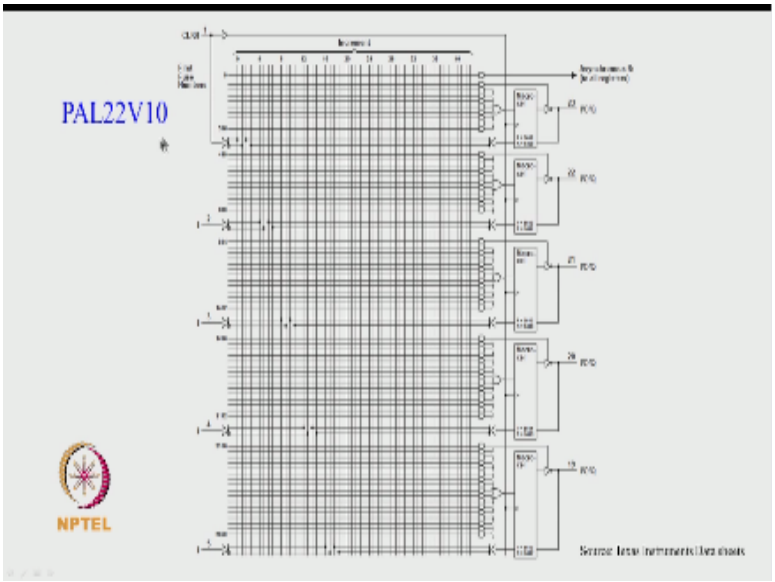
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-34
Complex PLDs

Welcome to this lecture on Programmable Logic devices in the course digital system design with PLDs and FPGAs. In the last lecture we have seen a fitting exercise in PLD22V10 but ah we have tried and constraints to see weather an 8 bit or parity generator can be fitted in the PLD. Then we have looked at the programmable technologies like EEPROM, EEPROM flash and how a wired and the wired OR is constructed out of these transistors which is very efficient.

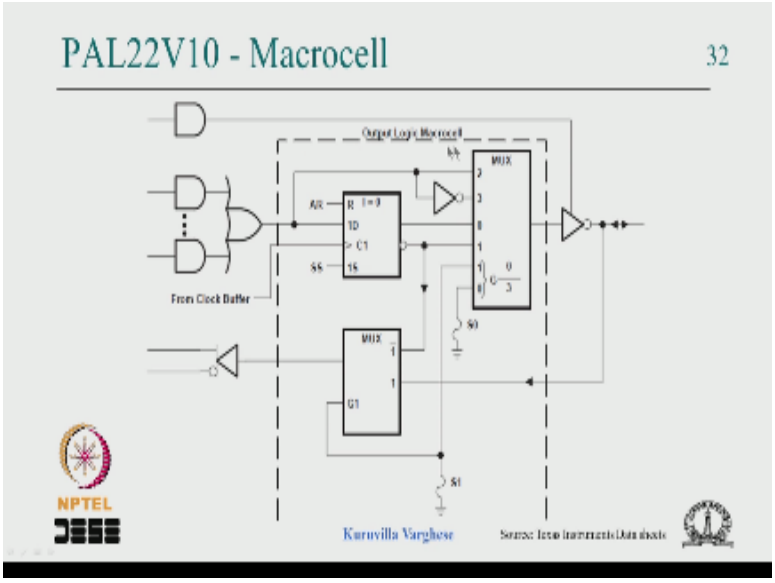
And we have an introduction to the complex PLDs which is a kind of build out of the simple PLDs ok with takes Q from the simple PLDs. So today our focus should be the complex PLD before that just ran through the last lectures material once.

(Refer Slide Time: 01:33)



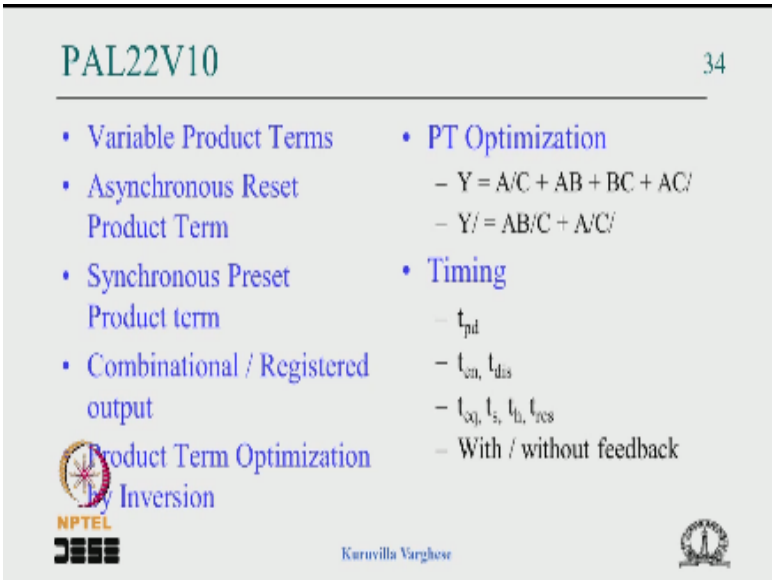
So we have basically see 22V10 structure.

(Refer Slide Time: 01:38)



The macrocell and the summary of it.

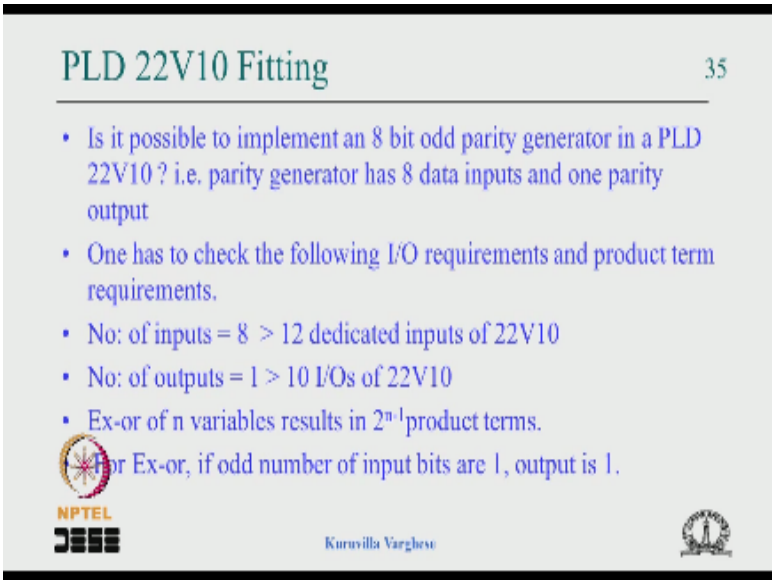
(Refer Slide Time: 01:42)



That you have variable product terms and asynchronous reset product terms, synchronous reset product terms, you can choose combination or registered output you can choose an inversion at the output of combination section or registered section to apply de-morgan's theorem and optimise the number of product terms and an example is shown here basic timing details combinational timing propagation delay.

And with respect to the flip flop and with or without feedback you know that is that means on time with feedback you know like you have extra time delay added to the array because of the input delay.

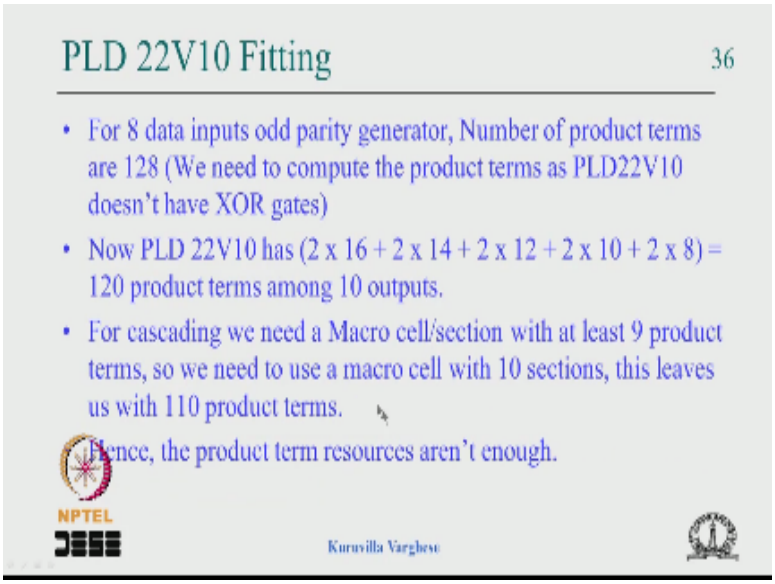
(Refer Slide Time: 02:30)



And all that ok and we have this is a question we ask 8 bit odd parity generator, so it is nothing but a 2 input exclusive OR gate if you consider like that with you know 1 output is like 2 input with and first xor gate output and third input goes to the next xor gate and so on, but main thing is that we assume that this is expanded in the product of sum in that is complete expression is converted in like A xor b xor up to 8 input is completely expanded into the product term which consists of 2 raise to 7 128 product term.

And I are requirements because there are 12 dedicated inputs, so 8 the requirement is less than that and one output but that time I used to that is also IO satisfies the resource, IO resources satisfies properly but you have to see the product term.

(Refer Slide Time: 03:43)



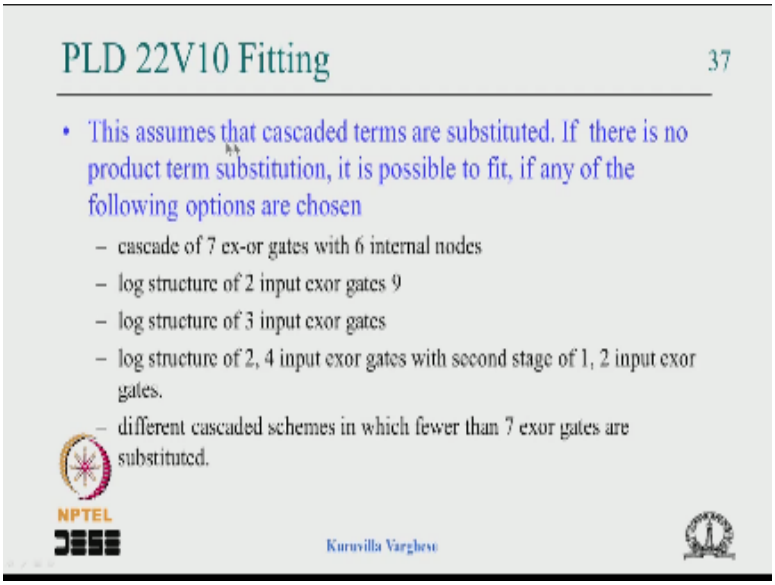
And when we can we assume that it is expanded then there are total 120 product as possible but we need to cascade, we need to combine everything together and 2 pass logic, so we need at least 9 product term to cascade but there are 10 total sections are there we take out one for cascading. So we have left with 9 section, so we need at least 9 product term to cascade. So we choose a section with maybe the 9 product term, so the closest one is 10.

