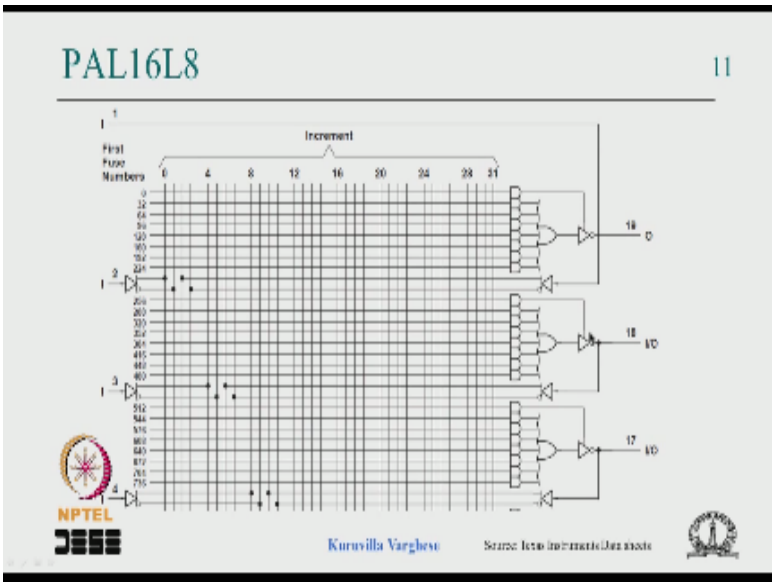
Digital Systems Design with PLDs and FPGAs
Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-33
Simple PLDs:Fitting

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 the simple PLDs which has registers which allows one to implement data path and sequential circuits. Also we have seen some kind of fitting exercises because not all resources are there, how some techniques are used to fit the design optimally in those structures.

And how like a what is available today ahead, versatile simple PLD like 22V10 we have seen that in detail, that as I said mostly we have seen this because when we go to CPLD without much explanation because we have learned all the basics and it is very easy to understand them looking at the evolution of this simple PLDs to CPLD that was idea. So before getting onto the CPLDs and maybe some slight exercises. We will look at the last lecture slides quickly.

(Refer Slide Time: 01:42)

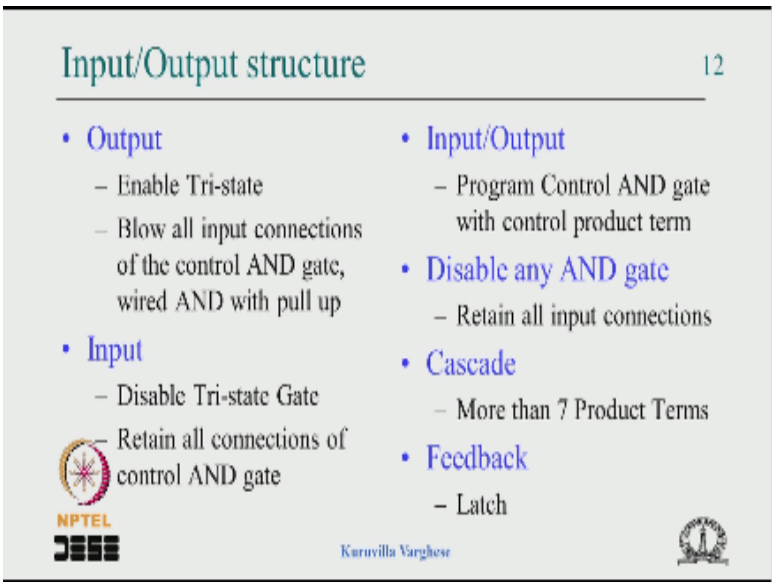


So the last lecture we have basically we have seen this structure we said this can be which can implement output by enabling this, disabling can input and this product term can control it as output as IO like a read bar can make it is a input then read bar is high to output and this

can be used for cascading like you have more than 7 product terms then 7 can be implemented here and inversion second inversion.

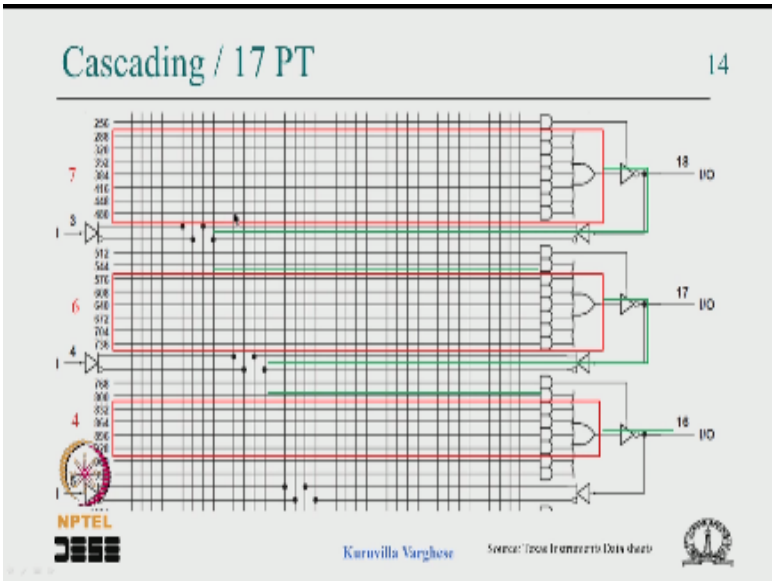
And that can be captured you can implement and so on. Also this can be used for cascading can use for feedback we have seen you can build a cross couple latch between these two section using an AND gate here, going to the to the NAND gate here and coming back to the NAND gate there and of course with second reset input it is possible.

(Refer Slide Time: 02:39)



So we have listed all that all those facilities and we have looked at the car skidding scenario.

(Refer Slide Time: 02:40)

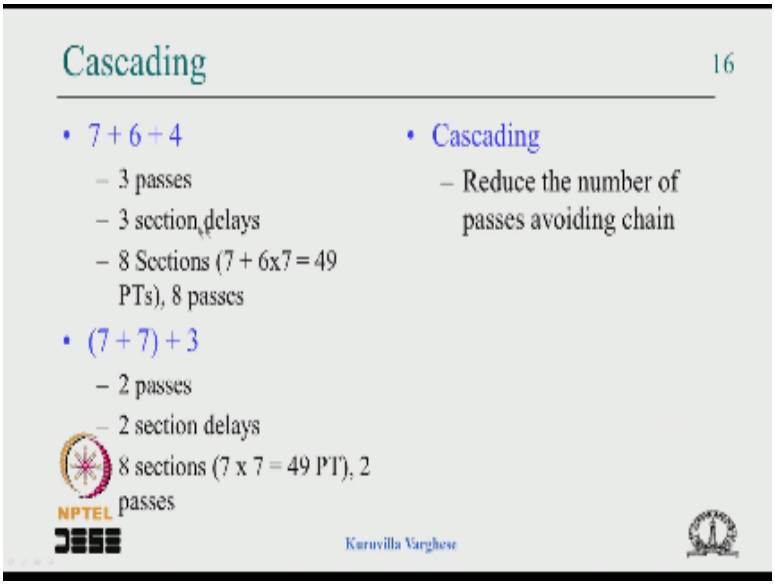


Suppose we have 17 product terms 1 way doing is that you implement 7 here take that 7 and put it on AND gate and then you are left 6 product terms that you implement in 6 AND gates

then that is mixed 13 and we need 4 more, so take this and put it here and you the last section will use 4 AND gates and these two are disabled by retaining the connection or the fuses. But this will incur 1,2, and 3pass delay, it is not a good idea, so the ideal thing will be to implement 7 here, 7 here and if you take both these 7 product terms to a third one and use AND gate to cascade it and use the rest 3 here ok.

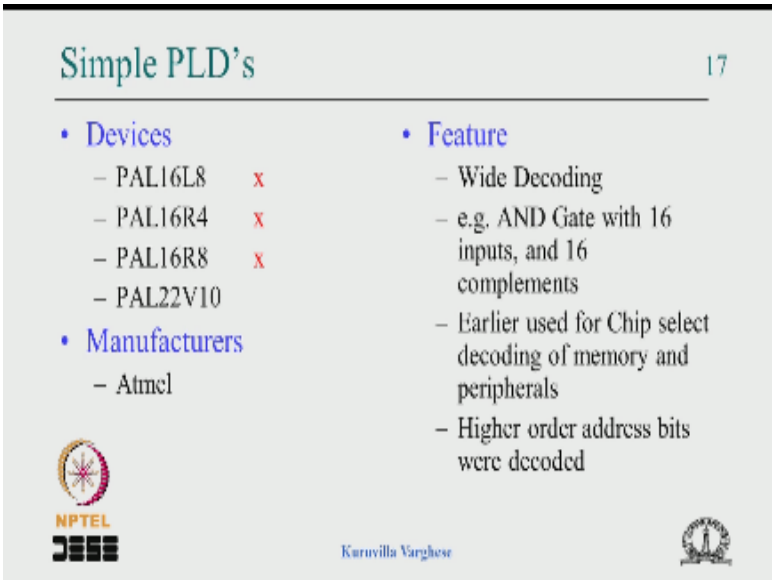
So that will make a kind of 2 pass, and you can go all the way in a 16L8 kind of device maximum you can use one last section with the seven AND gate for cascading and rest all section with the previous action with 7 AND gates to 7x7 49 product terms can be implemented with 2 pass delay.

