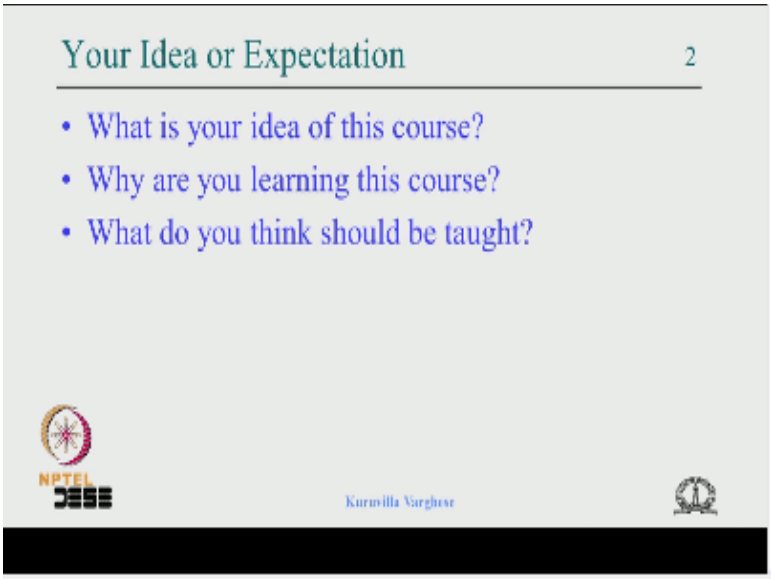
**Digital Systems Design with PLDs and FPGAs**
**Kuruvilla Varghese**
**Department of Electronic Systems Engineering**
**Indian Institute of Science - Bangalore**

**Lecture-01**
**Course Contents, Objective**

I am Kuruvilla Varghese from department of electronic system engineering Indian Institute of science Bangalore. I am going to give this course on digital system design with PLDS and FPGAs. As a name suggest it has something to do with the design of digital systems and we will be implementing the system in devices like PLDS and FPGAs. So over the next 40 hours I am planning to complete this course from the basic to advance portion.

**(Refer Slide Time: 01:08)**



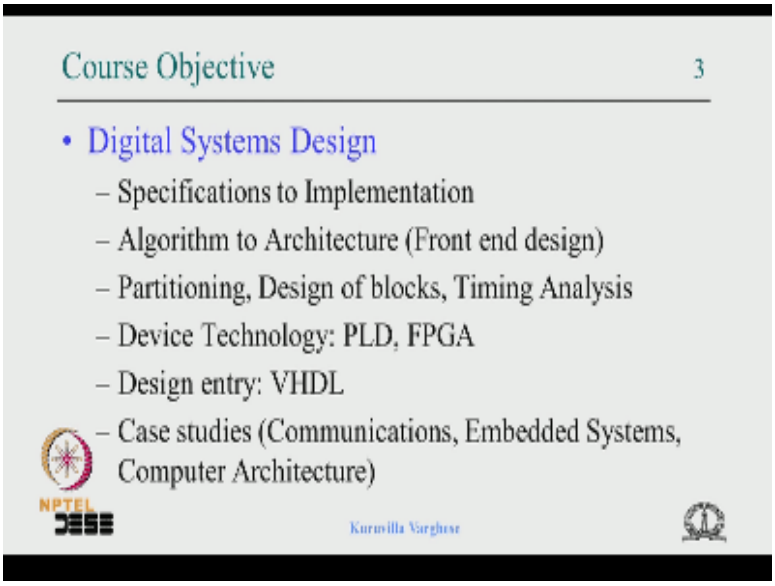

But before we start I wanted to do a small exercise, please take a pen and paper and write what is your idea or expected of this course ok. So basically what you think this course is about what is it contains and why are you learning this course?, maybe some people are taking this course to learn the basics of digital design and learn it thoroughly, maybe somebody is trying to get a job based on this course or could be somebody else has a similar course in the curriculum and want to get a complete the course with some good grades.

So whatever maybe your purpose behind learning this course please write it down. And the last question I want to ask you is what do you think should be taught, so from the like you have some expectation of the course that is why you have come to this course. So what do you think should be the content of this please write it down. The idea is there in the next few slide.

I am going to state the objective of this course what are its major contents, so that your career at the outset what you are going to learn by the end of the course, it is not good if I do not tell you are the beginning and you run to the course all the way till the end and realise this is not something what you wanted to learn, then there to be a problem.

So when you compare what is the content of the course, what is the objective of the course with your expectation or requirement and if there is a serious mismatch you can choose not to attend the course and waste your time, that is the basic idea. So let us move on and let us see what are the content of this course.

**(Refer Slide Time: 03:29)**




So mainly the course objective is digital system design, so the main focus is digital system design by which I mean that somebody gives your specifications of a system, from that specification how do you go about implementing the system ok. So the system could be a small system like counter and arithmetic logic unit or it could be a microprocessor or it could be a system on chip by complex system on chip whatever given specification.

How do you go about implementing this having only the knowledge of the specification and the domain, how do you go about implementing it that is the first objective. When it comes to the next one say very specific objective. Suppose you have an algorithm say you have a signal processing algorithm for a filter how do you design and architecture for that filter and in the balance of digital VLSI many times it is called front end design.

And front end means that you given the specification from the specification you go all the way to the logic design that means the design in terms of gates and flip flops and similar to the front end there is a back end design which takes this gates and flip flops all the way to transistors and the mask required for the chip or the integrated circuits manufacturing ok. So in this course we are mainly concerned with the front end design not the back end design.

So next step is that when you resign as such a complex system you have to partition the system into pieces in blocks and now you have to take each block and design the blocks in detail meeting the specification of each block and at least you have to do timing analysis of each block to be able to walk. There are many other analysis you need to do, but these are the basic level you should be able to make it work in a functional way and it should meet the basic delay requirement.

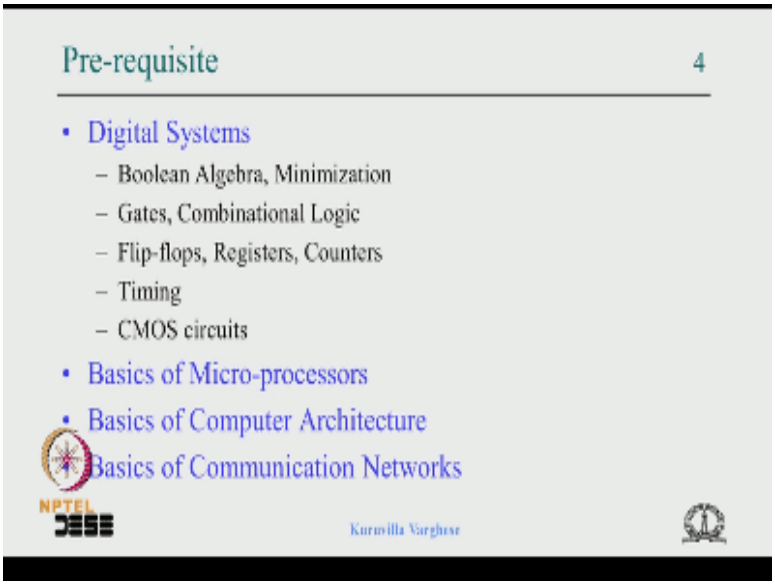
The third or fourth objective is the device technology. For this course we will be using two device technologies, one is called programmable logic device or PLDs. The next one is field programmable gate array or FPGA. So we will be using these devices to implement our design and the fifth point is that for entering the design for designing the blocks or designing the system we will be using a hardware description language called VHDL the we will see what is VHDL later.