One with the 10, so we have to remove that 10 from this 1:20 so we are left with 110 product term, so when you expand this XOR implementation it is not possible to fit that into 22V10 PLD. But if you like if you think of it as a cascade 2 input or XOR gate, we assume that the first 2 inputs come here and one XOR gate, that means 2 product terms are form and that xor gate output is cascaded to the to the next XOR 2 input XOR gate and so on.

You know like a chain in the each section then it is possible implement are you take care we approach to input XOR gate to input search for 2 input XOR gate which will consume all the 8 input 4 output that can be combined in the next two XOR gate and so you will get 2 output and that can be cascaded in the next section 1 XOR gate 4-1, that is also possible. So you can think of various others key maybe 3 input XOR gate, 3 input XOR gate.

Because 3 input XOR gate when expanded locally 8 product term, so 332 which can be you know given to a 3 input XOR gate and you get the output, so various possibilities are there. So if you the moment you assuming internal notes it possible to implement but if we expand it to complete product terms it is not possible to implement. So that is what I have written, maybe this was not there.

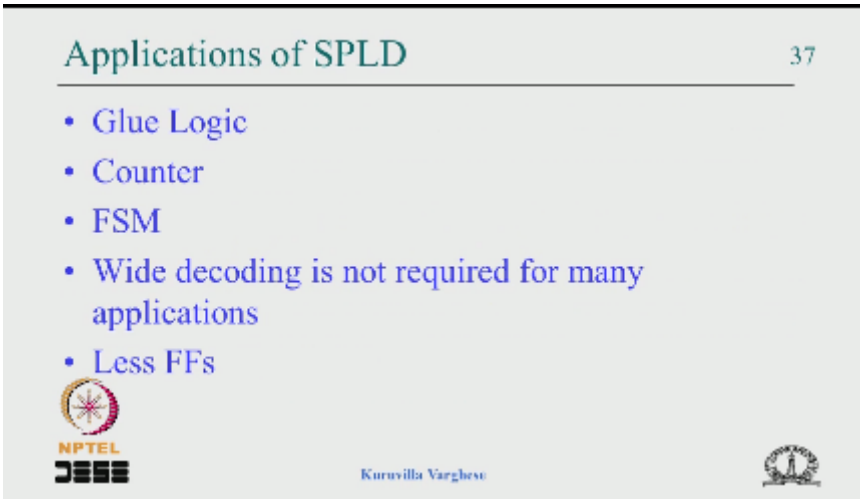
(Refer Slide Time: 06:01)



The last time I have added that you can think of cascade of 7 XOR gate is not it possible to set a log structure that is 4-1, 2 input XOR gates ok, you know this 9 and this is a mistake, log structure of 3 input cxor gates possible or log structure of 2, 4 input xor gate that means you have to 4 input xor gate but this will expand it to the 8 product terms you know 2 raise n-1 and then the output is cascade into 1,2 input xor gate.

So all these are possibilities various other stuff maybe there are other cascading scheme you can apply, so these are kind of low delay sensible structures which can be easily implemented, but a default approach of expanding into product will not fit into the PLD22V10.

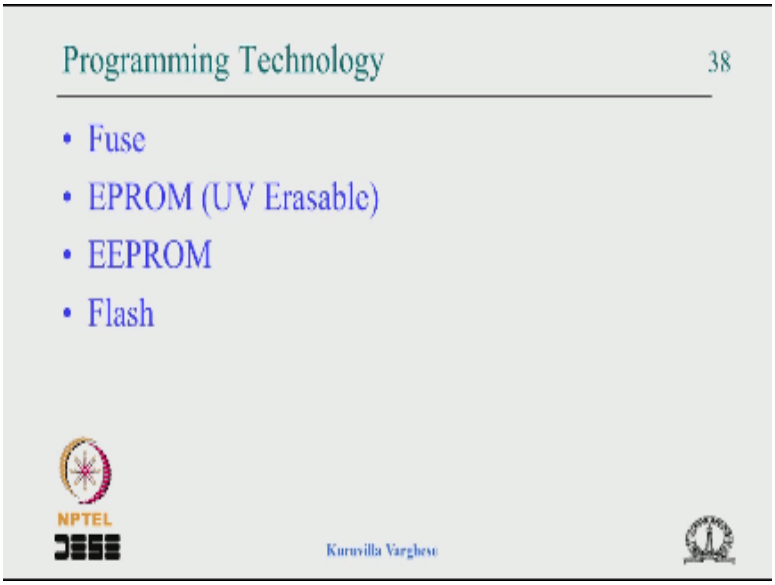
(Refer Slide Time: 07:03)



And application of SPLD we have see is true logic random logic you can implement counters finite state machine, but this wide decoding is not required for many application, 1

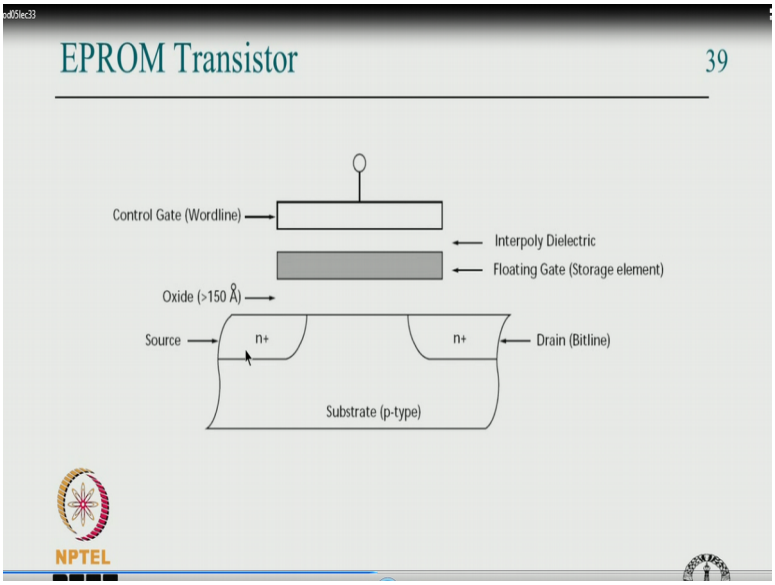
disadvantages it has hardly any number of flip flops just 10 flip flops which we can do very minimal but still it is useful when there is more of logic than the flip flops and application requires less number of flip flops and more logic than that can be used.

(Refer Slide Time: 07:39)



And we have looked at the programming technology EEPROM transistor.

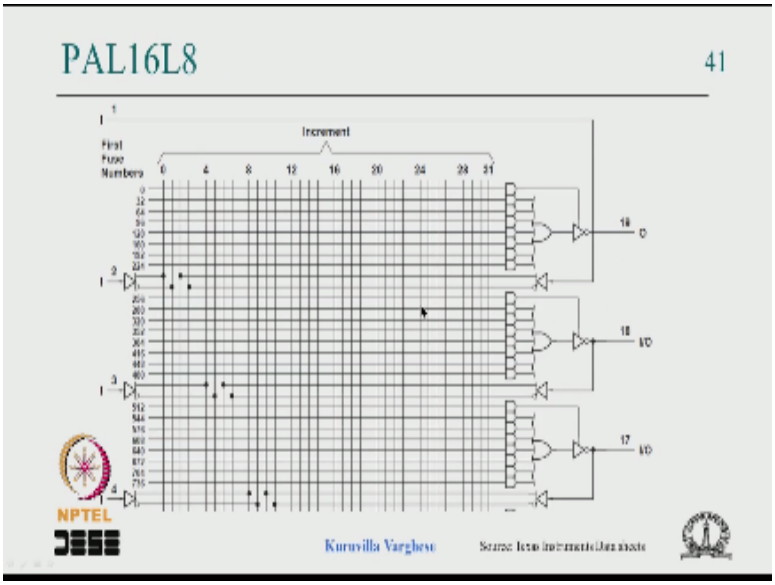
(Refer Slide Time: 07:44)



How the transfer is used as a switch in the normal case it acts as n transistor when charges attract it, it is kind of it does the gate has no control so it is switched off and rebuild the AND gate with this kind of transistor and once it is program it is permanent it is not volatile or non-volatile but if you are exposed to UV this gets out and it becomes a normal transistor, so 2 kind of remove the connection we charge here by applying a programming pulse here.

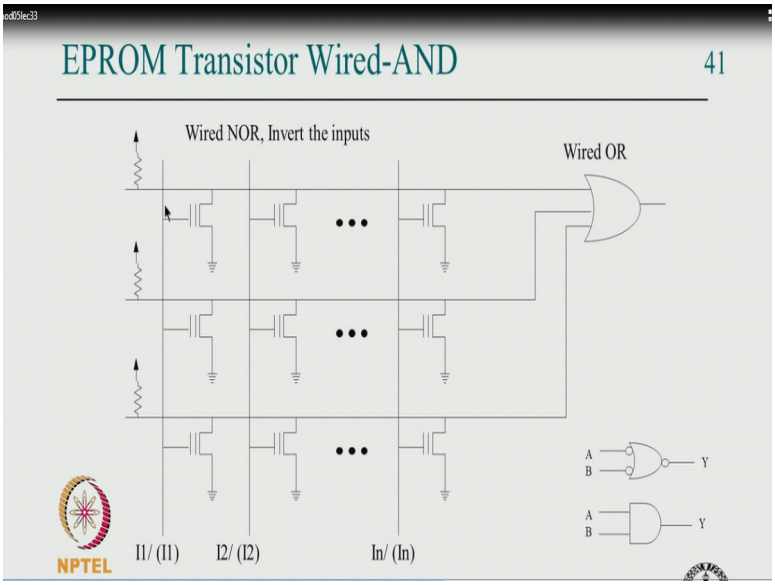
Grounding at applying the normal drain get you know trap and it is removed by exposing UV.

(Refer Slide Time: 08:27)



So we have seen how AND gate aa 32 input AND gate is formed with is transistor.

(Refer Slide Time: 08:35)



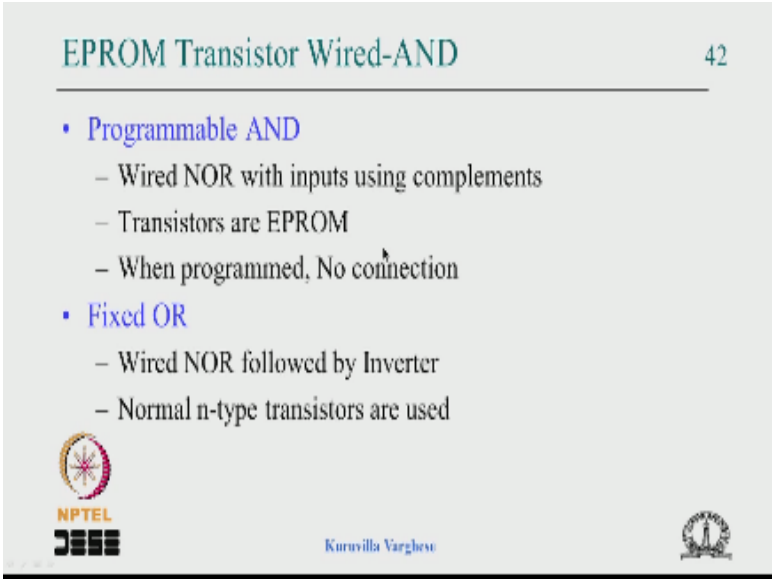
So it is simple you have one line which is pulled up this transistors are connected to the ground from the line, the gates are the one the inputs you know these inputs like a one input is 1 kind of is connected to the gate one transistor, second input is connected to second gate and so on like third fourth and so on. So when input is one this is pull low, so that means it is a wired NOR.