(Refer Slide Time: 03:59)



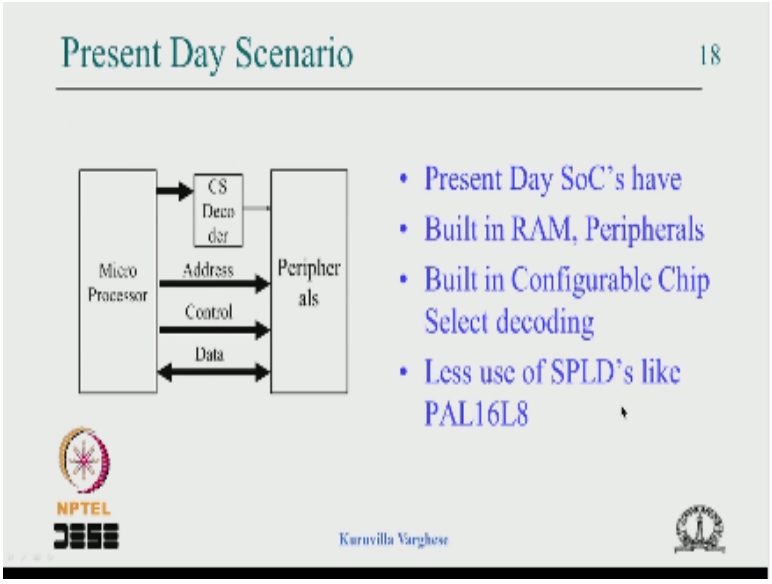
So that what we said if you change it you will get 8 pass with a 49 product term but if you do a 2 pass like that then you will get same 49 productive but it is very minimal delay, that is how you cascaded.

(Refer Slide Time: 04:16)



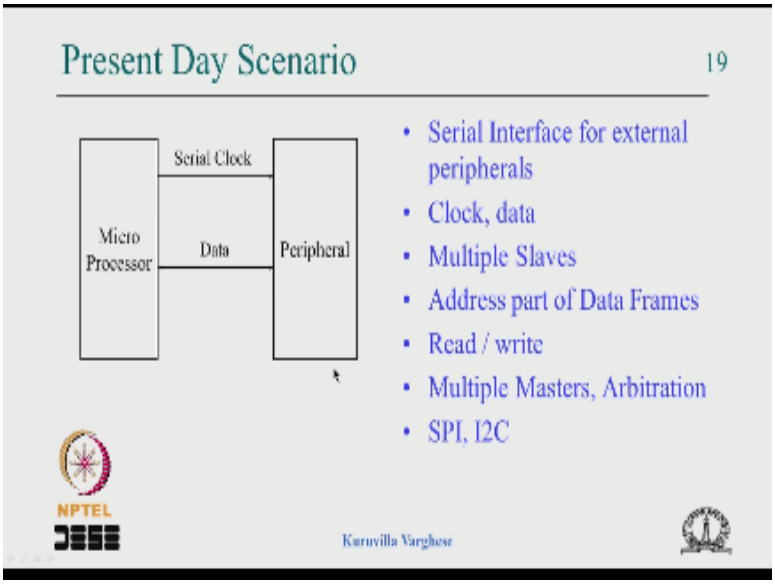
And we also have looked at the simple PLD like 16L8, 16R4 16R8 and these are all are with the registers and which is not used and at 22V10 and Atrnel manufactures them and basically the feature is that it is a very wide decoding that means a productive term can have the AND gate can have 16 inputs and its complement and this is used in chip select decoding.

(Refer Slide Time: 04:50)



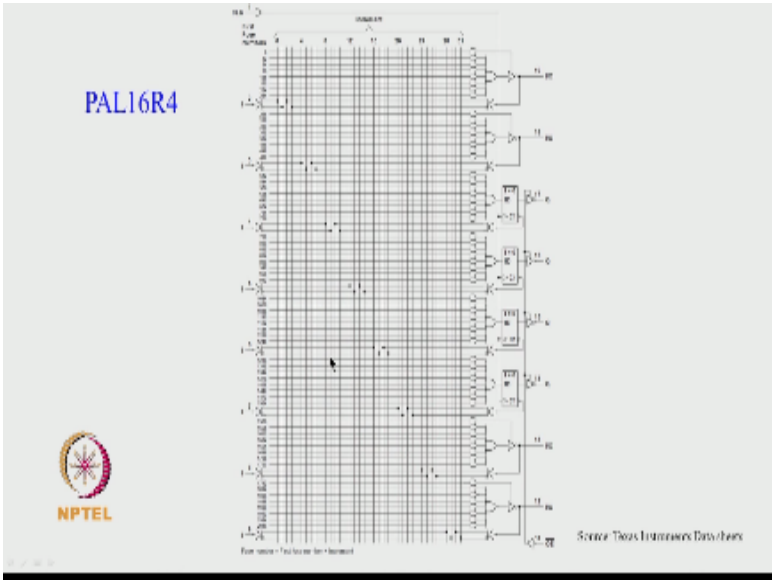
You know in a scenario like this you have an UP your memory or peripherals, so will connect the address bus maybe this has 20 address bus say 12 were used for addressing the location 8 were used for decoding, so you need to decode 8 address line higher address line and that is where this PLDs were used and this is the control the data, but nowadays what is happening is that many time these are built into the SOC and this is also built with SOC which kind of throughout the use of devices like these 16L8.

(Refer Slide Time: 05:29)



And much more than that you know nowadays peripherals are not directly map to the memory space they have a serial port, ok. So you have a serial clock and the IO date data in and out and this support way and the support is elaborate addressing through the data frame and you can have multiple slaves multiple masters all that, you know this kind of you know removed the need for the simple PLDs.

(Refer Slide Time: 06:06)

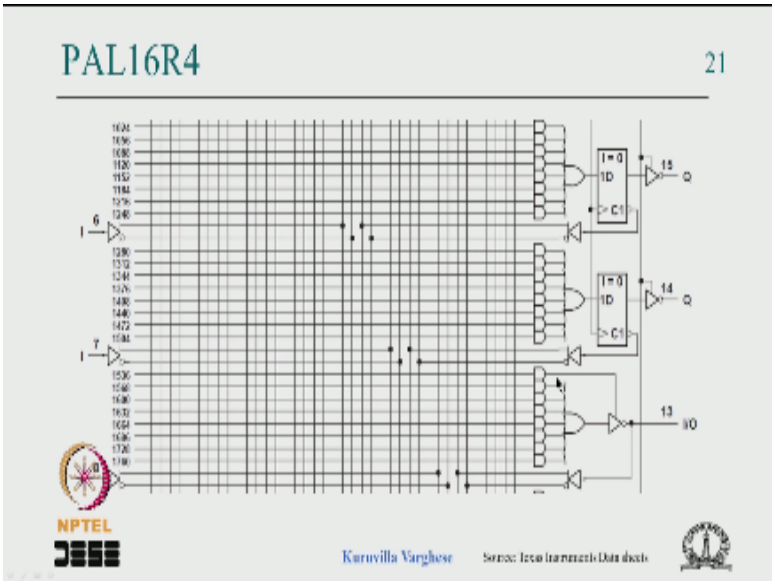


You know that is why it is not use very much and for understanding have looked at 16R4 it is nothing but 16L8 success but the output you have some 4 flip flops and recombination section and adjusted section a common clock and an output enabled, important thing is that the Q bar is feedback in this array. So essentially what can be done as that you can have a state flip flops.

Next state logic with all the inputs and the present state feedback. So you can implement the counters ok, with the next state logic and this is a the state flip flop and you can even implement state machine because this feedback the present can be decoded in this array and generate the Moore kind of output with combined with the input because everything is available here, combined with input you can make it Mealy output ok.

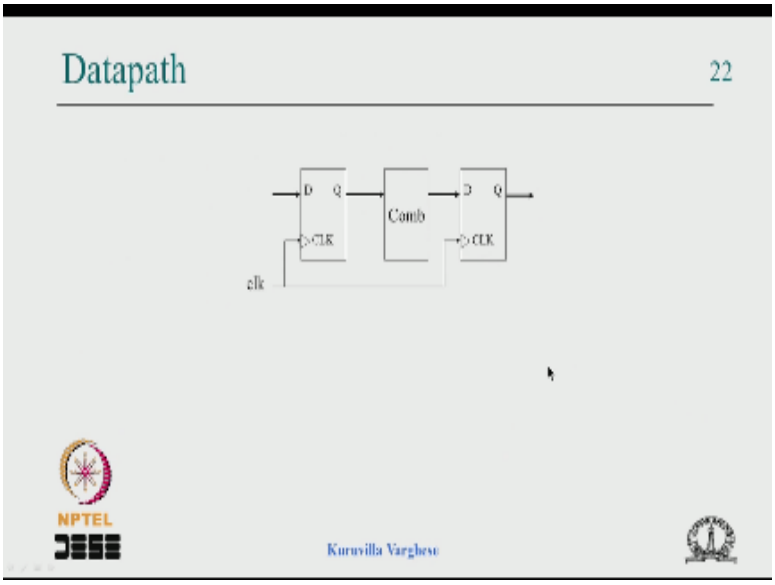
So that is how the data path and that if you have a register combination circuit to register even that can be done, you can start with input going to a register that output is kind of you know a computer with a combinational logic, that output can go to another flip flop and you can imagine parallel kind of data within the constraint, suppose you have a device like 16R8.