So the design entry in this course will be using VHDL. There is no particular reason to pick up this language one could use languages like Verilog but traditional the FPGA vendors used to support VHDL more than Verilog. And it is easy to move from VHDL to Verilog than from Verilog VHDL. So I hope that once have learn VHDL you can easily store to other hardware description languages when required and I must tell that the main focus is still digital system design not the PLDs or FPGA not VHDL.

We are going to learn this but the main focus is still the digital system design, not PLD, FPGA, VHDL. We will learn all these thoroughly but the focus is digital system design and in the course of the stadium we are going to have some few case studies. I will grow examples from communications, embedded systems, computer architecture and all that, these case study.

So I hope you have some basic understanding of these topics will help you to understand the course better and we do not have time for too many case studies. So I hope you have a background in these topics. So that is a course objective and let us move on.

**(Refer Slide Time: 08:25)**




And the next question to ask is what is a prerequisite for this course and/or what are the basics you should know before embarking on learning this course and I really assume that you have some background without with this course cannot be learnt. So let us see what are the backgrounds require. So essentially you would have gone through some digital systems course or digital circuit course in the undergraduate program.

I am not going to cover that part, I really assume that you are through with these particular topic to start with the Boolean Algebra. You should be thorough with the Boolean Algebra, some kind of minimization algorithm at least you must have done used Karnaugh map for minimising and you should have learnt gates, the combinational logic like the encoder, decoder, multiplexer, de-multiplexers, adder, subtractor and things like that.

And the sequences look like flip flops, registers, counters. All this should be known at the functional level and you should know the timing parameters of combinational logic and the sequential logic. At least you should know what are the delay parameters associated with the gates, associated with the flip flop. There after we can build complex timing parameters based on the basic timing parameter.

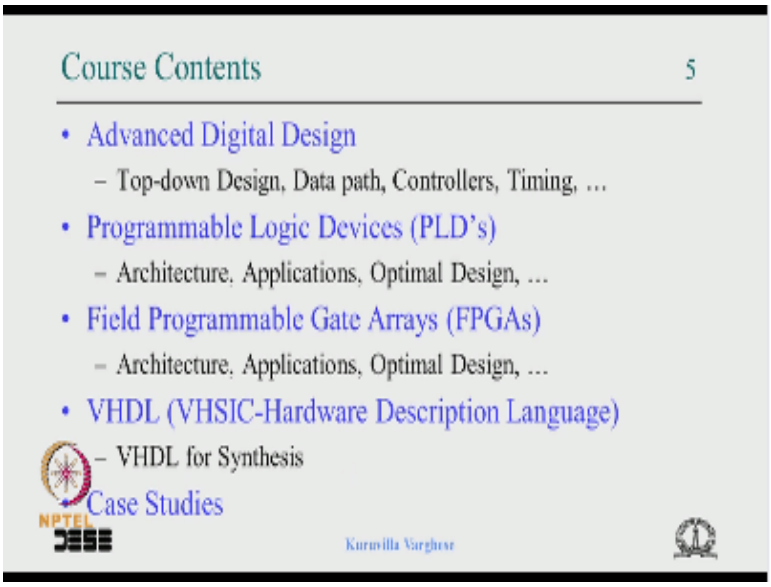
And nowadays you know that all the digital circuits built of CMOS technology or NMOS and PMOS transistors maybe in the undergraduate program you would have learn about some other technologies, but that is not important, the currently what is used CMOS, so if you are not through with CMOS circuit please take some time to learn the basic NMOS, PMOS, the CMOS circuits of various gates and so on ok.

So the next background I would require is some basics of microprocessors because when we try to learn design as an example I will bring in the microprocessor as a case study. So I hope you know some microprocessors like 8085, 8086 or anything you have learned, it could be some microcontrollers like 8051 or risk process as part of your computer architecture it does not matter if you know how the processes are designed, what are the blocks in microprocessors that should suffice.

And the next background I require some basics of computer architecture basically the Arithmetic Logic Unit instructions and so on. So if not learn you can refer to some good books on computer architecture and pick up the basics of the computer architecture. Maybe I am not sure I will if required I will grow some examples from the computer network of the communication networks.

I hope you would have learnt at least one course in communication network I will try to limit the case studies first 2 but if required may be I will site some case studies from this communication networks to, so these are the prerequisite if you are not through please take a break learn this from any text book, any good textbook and come back and learn this course.

**(Refer Slide Time: 12:18)**




So before getting on to review the basics I want to stay the contents of the course, various parts of the course this is essentially 5 parts. The main focus will be on advanced digital designer, I call it advanced digital design because I assume that you have the background, you know the basics of the digital design. In this advanced digital design part you will be mainly concentrating on the top of design or hierarchical design in particular in a serious digital design.

There are two parts one is data path and the other is a controller. And the data path talks about the computation the basic computation, so this is the part of the circuits were all the computation happens and the controller is one which moves or which controls the data movement within the data path and there could be the one controller on multiple controller and definitely after this we should be concerned with the timing and means how to get to meet the timing requirements part of the specification and many other things which when we go ahead with this topic we will see.

The next part is the programmable logic devices or PLDs these are used in a small way for glue logic in digital design and we will see the architecture of the PLDs, the evolution of those architecture, the application of the PLDs, how to optimally design a circuit using the PLD. The next part will be the field programmable gate array is on FBGAs and we will see the architecture of FPGAs, the application of FPS and how to optimally design using FPDAs and so on.

And the fourth part is this particular language hardware description language VHDL and the expansion is VHSIC hardware description language, that means very high speed integrated circuits HDL and I want to emphasize that we will not be learning all about VHDL, our focus will be something on synthesis. That means synthesis means that you write a code, you describe the hardware in a language and use a tool to generate the circuits.

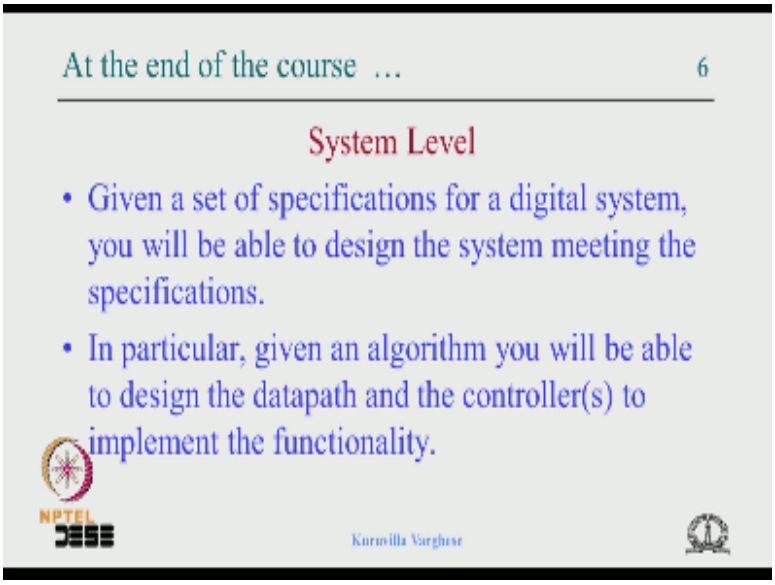
So our focus will be to write synthesizable code that means you write some description of a hardware and give it to a tools synthesis tool it should be able to generate the circuit that you indent to generate ok, that is basic idea of synthesis. Many times people write VHDL code which works very well in simulation, but it does not properly synthesizer or it makes no sense as hardware circuit.

So we are not interested in that this being a hardware design course. So we will be concentrate concentrating on syllabus is aspect of VHDL. Similarly this course contain many case studies, will start with the case studies minor one to illustrate the design methodology design steps and so on and when we come to the end of the course we will take more complex case studies.

So these are the five parts of the course but I am not going to teach it sequentially from say from the top to bottom I will start with advanced digital design, proceed come to a logical end then come back VHDL to cover the basics VHDL and then go back to this advanced digital design then maybe handle these two you know complete the remaining VHDL with case studies and ultimately tie everything together in this part ok.