To make a wire NOR we have to invert the input you see because in PLD the both the input and its complement is available so wherever the I1 is implemented use I1 I1 bar where I2 was

implemented I2 bar because both compliment and the signal is available just do a swap then it works like a wired AND and then we have a wired OR structure which is again a wire NOR followed with an inverter.

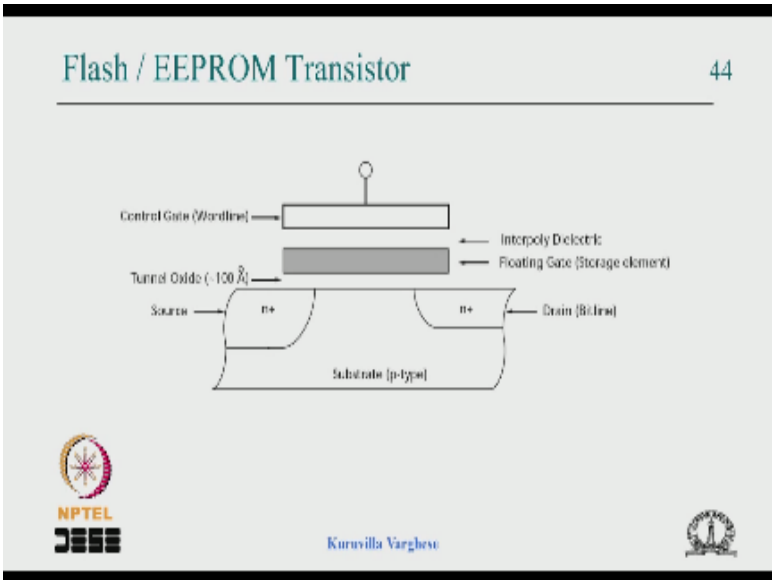
So it does not occupied too much you know does not consume too many transistors are the one thing because of this is wired structure, that is what you know the advantage of using the wired functions.

(Refer Slide Time: 09:56)



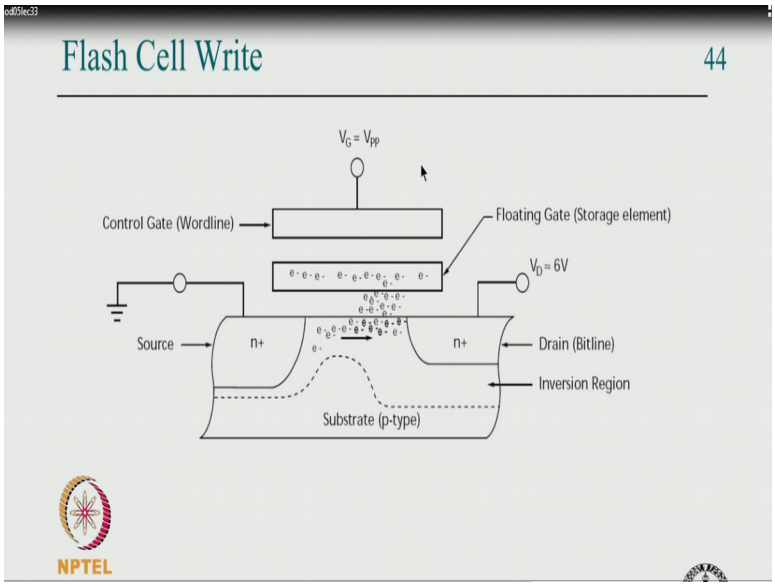
And the EPROM transistor it is similar to the EEPROM and latch is similar to EEPROM this silicon layer is thin.

(Refer Slide Time: 10:04)



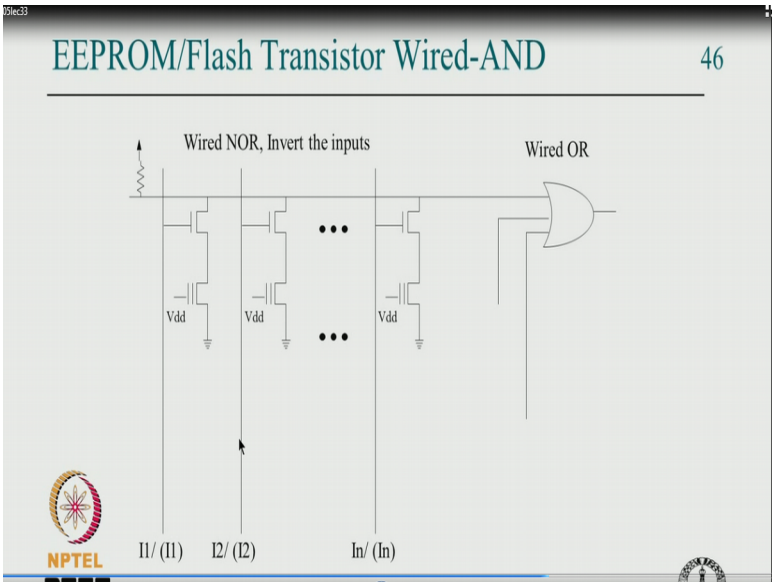
So it can be program by applying the positive voltage programming voltage on the gate.

(Refer Slide Time: 10:11)



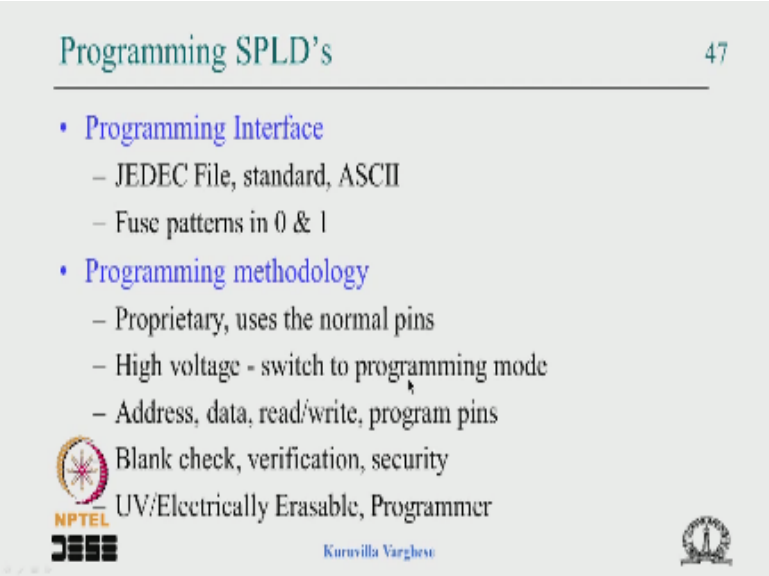
Electronic say flop can be also you erased by reversing condition, so the EEPROM get outside can be electrically program electrically erase only problem is that when it is not program this conduct normally because of its construction.

(Refer Slide Time: 10:26)



So you need to put this transistors in series with the normal entransistor and suppose you do not want I2 to kind of make a connection to this wired AND gate then you just cut it off, then it has no effect, now it is like whether it is one or 0 it makes an effect that is cut off. So that is how it is program and Y NAND and y AND was formed.

(Refer Slide Time: 10:55)

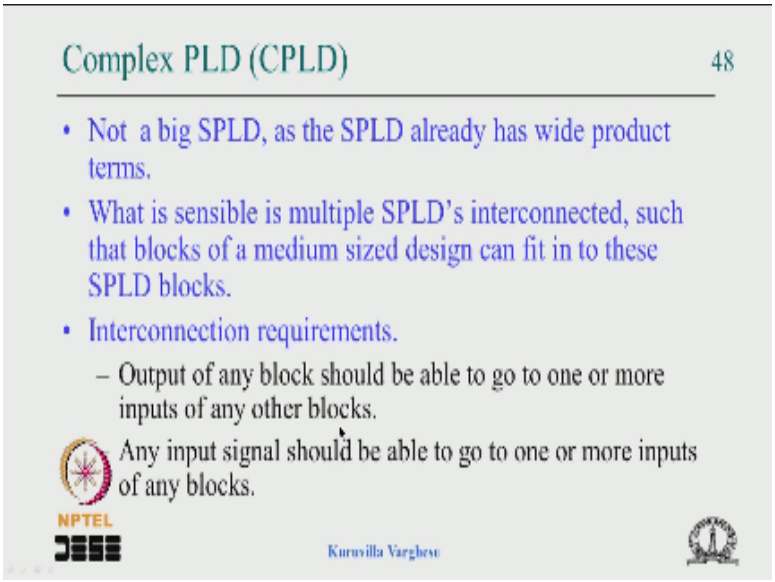


And the programming uses a standard file called ASCII file call JEDEC and the usual normal IO pins are used to program at the address pin, control pin, program pin and that so the programme will apply a programming pulse to program pin, then these functions of this program pins are exposed to the user IO pins that is program and then the normal mode this IO can be used.

That is how the SPLD program and that needs as separate program you cannot program it within the circuit ok. So is it cannot be programmed in circuit, it has to be programmed in a programmer raised in a programmer but put to the board, so essential it means that you cannot use some packages which can be directly mounted on to the PCB if it need to be reprogram.

If it is kind of one time programmable the design is frozen, the designers sure about it then it can be kind of mounted on the PCB directly.

(Refer Slide Time: 12:11)



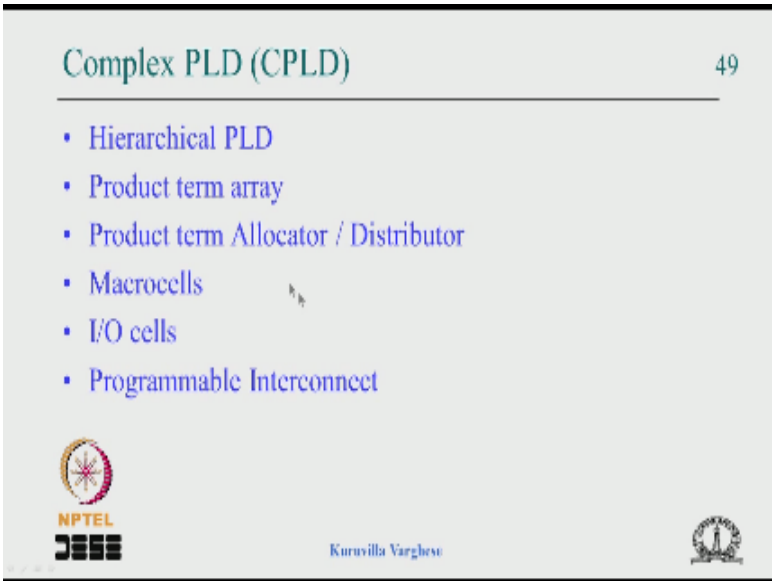
There is no problem and we look at the complex PLD and we said that there is no point in like you take a 22V10 kind of PLD and this already is quite huge you know you take there are section with 16 product comes and which many times one does not require that and each AND gate is kind of 44 input AND gates which is very wide AND gate with lot of product terms.

