top look up table, same d e f g is connected to the second look up table and that is you can see the output is going to F5 Mux, the select line is fifth variable which is h.

So our logic x implemented here ok, so that is what is the logic here that goes any of the flip flop, now we have to implement c that is nothing but so this flip flop output asked to go to the flip flop here we can see that this state 2 Mux through this Mux it comes the flip flop and you see the enable clock is connected to the to the c because that control signal as the code suggest, now d is a clock and you can see the clock is this pin and that goes to the b ok.

So assume that there are wires before the lookup table in the channel and this is these are connected to the wires are using some switch there ok. Now we are one more point A is a reset, so you can see that a is a set, reset is connected here and that blue line to the unit of the flip flops of that it gets reset. So you can even the idea here is that there is no magic in most people do not look at the CLBs ok.

So however complicated the CLB is nothing much it is lookup table, now you go to a higher FPGA you might have a more complex lookup table like 4 input look up table, they will have 6 input table and sometime within that look up table that lookup table itself is composed of

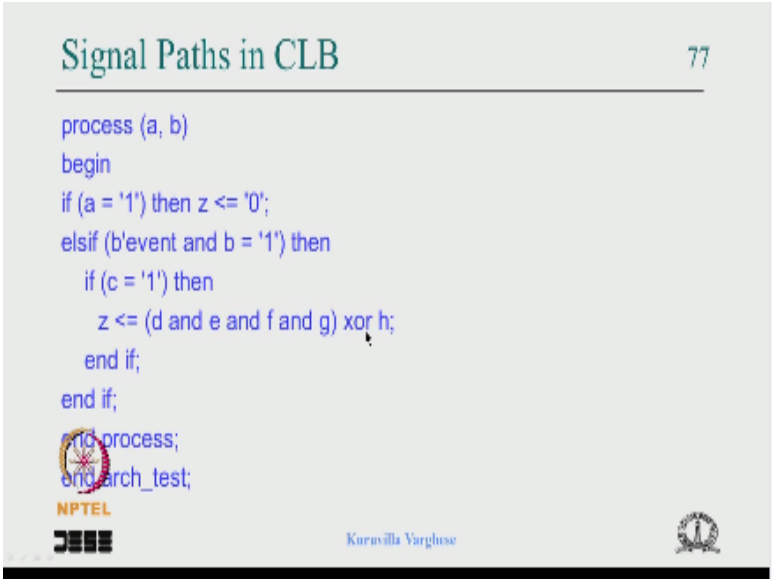
two 5 input lookup table. So here we are what we have is two 4 input look up table which can be combined into 5 input and 6 input.

But in a complex FPGA you might have multiple 6 input look up table which can be used as two 5 input look up tables like that they play with this kind of functionality will be able to understand this thing and you want to you can even I kind of a draw the synthesize circuit and you can kind of place and route manually and for your understanding to the slice of within the CLB.

What we can do is that you can write such a VHDL code using the tools can implement it you can place and round and there is a floor planning to you can go inside and can literally see that connection you can place this wires inside maybe when we discuss the tool when I show the demo of the tool which I keep it to them because other problem with that tools gets you know modified change all the GUI change, the menu change the more functionality is added.

And if I start introducing show things in the middle of the lecture middle of middle of this coach and after 1 or 2 years this course survive if found useful and then this will become that part become outdated. So I am keeping that towards a and I hope you will have the maturity to kind of you adapted that kind of thing, that we do not need to you know show it to learn make it work and things like that.

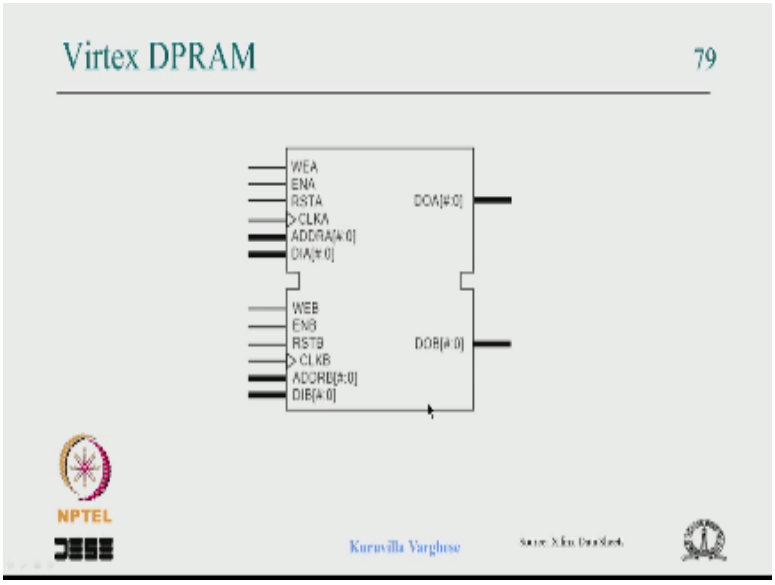
(Refer Slide Time: 52:07)



I keep that towards the end so that is how so let us come back to the slide. So that is how are given a VHDL code, how it is not here definitely I skip one step draw a diagram the

schematic the block schematic with the synthesis to synthesise you can do that because fairly simple that is why, so that is how things happened.