(Refer Slide Time: 07:37)



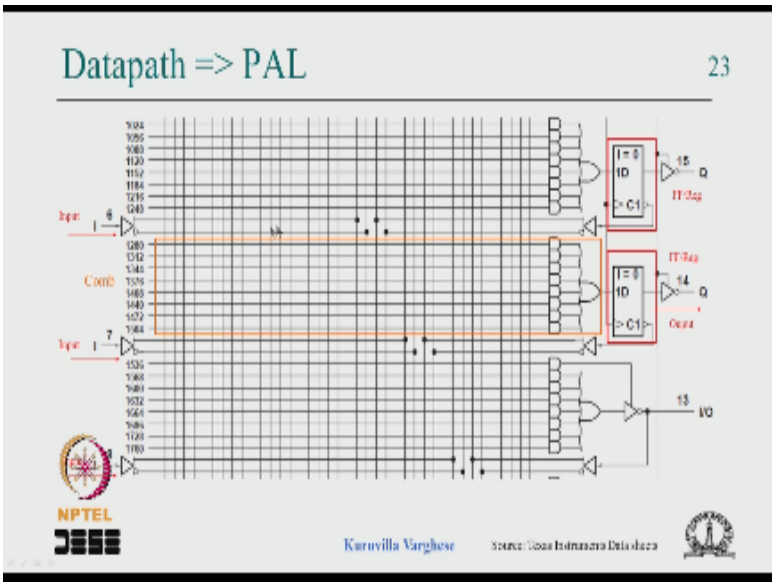
Then you know that is not shown here.

(Refer Slide Time: 07:40)



But then you have all the registers then you can have maybe little more date bus. In that case and we have see how this is map you know data path combination data path.

(Refer Slide Time: 07:53)

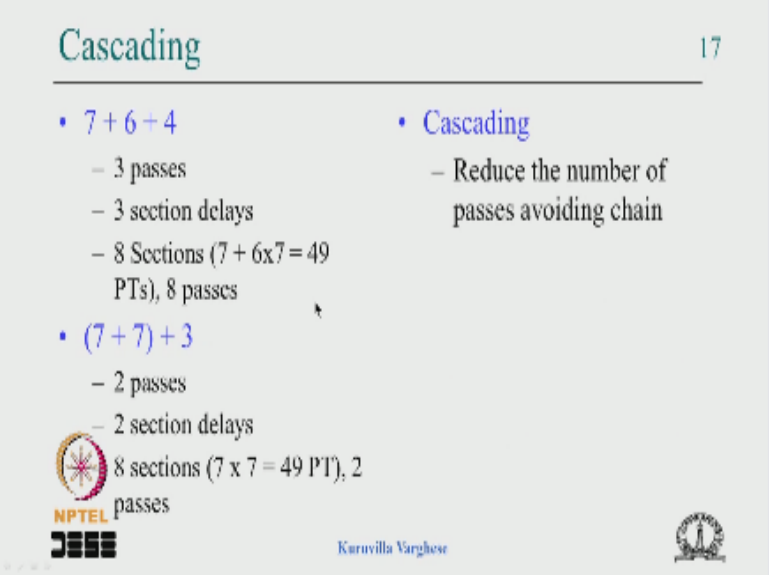


So you start with the source register with inputs coming because the inputs are available to every kind of section, then that present state is feedback and then you have an exceed the combination logic output go to destination register and if you have a state machine we have the flip flop input and the present state comes here and output on the present state goes to output logic.

So exactly same you have the import, you have the state flip flop, the feedback is here, so input and the the present state is combine in next state logic and goes to the skirt flip flop, the present state or input is again Decoder to produce output, so it say it look like you know this

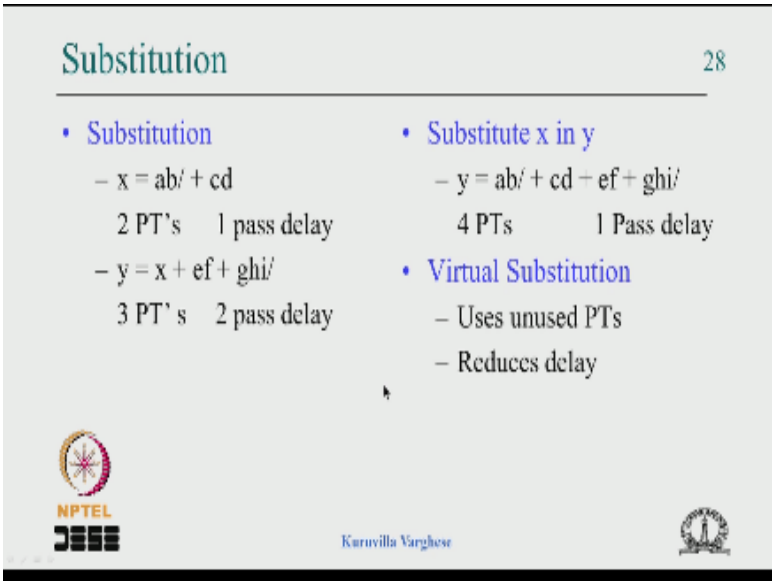
structure itself is converted into this form and it is very nicely, so it is a nice kind of architecture where in the sequence circuit can be easily implemented in a PAL.

(Refer Slide Time: 09:05)



Whether you look at this way or this way does not matter and may be this is ideal because you have a kind of say if you like that you know you have the next state logic and output logic here. All the inputs and the present state is coming, so this from the next state logic and rest of the path forms the output logic ok.

(Refer Slide Time: 09:26)

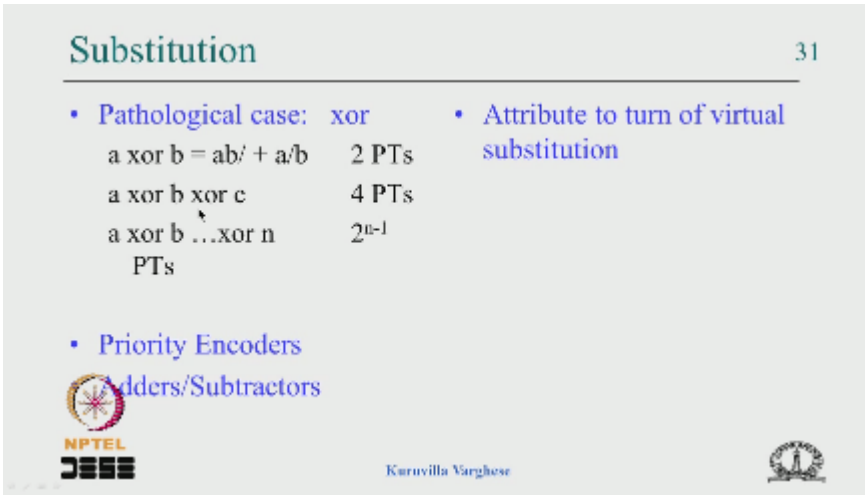


So that is how the sequential circuit and data path match with this and one thing I said that suppose you are implementing kind of say 2 product terms here ok say an example x is ab/cd and that x is implemented here and y is nothing but the x+2 are the product term, so one way

of implementing is that x you implement 2 product term, y you implement the feedback maybe x you implement here.

That x you combined in one product term and implement the next 2 products, but then the delay will be 2 pass, so it is easy we can replicate that product comes here substitute ok, that is what is done here.

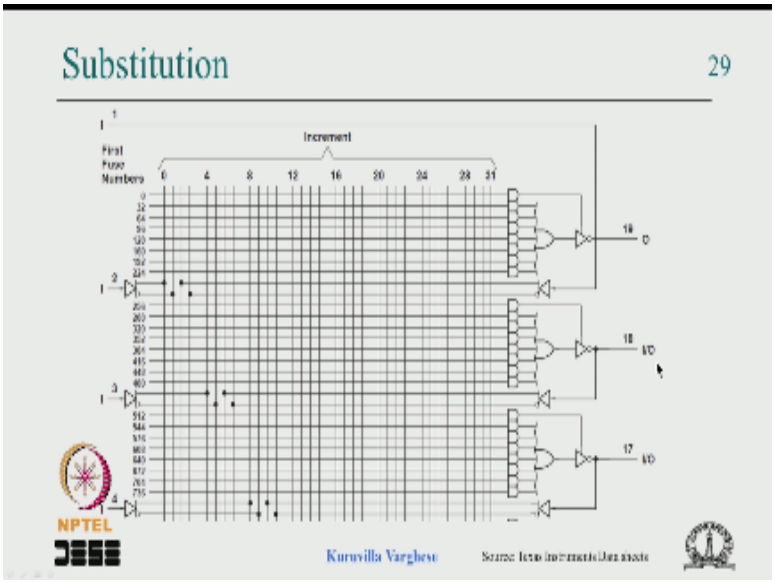
(Refer Slide Time: 10:16)



So instead of using a node here already implemented nor we substitute this product tern and straight away implement that because the product terms are available ok. So this is called virtual substitution but this can go wrong like in the case of an xor because in this simple structures there is no xor and a xor b is 1b bar or ab and when you take 3 xor then you end up with 4 product terms.