And now there is no point in blowing this up as I said you know there is no it does not make sense to have a 100V50 that means there are kind of say like 200 vertical lines and 50 output with huge number of product comes out what makes chances maybe replication of these kind of 22V10 structures which is interconnected ok.

So that is what a complex period is that simple PLD please add sections are interconnected , so the requirement is at output of any block like your way SPLD block out of any blog should be able to go to one or more input of other block. Suppose we have 4 blocks SPLD block one is PLD output any of the output should be able to go to any of the inputs of the other three blocks. Then only we can cascade and implement some useful function.

Similarly input signals okay, that those input signal should be able to go to one or more imports of any blocked ok.

(Refer Slide Time: 14:01)



And now so the complex PLD or CPLDs or hierarchical PLD, so that these are the simple PLD section interconnect okay. So there is a product terms array that is nothing but array of And gates not big numbers like 22V10 and where there is 8, 10, 12, 14, 16, these are small number say 5 product terms, but there are ways to expand it, so we will see that and the macrocell is nothing but a flip flop with bypass and choice of the clock lines choice of set and piece clock enable and so on.

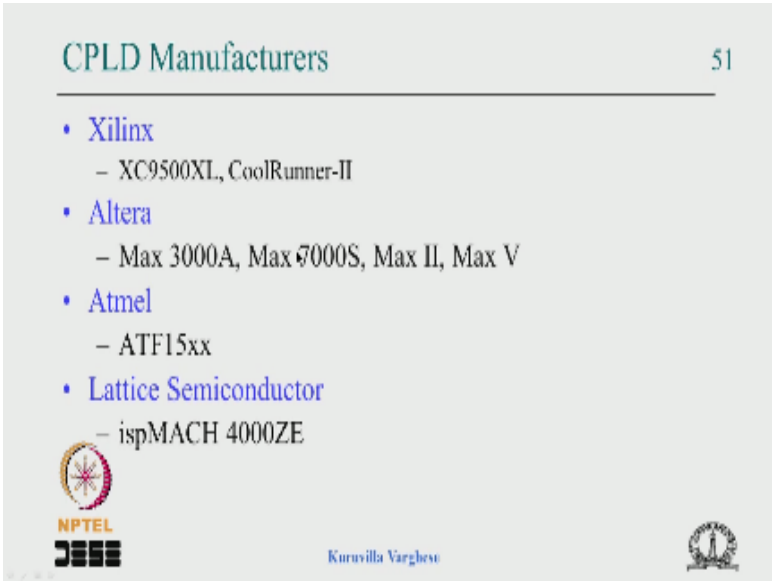
Now this the product term itself is not kind of permanently connected to an OR gate, so in between comes to switch called product term allocator or distributor which allocate its product term to the OR gate or to an XOR gate or to control a various functions of the flip flop ok. So we will see that maybe you can use a global clock into clock all the flip flops or we can use a product term as a clock ok or you want to reset the flip flop.

There is a global reset pin that can be connected to the reset of the flip flop or a product term can be connected it depends on the design maybe that in the design there is a FSM which need to reset a counter under some condition, then a Global reset helps during the power on at the power on you can bring all the flip flops required state but if FSM need to control a particular reset then you need to have the product term as reset and that is done by the product reset which is that this allocator does. This allocator allocate this product term for not only for AND gate and OR gates and XOR gate even to the control inputs of the flip flop ok.

So let us come back to the slide, then you have I/O cells basically which is this happens with tri state gate which can be used as output and input and at the end is this programmable

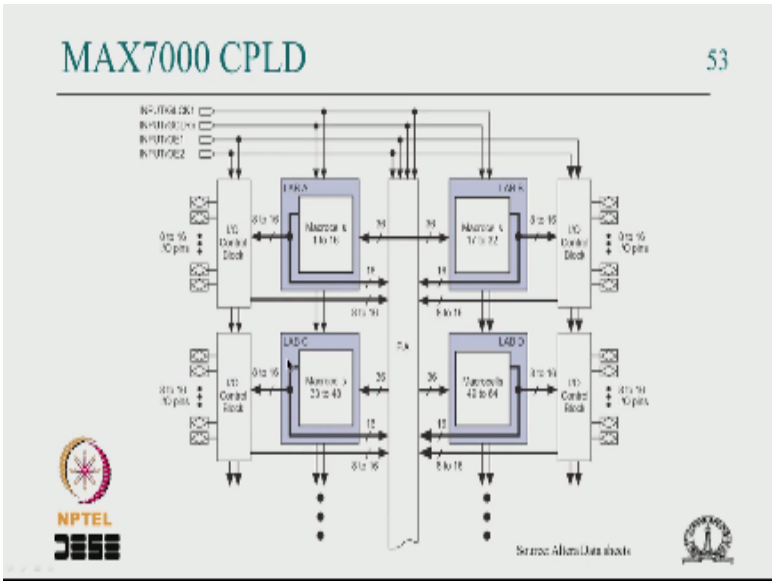
switch with the various PLD section ok. So we have various PLD section a programmable switch to interconnect them, not really the programmable section, the I/O cell also within each PLD section a product array, products allocator or distributor and macrocell ok. So let us so essentially means that 22V10 like sections are replicate ok.

(Refer Slide Time: 17:00)



So we have seen the manufacture Xilinx XC9500XL and CoolRinner, Alter Max 3000 and 7000 Max II and V. Atmel has ATF15, Lattice has ispMACH and things like that ok.

(Refer Slide Time: 17:18)



So here this is the Altera Max7000 CPLD, this is taken from the Altera data sheet I will be using the data sheet of both Altera and Xilinx because some diagram the good in Altera, some diagrams are good in Xilinx, the architecture is kind of a similar so you do not worry may be

specific the numbers input output and all may vary between the devices, but otherwise architecture is similar.,

So you will be able to **do** do you know understand by using a mix of Xilinx and Altera advice diagrams ok. This is what I call a simple PLD section, the Altera call Adlab logic array block a, b, c, d, so depending on the device that maybe two blocks, 4 block, 8 block, 16 block, 32 blocks like that. Now each block you see there are 16 macrocell that mean there are 16 sections AND/OR and flip flop ok.

But it is all fix it is not variable like 22V10 so you have 16 macrocell when you say microcell it includes everything the AND gates, the distributor, the flip flop, all the programmability everything is together is called macrocell in this data sheet, my description kind of little different I combine the flip flop and associated thing I have called macrocell.

But this is just a matter of kind of name, do not worry too much about it. So now you look at that this, you have a macrocell section the input of it so here we have 36 input comes from a huge switch ok. So you can think of it as a cross bar ok, we will see what are the cross bar, but this switch output is connected to the input of the AND gates, so that is programmable. So you have 36 means 36 inputs and its complement is available as a vertical line as in the picture ok.

So and you have an AND gate and the few AND gate go to OR gate XOR gate and flip flop and so on. So there are 16 flip flops, 16 sections, 16 output and that can be connected to 16 IO pins or it can be feedback you know it is not that it can be just taken output can be fed back to the array back, so that the idea is this how would you implement logic function output of it can go to the input of some other section where you can go here you can go here.

Some other can go here, some of it can go here from can come here and so on ok. So all that is available ok, in addition to that you can see that not only the output goes to the IO pin and it is fed back to the array you have IO pin itself acting as input and that inputs can come straight as input to the this huge switch ok. Now you see this switch if you take just assumed there are only 2 section ok to make the description simple this is not there.

This is a device with just two blocks to lab then we have you see 16+16+32+30 to 64 input to this which and 72 output 36+36 72 output from it, so when you say crossbar it means that in like all these 72 input can be any of them can be connected to this sorry this 64 can be connected to any of the day 72 maybe the first in can be connected to 10 then 15, 7 can be connected to 6 and 2 and 9 can be connect to 1 and so on.

If all possible connection supposed that is called and y it is called crossbar is that you know input come like a horizontal row and output goes off vertical row, at the intersection you can assume a switch, not possible connect possible that is white sauce for which essentially come from the telephone old telephone nomenclature it is those exchanges used to be crossbar because if there is a 10000 line exchange in principle of 5000 people can contact 5000 people ok.

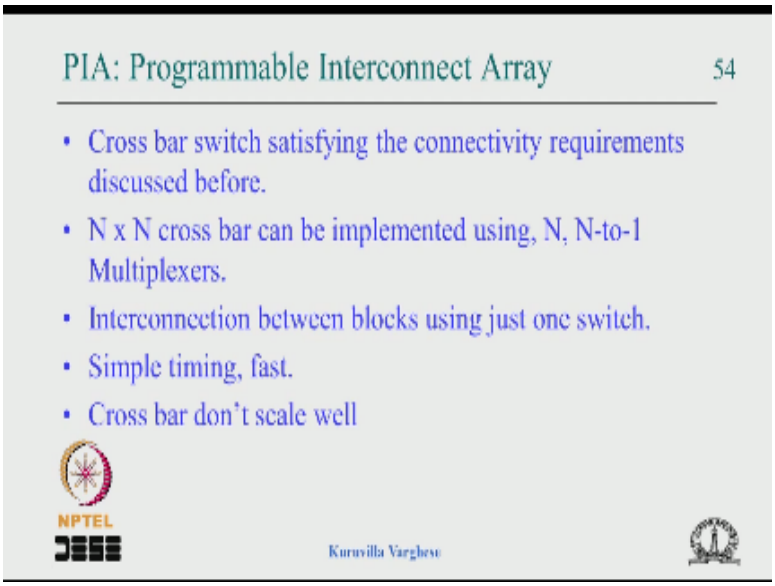
So that is how I like you is that means it is a if you imagine 10000 lien exchange then its 5000 by 5000 cross 15000 into 5000, so any 5000 can contact any other 5000 that was idea so that is the basic architecture we will see the internal detail but you see here there are additional signal one is a global clock in that could be multiple of them in this particular CPLD there is only one and that goes to this all the sections you know same thing goes to all the section.

Basically you can choose to copy or flop with this clock ok if needed is not that you are limited with it you can use a product on the pocket but you can use this also if you are not using this is a clock then it can be used as an input to the it is not wasted like as a dedicated pin, if there is no global clock in your using then it can be used as an input to the switch and be taken as input to one of the logic block ok.