(Refer Slide Time: 52:28)



And Virtex has the block memory built in okay, the Virtex FPGA has dedicated memory built in which you if you want use you should write the VHDL code for it what you can do it that though the tool allows it you can instantiate the template for this using a tool code for generator and if you see the 12 code memory is quite useful because it is unlike the normal memory with single code.

It has 2 codes ok, you can see that there is 2 address line 2 data line and 2 right lines to enable the lines and all, so this is a single memory and of course the data input and data output is separate which is 2 of every memory chip because you can imagine the memory cell inside or if you want to assume the flip flops inside in each location and the data input is D of the flip flop, data output is Q of the flip flop ok.

Only thing is that there will be multiple location depending on the location you want to read there will be a Mux there, the address goes as a select line of them and appropriate locations is Mux to the output ok and similarly the input, input goes parallely to all the flip flops, but the address line will clock the correct location, so there is an address decoder which enable the clock to the input of the flip flop ok.

So in principal you can have the multiple port you know you have the same location use another Mux with the second set of this line to read a another location similarly as you

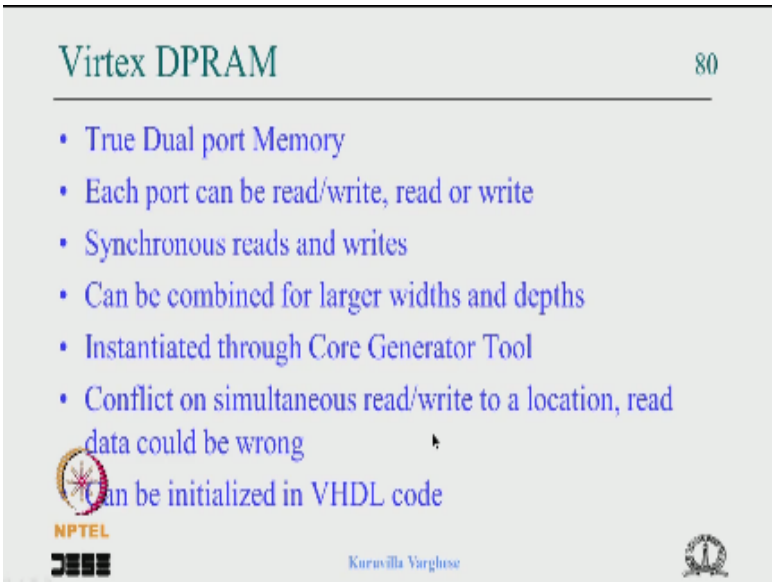
writing one location you can have an address decoder clocking location, so that is how the dual port rang the bell and I am not sure the internal decoding is not matter but it is not very difficult to image it is not very difficulty to imagine.

And in principle you built in number of ports like when have 3 port, 4 port, is does not mater but at least what is useful is a dual port because many time we can write using 1 port, and read using another port ok. So that is very useful because may be a somebody a some hardware it is computing a right thing to the memory location, at the same time the earlier output can be read and computed ok.

So you can see the computation can go and uninterrupted and if single port memory the this computation block has to write and then stop it, then the read computation though the computation here has to read using the same pot and read it okay. So there is a to put rules of the duel port will allow you to kind of write and read at the same time different location with same location of course they will be a conflict if one port is writing and 1 port is wearing the same location ok.

Then there is a problem what is maybe red may not be correct ok and you should also know that there is no great thing about input and output data line in a normal chip it is multiplex together to save the pin but it is very ideal to have separate input and separate output because you do not need Mux separate lead select and all that which how is that.

(Refer Slide Time: 56:38)



So that is how the dual port is implemented, so there is 2 dual port each port can be read or write or read or you know only read or write it also it works with the clock edge you know when a clock comes this gets latch, the address is use data is used similarly when clock comes the read happens ok, can be initiated through code generator tool the various blocks can be combined for large of width and depth.

And the same location is axis there is conflict, the rate can be wrong and the memory can be easy lies in VHDL code ok like you can instantiated and write some pattern start within the memory and so while configuring the FPGA gets return to that can be specified in the VHDL, so I think we have come to the end of the lecture what we have seen today is basically the carry chain how it implement the fast adders.

Then we have looked at how the sequential circuits the FSM get map into the basically you know to the CLB we have seen the I/O block, we have seen the hole circuit, we have seen as a little bit address bus line with some example of fitting a FSM counter and given a VHDL code how to even a trace the routing with CLB and we have seen the block memory usage ok. So we will wind up here we will look go ahead with the configuration of the FPGA in the next lecture. So I wish you all the best and thank you.