So that the essential course contents by now I think you should have some clear idea what this course is going to be, but definitely the is a content I cannot talk about the way which I am going to handle this topics which could be little different, we have to wait and see my treatment of the this particular subject.

**(Refer Slide Time: 17:09)**




So next in few slides I would illustrate or I will say what I am expecting you to achieve at the end of the course so or what competencies you will be hoping to develop or I expect you to develop at the end of the course. So at a system level I expect that you will be able to design a digital system given the specification, meeting the essential functional timing requirements or constraints you should be able to do that in particular.

If you have an algorithm you will be able to design an architecture with the data path and controller with all the issues, you know related to it sorted out. So let us look at each module level, so with regard to digital systems itself you will be able to design the datapath and the controller using the high-level combinational and sequential blocks okay. What I mean by higher level combinational.

And sequential blocks is that when you design in your undergraduate courses something like as a full adder you use some gates, when you design a counter you will use some flip flops and few gates and you literally you know start with the truth table work out the Boolean equation minimise it and implements the gates and flops. But when you design a complex system we will not be able to do it at the flip flop and gate level.

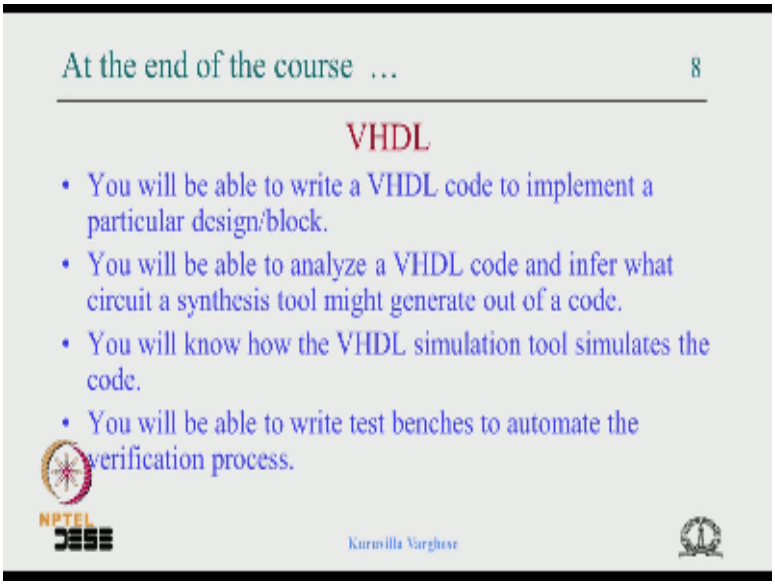
We are going to use the nomination blocks like encoder, decoder, multiplexer, de-multiplexer, adder, subtraction and sequential blocks like registers, counters and so on ok. That is what I mean by higher level combination and sequential blocks and you will be able to solve the functional and timing problems in datapath, many times with your background maybe you are able to solve some functional problem but not the timing problem.

So we will concentrate in this course quite a lot on the timing aspect of the digital design basically at the end of the course I hope you will be able to think timing okay. Many times people are able to think functional aspects but I want you to think the timing aspects to develop that capability and when you design a controller or finite state machine there are various issues functional and timing issues and you should be able to solve.

And the last point is that when you have a digital system you may have different parts of the digital system working with different clocks and or are there may be a part of the system working with one clock and you will receive some manual input from a limited or a keyboard to the system and unless this is handle carefully because an event happening in with regard one clock or 1 manual process reaches the another part unless it is synchronised to the receiving domains clock this would not be registered properly.

So that is called synchronisation aspect and that as we will at least the basics of synchronisation will be taught in the course. I may not have time to teach the synchronization issue in very detail in this course but I will make sure that the essential synchronisation will be handled in the course.

**(Refer Slide Time: 20:52)**




So let us move on to the VHDL, the VHDL part at the end of the course you will have the competency given a blog, given a design you will be able to write a synthesizable VHDL code to implement this block and in the reverse way suppose you have a VHDL code given to

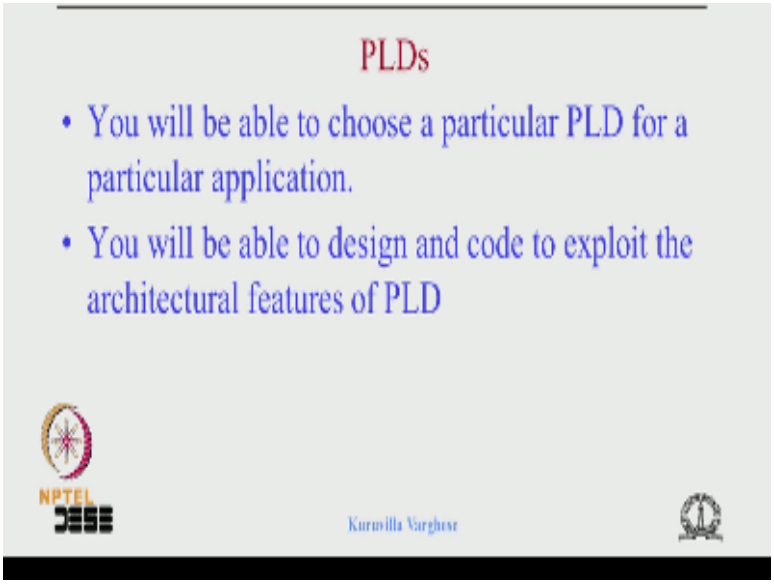
you then you should be able to infer what is a circuit that this particular code implement are given this code to a synthesis tool what probable circuit the synthesis tool will generate.

Many times this is required because you will be working in a project team, you may have to handle the code written by another designer who was worked earlier on the project and unless you are able to infer the functionality of the VHDL code you will not be able to proceed, so this is a competency which is required and we will also learn how the VHDL simulator or simulate tool simulate the code why this is important is that you know that the language is a sequential language.

But the hardware is concurrent suppose you have say five output from some 10 inputs in a combination circuit, any one of the input changes all the five output can change. But in a language when you write five output it has to be returned one after the other but when you simulate the functionality of this 10 input 5 output system everything has to happen suppose at 100 nanosecond one of the input changes after sometime all the output should change.

So we will learn how the simulation to handle the currency and from a sequentially return statement. It is not very complex but it is for our understanding that is not for us to break the head, it is for the simulation tool to do all the what is necessary but a clarity in understanding is important and in a complex you cannot manually verify or manual simulate and verify the circuit we have to automated, so we will in VHDL these are called test benches that you can automate the whole verification process.

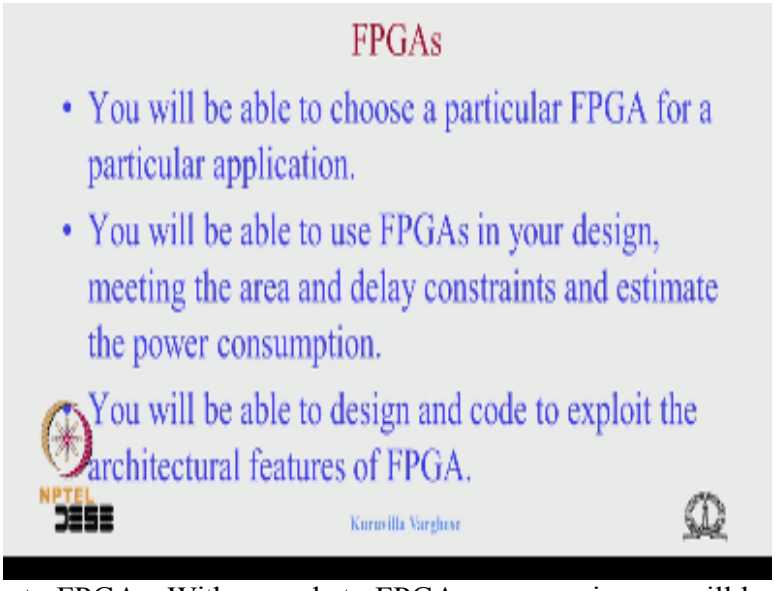
So we will learn how to write test bench in VHDL. So this is the competency I hope you will develop at the end of the course with regard VHDL. So let us move on to the next topic PLD.
**(Refer Slide Time: 23:42)**




So at the end of the course with regard ALDs I hope you will be able to choose a particular PLD for a particular application, this in the case of PLD it may not be very complicated but still you would not at least I choose a PLD which is too small for to accommodate the design or you may not spend too much money in choosing a complex PLD very complex PLD for a smaller application and so on.