Similarly there is a global clear that is a asynchronous reset it can be connected to flip flop once again if it is not being used it can be used as an input to switch ok and similarly you have output enable because there are tri state gate which can choose 2 output enabled either of them depending on your need or you can choose anomaly enabled permanently disabled so on. In case if it not being used again not always the pin that can be used as input to this crossbar switch ok.

So that you should keep in mind so this I know explain the architecture from the from the top level architecture how the logic blocks and arrange how it is interconnected how the output is able to go output of micro cell is able to go to the IO pin as well as to the input of various section. The IO pin itself the various IO pin can go to the inputs of any of the section you know that is what is shown here so let us move on.

(Refer Slide Time: 25:01)



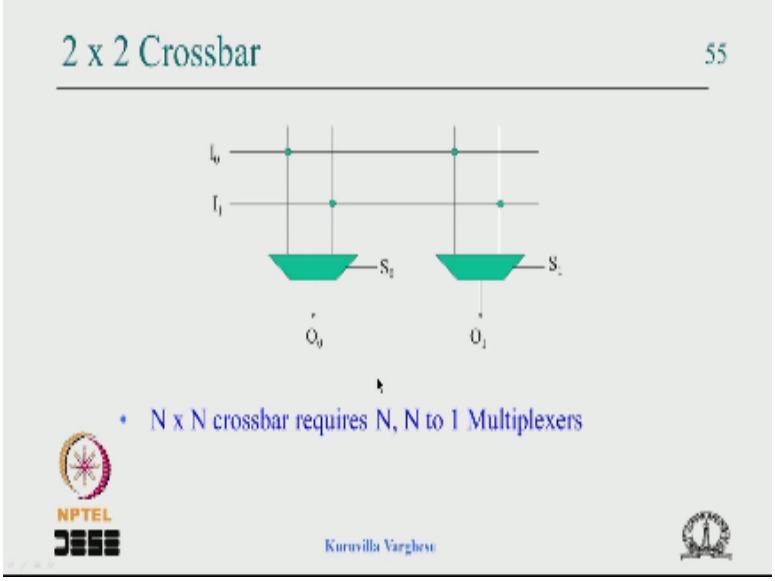
So it is a cross bar switch we have the this Programmable interconnect Array is a cross paths with satisfying the connectivity requirements before it means that any output can be connected to any of the input any IO pin can be connected to any of the inputs that is the requirement. N/N cross bar can be implemented using N/N multiplexer we will see that interconnection between blocks used just one switch ok. So it means a new take an output from a micro cell and connect to the input of a macro cell, it needs only one switch ok.

We will see that when we go to the FPGA when you interconnect some output input most of the time I get a lot of it goes through out of switches but in PLD like straight from 1 output 1 input there is just one switch Programmable switch. So the interconnection is very fast and the Timing Analysis also fast like when you want to do timing simulation the timing model of the device itself is simple.

So it is very fast the timing analysis is simple, when the problem is that this crossbar want scale well because you put this can become very huge because in one section itself it is 74 cross 32 switch and you can imagine if there are 16 sectors this switch will be very huge and so it cannot be kind of expanded to lot of macrocell you cannot say a millions of macrocell

then this switch will eat up all the space, occupy lot of area and then because of the area of the things can slow down ok.

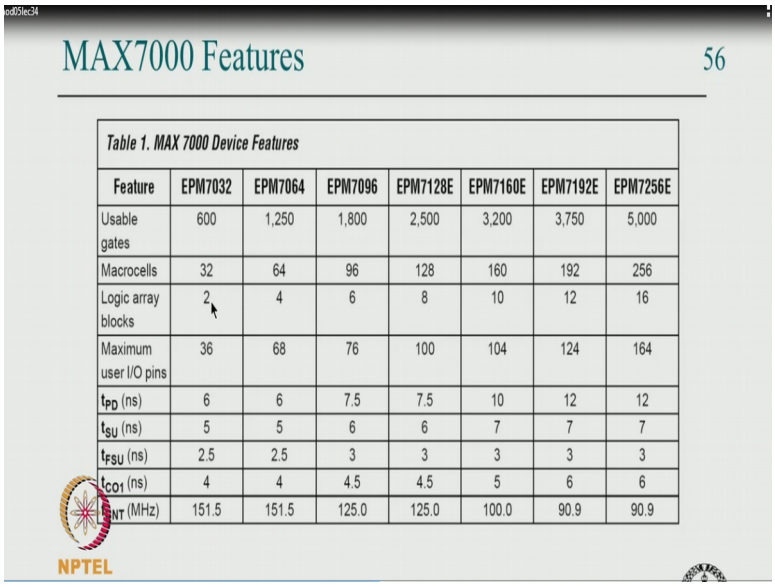
(Refer Slide Time: 26:59)



So I am showing an example of a 2 by 2 cross bar so you have to input size 0 and I would not put over and over to the requirement that any of the input should be connected to any of the output for you put a 2 to 1 marks for each output with the select line and inputs are connected to input now by selecting a so any of the input can be kind of taken to the output for both and if you want I0 to be connected to both its possible there is no problem.

There is no cleft, is not that you can connect one input 1 output you can go to multiple output. So if you have a you can imagine if there is a 16/16 in cross paths then you will have to ach output there are 16 to 1 mux connected and set 16 or so that is how N/N crossbar require N, N to 1 multiplexer. So you can imagine if you have huge switches to verified multiplexers are required and that can you keep a lot of area and the delay can be quite high in that case ok.

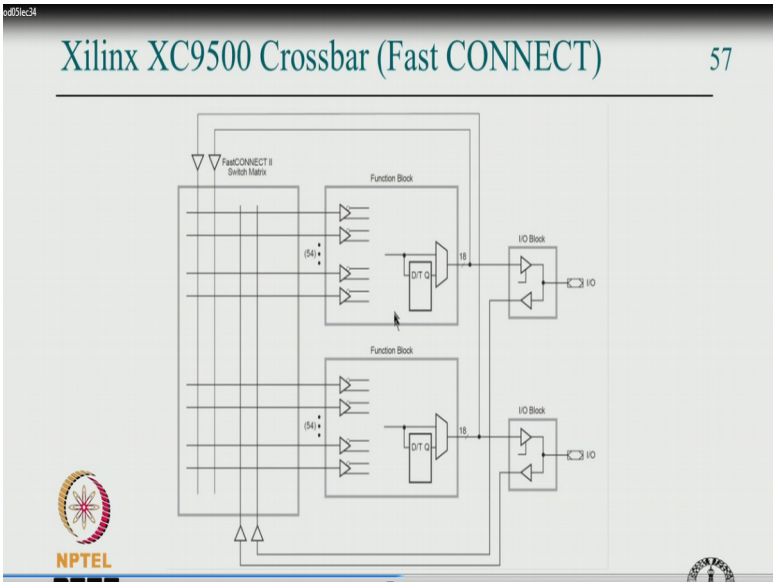
(Refer Slide Time: 28:15)



And these are the max 7000 devices you have EPM7032 it means that 32 macrocell, so basically each macrocell has 16 inch logic array as 16 macrocell, so 32 microcell you need to offset section for 64 you need 16 into 4 of them, so that is what is shown here 7064 means 4 logic blocks and it has 68 pin, so you can imagine this as I open the clock output enable and BCC crown and things like that.

So that is how it comes, so the highest one is there are 16 logic block which switches to 2 fixed macrocell is quite a you know bit device with lot of macrocell 256 macrocell is quite high and you can implement many medium complexity design and specifically.

(Refer Slide Time: 29:26)



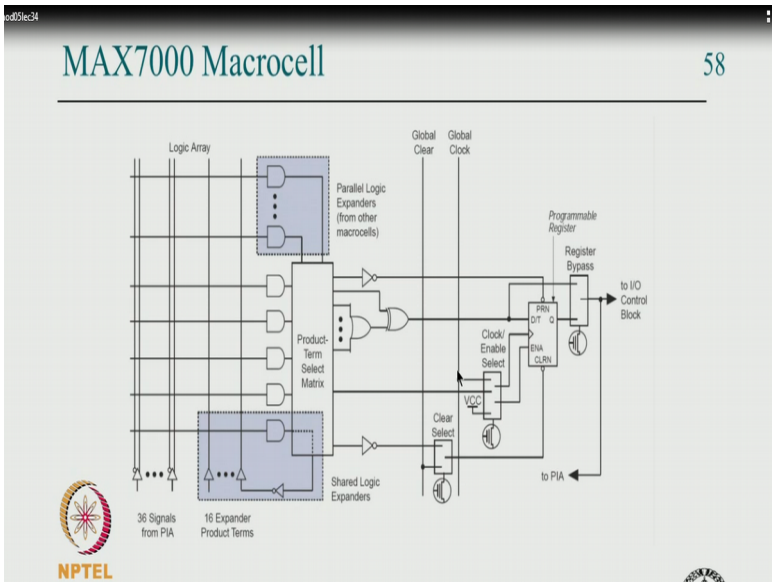
So let us move on this is the crossbar of the Xilinx XC9500 is not the crossbar of Max7000 it is Xilinx XC9500 criss bar just wanted to show the picture and that was clear data sheet, so

you have any IO pin there is an array like which is here and distributed flip flop and the bypass and such 16 output for coming to the 16 pin and then we can see that is also fed back to the crossbar switch to be kind of you know can take that into the various logic.

And similarly the input is also coming as input to the cross path switch which can be taken in inside the logic block. So that is that is out the crossbar is this the outputs of the macrocell as well as input IO pin input is going as vertical line and the AND gates can take you know choose any of them by programming the crossbar okay, so depending on how you design how you code these switch states we decided ok.

So basically you are VHDL code will be converted into this kind of what switch to program at the end ok.

(Refer Slide Time: 30:59)



So that is a basic game so this is the real microcell ok or other the logic part of the one section so you have one IO pin which is with tri state gate and all that which is not shown here, so what is the one section circuit shown here. So basically have to look at it 5 AND gates ok and you see one of the AND gate is fed back into the array ok. So this is kind of cascade it can be cascaded to other AND gates.

And there are 16 sections like that, so there are 16 AND gates from each section can be used by any of the section, it is not that this AND gate from the section can be used only by this particular section it can be used by other sections of macrocell services and you can see this is the input signal from the crossbar of a PIA Programmable interconnect, so there are 36

signal coming is taken for 72 vertical lines which can be program to the AND gate to form the product term whatever required.

So basically 5 AND gate, one AND gate is an expanded product which is feedback to the array, so there are 16 section so there are 16 expand product terms and now this is the normal OR gate which AND gate output goes but mind you this not permanently connected there is an switch here which is called product tern alligator or distributed or Xilinx call its product on select matrix.

