

Design Verification and Test of Digital VLSI Designs
Dr. Santhosh Biswas
Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Model - 1
Introduction
Lecture - 1
Introduction to Digital VLSI Design Flow

(Refer Slide Time: 00:23)



So... Good morning everybody and welcome to NPTEL video course on design verification and test of digital VLSI circuits. So, this is the first lecture of the course in which we will try to find out what is the, which is the goal of our course or what we are trying to achieve out of this course? So, I mean if you just look at the key terms of this course that is design verification and test of digital VLSI circuits. So, what is the circuit? So, if you just look at the basic terms. Circuit means it is a device or an objective transforms electrical signals you give one form of signals you will get other forms of signals. In all could and could format the term circuit means it is nothing but it is a closed form where electrons can travel.

Now, these are all undergraduate staff. Now, what is the digital circuit? So, in digital circuits all inputs and outputs are different voltage levels. We do not have any kind of continuous time current or continuous time voltages. If your circuit has continuous time kind and currents and continuous time voltages it becomes analog circuit. In case of

digital circuits we have discrete different voltage levels and in most of the cases we have 2 voltage levels that is 0 and 1. I mean for 0 it is around near about 0 volts. And in case of logic 1 which say it is 5 volts, in case of some technologies 3.3 volts in some technologies; that is we have discrete voltage levels. We all know that then it is called digital circuits.

So, in this course we are going to see about design; that is you are given a specification your designer circuit like if I say I want to built one with the other. So, you know that sum is a dot b and carries a or d kind of a thing sorry sum is a or b and carries a dot b. So, which means state for that is design you make a circuit. Then what do you mean by verification? That is you give some input to your circuit and then to check whether it performs as desired. Whereas, in case of adder you know that sum is basically a plus b and carries a generated when both of them are 1; that is a dot b will generate your carry.

So, if you have just give all input as 0 0 0 1 1 0 1 1. So, you can say that verify your circuit then finally, it is giving the correct answer sum or not. And then test means what in this case? You designed the circuit then you fabricated because you need to generate a circuit out of it hardware circuit. So, that is actually in this case is called VLSI because this technology is very large scale integration.

So, when you are fabricating a circuit in very scale technologies and finally get the hardware circuit out of it. And in case of VLSI we can put millions of transistors. So, I mean simple adder is a very, very minute example of a circuit. In terms of the VLSI you can have a multiple code processors as well as code quotes or even 128 code processors, you have embedded stop running on your mobile processors, you are have so many hardware processing units in your cars; all those things which you are all currently all listening about.

So, our circuit in these days are highly integrated, large number of transistors millions of gates are integrated in one chip that is actually called VLSI. So, VLSI digital circuit what do you mean? It means all digital circuits when you have integrated circuits millions of transistors do or meet our specifications; and in this course we are going to see design verification and test of those circuits. In case of testing once the circuit is fabricated. You give some inputs, hardware inputs from the signals from logical and electrical signals see if everything is fine or not.

So, in this course basically you see you can think about this design verification and test into paradigms. You can think that given I specification how I do a design? As simple adder I have told you and how to verify it is fine or not that you try with all possible inputs and see or not. So, once it is verified as then it fabricated and get the final chip and connect electrical signals and apply signal sense and see whether it is fine or not. So, that is actually called design verification and test in one paradigm. But in another paradigm if you think of the same problem as I told you adder is the very, very simple example or you can say minute example of a circuit I mean very small circuit example.

But all practical VLSI circuit are very, very large or very, very complex in nature like a processor like you can say graphics processor and so forth; which are doing large and very complex applications for us like Pentium processor or processor; which are using in videogame applications processor which is processing your mobile applications. So, they are very complex in nature. So, if I say just do everything manually that is from the specification design circuit like in case of adder for all and the addition some you do the or gate and for the carry you generate or gate.

These are the simple design you can do it by a dot b means if you have and gate a or b means you have an and gate. So, you can do it manually so you can test by applying patterns. And finally then the cheapest arrive you put in you all done you are all undergraduate digital lab. You put the chips on a bread board connect to the voltage supply and see the results in the lead. If you have very, very complex circuits; I mean hundreds of inputs and millions of gates and having hundreds of outputs. Then this cannot be done manually extremely difficult to do all these tough manually.

So, you required some kind of cad tools which will help you for which are automate the design for you give the specification. Then it will help you to generate the output and the final circuit. And then you find out mathematical you can try to prove that whatever I have designed is mathematically obeying your specifications required. And finally there will be some kind of algorithms which will tell you what patterns to apply in the hardware level? So, that you can know that your circuit is operating fine. In other words the whole design flow starting from design verification; and test flow can be automated or helped using some automated tools. And if you are go for VLSI designs it is mandatory could and could it is almost impossible to do without the help of tools.

So, there are 2 ways of looking at the problem. So, in this course we have to taking the second approach. So, in this course we will not teach you specifically how to design and verify or how to test the circuit? Whether what you will do this is actually you called cad for designs VLSI kind of course. In this course where we will try to teach you that given a specification how you can write automatic tools or how can you develop algorithms? Which can automatically designed the circuit for you, automatically verify the circuit for you, given a circuit specification and input specification you give a circuit description.

We will give the input specification automatically some algorithms have to be developed and automatic tools will be developed. You can tell that whether the circuit meeting the specification or not, similarly for kind of testing. You have to tell which are the test to be applied? If you try to apply all the almost test patterns testing millions of years. Because in case of circuit has n number of inputs all possible combinations this true forever. So, everything is exponential in this world of VLSI.

So, what you have to do? You have to tell how automated tools you which is the minimum numbers of test patterns to be applied, to get a desired quality of products or desired quality of production. So, we require cad tools to help in all aspect of design tools in VLSI design. So, these courses will not again I am emphasising the output or outcome after do this course. Our goal is that we will be apply to develop or you will have learned how automatically tools have developed or how algorithms are developed which can automate the design verification; and test flow in digital design VLSI circuits.

So, in the first lecture what we have going to see we gives an introduction to digital VLSI flow. That is what the steps that are required to go for digital VLSI design? And also tell you why the process are bit complex and what how tools and what can be developed? Once you know that these are the steps are VLSI designs of digital designs. You can think that what are the tools required or what the algorithms can be empowered or algorithms can be developed which can automate the steps? So, we will take a very small example of this lecture and take you through the different steps of digital VLSI designs. And the course will actually have 3 models one is for designing and one is for verification and one is for test. So, in that what we will do? You will go through all the different algorithms to meet different part of this 3 basic designs flow having 3 major parts design verification and test.

So, in the following lecture today first lecture is the introduction lecture; introductory lecture in which will tell you the different steps in the flow. In the following using a very simple example. In the following example module in the course you will be learning about the automated algorithms which will automate the different parts. Now, slowly moving into the digital VLSI flow.

(Refer Slide Time: 08:24)



So, now first see the introduction what is actually VLSI design? Now, I mean we have see all written programs when you have given a functional requirements like you have to add 10 numbers or you have to do multiplications, you have to do array addition or something like that; finally, arranging number in ascending order and so for. So, in all this cases you have very much well equipped write c codes for that. So, you have asked to write a counter for c codes. So, you can write I is equals to 1 to n i is equals to i plus one something like that. And you can implement up count or down count also. We have also learn the basic VLSI I mean digital design about digital VLSI design which is second year undergraduate course.

So, we have already seen that how the same thing can be done in case of hardware? Then you can connect some d flops and you can go for final present state minimisation which is the present state is 0; next state is 1. Then again you have first state as 1 and the next state is 0 and so for. So, you can use map or you can (()) method to minimize the circuit then we implement what you call circuit. Actually, what we call same like the hardware

which is the hardware. It was very similar nature as the software like the c programme.

So, whatever you have learned in the undergraduate VLSI undergraduate digital design; it is nothing but actually a miniature version or miniature part or you can say the small subpart or what is VLSI design. So, like for example, if you think of adder or multiplier or very simple like counter they can also be called VLSI designs. But only actually it is not a very large scale large scale integration. VLSI actually stands for very large scale integration. And in digital design actually we have seen the miniature versions or miniature part or small scale integration level stuff we have seen. That is the basic philosophy also remains same; that we have some specifications that is like accounting and number adding n numbers or like multiplying 2 vectors something like that or arranging numbers in ascending order and so forth.

So, flexibilities are remains more or less the same. But in case of hardware one in case of c language or Pascal or visual basic or whatever you used they are all (()) implementations. Now, the philosophy of VLSI or hardware is the same as the algorithms or the same specifications can also be implementing using a hardware. So, if you are using hardware then it becomes a dedicated application and it runs much faster now.

So, all these discussions see in details all these terms like how are specific applications, how you are making a application specific hardware? So, it runs faster and all these like for example, if you view a c code it works or it runs in your process. So, why do you all again require make a hardware block for that? Because you are microprocessor in your computer can do almost everything.

Then, why do you require to make some specific hardware to do a specific part of input? Processes are that is why called general purpose processes. That is whatever any specification you can give it will implement or it will have the implementation version for that or you can write it in any high level programming languages. Then why do you require hardware version for that? Actually hardware version if you develop a counter will be counter it will not do anything else. So, it actually application specific so it is a application specific it will consume much less power it will be much fastener and so forth.

So, if you require only up count or down count or add some numbers or some things or

there is a special applications which you want to put in your mobile phone, in your laptop or something or some other of devices. Then why do you want to put a general purpose processor? Because general purpose processes are expensive it takes more power and that is having other disadvantage.

So, if you have some specific objective in mind or specific application in mind, specifications in mind. It is always better than you go about designing a very specific or application specific hardware. So, that is the advantage of application specific hardware throughout the course will have much more detail discussion. I will explain why application specific hardware is more reliable than I mean more advantageous compared to general purpose processor for many of the applications?


So, that is what the very basic ideas that why do you want to go for hardware designer many algorithm? Hardware version manual algorithms are important because sometimes we do not require the general purpose capability of a processor. A person can learn any application we just have to mention the software version in a program and it will view it for you. Many case like for example, in digital camera we just want to do VLSI process. You do not want to do called array multiplication or maybe you do not want to do what to called some kind of printing purpose; and all these things may not be required in camera. If you just to equate to do digital processing taking the image from the lasers source so forth. So, if I connect the general purpose computer to the camera it is bulky heavy and power consuming.

So, we have to develop a very specific hardware which can do this only the required functionality for you. So, for different different specifications, we have to develop different aspects. These are application specific integrated circuits or application processor which will actually do only a sub part of the job you as much as lower cost as much lower power. So, these are the modification of digital I mean modification of VLSI designs whatever algorithms; we have we did not use of general purpose processes or whether you can make a simple hardware version for that and implement it as separate an sensor market.

Now, what is happen? In the last 10 years or last 2 decades the quality of integration that is if you look around 30 to 40 years back with a single IC you could hardly manufacture 10 or 100 gates in the circuits. In this case very large scale integration is possible or very

ultra large scale integration is possible because the advances in fabrication. We can put millions and millions of transistors in 1 chip. That is why our Pentium chips have millions and millions of transistors in 1 chip. And it can be done because of the highly sophisticated fabrication technology that is available.

(Refer Slide Time: 13:49)



The slide is titled "Introduction" in red text. The main text is in black and discusses the functionality of electronic equipments and gadgets, the reasons for their miniaturization, and the classification of integration scales. A red circle highlights the word "equipment" in the first paragraph. A red underline is under the phrase "single Integrated Circuit (IC) or chip" in the second paragraph. The NPTEL logo is at the bottom left.