And you will be able to design and code to exploit architecture features of PLD you learn the architecture features of PLD and we will learn how to design properly so that those features of PLDs are best used in our design, so that the resources are not wasted or we get the required timing performance. So the competencies you will develop at the end of the course.

**(Refer Slide Time: 24:45)**

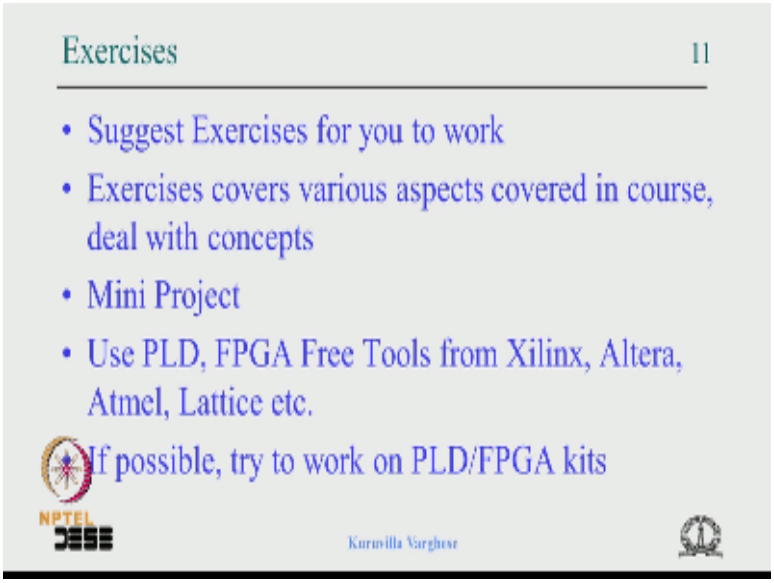



so let us move on to FPGAs. With regards to FPGAs once again you will be able to choose a particular FPGA for a particular application and as in PLDs you will be able to use a

architecture features of FPGA to design and fit a particular design within an FPGA and you will be able to design to meet the area and delay constraints and estimate the power consumption of your implementation within a FPGA.

So there are these are little more elaborate that the PLDs and these devices are complex more complex than the PLDs. So these are the competencies with regard to the digital system design VHDL, FPGAs and PLDs I hope you will be able to develop at the end of the course.

**(Refer Slide Time: 25:49)**




So now I want to in the cause of the lectures I will be suggesting some exercise for you to work of hand and this will cover the various aspects covered in the course and basically deal with the concert, many times students are little worry about the textbook kind of simple exercises but you have to trust me it is very important to do exercise which bring clarity to the concept okay.

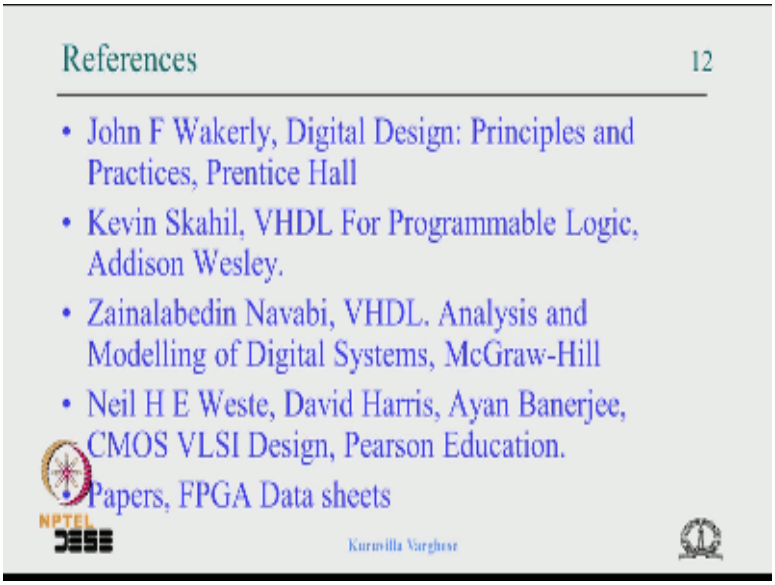
Many times people are in a hurry to design the real life systems but unless concept is clear you will not be able to sort out the problems you encounter, come out with elegant design and creative designs and so on. So it is very important to work with exercise of which bring clarity to the concept than some gimmick, so we can do all gimmicks LEDs or bringing some LCD display with some text and all that.

That is basically some time with the gimmick or it impresses people but in may not enhanced learning or enrich your understanding. So I will be giving exercises or telling you to do some exercises which bring clarity and I will suggest some mini project towards the end of the

course, so that you can try to apply what you have learnt in the subject to sort out some issues in a small case studies.

And for these exercises you can use the free web editions of the PLD and FPGA tools from major FPGA and PLD vendors like Xilinx, Altera, Atmel, Lattice etc. and if possible you can try to get some PLD FPGA kits which are low cost and if I to implement some of the exercise that we discuss on this FPGA kits. So these are the points I want to tell you about the exercise.

**(Refer Slide Time: 27:56)**




So let us move on to the last part of the introduction these are the references. So these are some of the references but I will not be using any of these reference very thoroughly I will be using my own notes, my own slides and my own way of handling the subject but these are very good books, the Wakerly, Digital design it is a very good book, so even for the basics and diesel design you can use the same book.

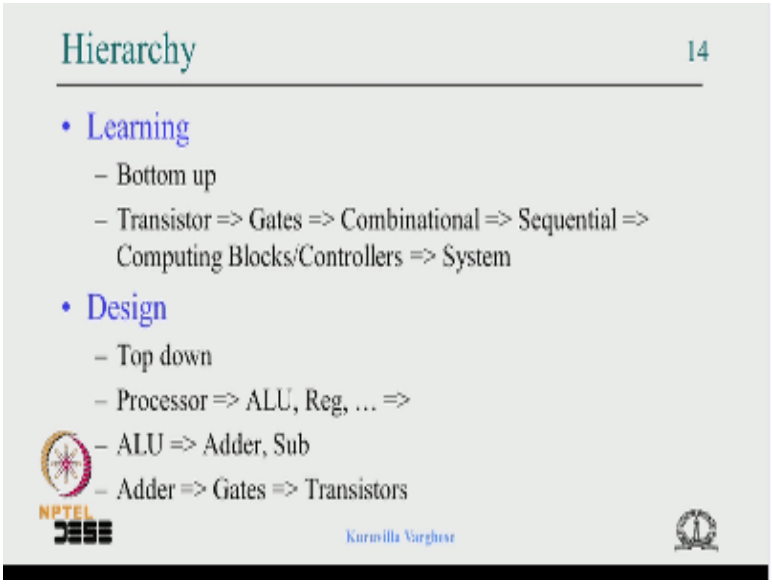
Suppose you are not through with the basic in digital design you can use this book to learn the basics in digital design VHDL for Programmable Logic by Kevin Skahil, this is a good book for VHDL for synthesis, little bit old but very good book. You could use any which is which handles the VHDL for synthesis then the VHDL book by Navabi it is a kind of complete Bible and then for CMOS we can use Weste, Harris, and Banerjees address and Banerjee's book on CMOS VLSI design.