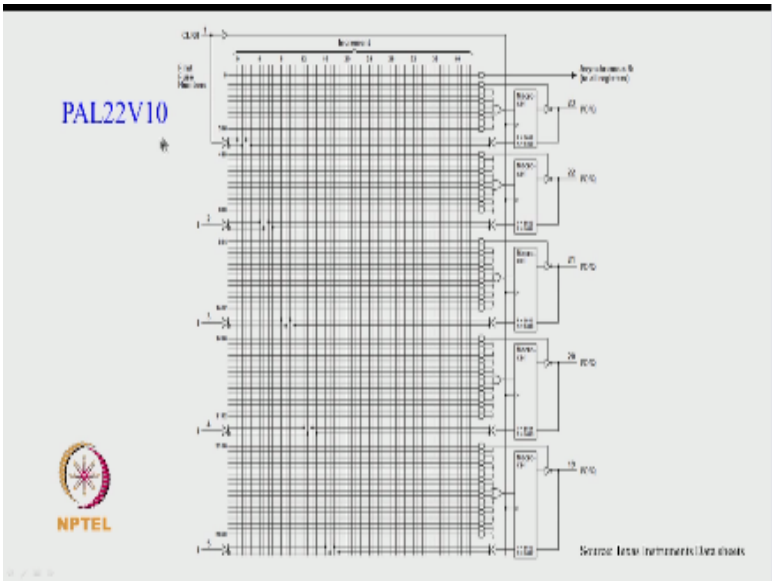
A xor to raise to n-1 product term and so if it goes on expanding then suppose you have a as a xor gate then you will end up 2 raise to 9 which is 5-12 product terms which were available in a simple PLD, so when you implement xor kind of priority encoder the arithmetic circuit where xor is there if substitution is a kind of implement then there is a problem and this is like it is better to implement kind of do not do the substitution like you implement a xor b.

(Refer Slide Time: 11:39)



Then in the next section u xor with c because u implement two product terms in one and bring it here and do that for another two product term here than you know expand here. That is why that the substitution may have to be turned off to avoid this kind of cases.

(Refer Slide Time: 11:50)

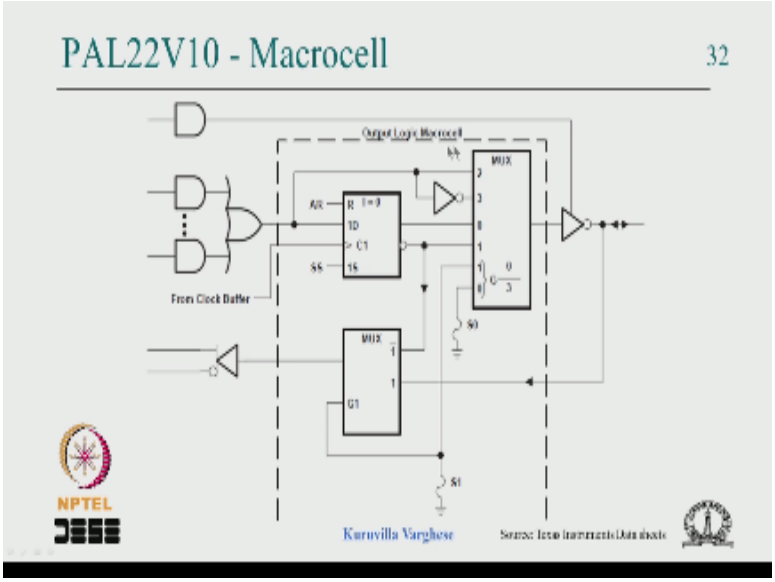


And the last we have seen is the 22V10 which combines the utility of simple combination device like 16L8 and 16R8 when you have something called a macrocell which is nothing but flip flops with bypass. So that this section can be used along with the register or directly ok and you have the clock common clock to the flop and the feedback can be from internally with the flip flop is used.

If it is bypass then the feedback is from here ok and if you are using all section like combinational kind of logic then this clock is not required that is used as input at the variable

product terms first section as 8, 10, 12, 14, 16 and then it is mirror down. There is a product term controlling the asynchronous reset and another product which is controlling the synchronous reset, so quiet versatile and we have seen what is inside this.

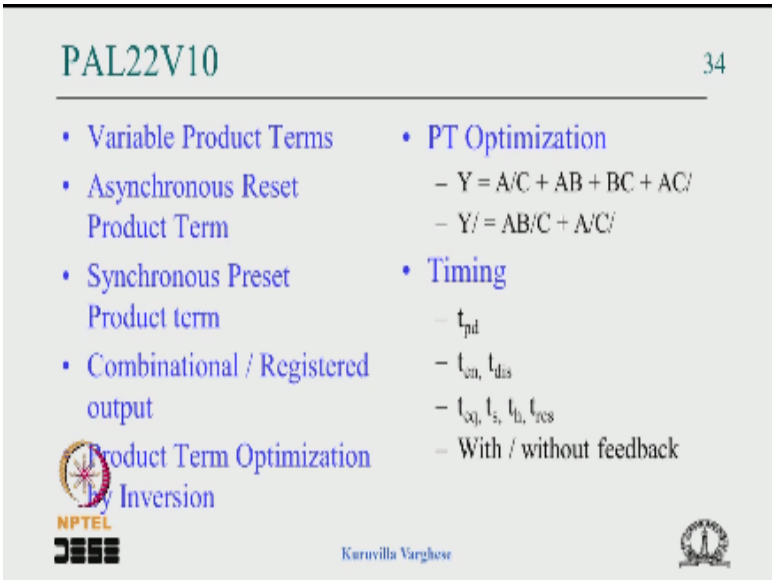
(Refer Slide Time: 12:57)



So basically and then there is register but there is a mux where you can use a combinational output directly or the registered output and you can use the combinational or x complement Q or Q bar and this allows you to apply the De morgan's theorem and optimise the product, suppose you are using a section with 8 product terms and you implement y, so normally say if used y then you implement y here go through this invertor.

This invertor you get it, but suppose that has more than 8 product terms you implement y bar here and y bar here it may reduce to say less than 8 and you can implement it here and you met here, so this section allows a product term optimisation and this is the feedback marks using the Q bar or the input and this is the product which is controlling the the tri-state enables. So this combines this can act as a 16L8 or 16R8 you know that kind of flexibility and these are available now, this can be used.

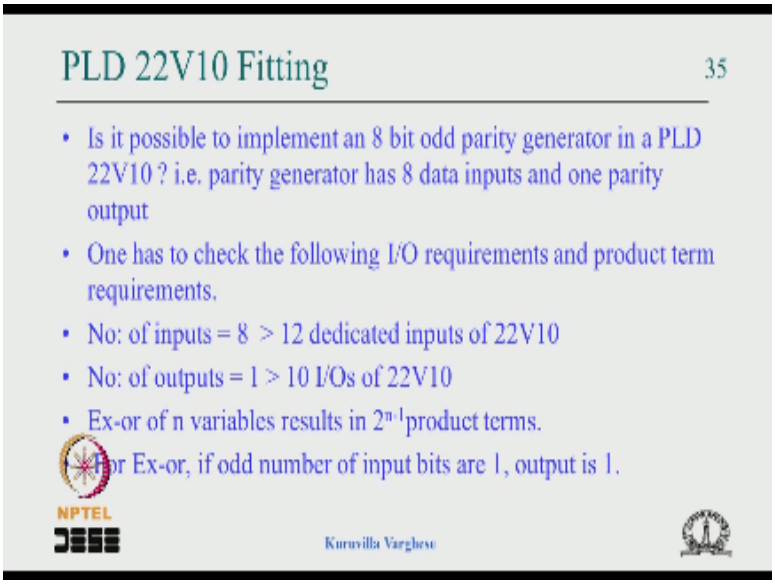
(Refer Slide Time: 14:16)



So the features we have seen your variable product terms asynchronous reset, synchronous reset you can use as combinational or registered output, product term optimisation and an example we have seen like there is a y with 4 product term, so you do the De Morgan theorem, then minimise it end up with two product and these are the kind of delays you have propagation delay when it is used as combinational kind of section.

This is the delay of the tri state gate, then you have the flip flop with setup and hold time Tcq we have discussed this and the kind of recent delay with asynchronous reset all that ok and then with or with without feedback we can kind of you can work out. Now suppose you have like recombinant section with feedback then you have to add all that delay and you know with feedback you will have some extra delay because there is a mux and that we added that, so that is the timing of the PAL22V10.

(Refer Slide Time: 15:31)



So I am putting a kind of question this is an exercise kind of thing, question is that is it possible to implement and 8 bit odd parity generator in PLD22V10 that means parity generator has 8 data inputs and one parity output. So you know that this parity generator is nothing but you know you have suppose you know I1 xor I2 xor I3 like that it goes ok. So first thing when you try to address that other two things to check.

You know you have 22V10 ok that you know the 22V10 internals so you have kind of 12 input dedicated input, 10 dedicated output we have to check whether input and output are you know sufficient to implement dysfunction and second thing to look at that whether the number of products are enough ok. So let us check that you know and the IO requirement is the number of input is 8.

Because it 8 bit parity generator, but we have 12 dedicated output so 8 less than so hurry I put greater, so 8 less than 12 so you have you know that is sufficient and number of output is 1 less than 10 output of 22V10 there should be enough, so I apologize just you treat as you know less than ok and we do not have any XOR gate in 22V10 neither in the combinational and or section nor the macrocell anywhere ok.