Introduction

The functionality of electronic equipments and gadgets has achieved a phenomenal while their physical sizes and weights have come down drastically. The major reason is due to the rapid advances in integration technologies, which enables fabrication of millions of transistors in a single Integrated Circuit (IC) or chip.

IC (used interchangeably with "chip" in this course) is a device having multiple transistors with interconnects manufactured on a single silicon substrate.

Integration with a complexity of 10's of transistors is called Small Scale Integration, with 100's is Medium Scale Integration (MSI), with 1000's is Large Scale Integration (LSI), with 10,000 it is Very Large Scale Integration (VLSI)

Systems of systems can be implemented in a VLSI IC. However, with this rise in functionality of VLSI ICs, design problem has become huge and complex.

 NPTEL

So, for now we have achieved the functionality of the all other electronic equipments and gathers have increased phenomenally; while their weights sizes everything have come down. This is because of the fabrication technology allows millions of transistors in a single integrated chip which this possible. So, this integration facility has made us enables us to put millions of transitions or you can say system of systems; in a single IC which is called system of chips or networks on chips. In which you can put large number of systems in a single chips and which can be actually a system in itself.

So, that is what actually helped us. So, that almost all algorithms which is initially possible in general purpose processor; and now can we fabricated the hardware. Then they are sold out as specific processors. So, I mean if you say that is why told you slowly we are move to the level say 30 to 40 years back you can think only tens of transistors able to fabricate chips. So, it is called small scale integration.

(Refer Slide Time: 14:17)


Introduction

The functionality of electronics equipments and gadgets has achieved a phenomenal while their physical sizes and weights have come down drastically. The major reason is due to the rapid advances in integration technologies, which enables fabrication of millions of transistors in a single Integrated Circuit (IC) or chip.

IC (used interchangeably with "chip" in this course) is a device having multiple transistors with interconnects manufactured on a single silicon substrate.

Integration with a complexity of 10's of transistors is called Small Scale Integration, with 100's is Medium Scale Integration (MSI), with 1000's is Large Scale Integration (LSI), with 10,000 it is Very Large Scale Integration (VLSI)

Systems of systems can be implemented in a VLSI IC. However, with this rise in functionality of VLSI ICs, design problem has become huge and complex.



Then actually we have gone for hundreds of gates in one chip; we call it as medium scale integration. Now, about thousands of chips, hundred thousands of transitions you have put in one chip you call it as very large scale integration. And above tens of hundreds of thousands or lakhs of transistors if you can put in a one chip it is called as very large scale integration.

And, what is the new point also termed which is ultra large scale integration in which you place millions of chips, millions of transitions in one chip. And you get what you called as system of or system on chips or network on chips or the sophisticated version multi code chips or all these things are coming up. You have ultra large scale integration but generally you call the term very large scale integration for both ultra large scale and very large scale.

Because of that VLSI I mean this as more accepted term. So, that is what we have seen that because the VLSI technology that is put in multiple or millions and millions of transistors in one chip have enables us to fabricate. System of chips or system of systems or systems on electronic chips making us capable of implementing it on any kind of software as hardware function. And when you can make a hardware version of software it will be faster low power. You do not require a general purpose computer and there are many other advantages which will slowly show in this.

(Refer Slide Time: 15:46)

Introduction

- To address this complexity issue, after the design specifications are complete almost all the other steps are automated using CAD tools.
 - However, even designs automated using CAD tools may have bugs.
- Also, due to extremely large size of the design space it is not possible to verify correctness of the design under all possible situations.
 - So techniques are required that can verify, without exercising exhaustive input-output combinations, that the design meets all the input specifications; this technique is called formal verification.
- In VLSI designs millions of transistors are packed into a single chip. This leads to manufacturing defects and all the chips need to be physically tested by giving input signals from a pattern generator and comparing responses using a logic analyzer; this process is called Testing.

So, in the process of manufacturing a VLSI IC there are three broad steps: **DESIGN-VERIFICATION-TEST.**

So, what next we are going to see this I mean as I already told you. So, that I mean as this complexity issue that is VLSI circuit as I told you we are not fabricating around tens of thousands of transistors in one chip. So, if you are designing you are making some designs it is impossible for a person to do that manually. How can you where to put that hundred transistors? What we will be the output or how can you get input of another block and so forth? That is not even in case it is impossible to think that manually.

So, you have a large amount of cad tools which will actually do that for you. So, we have developed this I mean VLSI industry will have actually 2 trends I mean 2 basic disciplines. Broadly 2 broad sense disciplines one is the design tool. We will take this specification tool to design and see how it is fabricated, how it can be tested and how it can be sold to the market and all?

Other parts of the people are software people who generate or who develop cad tools? Which can allow you to go for a very good VLSI design because if you see there is millions of transitions possible in a chips there complexity is very very high. So, it is impossible for a person to do a manually for find out for a given applications? How will be this level diagram will be how it will be made out in chip? So, that we can fabricated and all...

So, one set of people work for cad and that is called VLSI cad for designing cad tools for VLSI set of discipline; we very well learned to develop cad tools. And another set of

discipline actually we take care of in a given a digital specification. So, for any specification they want to develop what looks as a very good kind of a I mean very good optimize design and fabricated it and send it to the market.

So, in this course mainly we are looking into the cad aspect. How can we develop being this course of this computer science engineering department; of course mainly focused to computer science people. Our main goal will not be in the design part may be it will ideal to develop good cad tools. So, that they can aid the designer. So, that they can get very optimize or they can be able to generate optimal designs.

So, that is what becomes the complexity automated cad tools are very much required. So, this is actually called I mean designing part. So, when you say the design verification and test in this course it is the design does not directly imply that we are going to design some VLSI chips given a specification. Our idea here is that given a specification how can you write good algorithms or good cad tools? So, that you can automatically go for designer or which can help the designer to go about the making the designs. So, it is actually cad for design cad for design tool. It is the compute rated tool for design case of VLSI design. In design case you are not going to design any circuit, but you have to develop cad tool which can help the designer to design the circuit.

So, again what is the idea is that say for example, we will with the tools and all some designer is designing the circuit. But again as you thinking it as so complex always some error might coming into the picture like the cad tools may have errors or designers might have made some error. And actually something usually you should point out that VLSI designers are hardware. So, once you fabricate the hardware it is sold to the market then you cannot debug; like in a program it is unlike a program philosophy. So, when you write the program then you distribute it to the costumer then you can get lot of windings and errors.

This come back to you can patch up your code or again sell it to the or give it back to the customer. So, once you give it back to the customers it is very good. Actually they can I mean use that again they will report to you the bugs; they can again I mean correcting the bugs again you are giving it t them. So, in this hand in hand correct I mean finding of bugs and solving the bugs or patching the bugs that is always possible in the software industry. That is actually a very good I mean very good aspect of software I mean

software programs.

But in case of hardware once the design is done your fabric I mean it is fabricated and sold to the market it becomes a hardware component. And it cannot be change if anything goes wrong if you have to through it out you have to put a new chip. So, that is why the no question of any debug after the design is fabricated. So, when before fabrication of the chip when you are at the design stage; so you should be very, very careful that there is no error.

Therefore, we are something call formal verification step that is in this case what we have to do given a design if the chip has say n inputs. So, if you try to exercise all possible input patterns the number will be to 2 to the power n 100 in chip. So, your number of patterns which would be applied 2 to power n all possible cases find that there is no error. But that amount of testing or verification cannot be done by any physical human or any human being in as physical or logical amount of time. So, that is what we have to do we actually go for formal verification.

(Refer Slide Time: 20:12)

Introduction

- To address this complexly issue, after the design specifications are complete almost all the other steps are automated using CAD tools.
 - However, even designs automated using CAD tools may have bugs.
- Also, due to extremely large size of the design space it is not possible to verify correctness of the design under all possible situations.
 - So technique are required that can verify, without exercising exhaustive input-output combinations, that the design meets all the input specifications; this technique is called formal verification.
- In VLSI designs millions of transistors are packed into a single chip. This leads to manufacturing defects and all the chips need to be physically tested by giving input signals from a pattern generator and comparing responses using a logic analyzer; this process is called Testing.

So, in the process of manufacturing a VLSI IC there are three broad steps: **DESIGN-VERIFICATION-TEST.**

That means, we made mathematical models of the circuit. And we will try to prove mathematical design for the circuit actually obeys the specification or there is no bugs in the design or whatever you intended that you designed. That is checking of the design in trend that is whatever you intended to design that you have design. So, all these we required because if the hardware if you make a mistake in the design then it is fabricated.

Then there is no question of tracking back, because everything you have fabricated; you have to throw them up there is no question of debug. So, whatever you have to do all the debugs you have to do before you are sending it to the fabrication stage before manufacturing.

So, therefore it is very important step; second step is called the verification. They given design how will you verify? Again in this course I told you before it is not for not show you that you take a design and verify rather than what we will do? You see how tools can be developed or how can we develop cad tools that will help a designer to verifying the circuit? So, in the design phase you will see how you can develop a cad tools? So, that he can design your circuits very easy manner and it will help you. In the second stage that is called the verification part of the course we will tell you. How verification tools can be developed? Which can help a programmer which can help a circuit developer to verify his design whether they meet the design intend?

And, a finally a phase is called testing. So, whenever again millions and millions of transistors are put in the circuit. So, again the manufacturing process also it is not very reliable So, what happens if you that manufacture say 1 million chip and you have putting more and more chips more and more transitions in 1 chips. So, what may happening is that will enable us to the entire tough which I am telling elaborating all these; whenever you are going to individual modules. So, at the first lecture it will look more like a story rather than very inside depth. So, just mean takes the words which I am saying whenever fabricating millions and millions of transistors in 1 chip manufacture stage is not very reliable. So, why this not very reliable all? So, that thing we will discuss when we are going to the individual modules of the course.

So, that millions of transitions in one chip. So, fabrication process is not reliable; even if your design was perfect and verify whether it function very proper, when you fabricating in the fabrication stage, because of millions of transitions packing millions and millions of transistors in 1 chip it may happen that there kind of some manufacturing defect that remains in the chip. That is why you have to test each and every chip before you sell it to the market.

Again, if you have n number of input chips with you so you have to apply for 2 to power n physical factors. So, there is a difference between verification and test. So, in

verification what you will do? Your design is not at a hardware level it is only at the software level. You have written certain kind of code like a very log and see that. We do not directly go on for designing a circuit by taking chips and put resistors and transistors you do not do that. We write the designing some high level languages like c; in case of software the log and VHDL by using those languages.

You can develop your circuit in a software level first. And then you can do verification in the software level. That is if the circuit is fabricated whether there can be any defects or whether there can be any design bugs in the circuit. So, before manufacturing your circuit is still at a software level but which has to be implemented. So, once verification done verification also checking the software version of your circuit. If there is any design errors or design intend not matching in the design. So, that will be told you by verification part or verification cad tools. Now, once you have verify your design at the software level.

Then, you know that your design does not have any problem because you have been verified properly. Again you can sell it I men send it for manufacturing but here also if there are n inputs in the circuit. So, if you want to verify it using software because at now your design at the software level or program levels itself. So, it will take the order of 2^n amount of time. Now, testing is a much more complex procedure because now your software designed has been named manufacture or translated into hardware. So, there will be say 10000 chips has to be sell in the market; in case of verification software your design was a single program.