There is a book by Charles Roth it is a very good book on digital design. I am not listed here and I will be referring to various literature in this field and FPGA, PLD data sheets, so you can refer to them also I will say when I whenever I use this reference. So this gives an introduction to the course basically I have told what is the focus of the course?, what is objective of the course?, what is the content of the 5 contents of the course?.

And the competencies I hope you will develop at the end of the course at a system level and in each part that is what I have told and the reference books. So this is the basic introduction to the course. Now we will take some time to review the basics which have already learn which I assume you have but I will run through a given overview of the field.

And run through some basic not thoroughly will be a quick maybe one or two hours of lectures on the basics then we will move on to the real digital system design. So let us look at a digital system design with PLDs and FPGAs and overview.

**(Refer Slide Time: 30:33)**




So I want you to you to have some clarity about learning and design. In learning you always go bottom up that means see normally you start with transistor like CMOS, NMOS transistor, PMOS transistor, CMOS transistor and so on. Then after having learned quite a bit about transistor you learn how to build gates based on this particular transistor see AND gates, and NAND, OR gate, NOR gates, invertors and so on.

Then you will learn how to built combinational circuits based on this gates ok. Then built sequential circuits like the controllers of the data path, registers with combination circuit, all

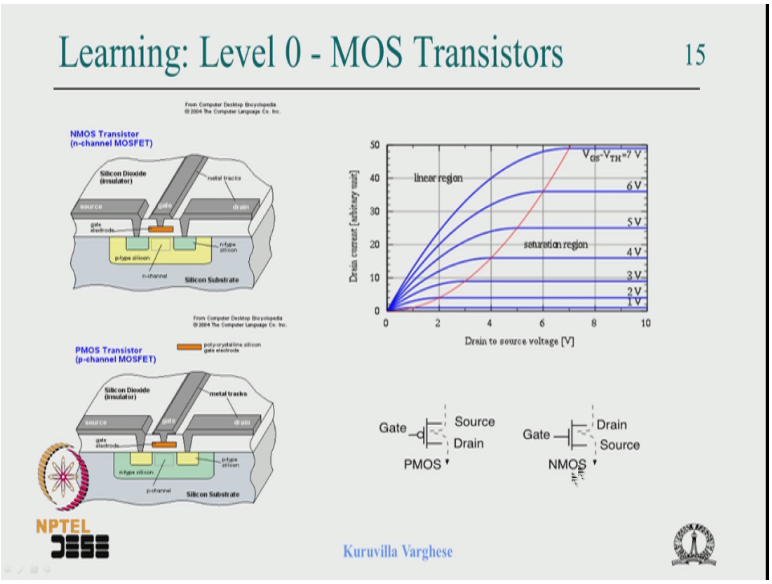
that you will learn. Then ultimately you learn how to interconnect all these in a system. So when you learn you go from the smallest pieces to the complex system, but when you design its opposite process we adopt we go top down, suppose you want to design a microprocessor.

Then you break the processor into pieces like ALU, register, say program counter, stack pointer and so on Ok. Then you take one of the piece say you take ALU and break down into adder, subtractor and so on and then you pick one of the pieces from that and the adder is converted design in terms of XOR gate, and gate and nor gate and ultimately these gates are converted to CMOS transistors. So the hierarchy in design is always top down, so anything complex you should try to do top town okay.

Suppose you take a nap craft you cannot start with somebody designing a wing, somebody designing a few slide and somebody designing an engine and ultimately bring it together with it together without any idea of the aircraft ok, so anything complex maybe it is true with the software suppose you have a complex software you cannot arbitrary design pieces and put it together or you organise a function a conference.

So you cannot you have to have a global view say somebody will handle the program, somebody will handle the stay arrangement, somebody will handle the finance, somebody will handle the travel and so on ok. So anything complex should be handled in a top-down manner but learning should be always going from the basic to the complex ok. So I am going to illustrate that in picture to bring some clarity to it ok.

**(Refer Slide Time: 33:33)**

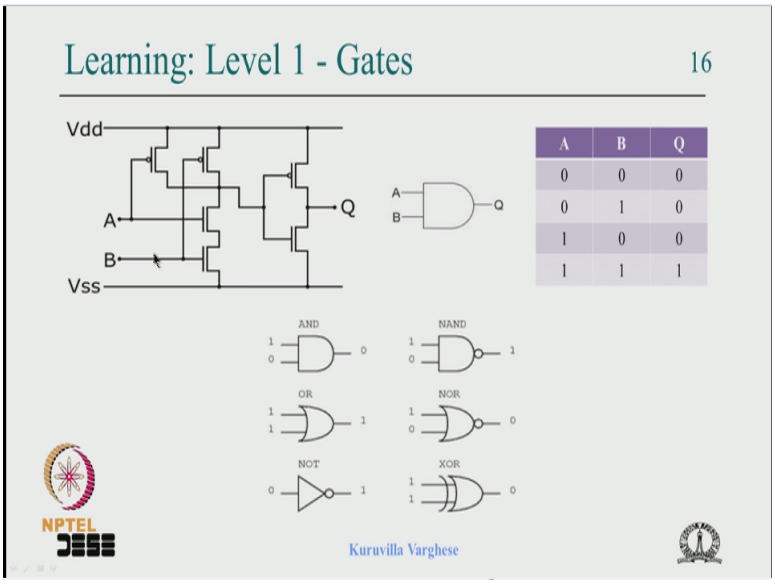


So let us move on to this, so say that the when you learn the basic level like college level 0 you will learn about the MOS transistor say you pick up this NMOS transistor then you know that there is a p-type silicon and I source and drain then poly silicon gate source and drain in the PMOS it is opposite you have the substrate of N type and the source and drain is of P type.

Then you know you learned ideas videos characteristics what are the other character, how the ideas change with regard to videos and for various VGS and so on and you learn various regions you know you have the linear region saturation region cut off and you learn sometime some symbol, so this is a symbol for an NMOS, you have a drain source and the gate and when the gate is at a higher voltage with regard to source it conducts.

And for the P was when the gate is at a higher voltage with respect to source then again it conducts normally it is opposite voltage we apply and we know that and then and NMOS is a good conductor of zero and PMOS is a good conductor of one and having learn this MOS transistor then you move on to the gates.

**(Refer Slide Time: 34:51)**

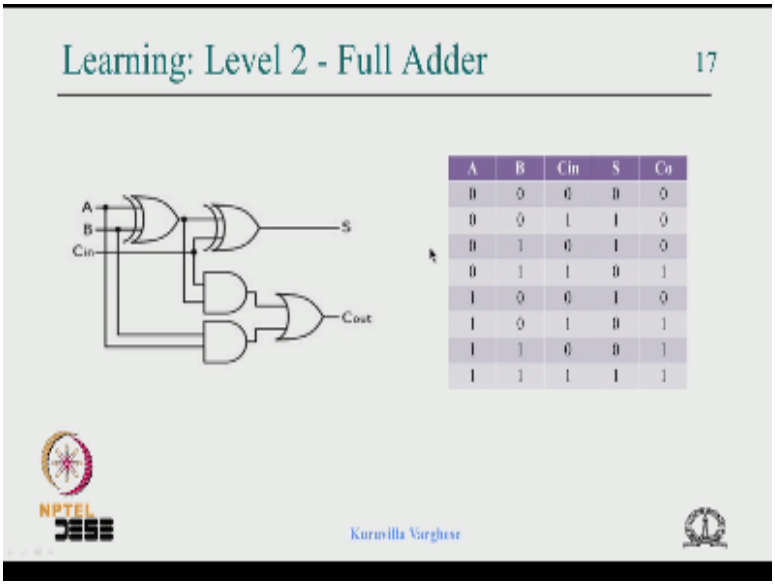



So take the keys of and gate, and gate is designed using PMOS and NMOS transistor, so if you look, you have learn this part is a NAND gate and this part is an invertor, so this is an NAND Gate follow with an inverter so you can see that if A and B are one, this 2 transfer should be off this will be on.