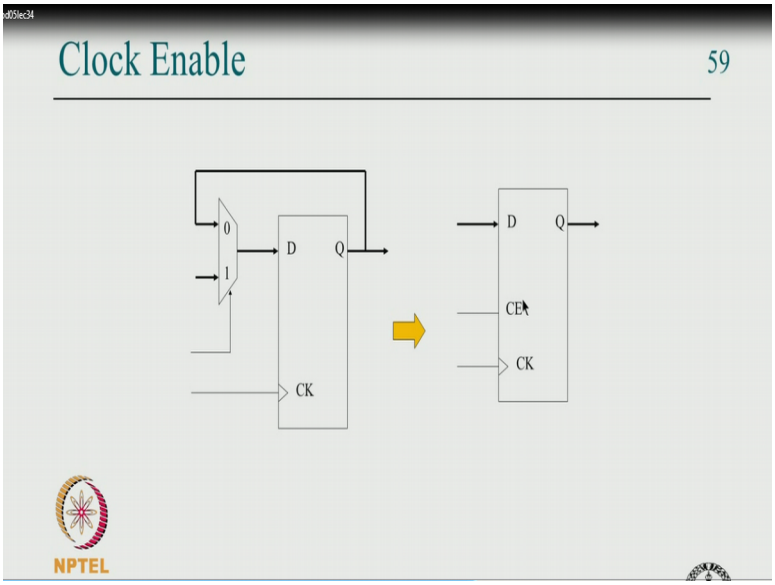
And that allocates the AND gates to the OR gate or XOR gate or various other purpose it we will see that. So this is a kind of a switch with you know you can choose an AND gate whether it should be connected to OR or XOR or this invertor and so on okay. Now let us look at the flip flop you see that the OR gate output is going to COR gate output input an XOR gate is output is coming to the flip flop input.

And the flip flop Q is going to a bypass Mux, so we are there is a 2 to 1 Mux with get the output of the flip flop as well as input of the flip flop, so if you choose this part then you are not registering the output your taking it directly. So if you want implements and combination circuit without register then you can use this path the tool will choose this path and you can see that this flip flop can be used as a d flop or a t flip flop.

So if you have counter application and program it is it a t flip flop it has a asynchronous reset which you see can be controlled by a product that is how this comes suppose you want to reset it using this product it is possible there is a connection with switch does and then you can reset that similar there is a asynchronously up in which now or still no one is there is a global clear pin for the PLD chip you can use it clear along with all the all other flip flops or you want to very and some condition you want to clear it then you can use this line.

And the product term can allocate an AND gate to the so depending on some condition you can reset it ok. And what we are left with this clock and the clock and above is nothing but a re-circulating Mux just select liners enable. So maybe I can show that we have learnt that when you want to control a register what you can do is are you can put a 2 to 1 mux and if that condition is satisfied this input goes there.

(Refer Slide Time: 35:22)



And otherwise it retains it now when the max 7000 flip flop as a inbuilt re-circulating marks and the select line of the max score talk anymore so you want to kind of such a structure just connect the select line here now the control signal here you do not need to do anything and automatic. So that is what is this clock enablers about enable A and now you can have permanently enable it or you can control with a product ok.

So it is quiet versatile you know you have 5 product one of it is a next 5 product term and this can be distributed to OR gate XOR gate set clear the clock enable also very flexible the device is very flexible and you can see there is a fuse which is a Programmable register that means you can choose depending on the value program here will become a D flip flop or a t flip flop. So you can imagine a simple circuit where it can buy.

Now so you switch between d and t maybe you can think of such a circuit very simple. Now not only that now know you see here this one issue you are just have 4 product has anyone is common so your product term or combined with this 5 or some common things but many times you want implement say 10 product term then what you can do is that suppose the previous macrocell has unused product term.

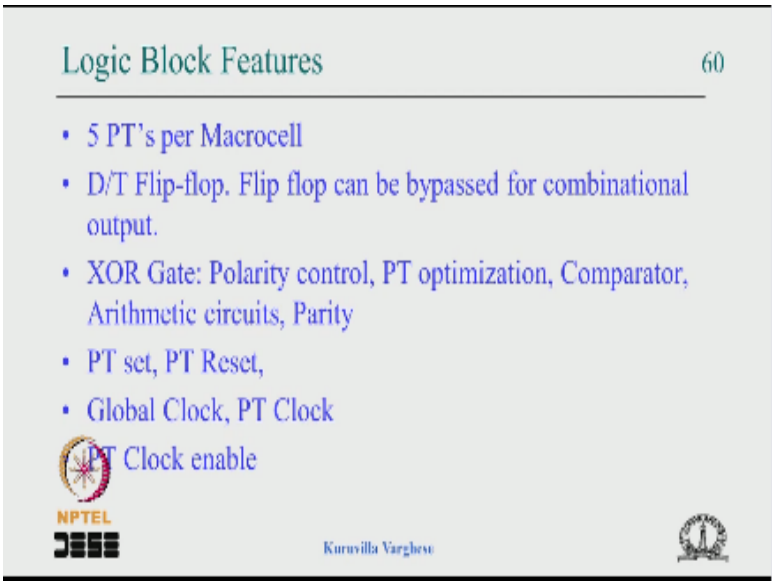
Say for some reason support the previous macrocell used only the flip flop, then what can be done is that that are unused product term can be stay here to the input of this OR gate. So basically it cascade the unused AND gate from the previous adjust section to the OR gate with a great advantage because if you are if you need more than 4 product and 5 product then

if there are some products which is left unused that can be kind of stay here to this OR gate through a cascade of steer.

You know that it goes all the way like if the previous section is not use your it can be used in the next section and so on and if can go in some device only in one direction in some advice we can go in both directions and so on ok. So I think you get up good idea and or and this XOR gate now can get the input from one of the AND gate, so XOR gate can be used for you know the product optimisation as I said you can apply De Morgan theorem.

And instead of y you have y here you can implement y bar inverted and take the y there, if the number of product are less it can use for Programmable polarity and it is one it is inverted to 0, it is not inverted, it can be used for arithmetic circuits like half or full adder or for parity all that in all kinds of you can make use of this XOR gate. So that is pretty much the architecture of the macrocell.

(Refer Slide Time: 38:56)



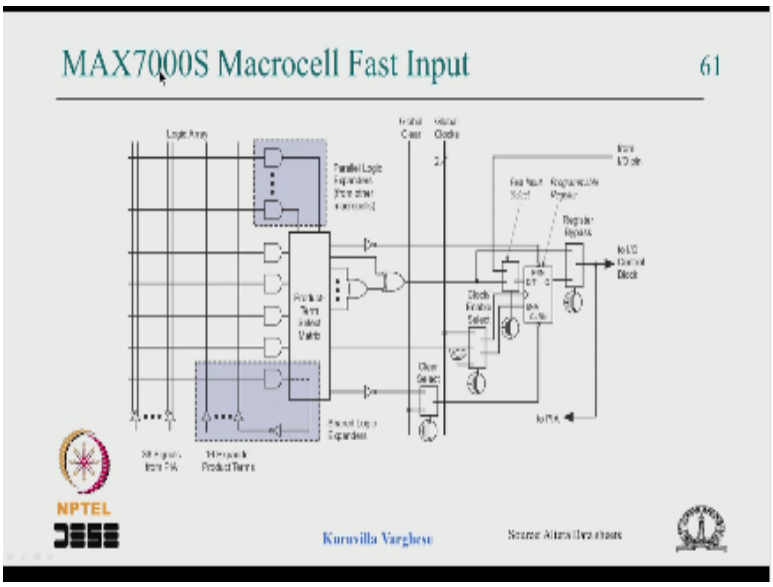
And so the summary of that is that you have 5 product term, 4 macrocell you can choose between d or t flip flop, so flop can be bypassed for combination output XOR gate for priority PT optimisation, comparator, mathematics circuit, parity you have program product term set, product tern reset of global reset, global clock of product term clock product term clock enable all that is possible in this architecture.

Now mind you in this device support you had say an input synchronisation as we have seen how a single stage synchronisation can be done. So there is a signal which has taken do a flip

flop to synchronise. So in this Max 7000 CPLD you want to synchronise it has to suffer an IO delay, it has come to the switch and switch delay and it does not get into AND gate.

Then the OR gate then it has OR gate then it has OR one of the AND gate output can be connected to the flip flop. So it before it getting the flip flop suppose lot of delay, that will be good for input synchronization directly one of the input can be taken into the flip flop. So such a thing is possible in Max 7000S.

(Refer Slide Time: 40:25)

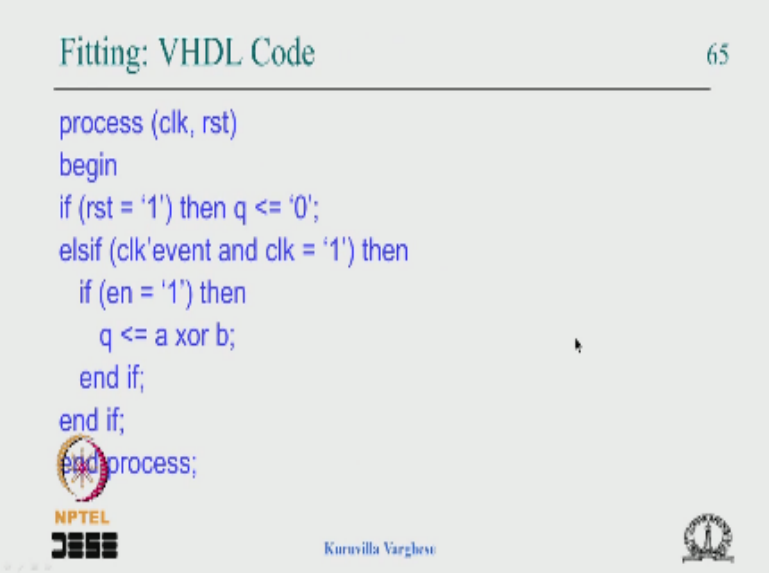


They have apart from all this connection from the IO pin does have direct connection to the flip flop which allows low setup time. Otherwise you will not taken through the crossbar have to set it up before the clock but with this set can be set up very fast in a little time before the clock at that can be set up that is advantage of this and that is IO pin that is the direct connection to the mux before the d of the flip flop.

So you get output XOR gate or from IO pin directly for the first input and that is re-shown like here we can see that the max 7000 CPLD is like this Max7000 S little different everything is same you have input output feedback IO pin but you can see that from the IO pin there is a direct connection to the flip flop inside ok.

So here, here, here like that. So it has some timing you have learnt it. So if you choose max 7000S it has some timing, but it has instead of 1 clock it has 2clock, it has a output enable which is little more elaborate and so on, that is about Max7000S.