So, you have a verified a single program that is the functionally proper. Now, when you make a hardware version for that you just fabricate it then 10000 chips have sold in the market; any of them may have a physical defect. So, testing you have to repeat for all the 10000 chips. In other words or verification a single code which we have to verify for testing you have to test all the chips that has been manufactured. Because the problem can develop, because testing looks for problems that can developed in the manufacturing stage, because a chip is not sent to the manufacturing stage if it is not verified. So, a chip is sent to the verification or chip which is sent for manufacturing or fabrication process; you can be sure that it does not have any kind of design intend or design error.

So, where from the error comes after fabrication? At the fabrication stage there can be

some kind of defects that come because you are putting so many transistors in chips or fabrication procedure also may not be very proper over. So, testing actually test that whether the design was proper with the circuit may not want. Because there can be some amount of manufacturing defects that has script in manufacturing I mean as the fabrication. So, for testing purpose what you have to do? You have to test all the chips which have been fabricated see the chips have n number of inputs. So, the order of complexity for that will be 2 to the power n into 1000 or 10000 chips that has to be tested. So, testing will take more amount of time than verification.

So, again what you have to do? You have to think how you can so everything you cannot apply all 2 to the power n test factors we will not take. So, why have to think which is the best subset of 10000 n number which should apply to your circuit. So, 2 to the power n test factor has to be applied. So, you have to think which the best sub set of the n is which you should apply to test your circuit somehow confident. So, the circuits are not having any manufacturing defect. Let me tell in the other way. So, 1 hour codes will be codes will not testing part of the code will not tell how task mean testing our codes?

We will not like a codes in designing like you make a 50 processor then how you verify your 50 processor? How you test that 50 processor? Our course will not be like that will not take a single design or set of designs or how it works? Whether it tell you how can we developed any design so that any design you make it can help you. First is the cad test for design next you can show for cad tool for verification for any test part of the model of the course. So, what you will see? We will see how cad tools can be developed which can tell you which are the best patterns to test your circuit? So, ideally speaking you should be able to apply all these tools for n test patterns job is done but as complexity will be very, very high you cannot do that. So, what will do is that? We try to tell you that which is the best subset for to do the n test factors which you can apply that and you can very much confident that your circuit is 99.9 percent sure that; there is no defect in the manufacturing stage.

(Refer Slide Time: 26:39)

Introduction

- To address this complex issue, after the design specifications are complete almost all the other steps are automated using CAD tools.
 - However, even designs automated using CAD tools may have bugs.
- Also, due to extremely large size of the design space it is not possible to verify correctness of the design under all possible situations.
 - So techniques are required that can verify, without exercising exhaustive input-output combinations, that the design meets all the input specifications; this technique is called formal verification.
- In VLSI designs millions of transistors are packed into a single chip. This leads to manufacturing defects and all the chips need to be physically tested by giving input signals from a pattern generator and comparing responses using a logic analyzer; this process is called Testing.

So, in the process of manufacturing a VLSI IC there are three broad steps: **DESIGN-VERIFICATION-TEST.**

So, again we will tell you cad tools for test. So, the circuit will have design verification and test of digital circuit at the cad tool level.

(Refer Slide Time: 26:42)

Introduction

- VLSI ICs can be divided into analog, digital or mixed-signal (both analog and digital on the same chip) based on their functionality.
- Digital ICs can contain logic gates, flip-flops, multiplexers,
 - Work using binary mathematics to process "one" and "zero" signals.
- Analog ICs, such as current mirrors, voltage followers, filters, OPAMPs etc. work by processing continuous signals.
- When single IC has both analog and digital components it is called mixed signal IC e.g. Analog to Digital Converter (ADC).
- The automation algorithms and CAD tools are mainly available for digital ICs because transformation of design specifications to silicon implementation can be accomplished using logical procedures (which can be converted to algorithms and tools).
- However, most of the analog circuits design is like an "art" which is best performed by designers with "aid" of some CAD tools (which provides feedback to designer if the manual design is progressing fine etc.)

How can you develop cad tools? That is processor can be automated and easy that is easy. So, we always saying digital design and so other than digital designing, what would be the type of circuits available? So, circuits can be basically divided into analog, digital and mixed signal circuit; mixed signal means more analog and design. So, what is digital circuit? Digital ICs contains logic gates whatever components we have learned in our

basic undergraduate VLSI digital design course all those you are have. And here signals are only logic 0 and logic 1 and you do a binary operations on that.

But in case of analog circuits like we have seen current, mirrors, voltage, follower, op, amps, filters etc. That is the components which we have learnt in undergraduate VLSI and they work by continuous processing continuous signals. Like in case of analog circuits we have sign gates, we have fall gates, we have rams. We do not have digital 0 and 1 and analog circuits and mixed signal circuits are something both analog and digital components are there like ADC, bug converters etc.

So, in our course we only are looking at the digital design part of the course. Because it is said that digital designs are very, very complex as millions of transistors are there designs are very complex. So, you require a cad tool support for that. It is that indispensable that you cannot make any practical VLSI design without a cad tool. But in case of analog design you also require some cad tools but compared to digital design support required is bit less. Because in case of analog design circuits sizes are small and generally people design by hence it is more like a analog call it as more like an art.

So, we have to do good VLSI analog designs. So, you can layout the transistors layout design of your gates, sizing of your transistors etc. In analog everything is done manually as of now. But only want to verify because some simulations cad tools are basically simulations stages in this. So, you design some circuits manually and then you see by simulating the circuit if I gives this input whatever will be the output that you are getting whether they are matching my specification or not as. In case of analog circuits unlike digital circuits the input and output takes are not very large. So, they may give reasonably good subset of all possible input cases and verify them with the output manual components or some kind of preliminary comparison tools.

So, therefore analog cad is basically made for basically based on simulation which will takes some inputs and so the outputs. Then you can use it for whether it matches all the specification. But in case of digital designs the complexities are exponential because if the 100 beams in the circuits complexity of most of the case would be foreign which is impossible to do manually. Therefore, cad tool support is very much required for digital design. So, in this course we will be mainly talking about digital designs and cad tools for design verification and test. So, wherever you use the terms design VLSI design the

course. So, it will be only for digital design. And whenever say design verification and testing any of the required terms.

(Refer Slide Time: 29:23)

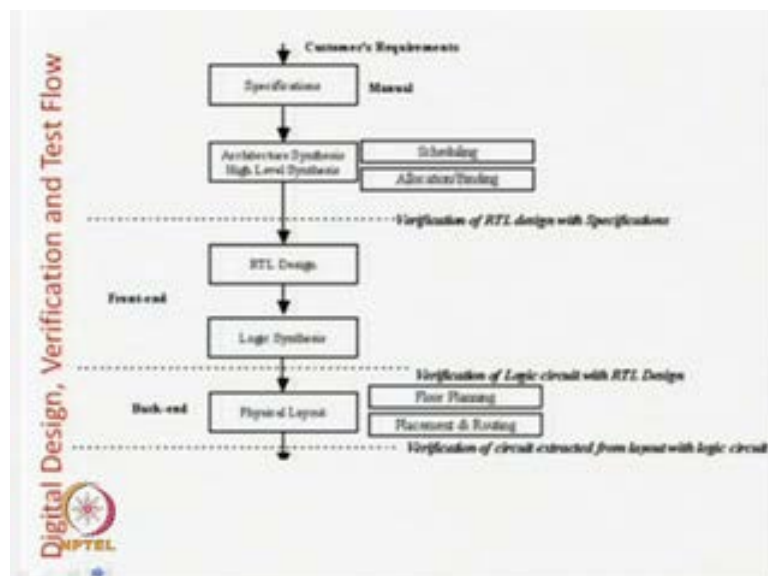
Introduction

- In this course we will deal only with digital VLSI circuits. Henceforth, in this course VLSI IC would imply digital VLSI ICs only and whenever we want to discuss about analog or mixed signal ICs it will be mentioned explicitly. Also, in this course the terms ICs and chips would mean VLSI ICs and chips.
- This course is concerned with algorithms required to automate the three steps **"DESIGN-VERIFICATION-TEST"** for Digital VLSI ICs.

NPTEL

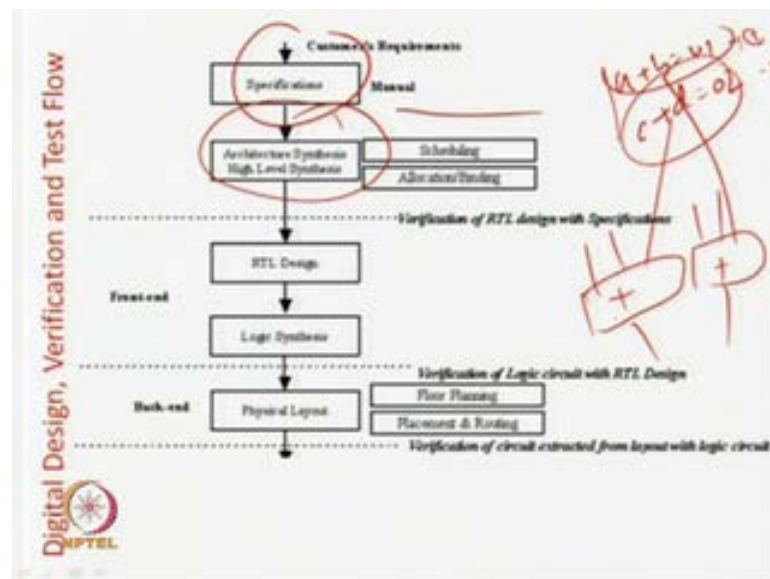
So, they will be generally for what? They will generally form cad tools to support this test. We are not going to do any kind of specific design. So, that is what we are say henceforth in this course whenever we use the term design digital design and it will be actually cad tools for this one. So, in this video course we will be looking at cad tools for design verification and test phase for digital VLSI ICs.

(Refer Slide Time: 29:43)



So, today as I told you in the first class we will we are not looking into what do you called this details of which the steps rather what we will not take a single step and going to details. That will doing from next class onwards or from the first class what we are going to do? We will tell you what are the basic simple examples? We will cover up all the possible steps all the stages all the stages that are taken. I mean they are handle or in other words take the simple example. We will cover out all the cases which are followed in I mean digital VLSI design procedure.

(Refer Slide Time: 30:22)



So, let us see this is the first part of the VLSI digital VLSI design. So, first we have specifications see these are manually developed specification like you can say that a plus b is equals to out 1 and c plus d is equals to out 2. So, this can be a very simple. We can say all this case you can say all these numbers are on and o2 are sum and carry with this number something like this. So, these are the design specifications. So, that is done manually. Next what do we do with that? We will go for architectural specification high level synthesis like in this case we will take another example.