Because they could be some PLDs where there is an xor gate you know at the output we will see that CPLDs ok. So there is no xor so we have to the tool has to kind of expand this in terms of AND/OR gate ok. Now we assume like for this we assume that we are implementing the substitution we are expanding it ok that is that is what we assumed that we are assuming that is expanded, that is substituted.

So xor or n we are not implementing intermediate notes ok we extend maybe that assumption can write here so for n variables we have seen just in the previous lecture there are 2 raise to n-1 product term. So that is why is the odd number of input with someone and output is one soon from there we can understand that so for 8 bit inputs we need 128 product terms because 2 raise to is 7 product terms are there.

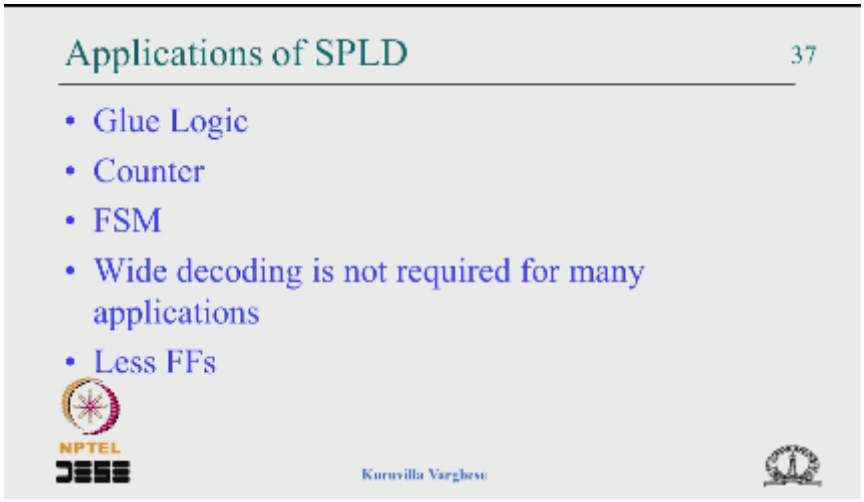
But if you look at the total product terms available in a 22V10 PLD is 8+ you know 8x2+ 10x2, 12x2, 14x2, 16x2 ok and that we will end up with you know 120 product term, but we have 120 product terms much more than that we have to cascade ok and we have 10 section so when we use for cascading then we have 9 section for cascading, so we have to use a section with 10 product terms for cascading, so we are left with 120-10 product term.

So we have only 110 product terms, hence this resource the not enough ok so it cannot be fitted ok. So provided and we are assuming that we are expanding we are not we are keeping the delay minimal, we are not kind of implementing the intermediate input you now possible that you implement the xor gate then cascaded to the next xor gate, into the next XOR gate then it will be possible to implement.

But then it will incur lot of delays ok, we can approach various other ways you know you can think of a 3 implementation you know, where in you can start with some 4 xor gate 2 input XOR gate then it goes to the kind of a logarithmic reduction can be applied, then it is possible to implement. But I am discussing the case where normally the tools kind of substitute expand it.

In that case it will not be able to fit it and that the case with minimal data, a cascading some kind of chain cascading or kind of logarithmic kind of cascading or binary kind of you know 8421 that kind of cascading will work with you know this 22 bit and fitting maybe have to kind of manually little bit 3 attributes to do that.

(Refer Slide Time: 21:01)



So that is one advantage of knowing the architecture of the tool is that you can work quickly work out whether something with and that I want to highlight because if you have no idea of the architecture you will not be able to exploit this kind of situation ok. So if you know that using the tool attribute using some kind of coding techniques you can get out of this kind of you know bad situation to implement something and which many times people ignore.

And that is tool when we go to PLD or FPGA you know the architect and we also know that you want something whether it really fits and whether the device is able to kind of that thing in the resources can do a rough kind of estimate where in whether the given circuit given logic fits into the to the device. So that is possible, so that is advantage of learning the architecture learning how the secured maps into that.

And I have taught you when we discuss VHDL we know given a VHDL code how an architecture circuit or the equal and logic circuits can input, so if you apply that knowledge has given a VHDL code for I am sure you can work out the logic which that the tools in the size and you check in the target device whether it can fit in all this can be kind of even and a complex design when you break into pieces you can estimated it does not take much time.

And you can rightly choose the device and something goes wrong like you try to fit something in i20 to be done and it says that it cannot fit so looking at other ports you can make out what is going wrong you know maybe the and implementing it as an internal node using some different cascade structure you can implement old thing with minimal delay. All this can be done it is very simple kind of device simple cases.

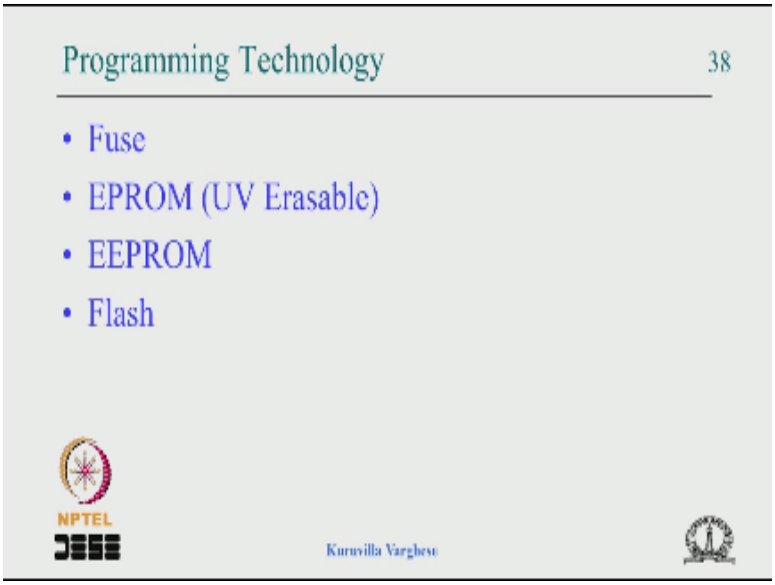
But having the knowledge will enable you to do many optimisation which a novice or those were just walking with the tool which some kind of Verilog VHDL will not be able to do it okay. So applications of the SPLDs that you can implement some blue logic blue means that you already have chips for a microprocessor memory some enter to some random logic, some simple logic you need to implement for whatever reason.

Such things can be combined in SPLD, we not be related maybe while interfacing want if you need some kind of translation of the bus protocol or as some small thing or that can be implemented many things together can be implemented because there are you now 22 IOs and 10 output so many things can done ok and definitely it implement counters or small state machine.

The disadvantages that such kind of wide decoding, the AND gates with lot of time would not required in many cases and the number of flip flops unless you know that I used to be done there are 10 flip flops and practically you cannot do much with the 10 flop you cannot have any kind of memory implemented here. Suppose you need a shadow register that cannot be implemented.

So that because of the lack of registers this do the logic is quite the logic section gives you lot of products terms but lack of registers limit the use of these devices ok. So that is about the applications of simple PLD like 22V10.

(Refer Slide Time: 25:24)



Then let us look at the programming technology that means that say 22V10 like devices can be fuse programmable, but nowadays it is not used to have not very of any kind of practical

interest earlier that fuses I have shown in the initial slide fuse and you know passing a large current and blowing it. Then there were EPROM again is not very much used. These were the kind of devices which can be electrically program.

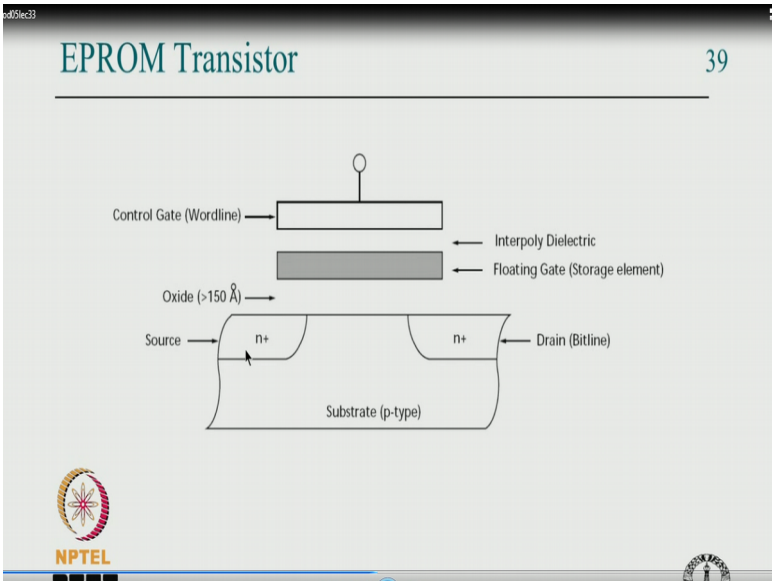
And once you program it to raise it you expose the silicon die through a glass window through to the UV light and some 5 to 10 minutes of exposure will kindly remove the programming ok. Once again this is quite famous may be 20 years back or 15 years back where and all the memories with the actual our window and you program as and when you want to raise it you exposed to UV light for sometime like you cook the device then it gets raise.

And not very much used to not use that all now not required but then we will have a look at those transistor just for the learning purpose because the next technology is kind of related to it ok. It is a variation on that so the another technology used is EPROM where it is electrically Programmable erasable ok. Here it is electrically Programmable UV rays but here is electrically Programmable electronic ok. Basically and there is a flash transistor which is kind of similar is not not much of a difference construction wise why it is different.