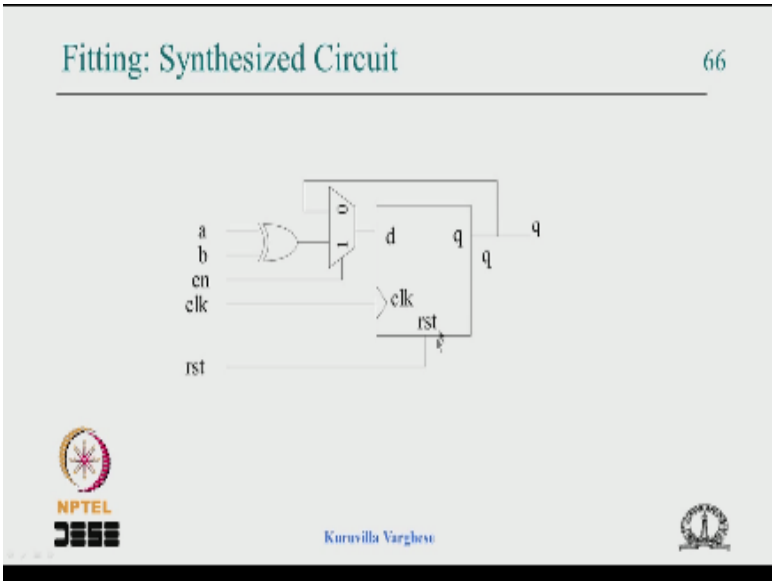
(Refer Slide Time: 41:52)



Now let us take an example we want to see like so we are showing a code how this code get translated into that CPLD architecture. So look at this which say process clock reset begin I hope you remember to VHDL will be able to discuss. So each reset is 1 then q0 else if clock even clock is equal to 1 then under that if unable is 110 Q get a xor only ok. So naturally and upon decision a flip flop with asynchronous reset.

And a clock and this enable naturally we know that this a xor b should be connected to the d input of the flip flop because it is synchronous up on the clock if you have to get that it has to be connected to the d, but there is a control if enable is 1 then only this should happen. Otherwise the Q will be the previous Q ok. That is the meaning of end if there is no else and this shows memory ok.

(Refer Slide Time: 43:08)

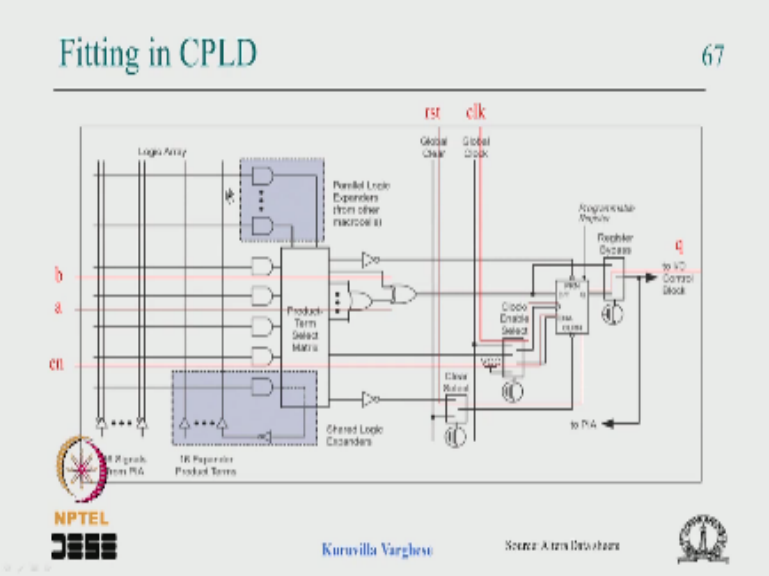


So that is implemented easily through a re-circulating mux, so the synthesizer circuit look like this you have a flip flop, the clock goes to the clock, reset goes to the reset and the Q is coming back to a 2 to 1 mux and enable select line of the mux is connected to the enable. So when a enable is 1 this a xor b goes to the d. Otherwise it retains and if the reset comes it becomes 0 ok.

So that is how it is basically implemented now if you look back at the architecture you can see that we need an XOR gate, so XOR gate can be implemented here, so taking 2 inputs here a and b and a can be connected here, and b can be connected through the OR gate here ok and enable signal has been implemented that can be connected to re-circulating mux built in.

That can be connected to another and gate, so that we have the clock and the reset an all that. So that is what I am going to show that this is the code upon the reset q0 up on the clock is enabled 1, then Q is a xor b and that is a circuit a flop is there re-circulating mux select line is enabled and to the 2 to 1 mux you have a xor b and naturally this is built into the to the flip flop.

(Refer Slide Time: 44:51)



So we connect the a xor b the output to the d directly ok. So look at this diagram so we have the flip flop, we have a reset, global reset. So that is connected through this switch to the clear, so that it is when it is reset asynchronously this cleared. We have only one clock, so we cannot seem it global clock. So this which is program for the clock and you have a and b pin which I am not showing which is going to the crossbar switch and the cross power output is coming here to say a is coming to this AND gate and these coming to this AND gate.

And this particular AND gate goes to the one of the OR gate input and go to XOR gate and d directly goes to this AND gate and go to the XOR gate here and we are enable signal which again comes through the IO pin and the crossbar switch and come directly through this switch to the enable because that is re-circulating mux built in and q is taken directly to the pin.

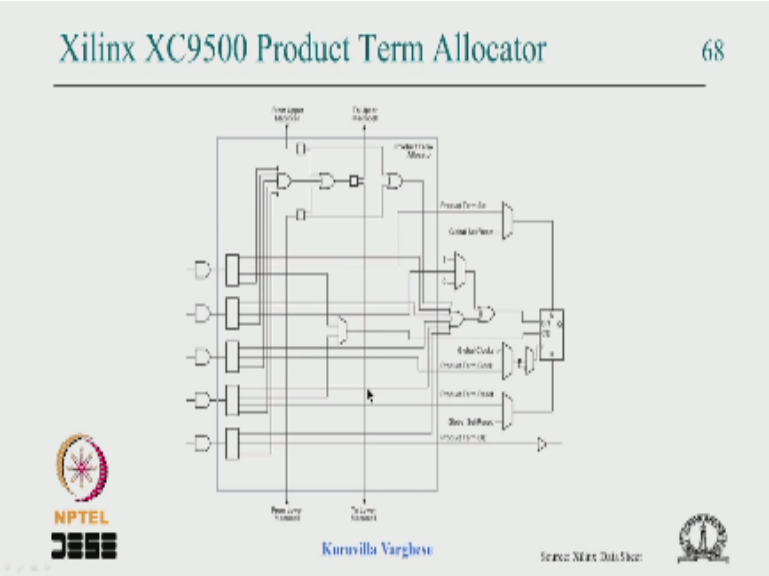
So if you write a code like that then this is all that the macrocell gets program. In addition to that you know that there will be a and b pins connected here and we should be taken through this the crossbar into this XOR gate ok and similarly enable, so that you have to remember there is more programming within the crossbar switch. So that gives you an idea can be easily what doubt know given a code.

We have seen have given a code you can in for synthesize circuit and we have learned enough about architecture of PLD and CPLD and you can work out the connections and he can you an estimate how many water resources are used within a CPLD. So that the power of it by understanding the low level details and you know just writing the code and hoping that everything works properly.

So you are able to suppose I give you a counter and if you work out the equation and you will be able to come and you know in for the circuit out of that VHDL code then in for the connection in CPLD and you can see how many set how many pins of the CPLDs use, how many macrocell, how many product terms to that detail you can go and some of the tools allow you to see this detail within the device.

And you can view that and then verify that it is it is right you know that what kind of estimated is right ok. So this shows let us move to this kind of this product on select matrix I am going to this is Altera data sheet.

(Refer Slide Time: 47:56)



For that I am going to the Xilinx XC9500 product term allocator which may be little different from the Altera because there may not be an expanded product term but nevertheless it shows the idea of the product term allocator, so you can see there are 5 AND gates and there OR gates, so you can see that this AND gate output one of the like there is a d mux 1 input is connected to this AND gate this OR gate.

And also can be connected, so all the 5 are connected to 5 input OR gate, but not that you see that is not using this product time for this OR gate it can be used for setting it okay can also use a global sector similarly you take this product term it can be used as this can be used with the OR gate or you know to as an input of XOR gate. Now XOR gate itself can be one of the simple XOR gate can come from a product term.

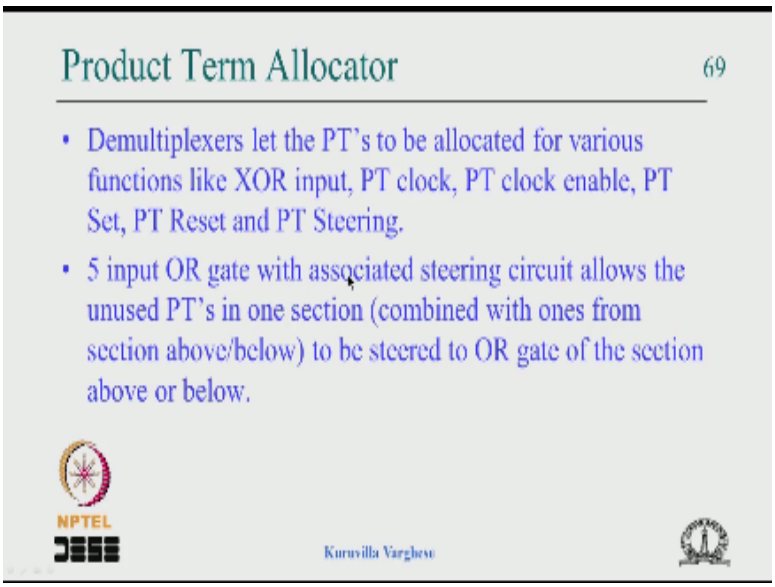
It can be permanently one by you know kind of control inverting the signal or 0 by letting it through a similarly this one can go to OR gate to clock edge or and this one can you know go to the OR gate it can go small enable it can go as product term reset and so on ok and this can go as a output enable. So the design is architecture is a little different between the mux7000XC9500.

But this was the steering product and steering all these five products are old and given to this fact which can be taken to the upper macrocell and lower macrocell and we can see that what happens in that upper macrocell, that means that there is a section there with 5 product term and there is an OR gate XOR gate that is kind of diverted here and now you see you can take that and put this is connected to this OR gate.

So you can cascade the previous AND gate through an OR gate write down, so if more than one product term and this goes through so it can be like you can you can see that it gets its gets combined there. So you can so if you look at this section you see this 5 products can be combined with the previous one the top one and bottom one, so you have 15 product terms that can go up or down which one it can be taken to the OR gate of that section.