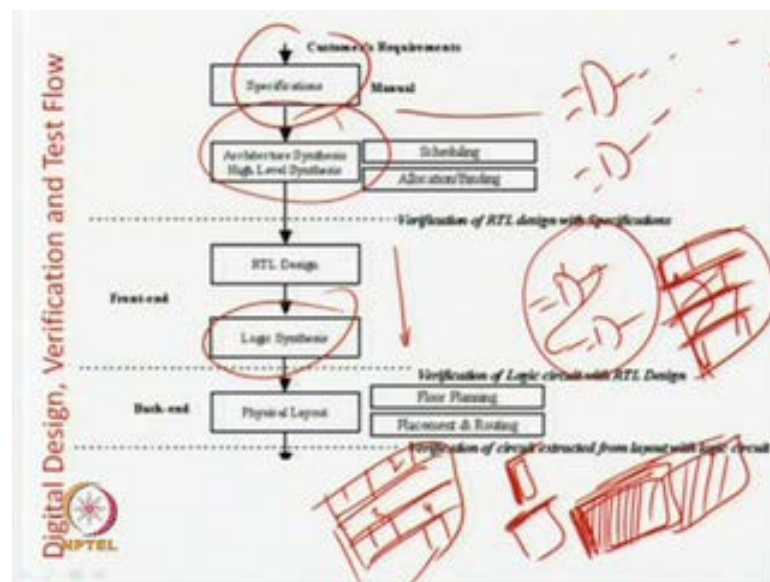
So, will elaborate this example. What you say so architecture example means you will say tools take 2 adders say a b c d these are the outputs. So, these are very broad level. Now, what is what are doing in architecture level very broad level. Now, the question and answer that this is a operation 1 and this is operation 2. So, required to adder first actually the answer for question is like that whether these adders will allocate to this one

and this one to this one or vice versa.

Sometimes we can also said like 2 adders are not available single adder will be there. So, first do it for you then the second adder will (()) done for you. So, wherever we talking about the very broad level in the architecture level how you can meet this specifications? So, that is actually called as the high level synthesis; which actually comprises scheduling allocation and binding scheduling allocation. So, scheduling binding will elaborate it shortly.

So, the basic idea is that we are making a block level diagram for that specification to meet that tools that is the specification. And when we are answering questions like how many adders are there? How many multipliers are there which operands goes to which operator and so forth? Once that is done so we will go for RTL design and logic synthesis. What do you mean by RTL design this is actually resistor transfer level design. So in this case so we seeing that adder, then we say the resistors, then you say that multipliers, then you say that clock. So, that means what we do over time of adder do you want to (()) those things we decided.

(Refer Slide Time: 32:30)



Also we design registers do you want to use deep leaf type of resistors, do you want to check digitals of adder registers? So, what will be clock will be gated. So, many other resistor transfer levels this is actually what we will do this will be at the resistor transfer levels. That are we take the résistance of the registers and the blog level like adder

multiplier or subtractor what type of staff you want to use and so forth.

And, then we will go for logic synthesis. What do you mean by logic synthesis? From the RTL level now we have to go for gate level implementation, if you are in (()) so there will be different grade implementation. And if you want to carry look adder implementation will be different and so forth. So, in logic synthesis what we will do? We will take the RTL level like for example flop we have to convert this into gate level. If you have a adder we have to convert it into a gate level adder is the RTL level a is equals to b plus c . Now, when we go to the RTL level a plus b is equals to a dot b and sum ax or bx it will carry so we will get the gate level.

So, the logic synthesis is actually converts into the gate level. Now, one everything has been done at the gate level now your circuit has to be fabricated and then sent to the market kind of a thing. So, that will go for a physical layout. That is once the gate level are there so actually this is the gate level several kind of designs are there. So, actually see the VLSI circuit or the chip generally in the chip you do not have you never have gates like this. These are only for your pictorial view gates in the circuits are nothing but some rectangular chips.

So, in this course we are not covering any kind of physical design. So, you can take any standard material and you covered the physical design but I will give you the idea that what the gate does not looks like this. Gates are basically rectangular blocks chip is not actually the die or the circuit die like this it is a will have something loop like this. It is a rectangular square inside it has some rules. So, whatever the gate you require because the gates in physically gates are some rectangular blocks different in width. So, it corresponds to gate 1 and then it will correspond to gate 2; then may be or the gates 3 it will be gate 4. So, in physical design what do you do? So, you know that gate is correct connectivity and all the gates are available from the logic synthesis. I will tell you which gate has to be used? How it has to be connected and all this information are given in the logic synthesis stage?

Now, you have to put these gates and interconnections in what do you call rectangular Square where there are some rules; and the gates looks like some rectangular block. So, your job is how can you optimally put these blocks or these gates on this or what do you call this die or the chip. So, how can you put the gates where you will place the gates?

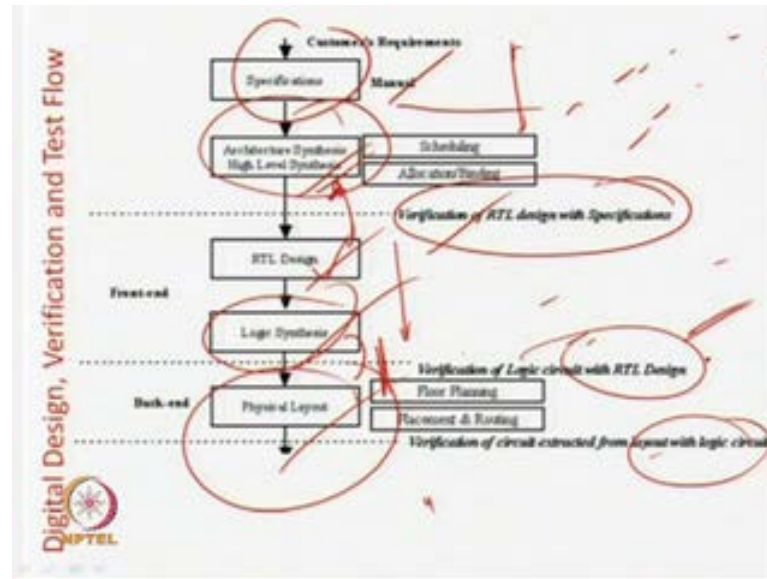
So, that all the gates can place the minimum amount of area. It is like a beam packing problem kind of a thing. You have a big box, you have different size of oranges which arranges how do you arranges the oranges in the box?

So, that the maximum oranges can be placed in this one same thing like here. So, there are some rules and it will be some gate will be this thick may be or gate will be this thick; xor it will be longer and inverter will be thinner. Generally with this I mean height is same only the width varies in a very broad spectrum, so very broadly speaking. So, it takes more area it will be long in this direction this one you want to keep fix this, because this width and width of each row in the area accordingly. And then what you have to do? You have to find out that different width with different length gates are there you can think.

So, how different length blocks are available? So, how you can place this blocks here? So, that the minimum maximum number of blocks are seated over here you can think minimum area. Again it is an interesting computational problem how does you that? So, cad tools are available which can also help you in with this physical layout. So, one this gate level connectivity is in the logic synthesis in physical layout what you have to do? You place the gates in the die in this way and do the interconnection. Because these point has to be granted with round this and again this point is integrated with without any kind of shorts and all those things. So, I mean what is the physical layout will do physical layout will do proceeds with through all the gates in the die.

So, that the minimum area is taken. So, it will also round the input I mean round the connections in the gates. So, that is actually done in the physical design obviously we are having millions of transistors. So, millions of about lakhs of thousands of gates has to be input in the circuit or transistors have to be put in this die. And in this and interconnection has to be done in between them. This is like a graph or millions of nodes and you have to node one node of the graph or the other. Again, that is very, very computationally tough problem you cannot never do it manually.

(Refer Slide Time: 36:31)



So there are cad tools which can help you in the physical design in the process, and at which states, because these are basically first few steps and next few steps to be explained. So, these are the first few steps of digital VLSI design. And each partition dotted line you can see that the verification of RTL which takes; verification of logic in this case are RTL design. verification of circuit abstracted from layout.

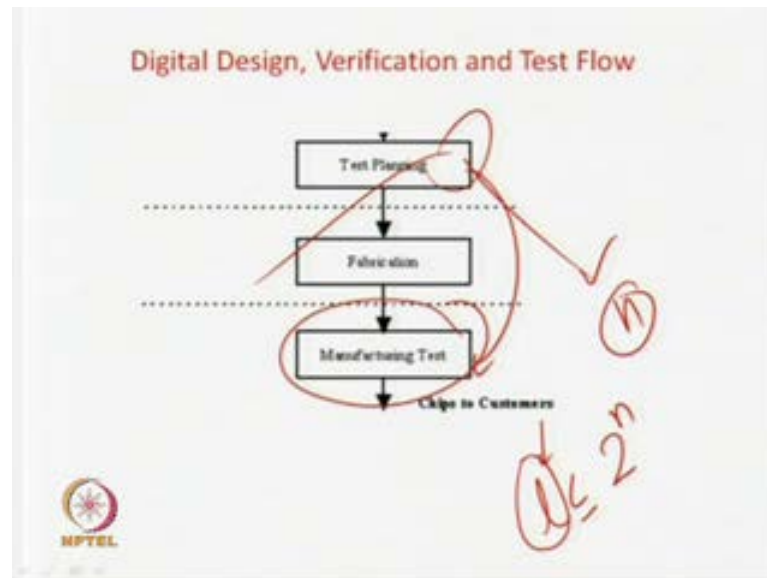
So, what is verification implied? That means, at this level well your design was some logic or some manual aspects taken place or it was at the block level or architecture level. From here what you have done? So, you have got the RTL design that. So, here at the block level, it was at the architecture level. From here you have done what you call RTL level. Now, there are some translations so you have to see that whether the translations are valid or equivalent or not.

So, that we require verification here; like in logic synthesis that you have gate level what do you have here? Physical layout everything has been laid out at circuit and interconnection has been done. Again also you have to find out whether the translators has equivalent or not. If translator has not equivalent in balance then there is a deviation in the design because you design something it has changed in next stage may not be adhere to meet specifications.

So, at every stage when you are going to form one form of design with the other like it was takes then it have go to RTL high level designs there is the RTL and then lay out.

So, same design same specifications you have combining into different types. So, when you are converting from one type to another verify that they preserve the equivalent some where you get the change problem.

(Refer Slide Time: 38:02)



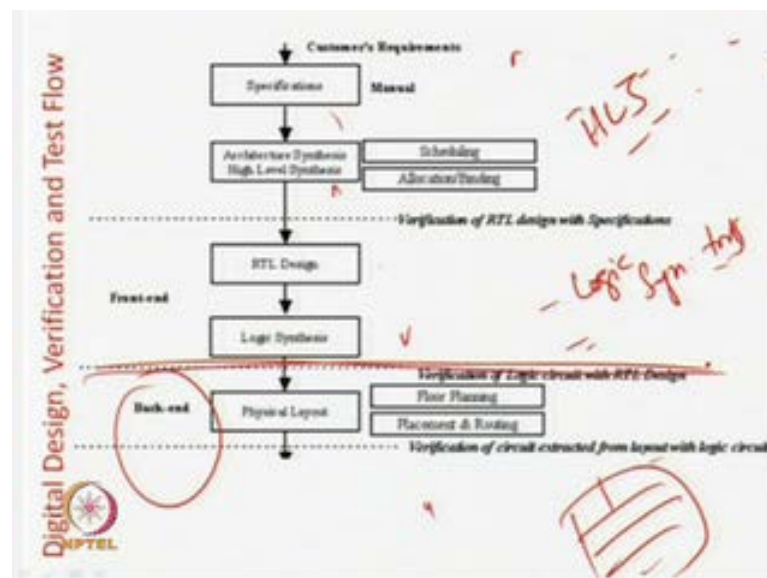
So, once I mean all these things are done before sending it to the market also go for a test planning. That is given the circuit because you already have the software version of the circuit and it has gone to the market for fabrication. Mean while what we can do? We can find out if there is n PCB circuits we have 2 to the power n vectors to test it get us to test very immerge impossible. So, you have to find out a subset of this test vectors; then you can apply like then you can readably confidently there is no defects in the circuit.