So this essentially connect this point to the ground that which converts it so you both are 1, 1 you get 0. So any of the input is zero then you can see one of the transfer will not be conducting so this path will be off and one of these transfers will be conducting so you will get a high here, so accordingly you will get low there. So and that is the symbol of a gate. Similarly you have like and gate your have NAC or NOR, XOR gate and the inverter.

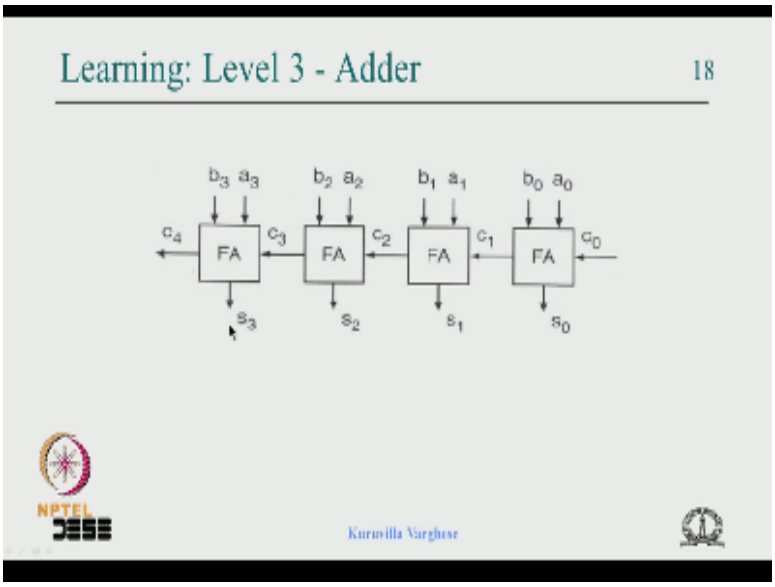
So the level 1 having learn the transistor, you learn about the gates what are the input output relation in terms of the binary values this gate implement.

**(Refer Slide Time: 36:06)**




Now once you know the date you are able to go to the next level, level 2 they take the case of a full adder and full adder has three input a b, 2 bits and carry boot normally from a previous state which gives a sum and carry out which you can use it in the next stage. So this is a full adder is a modular adder slice which can be combined to form bigger adders. So this is the truth table and you know that if any one of the input is 1 sum is 1, any 2 or 1 then the sum is 0 and the carry is 1.

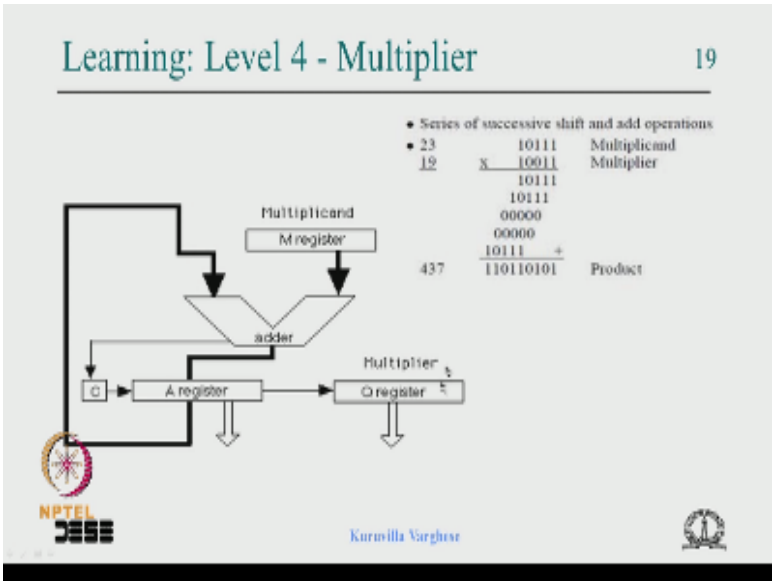
**(Refer Slide Time: 37:10)**




All the 3 are 1 then both are one ok this if studied and if you work out in a minimise then you will end up with this expression. So this is the next level, in the level 2 you go you move from the full adder to say A4 bit ripple adder so using for a platform full address we are building a 4 bit ripple adder this is the A0 B0 the least significant bit of the inputs A3 B3 are the most significant bits of the inputs.

And these are the some this is the carry into the first stage does a carry out of the next stage is connected as a carry input to the next stage and so knowing the full adder we will be able to build a repel adder at the next level.

**(Refer Slide Time: 37:36)**




So we go further maybe we go to a multiplier so this is the this is how the architecture of a multiplayer looks so if you get to look at the multiplication algorithm the paper pencil method saying here we are putting a 5 bit multiplication and a 5 bit multiplier for each bit of

the multiplier you from the partial product and you know that when it comes to the second bit this is the power of you know 2 to raise to 1.

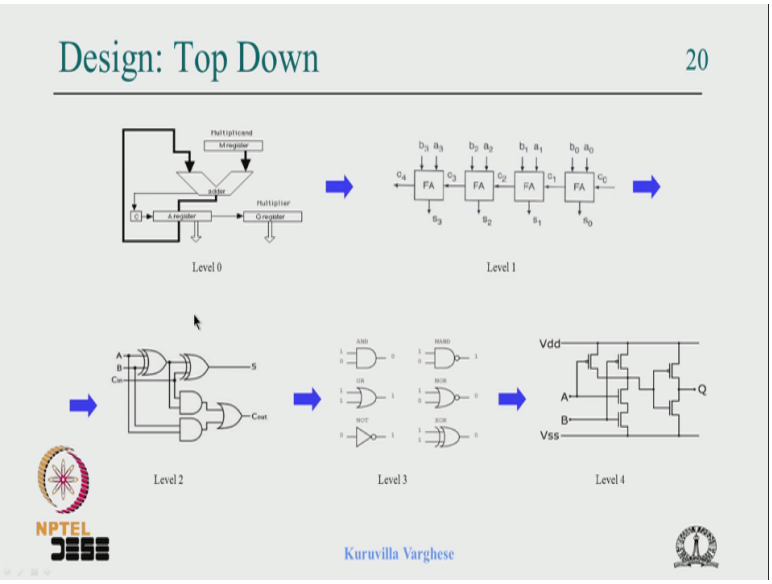
So the partial product has shifted, the bit is 0, the 0 shifted so ultimately you from the 5 partial product you added together you get the product, but then it in was lot of adders in a realistic design to save the area we use a single adder and form the first partial product add to the accumulator which is zero then from the next partial product add it. So here have an accumulator you have a multiplicand, you have a multiplier ok.

So look at the least significant bit of the multiplier if it is one they usually the accumulate is zero you add the multiple there, then you instead of shifting the partial product left you shift the accumulator right and then this the least significant bit of the multiplayer is gone the next bit comes here, if you look at it with is zero you re-circulate, you take this result itself put it back and shift it.

Because it is equal and to adding the zero, instead of adding 0 we take this and put it back and so. If I do this 5 times then you will get the desired result, so which involved a 3 registers and adder and some control which is not shown here. So we can say this is the datapath of a multiplayer which basically used idea of the adder ok, so that is how the learning happens you started with the transistor.

Then move on to the gate and then we have moved on to the full adder and we made the repel adder, then having some idea of the flip flops you are able to build a multiplier or you are able to understand the functioning of a multiply very thoroughly. But when you design so let us look at the design how to design this multiplier you do the opposite.