And there are a number of raise cycles is more in flash than the EEPROM number of times you can reprogram is more in flash, but then kind of the data stability will be more in EEPROM where there is no it is less susceptible to know is this is less acceptable noise and but then again this bit is storing nowadays everybody use flash. So if you use if you ask me current technologies flash all these are kind of bit historical interest.

But nice to look at it because the way will happen will teach you something. So we will look at those these the fuse we have seen these 3 we will look at it ok. So this is the EEPROM transistor.

(Refer Slide Time: 28:21)



This transistor is used to construct the AND gate and OR gates ok, and as I said as I mentioned it is a wired AND gate, it is a wired OR gate ok. So we will see how this enables one to implement that wired structure, so I think you are familiar with the CMOS transistors. So this was an enchannel or NMOS transistor where there is a substrate is P type, so normally you will have a big substrate and you are P well in that there will be n channel, n channel.
And normal transistors will have a insulator here silicon oxide, then there will be a gate polysilicon or copper get on top of it and normal keys only be only one gate and you what you do is that source ground, drain you give a supply say 3 volt or 5 volt, then you apply 5 volt here because of this electrical field are the electrons will come here and n channel is form conduction happens ok.

Now in this EEPROM transistor there is Silicon oxide, there is a gate which is poly silicon which is floating which is not connected to anything, then there is Silicon oxide, there is a normal gate control gate. So the normal case relaxation en transistor you ground the source, you apply the drain voltage, you give gate voltage which about the threshold voltage because of the electrical field which is to the positive side in electronic come here for my channel is conduct ok.

Now this floating gate allows you to completely turn it off ok, when you turn it off it acts like the blowing the fuse ok that is basic idea ok we will see that, but why we turn it out is that this control voltage will have no effect on the conduction ok that is how we kind of blow the fuse ok. So what is that you ground the source and you apply the positive voltage to the drain

and a brief positive pulses given here like say supply voltage is 5 volt maybe you give a 6 volt or 8 volt pulse for short duration.

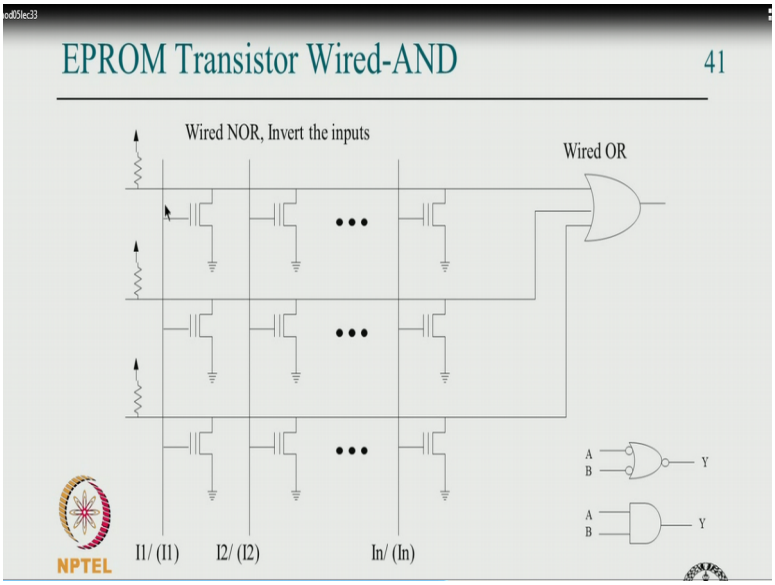
Because the electric field is high and this layer is very thin actually very you know kind of depends on the technology and this shows 150 Armstrong but it depends on the current device technology compared to the normal Silicon oxide thickness is less ok and what happens is that because of that feel electrons traveling to the drain will tunnel through this thing insulating layer and get trapped here ok.

Because it is a conductor and then you stop it what happens is that because some electrons are sitting here, how to format channel here you need to apply more voltage here, so if you apply 5 volt the gate voltage channel would not be formed and there is no conduction, so it is like a transistor which in the absence of this chapter it act like a normal and transistor and when you program it to be off by capturing the electron.

Then it does not conduct even if you apply sites like going the fuse. Now we use this transistor to build a wired and wireless of structure, that is how to program off ok. So suppose you would not say a particular input does not affect then you just program it often and we will see that structure to raise there is no easy way in the keys of EEPROM transistor as I said there is a like these Silicon dye is a in a chip will be a window on top of it and you are exposed to UV electrons.

Because of the UV radiation will get enough energy and tunnel back and get out know that that is how it is raised ok. So we will see how a wired structure us form ok. So now take this AND gate so what we are trying to bill is AND gate okay which say 32 input ok or some kind of some number of input and so many AND gates goes to AND gate ok now these AND gates are built using this transistor ok.

(Refer Slide Time: 33:34)



And it is a wired structure, you will see that, so this is how it is formed. So this shows assume that this line is AND gate ok, this line is another AND gate ok, so there are 32 inputs then there 32 vertical lines as shown in the picture and between those vertical lines are connected to the gate or this EEPROM transistor, but the drain is connected to this common line the source is grounded ok.

All NMOS transistors with floating gate and this line is pulled up through a resistor on a transistor acting as a resistor you know you can have NMOS transistor with the gate connected to the VDD can act as a resistor. So do not think that the register is implemented as a pass which can be an inactive circuit, because in a single dye you can implement transistors easily than again bringing in the passive elements ok.

Now suppose and these vertical lines are connected to the input suppose this is connected to I1, I1 basr, I2 I2 bar and so on. Now if you if you do not program it of you see when this line is high any of the vertical line is high, this is transistor will be on and this line is pull though. So the logic is that if you retain some connection if you do not program at of if that line is high the output is 0 ok.

So if input is high any one of the input is high output is 0, which are wired node ok, so it is a NOR circuit ok so NOR is like you know there is a OR gate and NOR gate okay, so this bubbles are not that but what we want is an AND gate okay, so we know that again we have a you know study the gates and function, but if you invert the A and B ok then you will become an AND gate automatically.

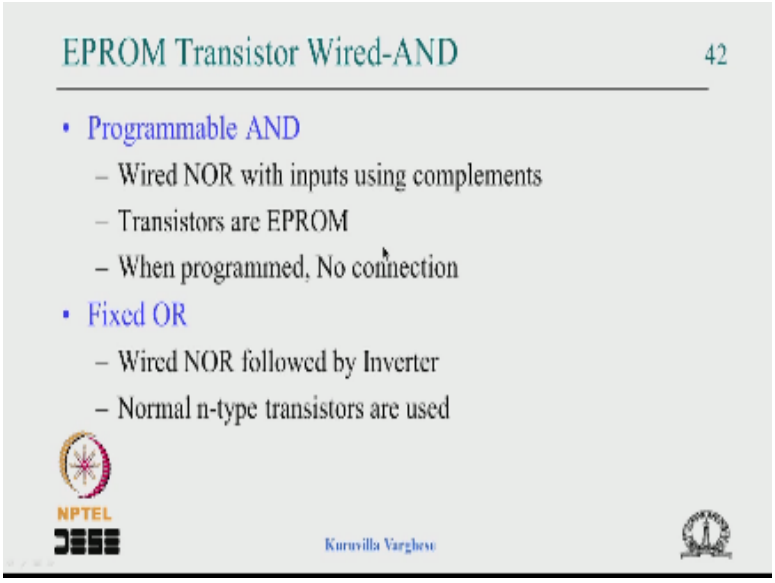
So this structure if you give one input directly it will implement a wired not circuit because any input is this line is 0, so that is NOR gate, but if we invert input you say I1 you connect I1 bar then instead of I2 you connect I2 bar then automatically it implements AND function wide AND function ok. So that is what is shown normally it is a NOR gate, but you invert the inputs.

Then you will get an and Gate as far as the active circuit is continent is nothing but y bar is A bar or B bar ok, which is nothing but you know y is you know if its applying the De Morgan theorem if you can you get AND gate ok. Now that is very easy to do in a PLD because you see that all the inputs are inverted both input and the complement of the input is there. So how it works is that suppose you connect A this line act as A bar and this inverted line act as A ok.

So you want to program A to this you know this AND gate then what you do is that you retain the connection of you know A bar and you blow the you do not you know put off the connection for you know A, so instead of where you want A use A bar where you want use A bar you say A I am seeing both are available a matter of swapping the position of the programming ok.

So that is what is I have shown for I1 the I1 bar is use I2 I2 bar is use, so that is very simple for a like an AND gate with 32 inputs you need 32 EEPROM transistors with a single line with a pull up transistor and a similar to replicate that.