So, the subset like 1 is the subset equals 2 to the power n . So, that subset has to be found out that is actually called as test planning. So, given the software part of the circuit itself because it has gone to be fabricated. So, you do not have to fabricate that is also required for test planning. So, you can look at the gate level diagram nothing at the layout how you are planning to put in these dies and all those things. The software version is always available to you after that. So, looking at the fabrication level which are the best subset of vectors 2 to the power n vectors which we can apply you can reasonably happy that circuit. I mean circuit in what do you call your circuit will be available with the subset and any manufacturing defects.

So, this is actually called test planning. So, once the test planning is done after that you

can go for fabrication. But very general I mean very broadly speaking this is test planning. You can go involved in and you can go for the fabrication and manufacturing fabrication is another stage because once the chip is designed. So, you can sell of it to the market while manufacturing send it to the vendors for manufacturing and mean time you find out your test plan. Now, once you have the circuit is ready after fabrication you will get your circuit. So, that you have to test using this test patterns. Whatever chips you have find out there any affect give it to the customer. And for the chips you can throw it up the market because we will see at the end of course of the testing. Inspecting is not very essential part of VLSI testing we will see that.

(Refer Slide Time: 39:54)



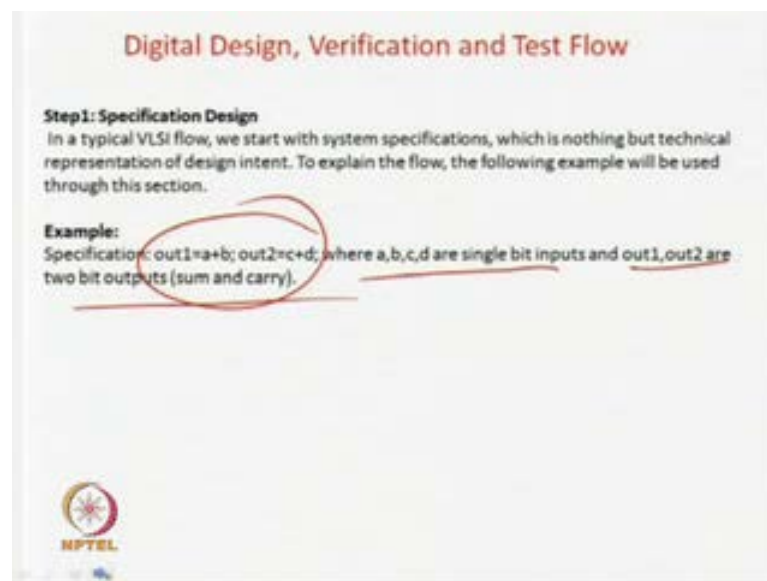
So, at the end so that is what is the basically design verification and test flow for digital designs? Now, what will be standing in this course? Again repeating we will be looking at the cad tools which can help you in all the phases of the design. For this phase we will have some high level synthesis design tools means what we call cad tools we will see. In this phase we will see logic synthesis tools. Now, this part I mean many of you already learned like (()). There basically takes your design and you are like counter adder and all those things you generate at the gate level design. So, you cannot you can walk up to 10 variables or 20 variables but in VLSI you have 100 variables or 1000 variables.

So, how can you do a kind of optimization for on1000 variables? So, you required different types of cad tools for that; that will be landing in the second stage. So, that this

also gives a die like this an interconnection of gates. So, how can you put the gates in different roles? So, that minimum area is taken and this can be connected. So, this CAD tools for that. So, this is actually called as back end part of designs. And this course not to be looking into backend part of your digital designs because we are covering some spontaneous aspects of the designs. So, in VLSI backend spontaneous means designs design verification and test plan development, and back end means how you are going for layout for routing and fabrication that part.

So, as we are merging 3 broad courses in one small; basically we have separate tools with these separate courses for VLSI design verification test and back end. So, in this course we are trying to give you a brief overview of the spontaneous part of VLSI design flow. So, what we are going to see we will be telling you valuable design verification and test flow. How CAD tools can be manufactured or tools can be developed for that? We will not be covering for that manufacturing stage manufacturing processes as well as we will not be covered in details about physical layout.


(Refer Slide Time: 42:30)



Digital Design, Verification and Test Flow

Step 1: Specification Design
In a typical VLSI flow, we start with system specifications, which is nothing but technical representation of design intent. To explain the flow, the following example will be used through this section.

Example:
Specification: $out1=a+b; out2=c+d;$ where a,b,c,d are single bit inputs and $out1,out2$ are two bit outputs (sum and carry).



How the physical layout is done it can be developed or because for verification or tests for manufacturing whatever models will be studying over here. So, there can be separate this semester course was been done for this courses. But the aim of this course is if you want to understand the (()). How digital design is done or what are the CAD tools are designed? We have developed a course which can give you a flavour of all the parts of

the front end parts that is design verification and test. So, as you are covering 3 models in overall courses into one course it will be heavy load or it will not be proper into one more model that is the back end part in which same course. So, we will be covering mainly front end design cad tools.

So, now whatever we have told you said that using a example. So, what is the specification? Let this be the production specification. So, out 1 is equals to a plus b equals to out 2 plus c plus d; a b c d be single input outer tool. So, we have to do that is new specification how you go about it?

(Refer Slide Time: 42:50)

Digital Design, Verification and Test Flow: HLS

Step 2: High level Synthesis
High-level synthesis (HLS) algorithms are used to convert specifications into Register Transfer Level (RTL) circuits.

- HLS, sometimes referred to as architectural synthesis is an automated design procedure that interprets an algorithmic description of the design intent and creates hardware at RTL that implements that behavior.
- The input to a HLS tool is design intent written in some high level hardware definition language like SystemC, System Verilog etc.
- The HLS tool first schedules the computations (required to meet the specifications) at different control steps.
- Following that, depending on availability of hardware units and time constraints, the scheduled computations (comprising instructions and variables) are allocated and binded to the hardware units like adders, multipliers, multiplexors, registers, wires etc.

The slide contains several handwritten annotations in red ink: a box around the title, a box around the definition of HLS, a box around the word 'schedules' in the third bullet point, and a box around the phrase 'allocated and binded' in the fourth bullet point. There are also some scribbles and lines in the top right area of the slide.

So, this is called the high level synthesis that is what I told you design it at the block level? So, what is high level synthesis? The high level synthesis algorithms are used to convert specifications into RTL or high level circuits. That is like adder and all those things even much more than the RTL level. We generally call it as RTL level high level synthesis is actually converts specification to you can say black box type of levels like adder, subtractor, registers and something like that up counter, down counter some registers. And so you can call it as very near to RTL design. RTL means resistor transfer levels that is you have between 2 resistors you have referring that functionality will come so and so will be the example. So, that is why the RTL high level synthesis is called as architecture synthesis.

That is why just say in blocks adder, multiplier, divider, registers interconnectors in

between the hence so forth. It is actually it is called as architecture. So, high level synthesis basically sometimes architectural synthesis and automated design procedure because we will developing cad tools. So, high level synthesis tools what it will do? It is actually do that nothing but automated design tools that will take a specification algorithm description; and it will convert into an architecture design. So, that is what is the that is what we are going to see at high level synthesis cad tools. Basically high level synthesis means taking a specification and converting into something like a architectural design or what do you call near about RTL designs. That is what the high level synthesis is.

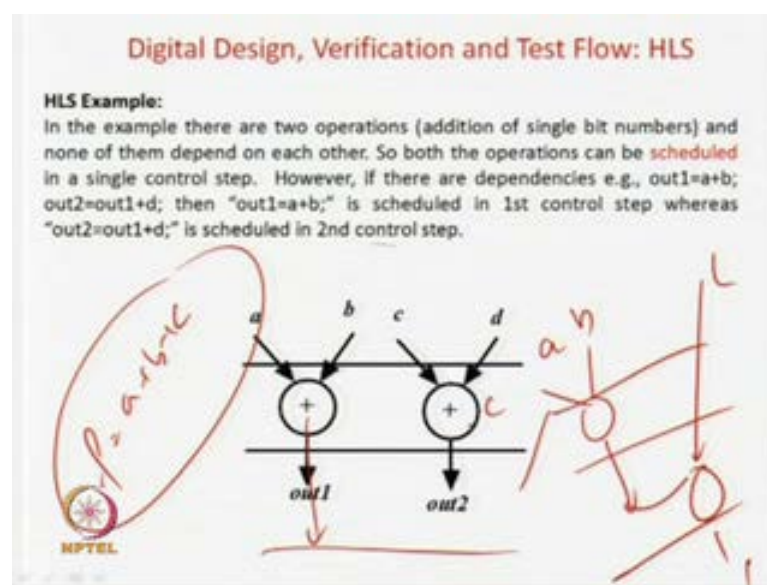
Now, here in this module we learn about tools; cad tools which can automate that. So, how do means? So, how do you write your intend? In this case you have written our intend like this one. Generally it is written using some kind of languages which are called system c very large or some other languages. We should not be going into depth in the languages because whenever required give connections to that one. So, basically we requires we write in some kind of high level languages with there is not like that c system. So, it is very easy to write I mean hardware specification synthesis.

So, what is this high level synthesis tool? It will do it is finally, generate RTL design all those generate what we can say architectural bound diagram. So, first what we have to done? First we have to schedule the competition at different steps. So, will see what do you mean by scheduling? We have to add a plus b and c plus d there are 2 adders. That means, 2 operations you have to do. So, you can schedule both of them in 1 step or what you can do that is you can schedule a plus b in first stage and then b plus c in second stage.

So, this is actually called scheduling. What operations you are scheduling that is called scheduling. And secondly what you have to do? You have to use hardware like if I have 2 adder then you can do a plus 1 and a plus b and c plus d parallel. If I have had 1 adder in the first stage I have to go for a plus b or c plus d and in the second stage you have to go for other way. In the first stage I have to go for a plus b and second stage c plus d. So, that is why if you allocate which operations you do when? And similarly, that is you have to allocate your operations in the time stages or different hardware because only single hardware; single adder a plus b and c plus d will be it have to done in the same order here allocation does not mean anything.

But if I add 2 adder then a plus b will be in 1 block and c plus d will be in other adder. So, either one you have to add a plus b or adder 2 you do the c plus d or vice versa. So, in which hardware it will to which operations. So, that is actually because of allocation and binding. So, here what we have to do availability based on the availability hardware needs and the time scheduling computations. I mean what do called schedule computations because you already scheduled operations in time because you schedule your operations as 1 2 3. Schedule computations are allocated and binded to the hardware. So, that is actually called as binding allocation and binding.