**(Refer Slide Time: 39:59)**




So knowing the algorithm of the multiplier you design and architecture for the datapath of the multiplier consisting of 3 registers and an adder. Now we have to design this adder and the registers in a detail way. So let us speak for example the adder say assumed that is the four bit adder for bit multiplier then we will design this 4 bit adder with 4 full address cascade. So the adder is broken down into 4 full adders.

Now you take full adder design using gates, using XOR gates and AND and OR gates, so this is a majority logic any two or more than one input is 1 then the carry out will be one. Now the moment you do that we know all the gates and then go to transistors and ultimately the chip mask are built from the transistor layout, that not shown in the picture, so up to hear from here to here to here is the front design that from the spec the multiplier algorithm we have come to the gates and flip flops.

In this case there are no flip flops shown, but you know essentially a register is composed of flip flops suppose a 4 bit register is nothing but 4 flip flops in parallel and so this is very easily designed as 4 flip flops in parallel. So this is how the design process in a top-down approach and of course you should have the domain knowledge to design, we should know all the pieces it is very important that you are through with all the building blocks.

So that you will be able to design it thoroughly and you should be able to sort out the problems as you know you encountered as you design ok. So this was the hierarchy of learning and hierarchy of design.

**(Refer Slide Time: 42:08)**


So let us move on so let us ask what are the major constituents of a design that means suppose somebody tells you to design a multiplier what should we focus on ok and many times students the smart students say it should be low power, it should be low area, it should work at 1 gigahertz within the high clock frequency and so on ok, but think for a moment you design a multiplier and you say it works at 1 gigahertz but given some input.

Say you give to the multiplier 7 and 5 and it gives an output like 42 but you said it was that one gigahertz it is of no use ok or you say it work it consume hardly any power only microwatt but the answer is wrong then is of no use so that tells you that are the primary part which we have to focus on is the function of the logic ok. So when you design the first part or the first constituent the first focus should be on function and logic at the power area timing everything comes later ok.

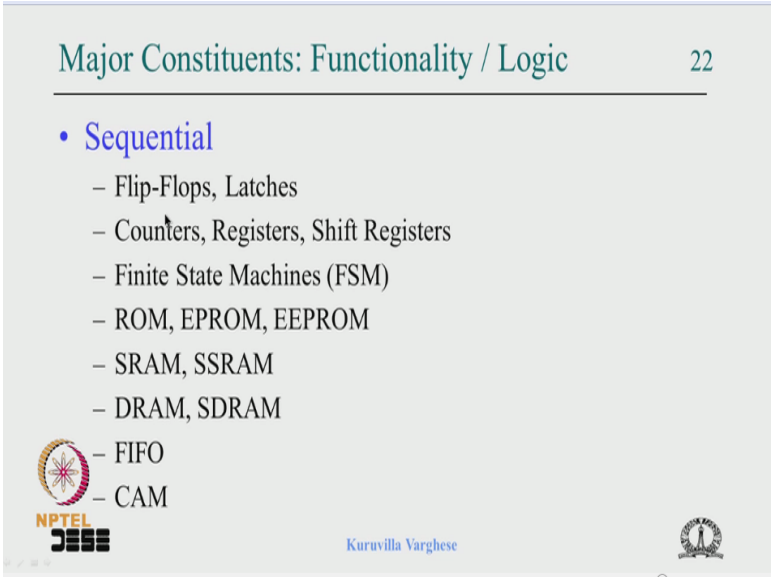
The first will be functional logic and you are lucky in this way because you have learnt all the building blocks in an basic course and I will list the two parts combination logic and sequential logic. So the combination logic you would have learned all these you know you would have learn Boolean Algebra, you would have learn minimization sum algorithm like (()) (43:56) and things like that you would learn various functions like AND, NAND or NOR, XOR.

Inverter the gates implementing this functions something called encoders and decoders multiplexers and de-multiplexers, Priory circuit, comparators, priority encoder, open drain output, tri state output, Schmitt-trigger inputs, Adder, subtractor, increment, decrement runs

on. So we are going to use all of this ok some may not be explicitly like we may not do any minimization most of the time this is done by the tool in high level design.

But an understanding of this is very much required to grasp the concept and bring clarity into the whole game and so all these are very important what are the is composed of, all these are design and so on.

**(Refer Slide Time: 44:52)**




So let us move on to the sequential logic in this when is when we come to the sequential circuit the basic building block is the flip flops various type of flip flops like you would have learn D flip flop, SR flip flops, JK flip flops, T flip flops and latches and there is a difference between latch and flip flops. Latches are by definition transparent when the clock is I the input output follows input and when the clock is inactive the last input is latched onto the output.

In a flip flop normally works on the clock at when a clock active clock at come the input is transferred to the output provided some input meet sometime in requirement we will see that and you would have learnt some kind of counters like repel counter, synchronous counter, repel counter is of no great interest to ask, so forget about it, will be talking about synchronous counters, various registers, parallel resistors and shift registers and so on.

And the finite state machines I am not sure whether your have learn this in the basic course but we will I assume that you may not have learnt and we will put some time developing the concept and the design of finite state machine and various memories are important various

kinds of Rom, read only memory, erasable programmable read only memories are electrically erasable PROM.

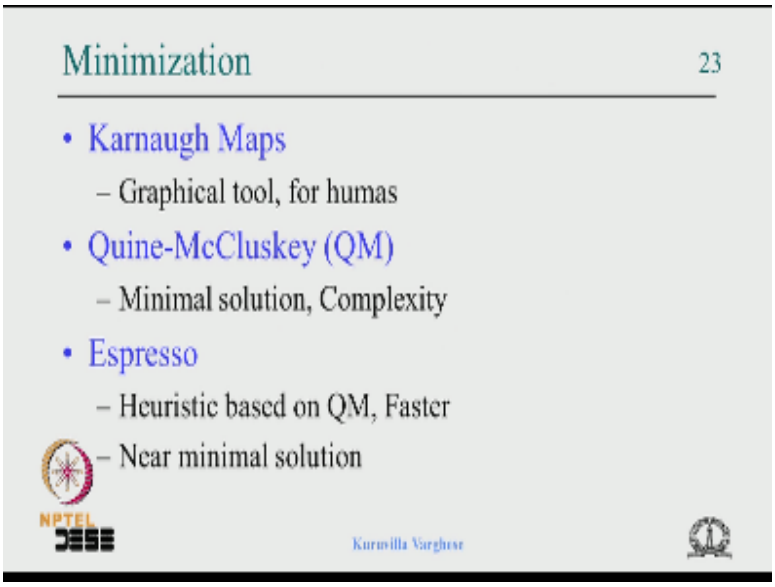
Similarly static RAM these are fast memory which is used in a computer system at the level 1 as cashes and level 2 cases and all that synchronous RAM nowadays everything is synchronous with the clock. DRAMS is the secondary level of storage in a computer system. These of the large capacity but it is not have asked SRAM and FIFO which is first in and first out memory that means a 2 ports.

One port you write and one port you read many times address in is implicit that means you don't mess with address for the first data you write will go to the first location and the second you write go to the second location and when secured read output reach starting with the first and the read point is in right pointers are increment depending on to read and what has to take care that one does not overtake the other and so on ok.

That there is no overflow or underflow similarly CAM normally it is defined as an opposite of a memory, here you have data stored in a memory can you look for you search for some content that means you provide data it says the location where that particular data is stored soap in a normal memory you provide the address you get the data but here you know your data then you get the address which is used in the cache memory is and also search algorithm lookup algorithm used CAM for it implementation.