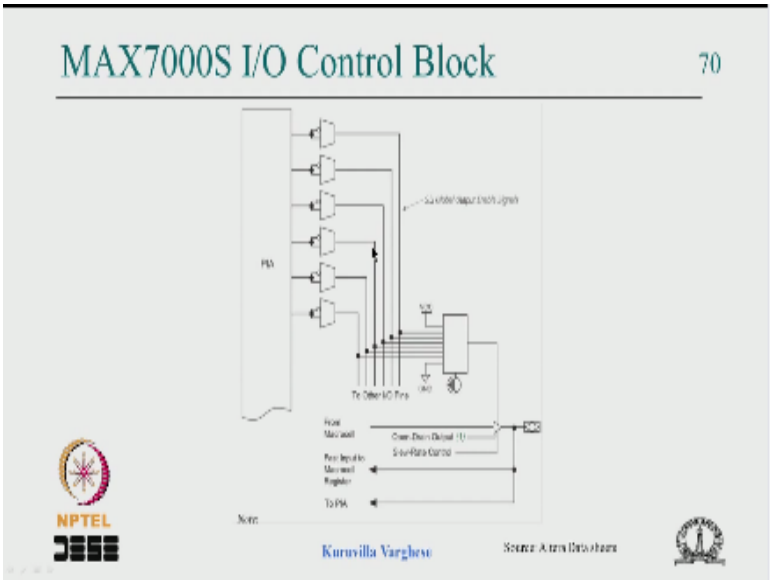
So that is what is this product on steering is what we're calling for you can if you are left with if you need more than 5 product that is possible for steering.

(Refer Slide Time: 51:07)



So basically these are de-multiplexers and output of products term which then can be steer to the XOR input product term clock enable set reset steering and all that basically steer you know that steer from the upper to lower combined various action and use it and all that shit quite useful.

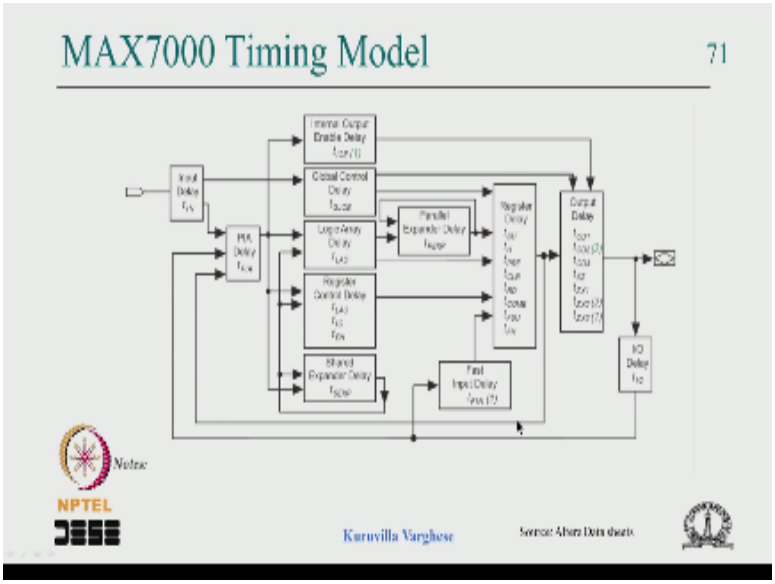
(Refer Slide Time: 51:28)



And if you look at the Max7000S I/O control will be a tri state gate as IO pin and this output if it is enable input it is disabled, so there is a mux by which we can permanently enable, permanently disable or you can choose the various output enable from this future because we have seen the output enable multiple output enable pin switch is coming to the to the product this crossbar.

So either we can come through the crossbar to this OR some product term can control the I/O status of particular product which is in control a particular know that adds to the flexibility.

(Refer Slide Time: 52:26)

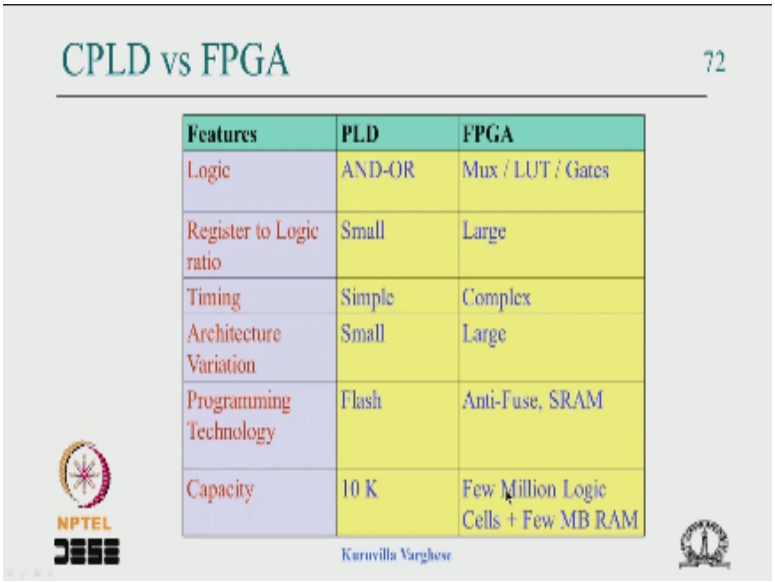


So this is the model of the CPLD Max7000 so it capture all the delays, supposed I/O signal which is coming to an input pin so there is a delayed incur goes to the crossbar in cause a delay then it goes to the logic array so it is an and gate delays incur and if there is an

expansion by parallel cascading is added benefit goes to a flop set up time is added and if it goes to the tri state gate there is a delay.

So similar there is an input there is an I/O delay then again it period for fast input it directly come that much delay and come to the flip flop. So the timing is timing model is very simple and the tool basically to use this journal to estimate the time but you yourself can put this picture and estimate the timing manually without even writing code in that is possible to do that if you have the idea of the circuit you want implement basically the timing model of the PLD is very simple.

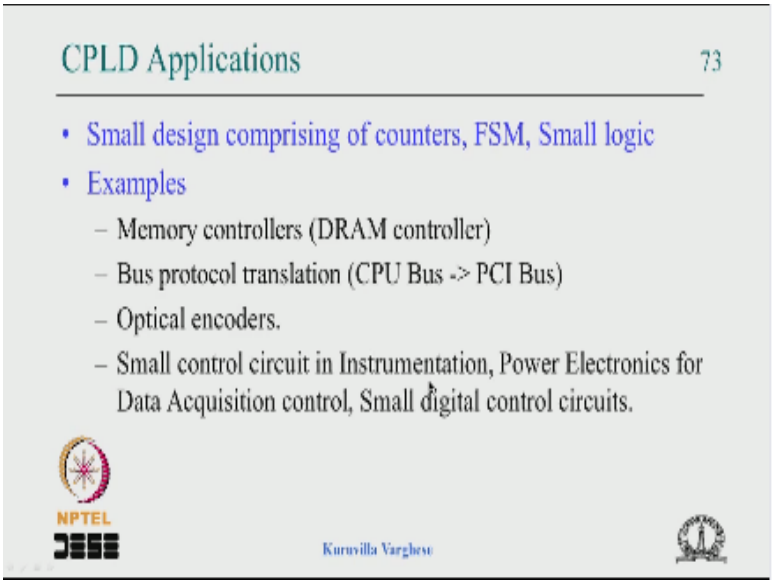
(Refer Slide Time: 53:36)



That what I want to convey so when we compare CPLD versus FPGAs, so do we have not seen that FPGA basically the PLD still using handover structure to level number of resistors are very small with not case with FPGA, timing is simple PLD which is mill complex in FPGA station that means closer manufacturers they have similar architecture and programming technology is flash and the capacity is kind of 10K.

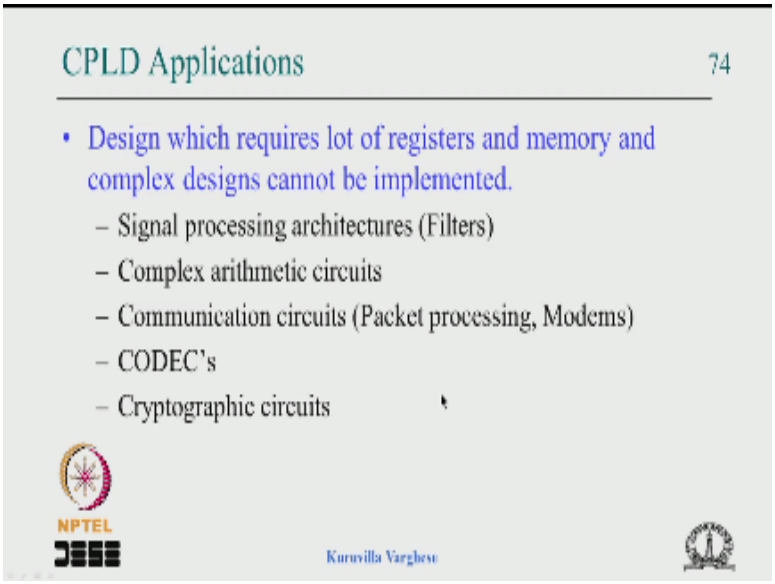
Maybe we will look at the FPGA part after going to the FPGA, but the basic thing is that it is non-volatile up too kind of medium complexity low complexity can be implemented and it is very fast you get recently fast implementation and number of registers of small. So you cannot think of the FIFO and memory is and things like that is very difficult implement.

(Refer Slide Time: 54:36)



So are the applications that very small design comprising account finite state machine small logic all those are the application may be can build a memory controller Ram control protocol translation some optical encoder small control circuit for instrumentation power electronics data acquisition and things like that. So small and medium kind of application cannot you can use this CPLD.

(Refer Slide Time: 55:10)



But designs which look like lot of registers and memory cannot be implemented in this like Signal Processing like it filters, complex arithmetic circuit packet modems codex like the cryptographic, so all that be difficult implemented in CPLD. So that the limitation of the CPLD. So I think we have come to the end of the CPLD what we have seen today is basically the CPLD architecture.

How it is a hierarchical PLD is formed out of simple PLD section for their logic blocks, a cross for which connects which allows the output to go to the other inputs IO pin to go to any of the input and then there is a product products which are allocated for various things flop with clock enable the re-circulating Mac fix, you have the set reset which can come from global your product term.

The clock and the Global Clock to the product time clock enablers it can be global or the product and there is in some cases there is a fast input and we have seen the crossbar is nothing but multiplexers and output for NXN means N to N multiplexer product term allocate is nothing but the d maximum, so that it can be connected to various other inputs like or xor and flip flop controls and things like that.

And then additional OR gates which can be kind of used to steer this unused product term the next section and change it you know like that to built quite a bit of product term and it varies from device to device how it can combine can refer to the data sheet and we have seen timing model which is very simple and we have seen the application you can use kind of best product or translate be Ram controllers.

Small electronics instrumentation, data courses in control circuit can be implemented using CPLD. CPLD is also ideal for the counter for some small randomly all that but it cannot implement lot of memory Complex arithmetic and so communication Signal Processing Computer Architecture application out of it networking all that will be out of it and CPLD has used as advantages for we should focus on the advantages and use wherever it is required.

So I wind up here the CPLS part in the next lecture we will move to the FPGAs and please revise it try to understand that there will be very useful and I wish you all the best and thank you.