(Refer Slide Time: 46:33)



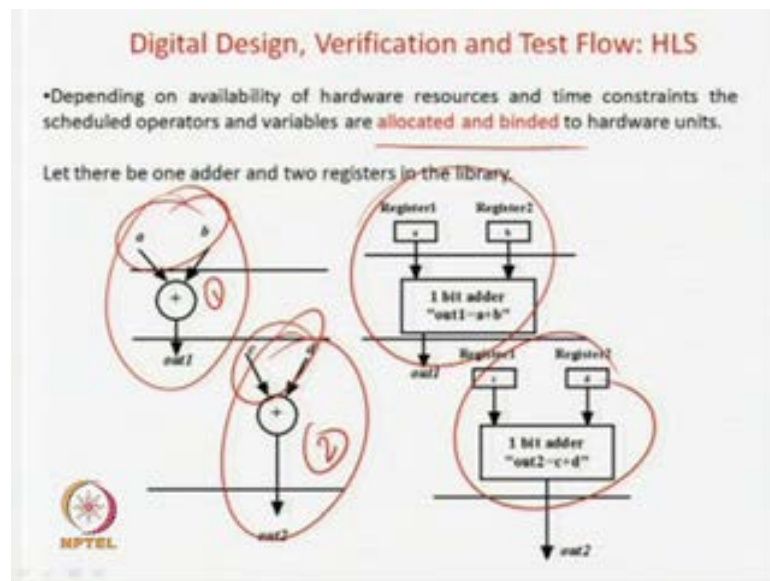
So, we will see with an example we can maintain clearer. This is you see these are called specifications a plus b equals to the sum c plus d. Here we scheduled what is the scheduling over here? It is schedule that a plus b and c plus d are done in the one control. So, this is the one kind of scheduling. You can also have another kind of scheduling like you may not have EDL. You can also have this one this c plus d you can this is another kind of scheduling. But let us see if you have a something from other specifications this s equals to a plus b plus c. Then you cannot schedule a plus b plus c one time pair.

It is impossible; first you have to do a plus b you get some output. In the second stage you have to use this then you will get a and then b. So, for this specification obviously you require 2 times stage for scheduling. But here you can have flexibilities you can schedule this here and you can schedule this here. So, in this example like in this a plus b

plus c.

So, in this case you cannot schedule within one set this actually minimum that is 2 times. But you can schedule this plus this 1 stage or b plus c in first in first stage and with the temporary you can add a or other way round you can add a plus b in the first stage. And with the temporary result you can add a in the second stage or the other way round you can employ variables in this scheduling.

(Refer Slide Time: 47:58)



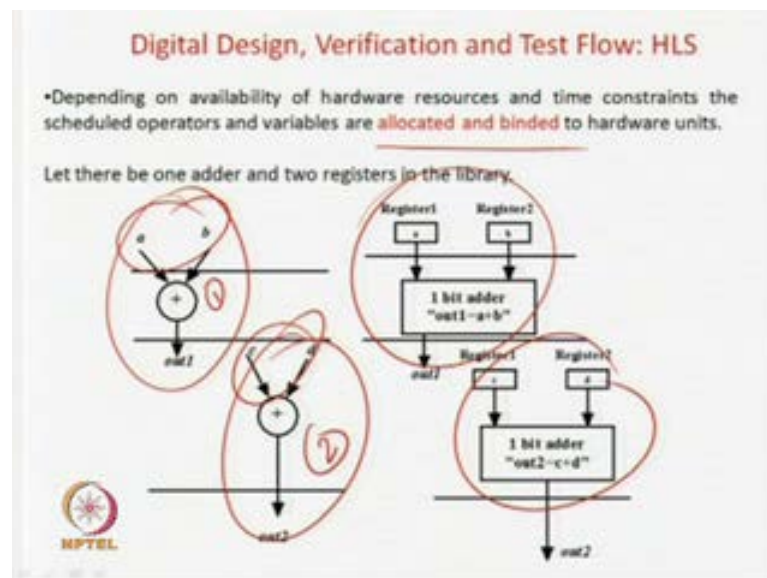
Now, actually what do you mean by allocation and scheduling? Let us see this in this case we see different examples. If you want to go for scheduling allocation and binding want to go for that it is very simple. I mean just if you have 2 adders then you can just do it a b and this is c and these are the outputs. Now, what is allocation of bindings? In this one you have allocation of allocate a and b in this one you have done a and b plus c. Now, you can also do some other thing that you can take a plus b in 1 adder and b plus c in adder 2 for where binding a plus b to 1 or c plus d to 2.

You can do it in other ways. So, if you are directly mapping your scheduled operations to be hardware. So, this allocation and binding are very clearer but now you think that I have only one angle then it is a problem. Then actually what you have to do? You have to schedule this one here and you have to schedule this one here. So, in this case in the same adder what is happening you are allocating both a plus b and c plus d depend on the time schedule? This one you want to do a plus b one over here and in the second step you

want to do c plus d or vice versa.

So, depending on the hardware resources and constraints scheduled operations are allocated and binded, in this case only one hardware one adder. So, you can you have to allocate a plus b and c plus d they have been same adder use. Then obviously the numbers are stored in registers. So, you cannot directly bring them you cannot directly doing them at the hardware multiplier or adder subtractor generally talking about whatever variables in hardware which stored them in registers. So, in this case there are 2 registers on the you have to mention that not only what do you call adder multipliers or subtractor can help you out. So, also have to registers to store them. So, in this case you have as there are 4 variables we have to assume that there are only 2 registers.

Refer Slide Time: 50:01)



So, first you do this and then same registers and adder you used for c plus d. But you are taking this type of case you have to have 4 registers to do this because wherever you have variables. So, again you can think that r 2 b, r 3 c and r 4 d again also b interchanges in between them. So, actually this is called allocation and binding. So, variables are allocated and operations are allocated allocate to hardware.

(Refer Slide Time: 50:11)


Digital Design, Verification and Test Flow: HLS

There is one adder and two registers in the library. So the two operations (addition) of the example, even if scheduled in one control step, cannot be allocated to the single adder. Similarly, the four variables cannot be allocated to two registers.

In the running example with the given resource constraints, the two operations can be done in two control steps:

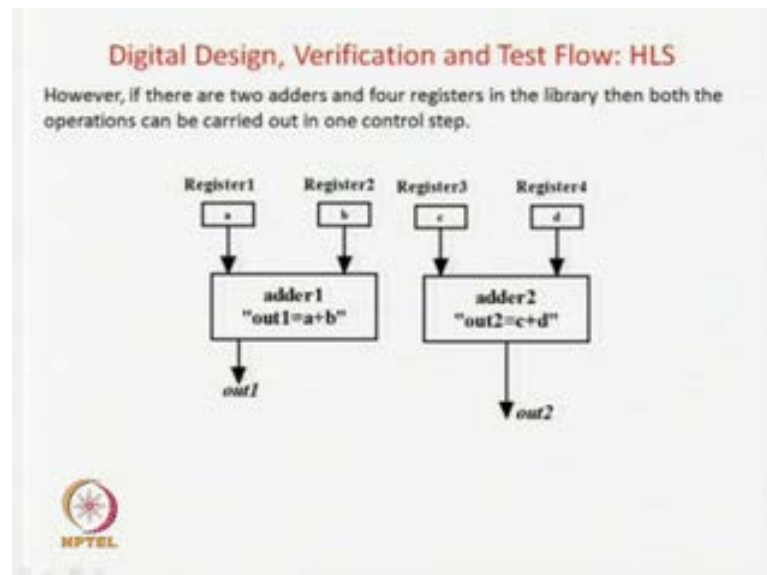
Step 1- variable a is allocated to Register1, variable b is allocated to Register2 and operation "out1=Register1+Register2;" is allocated to adder;

Step 2- variable c is allocated to Register1, variable d is allocated to Register2 and operation "out2=Register1+Register2;" is allocated to adder.



So, that is what I have told you? So, I mean in the running example there are 2 operations that is are doing in 2 control steps that is 2 control steps, because we have only 1 registers, 2 registers and 1 adder. So, first one you do this, second one you do this that is what has been told.

(Refer Slide Time: 50:26)



Now, whatever is talking if you have straight forward 2 adder and to 4 registers. So, there is no question of any kind scheduled in first stage only you have to decide in which adder you have do a plus b or you want to do b plus d. And similarly for the registers

variables which you want to allocate and buying fair.

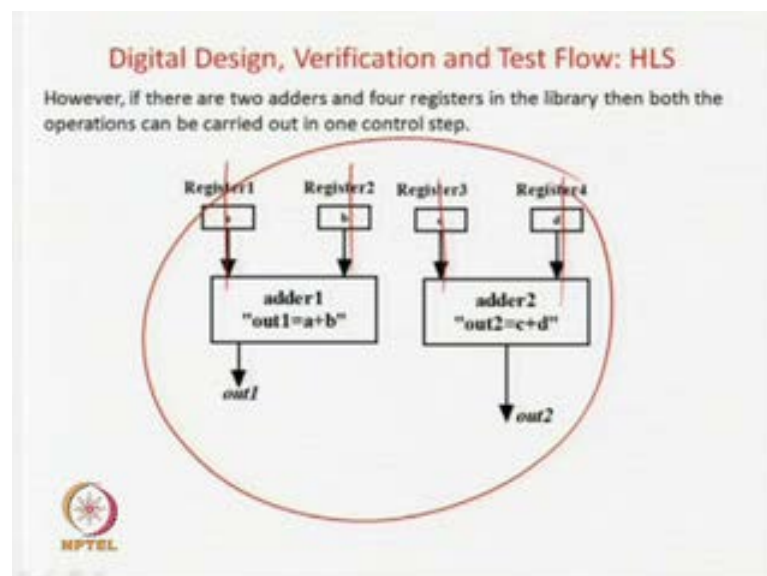
(Refer Slide Time: 50:42)

Digital Design, Verification and Test Flow: HLS

- Finally, based on allocation and binding, the control unit is to be designed (at high level).
- If the allocation/binding is according to {2 adders + 4 registers}, the control is trivial.
- However, if the allocation is according to {1 adder + 2 registers}, then the control circuit needs to provide signals that can do multiplexing between a and c , b and d :
 - In 1st control step, a should be fed to Register1 and b should be fed to Register2.
 - In 2nd control step, c should be fed to Register1 and d should be fed to Register2.



(Refer Slide Time: 51:14)

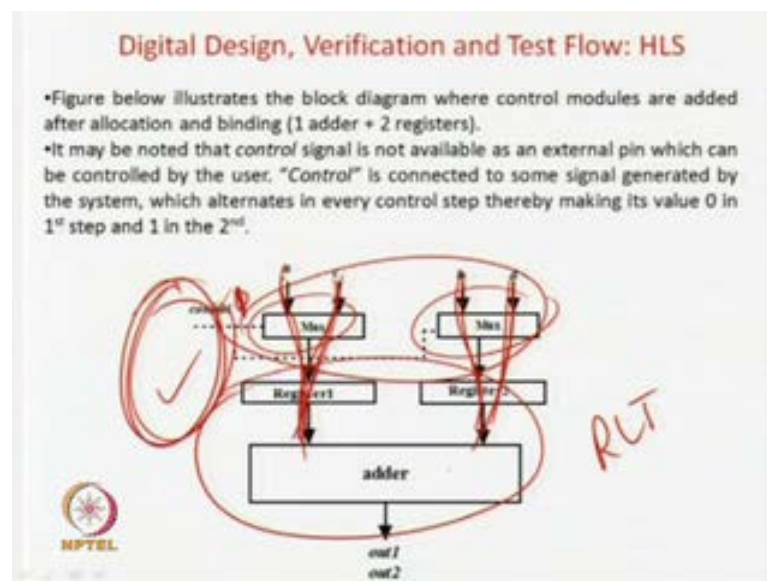


So, that is what the idea? In case of scheduling or schedule which operations have to be done in instantly? And allocation and binding given different hardware in which hardware you allocate variables and which operations, that is actually called allocation and binding. So, in high level synthesis what do we do with these steps. We schedule allocate and binding. So, at the end we get what we get? You get a circuit like this or you will get a architecture design like this. So, you get a architecture design like this. So, that

is the output of high level synthesis at this stage.