So this is the major constituent of a digital design that is combinational logic and sequential circuits and if you care to look at a data sheet maybe of a gate or PLD or microprocessor fasting in a data sheet stated is the function of that integrated circuits, so that shows the priority the function as over other factors. So we have little time left, let us move on to the sum of the blocks we have discussed I am not going to deal with all the blocks. Let us pick up some of this combination of blocks and just revise so that it brush up your memory.

**(Refer Slide Time: 48:57)**




So let us move on so with regard to minimization all of you must have learn Karnaugh maps ok. This is a very systematic method of minimization from the midterms to minimal product comes and this is a graphical tool, it is for humans to work out it is not for computer work out, the equal computer algorithm is called Quine McCluskey. As in Karnaugh map it provides minimal solution, but the complexity is very high.

It has exponential complexity because it start with the window suppose you have 5 variable problem like a b c d e and you already given expression to Quine McCluskey like AB bar it you know that it expand in terms of the 5 variable. So already minimise expression is taken back to get the absolute minimal or optimal that the product that idea of Quine McCluskey, but you know that given an input that you have to raise to end min them.

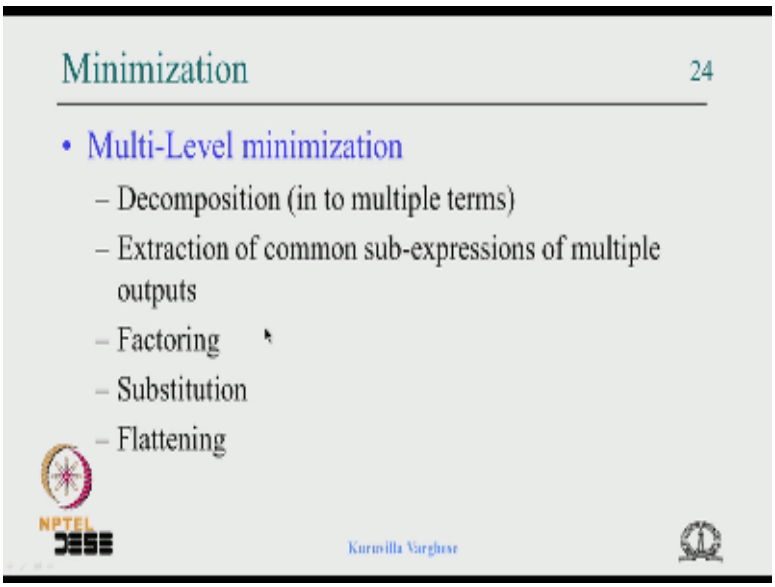
So the complexity is exponential and is very hard to computer using the Quine McCluskey, so mostly the tools used heuristic algorithm called Espresso. This is based on Quine McCluskey but is faster. So that means it as a product term already which is simplified is not going to expand all the way to the midterm and start work, reworking back. So it uses some kind of heuristic method or shortcuts, it is not going to produce an exact optimal or minimal solution as in Quine McCluskey.

But will give a near minimal solution and we do not take because many times we have not worried about the area so much nowadays like it does not matter instead of adding up with 25 product on you might end up with 27 or 30 product on it should be ok in the present technology, so we will these tools will use Espresso kind of heuristic based the tools for

minimization algorithm for minimization and the next thing it should realise a little bit about the real life is that.

All these applies many times to the two level implementation that you talk about and/or or or/and or sum of product and product of sum which have learn in your the undergraduate program of the basic course.

**(Refer Slide Time: 51:45)**



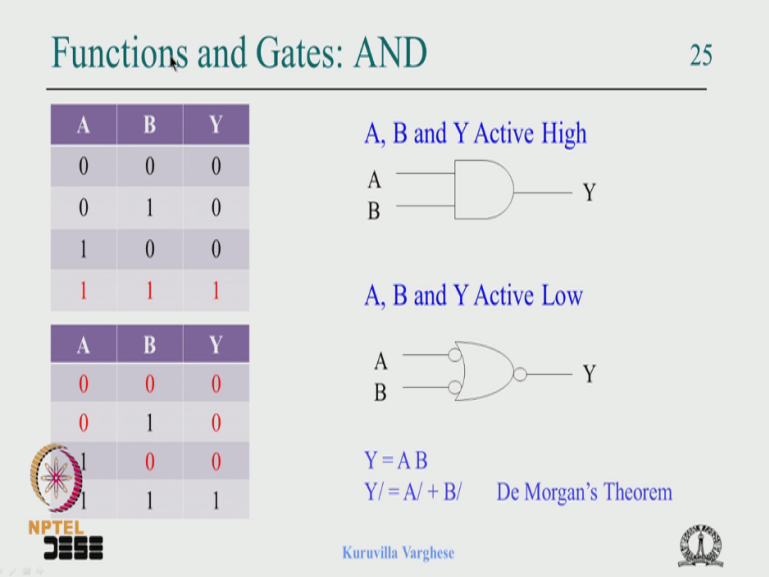

But in real life when you come to the FPGAs or ASIC you do not stick to the two level implementation because it not possible, so you will have to implement a particular circuit in multiple levels of logic. So you will have to apply some decomposition of a Boolean expression or a circuit in multiple terms, so that multiple levels can be implemented and suppose when you have a multiple outputs.

Suppose you have 10 inputs and 5 outputs then you will have to for a minimal expression in term for the multiple output will have to find the common sub-expression like maximal common sub-expression of all the multiple output. So you need algorithm to find the common sub-expression across the multiple output which minimises the area and many times these involved the steps like the factoring, substitution, flattening etc. for the multi level implementation of the digital logic.

So the minimization is more complex than you have learn, it uses heuristic algorithm and it concentrate on the multiple output minimization and multi level minimization and so on. So

these are complex those were interested can look at the synthesis of digital circuit, these are that the subjects which looks at this minimization algorithm.

**(Refer Slide Time: 53:20)**




So let's move on so let us look at this part functions and gates. I just want to tell you that you would have most of the time you confuse the gates with the function ok. So you take an AND gate then this AND you take it to implement the AND function, but many times the AND gate can implement many other function than the AND gate and function. So let us look at the truth table of AND gate say like in an AND gate when the both inputs are one that output is one.

That means A and B are 1, the output is one. In the case if AND gate so here we are treating the inputs are active I and output is active I, so then it AND gate implements and AND function but if you treat the AND gate inputs and outputs active low look at the truth table if A B are active low look at the truth table if any of the input is active any one of the input is active than the output active which is nothing but an over function.

So if for the same AND gate if you treat inputs and outputs are active low then in implement AND OR function so that is shown as an OR gate with showing the active levels. So AND gate can implement or function when the inputs and outputs active low and if for a smart student it can easily you can easily know that this is nothing but applying the De Morgan theorem so Y is a b.

We take Y bar is nothing but AB whole bar which is nothing but A bar or B bar, so I am showing bar by a slash because I can use a text for that so that is shown here. So AND gate can implement AND function OR function and which is basically applying the de-morgan's theorem or building the De Morgan theorem in the concept ok. So in the lecture we will continue with this.

So today in the second part of the course handled basically we have looked at the major constituent of design. There is function basically what all you have learnt in the basic undergraduate course the combinational logic, the sequential logic all that is important, then we have looked the functions and gates before that we have looked at the minimization the multi level multi output minimization and algorithm used for that.

So in the next lecture you will continue with this function may be in next 1 or 2 lecture I will be able to cover the complete the basics needed, and through the basic then I will also give certain the current state of the art in this particular field to give you an idea over there after we will start with the main focus the real digital design showing the example how to go about designing and the concept.

I hope you enjoy this session, please go back and work on the basics have covered. Those are not learn the basics please go back to the reference book learn the first few chapters covering the combination sequential logic, learn bit about microprocessor, learn bit about the computer architecture, so that when we move ahead you are in sink I wish you all the best and thank you.