(Refer Slide Time: 38:15)

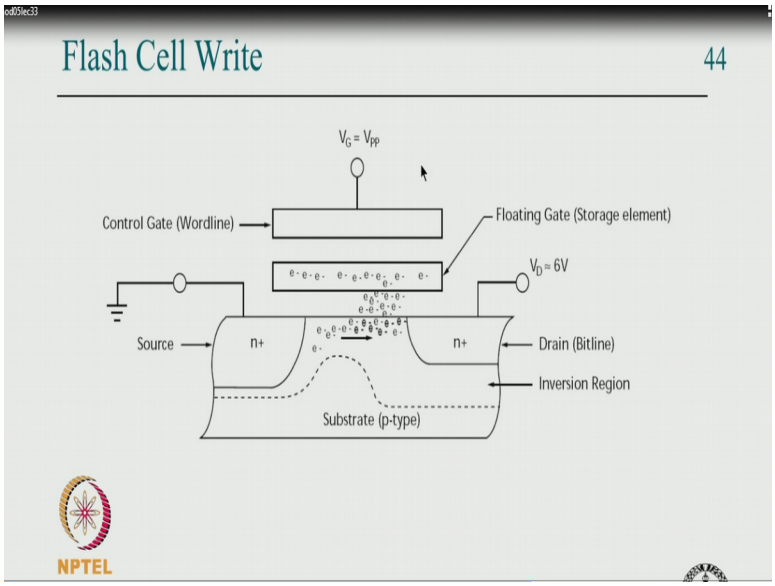


You have kind of 7 product comes then 32 into 7 which all goes to and wired OR gate and the same structure is that it is a wire NOR gate, so you put a kind of here you have says 7 input OR gate, so same thing in all these 7 are going to 7 and these kind of transfer with a single line pulled up and you put an inverter, inverter you can put with NMOS and CMOS and then you get any inverter through this wired OR is nothing but a similar wired NOR with inverter.

You see how this is program using the transistors alone is very easy that is how it is implemented in PLD or CPLD and things like that and so wired NOR with input using compliments transistor are EPROM, so when program there is no connection when you program it off there is no connection and the fixed or that is a Programmable and, fix OR is nothing but wired NOR followed by invertor, normal Ntype transistors are used with very easy to implement.

It does not mean you do not have to worry weather how to do kind of implement the 32 input AND gate is not very complicated logic in terms of transistors ok because the wired versions are implemented and I will let us look at the flash on EEPROM transistors the construction of similar but the gap between the floating gate and this channels are low ok and you can also see that like if you see the construction will be little different it is kind of little more extend over the channels this source and drain.

(Refer Slide Time: 40:28)

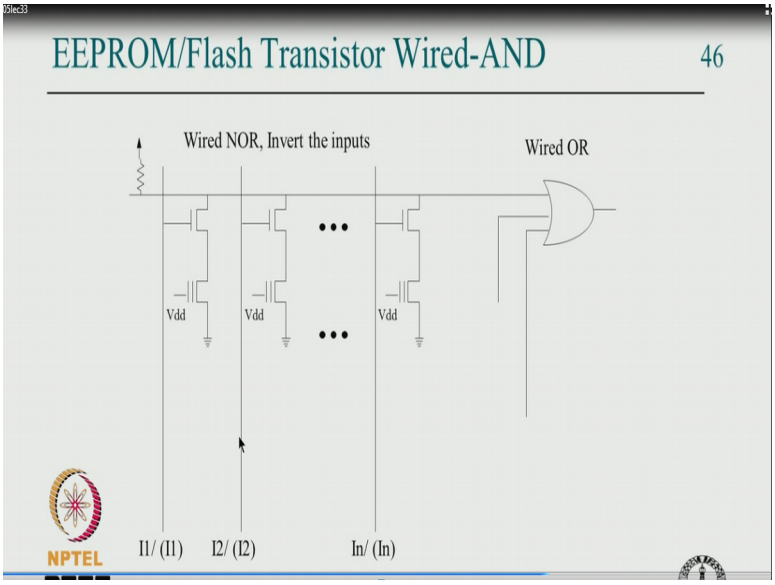


Now what happens is that similar operation you know you just ground the source apply the drain voltage and you give a programming of little higher voltage to this gate, so these electron there is a feel from drain to source there is a feel from sorry source to drain electrical field from the substrate to the towards the gate and because of higher voltage the electron gets stuck here. So it is programmed of but you want to erase that what you do you do not have to expose UV you do the opposite ok.

You ground great you ground you know let it be floating and you apply up higher programming voltage source then these electrons channel back and you can electrically erasable in the circuit, the advantage with this transistor is that it like you kind of you can electrically program it and you can electrical it is very fast than exposing it UV, the flash is kind of flash and EEPROM.

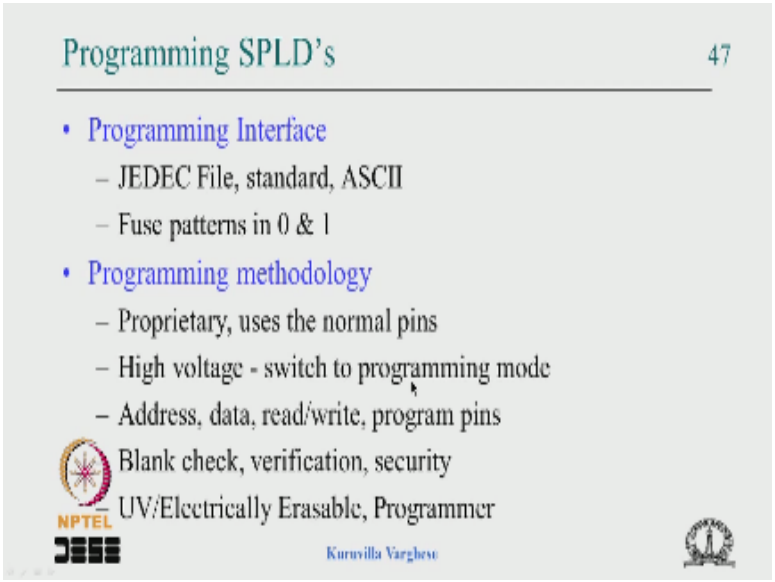
The construction wise little difference is there, but one only one problem is that when you program it off this because of the construction even if the gate voltage is less than the threshold voltage it conducts ok. So this can only be used kind of cut it off and you need a normal transistor and series for really controlling the wired implementing the wired and NOR function.

(Refer Slide Time: 41:52)



So this is how when use flash or EEPROM using normal and transistor for forming the wired NOR structure or wired AND structure with inverse preparing inversion and you in series you connect this kind of the flash cell order from cell to cut it out when you do not want one input to be connected to this and get them that you program it off know that is how are the flash or EEPROM structure is used.

(Refer Slide Time: 48:29)



So and the programming in the face the programming standard is gives JEDEC file, it is a company's name, it is a standard ASCII file if use pattern will be 0 and 1 where there is connection in busy over there is no connection B1 and the normal pin of the PLDs used for programming for that are the hardest basically of to address this is location or the transfer location to program it.

So there is an address line data line read write line and the program pin. So how is a higher programming parts is applied to this program pain then it gets in the program mode then you give address data rewrite signal in the appropriate kind of with timing it gets in a program then you can use it is it normal device with iOS pin used as user IO ok and the progress you put in the programmer you can erase it and we use it so that is all about this SPLD.

So what we have seen today is in the just previous part is that we have looked at the various me know the one example of fitting and we have taken a kind of parity or parity generator and we have assumed that your kind of expanding at in terms of product of the unemployment in node and they found that 8 bit odd parity generator with expansion cannot be implemented in a 22V10, but with internal node you know one to one picture of a chain of 8 2 input XOR gate or are you know you can cascade 4,2,1 of 2 input XOR gate.

All that can be implemented, so that gives you kind of freedom and as I said that kind of give you an idea how to know the architecture of a device and how to map from me when from a VHDL code down to the level of fitting, you can after some practice you can even start kind of working days in the mind even in a complex circuit by party know you can take a paper and start working out whether how many flip flops are required how many sections are required how much delay can incur.

And all that can be calculated you can use estimated of estimate can be worked out without blindly going to the to learn coding in applying with the two line already it is all possible but you know you can do that yourself you know that advantage then we have seen the programming Technology the fuse have seen we have seen from transistors how a floating gate allows the transistor to be cut off.

That means it you can program of the transistor and the same transistor is used to implement a wired AND get and wired OR gate essentially wired NOR gate for AND gate we kind of inverter inputs and it is very easy in PLD because all the input and their compliments are available, so wherever fuse for or programming for A is done in the A bar section and for A bar you do in the A and A section.

That is all the different, so app for free tools and you take care of programming it and we have looked at and that the problem is EEPROM is that yours UV rays it takes time more

than that you have to take the expose the device after sometime the device will become unusable and EEPROM and flash allows you to run program in electrical AZ and we have death scene that is plucked also but it conducts normally so we put a normal form wired and get and wired OR gate.

And put this in series to program it off and that is basic idea and simple PLD applications we have seen it is used for blue logic array as you can implement counters and FSM and it can be used for **very** very small logic but the lack of flip flops does not know enabled to be used in complex application. So that is the simple PLD with this knowledge we can quickly grasp the complex PLD and that is my idea.

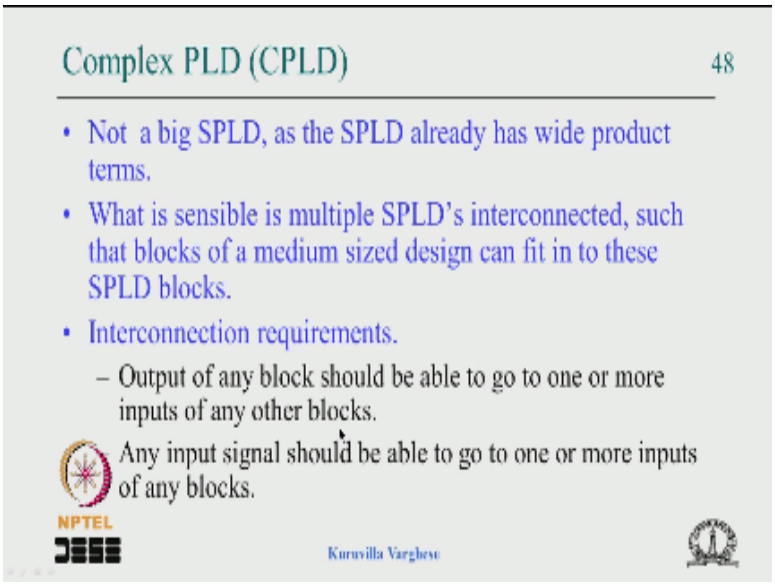
So let us look at the complex PLDS ok now the first thing to look at is that you see that when you say complex PLD say you take this 22V10 structure okay, already we have a section with 8 product terms 10 product terms, 12 product terms, 14 and 16 product terms and as I said most Boolean function does not need all these kind of product terms and the product terms are very wide and this can implement up to 20 variables and its complement.