So, in this slide saying that we have 2 adders or 4 multipliers in this case it says that. I mean allocation time scheduling and binding can be done; in adder one you have a plus b and c and you can translate them all. But now there is one very important thing. So, in this case you see that very good looking staff but everything is done in single stage. So, you can see here there is a problem. You require more hardware here then here. In this case we require only 2 registers and 1 adder. In this case you require only 2 adders and 4 registers. So, a person may not be happy with this because these are the you will get the answers here you require 2 stages. But still somebody require very fast operations. Then I do not want to go for these designs. I always want to go for this design do not need very fast operations but this is the big part over here. So, in this case there is no controls you directly a comes b comes c comes d comes.

(Refer Slide Time: 52:21)



But if you require in a single adder and 2 registers you require a circuit for that. What are the control circuits here? You see first a will go then b will go then the same resistor c will go and d will go. So, you have to have a multiplexer and initially control will be 0. So, a will go, b will go and in the second stage control will be made 1; in that c will go and d will go here you require a control circuits.

Which will first generate 1 and then generate 0 first generate 0. So, that a will go here and b will go here and once that c will go here and d will go here. So, you require a

control circuit for that. So, even if you are saying that I do not require any fast operations use single adder registers and 2 registers and tool. So, with these what I want to do? That you require a control circuit and you takes area require some multiplex. So, nothing comes at again if you want to go for a very fast design fine. If there are very big circuits you require any control levels over here.

There is nothing to multiplex, but if you are going to go for minimum design here minimum hardware design. Then control circuit and you also take some area. Obviously the area will not be as large as this one but some penalty you have to pay that is what I am saying. So, hardware also high level synthesis not only goes for allocation serially allocation and binding but also it will tell you how the controls are behaved. Because if you are going to these moral structure with 2 multiplexer and 1 adder.

You also require a controller so high level synthesis will also tell you how the controller will behave? So, that is what been saying? These allocation and binding is according to 2 adders 4 registers which are using for this small structure. So, first stage a and b should be set second stage c and d should be fair and so forth. You require the control circuit first. So, this is how what the high level synthesis are done?

(Refer Slide Time 54:06)



So, next what is the case? So from the high level synthesis you have got the structures as well as the controller. So, in logic synthesis what you will do? You have to confirmed that this is actually also called as RTL levels or architecture level you have to convert it

into gate level.

(Refer Slide Time 54:25)


**Digital Design, Verification and Test Flow:
Verification**

Before the starting of logic synthesis, one needs to verify if the RTL is equivalent to the specifications.

In the running example, we can verify by applying all possible input conditions of a,b,c,d (along with control) to the RTL and checking if out1 and out2 are as expected.

However, if the RTL has about hundreds of inputs then exercising all possible inputs is impossible because of the exponential complexity (I.e., if there are n inputs then all possible input combinations are 2^n).

So we need to have formal verification methods which verify equivalence of RTL with input specifications.



(Refer Slide Time: 54:44)


**Digital Design, Verification and Test Flow:
Logic Synthesis**

- After the RTL is verified to be equivalent to system specification, **logic synthesis** is performed by CAD tools.
- In logic synthesis all blocks of the RTL circuit is transformed into logic gates and flip-flops.
 - For the running example all the blocks namely, adder, multiplexers, control logic etc. need to be synthesized to logic gates.

Will illustrate synthesis only for the adder module and for the rest, similar procedure holds. Details will be explained in the "DESIGN" module of the course.

We first determine the Boolean function of the adder module, in terms of mean terms.

| a | b | $Out1(sum)$ | $Out1(carry)$ |
|-----|-----|-------------|---------------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

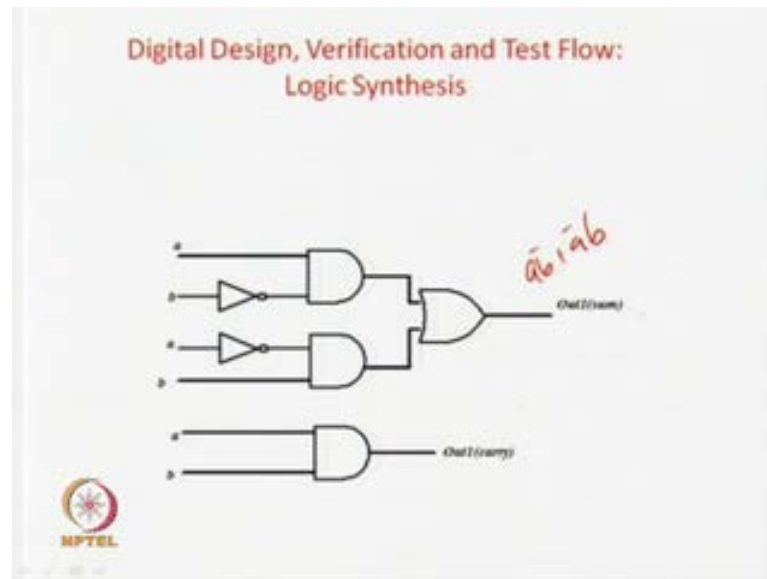


So, next what is the case? So from the high level synthesis you have got the structures as well as the controller. So, in logic synthesis what you will do? You have to confirmed that this is actually also called as RTL levels or architecture level you have to convert it into gate level. So, how to do that this is we have already done in the digital design class that is what gives? So, hardware requires how many blocks? So, you require registers multiplexer and adder I have give you the example of a adder. How to do that there is

straight forward.

So, I will just show you do that and I will come back so in this case you write the route schedule of an adder. So, this is a b this is outer and sum then you can easily go for an optimize design whatever you say.

(Refer Slide Time: 54:48)



Something you can say that this is the $a\bar{b}$ plus $a\bar{b}$ this is actually x or B for the sum and this is for the carry. So, you can easily generate from this table. This is sum and carry. And what we can do that there is some technique (()) because the standards you have already done but the problem is then if the numbers of inputs are less. So, in case of VLSI design number of inputs can be (()).

So, how will you do that? How cad tools can be developed they can be they can automatically minimize circuit when the input is thousands given you more than that. So, that will be landing in the design module of the course. And so this is about the from where you can what do you can say from the RTL design. So; we have gone through the logic this is actually called as logic synthesis. So, again there could be a but it should be a equal balance preserving step. So, what do you mean by equal balance preserving step? Some adder description here some adder description was there. So, what is say that then $a + b$ is equals to b equals to c and sum carry it will also do c plus d . So, that is the basic job of the adder. Now, you have to verify whether this dl is doing well by this circuit because this circuit you have got the logic synthesis. In the same adder logic level

that a plus b or c plus d is not converted into something like this.

(Refer Slide Time: 56:26)

Digital Design, Verification and Test Flow: Verification

Broadly speaking, for formal verification we need to model the RTL circuit and the specifications using some formal modeling techniques and verify that both of them are equivalent.

In other words, equivalence is determined without applying inputs.

Control and Data Flow Diagram (CDFG), a formal modeling, to capture the RTL. Finite State Machine (FSM) to model the control logic.

This example being very simple, we can see that both specifications and the model are equivalent. Formal techniques for checking equivalence can be will be elaborated in "VERIFICATION" section of the course.


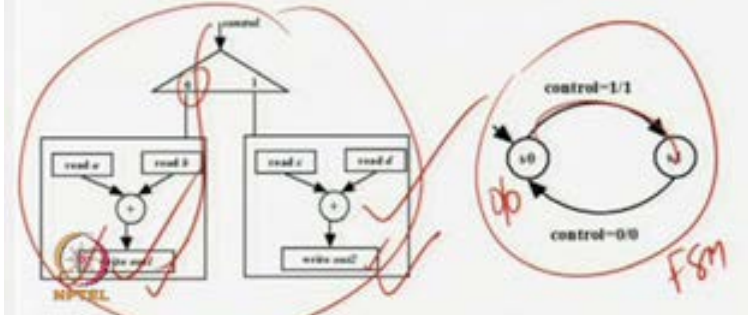


So, we have to formally say that or you have to somehow say that both of them are equally balanced; for that we have to do actually do something like a formal verification. So, in case of adder it is so we cannot try 0 0 1 and 0 but if in the on 1000 input case, then if you are had a big problem. So, in this case what we have to do? You have to do the formally there can be some kind of a landing in a formal verification model.

(Refer Slide Time: 57:09)

Digital Design, Verification and Test Flow: Verification

This example being very simple, we can see that both specifications and the model are equivalent. Formal techniques for checking equivalence can be will be elaborated in "VERIFICATION" section of the course.



So, what we will do is that? We will model the circuit in some formal model then we will

write to verify mathematically with that everything is proper or not. Some important models to capture the RTL design or the architecture design something is like FSM. For example, you look at this so what is the data flow? So, what you do you do? With c read and d and these are controls 0 and this is going to happen and that is one. And this is actually the mathematical model of the whole circuit. These whole circuit is being mathematically modelled over here; this circuit is mathematically modelled the control and the flows are here.

So, this is the case. Now, what say then is the financial statement of the controller. So, in stage 0 it is 0 and 0 it is activating this one. And in the second stage it goes here and the activation is 1 here. And this is the financial machine model of the controller. Now, you will see in the formal modification controller how can you use these models to verify that what we required in the design? So, the requirements of the design was done in this one a plus b b plus c this was the requirement of our design. And this is being possible why this circuit. So, this can be verified by this model. So, how will do the verification? And all the mathematical possibilities are there we will learn in what you call as verification class.

And, from this for the adder for the registers and all we have to get something called as logic gate level diagram. So, again we have to mathematically verify that this tough and what you can say? This model or mathematically equivalent. So, again you have to make a formal model for the gate level circuit and then you have to see that whether they are equivalent to the adder. So, in this case out 1 equals to out 2 equals to c plus d. So, you want to implement this one. This one we have converted into formal model like this verify. Next step what we have to done? We will take this adder add registers and all and for the adder we have got a circuit like this some kind of optimization like this one. And we have to make a formal model and we have to show that this is equivalent to this functionality.

(Refer Slide Time: 59:02)

Digital Design, Verification and Test Flow:
Backend

- Once the logic level output of the circuit is obtained we move to **backend phase** of the design process.

In backend we start with a software version of the silicon die where the chip will be finally fabricated.

- Broad plan regarding placement of gates, flip-flops etc. (output of logic synthesis) in appropriate places in the software representation of the chip; this process is called **Floorplan**.
- Exact locations in the die (software representation) where the circuit components are placed; this is called **Placement**.
- Required interconnections (as given in the logic circuit) among the **gates** that are placed in exact positions in the die; this process is called **routing**.

Again equivalence of output of Backend process should be established with logic design.

And, this is what then we will go for back end. So, for back end that is I told you. So, what do you do? Actually make an idea where you will place this gates; this is called the floor plan this is the overall plan. And here I put the n gates and then I will put the adders put the multiplexer and here I will put the memory block and so forth. They are actually called as the broad level placements or the gates in the die. That is called for plan think about individual gates in which row of the die.

How will you make the connection from placement and routing? So, once you have done this again you have to verify that the placement and routing is equivalent to this one. So, in that case what we have to do? We again I mean abstract out what we have laid out in the circuit. And then we try to again go for formally with this in verification. And this source you will not be backend design I am not talk about more about that backend.