It is quiet you know complex now when you say complex PLD we do not need to scale this up it does not mean that we make a as a 100V50 implement it makes no sense you know you have hundred dedicated input with maybe as a 300 vertical lines and some 25 product terms with 100 section and it makes no sense will be wasted, it is no use ok. But we already know that this structure can implement counters, FSM, and you know it can implement the combinational logic when you bypass it.

So it makes sense to have you know 22V10 like structure as a building block put multiple of them together and some more in the connector, that means that there are set of IO pins, there are set of 22V10 like structures, you should be able to connect the input to here or some other sections, take the output of this section to the input of the next section. So the idea is that maybe in 1 section you implement a counter.

And you decode it and that goes to FSM in another section through you know kind of IO switch and that section goes to some another FSM output goes to control some other thing and so on, so instead of a big PLD the CPL is we can say is not a big PLD it is a kind of you know hierarchical PLD you know like you have multiple SPLD sections in the connection.

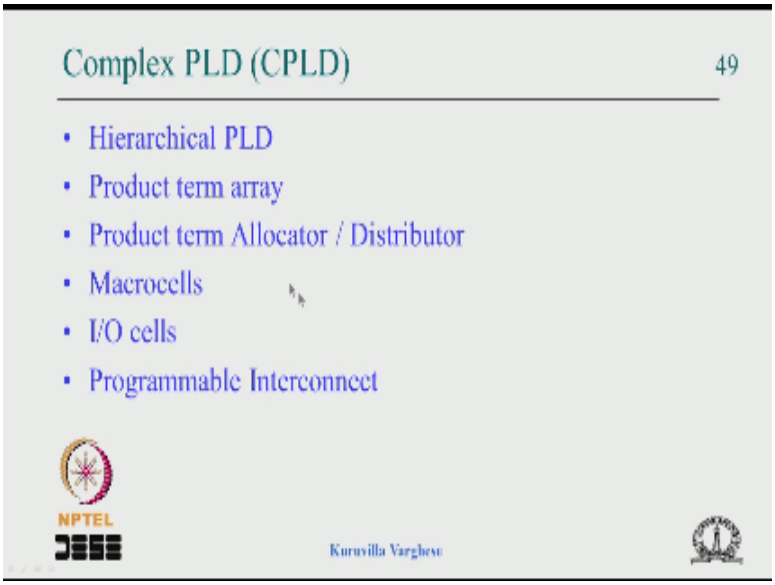
(Refer Slide Time: 49:56)



The requirement is that output of any block SPLD block should be able to go to one or more input of other block to cascade and any input signal should be able to go to the one more one or more input clock ok. So basically you have enough complex PLD of CPLD is hierarchical PLD so you have simple PLD sections interconnected by a switch you know that is called CPLD and you have a productive term array.

That means basically AND get this clip product and product number locator a distributed that means this product comes and not permanently connected to OR gate depending on the need it can be connected to OR gate it can be connected to xor gate because we have seen the simple PLD kind of lack in XOR gate and that is the major disadvantage because when you try to implement expand XOR gate in the terms of product terms it goes out of it.

(Refer Slide Time: 51:16)



So it makes sense to have dedicated Xor gate for parity for product optimisation for arithmetic circuits, all that it is useful so XOR gate and so this product term allocator allows you to use AND gate to OR gate to one of the input XOR gate or there are macrocell wearing flip flops are there, maybe one and it can reset the first one I am get can set the from depending on your name. So this allocator we can say the small switch which allocate this product term to various need .

Weather can be used directly out outside to the OR gate xor gate or the flip flop controller, so that nothing is wasted, so if you permanently connected, connect some AND gate to an OR gate I say there are 5 AND gate it is connected to an OR gate but we are only need one AND gate, but we could use that another AND gate to the input of an XOR gate ok. So this allows flexibility this allocator or a distributor nothing but a switch with connect the output of AND gate to the various inputs of the OR gate XOR gate flip flops and think like that.

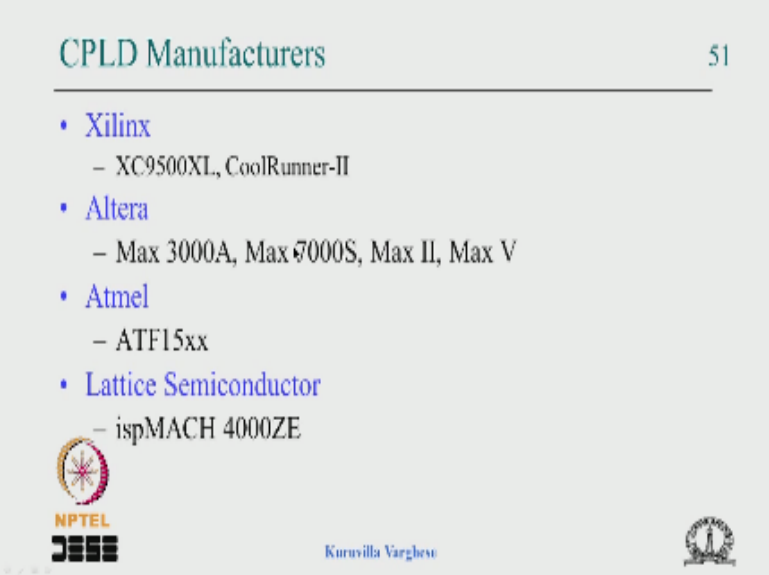
And macrocell is like a 22V10 which is a flip flop with bypass and so on and IO Cell is nothing but IO pin with the tri-state gate with the input you can be used as output input IO can be used for prep feedback and there is a Programmable interconnect which interconnect this simple PLD section, so use switch with take the output of one section to the input of other sections.

Any input signal can be taken to the input of any section and so on. So that is the programmable interconnect, so basically any CPL you look at it will be multiple simple PLD section a interconnect which interconnect the IO pins and the PLD section, within this PLD

section you have a product term array basically AND gates a product term allocator which allocate the AND gate to the various combinational and register.

You have registers with feedback then the IO pins you know that can suite the CPLD and we will see some example of the commercially available today and see how this is kind of this architecture is implemented ok, so ah basically as I said these are kind of simple PLD section interconnected and if you look at the xylene manufacturer then you have tool use of devices XE9500XL ok.

(Refer Slide Time: 55:01)



This is a CPLD which was there from very beginning and the CoolRunner 2 series of family the architecture is similar only thing is that has a name suggest it has very low power dispassion that is why the cool runner and when it comes to Altere they have a series call Max 3000A Max 7000 and their way different various max2 max5 that. So there is at least 4 families and Atmel has ATF15xx series no with some numbers.

Lattice semiconductor has ispMACH 4000ZE family so you have quite a bit of choices between Xilinx, Altera, Atmel and Lattice semiconductor and if you look at the architect it is all kind of similar architecture at least I know that this Xilinx architecture and Altera architecture are some kind of similar support. So for this kind of discussion I will take the max 7000 as an example to show you the architecture of which is can be easily applicable to the **the** Xilinx architecture.

There is no much difference at all you look at the data sheet you would find that quite similar you the whatever I am discussing for this can be applied to this so maybe that we do in the next lecture because we are kind of coming to the close of the this lecture. So the complex PLD is not a big simple PLD because it makes no sense. The simple police already has lot of product terms with which support lot of input.

And what we need is and we have seen that account state machine can be implemented. So when there is a medium kind of or low complexity some particular circuit we will discuss that what kind of application can be used then you it makes sense to have multiple simple PLD section interconnected and we have seen what kind of interconnection IO pin should go to the input of any section output of any section should go to any of the input of the other sections ok.

So that we can kind of interconnect this section to implement some reasonable logic, so it has a c-section multi section interconnect array, a product array which compose some product number locator with the AND gate to the macrocell which consists of various you know the Ex Or Gate nor Gates the flip flops the bypass and so which allow with lot more control more flexibility than a CPLD you are able to connect a global clock or a product import Global Research or a productive reset.

So you are more flexible in CPLD than in a simple PLD, so we will and we will see some kind of available products from you know available products from Xilinx Altera Atmal and lattice semiconductor and as I said in the next lecture we will take an example of Max 7000 Series of architecture will see some kind of code how it maps into this device and so on and then we will wind up we will see the application where can be used and then we will wind up the PLD part.

And we will move on to the FPGA and we will find what is the difference between the PLDs and FPGAs so on, so I add you to go through the portion for yourself you imagine some kind of logic work out how it fits maybe I have told an example of an priority maybe you can think of a counter how it fits and what is the basic Boolean equation how it the delay so on, so I wish you all the best and thank you.