So, next stage so all these things are done fabrication has been done. What you have to do? You sell of your circuit send the circuit for manufacturing and at the same time we are also go for test planning. So, what is test planning and why it is required? Already I told you that in manufacturing now days that what do you do? You do actually putting large number of transistors in one chip there can be physical defects when you are fabricating.

(Refer Slide Time: 59:53)

Digital Design, Verification and Test Flow:

Test

- In VLSI designs millions of transistors are packed into a single chip, thereby leading to manufacturing defects. So all chips need to be physically tested by providing input signals from a pattern generator and comparing responses using a logic analyzer.
- As in the case of verification, testing by applying all possible input combinations is prohibitive, due to curse of dimensionality problem.
- The testing problem is more time hungry than verification because all chips need to be tested while only "one" design is to be verified.
- Testing by applying all possible input combinations is called exhaustive functional testing, which is avoided because of prohibitive time requirements.

 NPTEL

So, what we have to do? You have to test them by applying some test patterns. When the circuit it has been fabricated apply some test patterns. And you see they are proper or not. So, again if you have of dimensionally problem is there if you are to do for with 2 to the power of n and where you have to apply for that for 2 to the power of n test factors. Now, we have to open for all the chips we have to do.

So, there is a big problem over here more big problem than verification because verification has to be done only one design and testing has to be done for all the chips manufactured. So, somehow test what is test testing what you will do? If there n inputs in the circuit or if we are can trying for all to 2 to the power n vectors that is actually called functional testing of the circuit. This is actually called the functional testing.

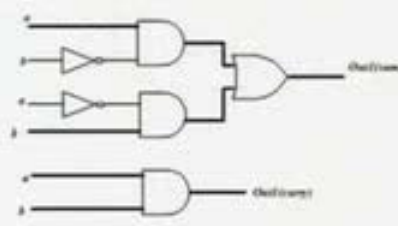
So, this well exhausted in functional testing that is n inputs are there and you are giving all tools for n possibilities to test the circuit. So, this exhausted functional testing which will provide here you see n is about more than 1000 is impossible to do that. So, what is the test flow? Of the course in VLSI flow does it takes you with the subset of 2 to the power n . You have to apply anything about the functionality of the whether these circuits are structured level. Let me explain that you can be that your time will be less as well as you can really very confident to do that. So, we will see in the test model test model of the course there is something called structural testing. In which testing what we will be done you do not know anything about the functional testing whether we treat this


structure as (()).

(Refer Slide Time 01:01:31)

Digital Design, Verification and Test Flow: Test

- Test planning for the adder module of the example assuming that fault model is "stuck-at".
 - In "stuck-at" fault model each line of the circuit is assumed to have two types of faults i.e., s-a-0 and s-a-1.
 - So if there are n lines in a circuit then in all there can be 2n stuck-at faults in the circuit.
- In test planning we need to find input patterns which can determine that none of the stuck-at faults are present.
- In the circuit of Figure 8 as there are 12 lines (9 lines in circuit for "sum" and 3 lines in the circuit for "carry"), there can be 24 stuck-at faults.





Let me explain in the circuit here there is sum n that carries. So, in functional testing you will have to apply that a and b are there. so you have to apply to do the part 2 all the 4 patterns. We have to apply and test, but in case of structural testing you do not think the circuit as adder as subtractor. Here you say structural characteristics of the gates are sometimes called as fault models. So, we will assume that structured formed well known fault model. What is the structural fault model? structural fault model assumes that this line can have stuck at 0 and stuck at 1 second generation circuit means 0 to this line is always sit at 0 warn over here it will not go to here and it will remain as 0. For stuck at 1 in this line always at 1. If you are given 0 over here it will not hold good to 0 that is the stuck at 1 and stuck at 0 fault models.

What is the structural testing will do? You have to apply the test vectors. So, that you can be assure that there is no stuck at 0 and stuck at 1 here and for all the that is the stuck at 0. So, in the circuits how many lines are there 1 2 3 or 4 5 6 7 8 9 10 11 12. So, you can there are 12 lines nests in the circuit. So, 12 into 2 so there will be 24 stuck at 0 and stuck at 1 faults. So, 12 lines means stuck at 0 and 12 th stuck at condition because stuck at 0 and stuck at 1 condition see for all the lines. Here you have to verify that all the stuck at 0 faults here you have to verify that no stuck at 1 faults similarly, for one at a time.

So, you need not take a all the combinations I mean you need not take this is stuck at 0

and this stuck at 0 here no. You take 1 case at a time stuck at 0 and you have to verify them no stuck at 0 faults this stuck at 0 here no stuck at 0 for that. Similarly, you have to take all the stuck at faults one at a time in all the units of the circuit; then what do you do? Then you find out the test vectors which can actually assure that there is no stuck at 0 faults here or no stuck at 0 faults are here something like that.

(Refer Slide Time: 01:03:22)

Digital Design, Verification and Test Flow: Test

- Test planning for the adder module of the example assuming that fault model is "stuck-at".
 - In "stuck-at" fault model each line of the circuit is assumed to have two types of faults i.e., s-a-0 and s-a-1.
 - So if there are n lines in a circuit then in all there can be 2n stuck-at faults in the circuit.
- In test planning we need to find input patterns which can determine that none of the stuck-at faults are present.
- In the circuit of Figure 8 as there are 12 lines (9 lines in circuit for "sum" and 3 lines in the circuit for "carry"), there can be 24 stuck-at faults.

NPTTEL

So, in this case like that so in this case here 12 lines 24 cases will be there which will tell you that there are 24 cases of faults in this. So, if you are thinking about general fault case. So, there can be generally n number of defects like this line can be short, this line can be short with this one this or gate may be converted in to and gate because of fabrication defects. So, this not gate becomes an inverter. So, there can be infinite number of faults possible in the circuit. So, if you want to test for all then you will go mad in time you cannot do that.

(Refer Slide Time 01:04:15)

Digital Design, Verification and Test Flow: Test

- Here we will illustrate for only one fault and the same holds for all the other 23 faults. Let there be a stuck-at-0 fault in the output of one AND gate of the circuit for "sum".
- If $a=1$ and $b=0$ is applied as inputs, then "output1(sum)" is 0 if fault is present, 1 otherwise. So $a=1$ and $b=0$ can verify the absence of fault by comparing output with 1.
- Algorithms and techniques to perform test planning will be covered in "TESTING" part of the course.

So, this structural testing do not think about the functionality and do not think about all kind of defects. Only think that some particular type of faults are there that is stuck at fault that is actually called the fault models. Different types of fault models are there stuck at wheezing. And that we will be in the elaborating in the fall when we go to the testing model; for the timing you just assume that only stuck at fault model is there which can give very good assurance is there does not have any faults. So, none of the lines have stuck at 0 faults. So, how can you verify that? You have to find out the test patterns which can be assure that there is no stuck at 0 faults here or there are no stuck at faults.

Here repeat all of the lines for that one at a time. So, let us explain me with an example. So, like a assume that there is stuck at 0 faults here somehow you have to verify that there is no stuck at 0 faults. Here something like that values here in that stuck at 0 means that is electric bulb in your room that it does not become glow. So, that is always it is 0. So, how can you expect that the bulb is not fused? So, you have to put it 1; at this you have to make a reverse make a values stuck at 0 means try to put up 1.

So, how can you put a 1 over here? You have to put a 1 over here and 0 with the inverter. So, one and one you should get a one, but it may have that because of the fault it may become a 0. So, 1 0 fault is there it will be one. Similarly, or gate so you can put a is equals to 1 you will get the answer 0. So, the answer is 0 over here. So, sum is this is the sum that the functionality. So, if the fault is not there what will be happen? 1 or 0 the

answer will be 1 but if the fault is there answer will be 0 or 0 the answer will be 0. So, you apply the pattern 1 0 1 0 and if you get the answer 1 when you assure that we stuck at 0 faults not at all presents. If you get a 0 when you can know that stuck at 0 fault is there which will have a defect. Now, in this case how many faults are there are 24 faults so 24 faults are there. So, 24 patterns will be equated to test your circuit.

This is a small circuit. So, if you want to apply go for a functional model a b there are only 2 models that would be for only 4 patterns are there. So, if you are go for structural testing we have to apply for 24 patterns one for each faults but actual this is not the fact. So, when we see a large circuit as we go on test model will find then actually reversed. The number of plates in a circuit is n faults can be possible to n. If the practical circuit if than 100 inputs so you can think that there is around 10000 lines in the circuit.

So, there is an approximate numbers. So, on 100 circuit in the there will be around 10000 lines. So, how many faults are there stuck at 0 and stuck at 1; 2000 faults are there 2000 faults are there 1000 lines and 2000 faults; then how may inputs we have to give? You require some 2000 input patterns to verify them at stuck at 0 stuck at 1 faults are there. In this case we have seen that for stuck at 0 faults you require this faults to verify that there is no stuck at faults. But how many input patterns you want to go for exhaustive functional test? It will be 2 to the power 100 which is exponential problem which is it cannot be solved. So, that is what actually called as structural testing.

So, what do you mean by structural testing? We say that we forget about the functionality of the circuit. We take a fault model stuck at model is the very well accepted fault model when we apply patterns or find out patterns like this. So, how many n lines in the circuit because so that which can assure you that none of the lines in the circuit as stuck at 0 and stuck at 1. And it is establish experimentally that this is equivalent to level of confidence equivalent due to fault number of inputs into this is we apply to the circuit that confidence you get.

So, if you have n inputs in the circuits if there are I inputs in the circuits and apply to the I input vectors. And thus confidence you get that confidence is almost equivalent to 2^n test patterns apply for testing the stuck at faults. So, structural testing is widely accepted test fault model or test methodology. So, see in the test model of the course we will see how you given a circuit? How you can find out which are the very important vectors to

test this model? So, which are the important vectors let us 2^n to the power n is actually the super set of nothing but 2^n . So, n is a very small amount of test pattern compared to but how can you determine this 2^n or subset of this? That is we have to apply you have to give faults and all the points in the circuit.

And, then we have to find out automatically by using tools like this those patterns. You have to apply and you will also surprisingly see that this 2^n will actually come through a very less. Because many of the faults level is equivalent and same pattern can be multiple number of faults. So, you must surprise that 2^n are eventually come to very, very less than n by k . There is small fraction of n because of faults collapsing and equivalent of faults this vector can this multiple number of vectors. So, will find out from 2^n to the power n will come to n by k level. So, testing model what you will do? We will get a circuit automatically apply this stuck at faults in all the points in the circuit or whatever form model used an automatically cad tool will tell you; which the vectors to be applied in the circuit it is to be tested. So, this is called the test planning of the circuit.

(Refer Slide Time: 01:08:55)



Now, how was the test plan has to be done circuit is manufactured verified? And it is given to manufacturing and when it is fabricated it will come back to you. So, whatever test plan you have done. So, in this case test plan here means you have to found out the test factors so whatever test plans or whatever test vectors are already we are pre computed and kept in your memory. So, when the chip will be fabricated and brought

back to you so will apply those test patterns which is now 2^n or more precise the n by k . You apply to all the chips and if you find that somehow there is a fault found in the chip. So, you will throughout and all other chips will sell out to the customer.

(Refer Slide Time: 01:09:28)

Digital Design, Verification and Test

The breakup of the modules in this course is as follows:

Design

Module I: Introduction
Lecture I: Introduction to Digital VLSI Design Flow
Lecture II: High Level Design Representation
Lecture III: Transformations for High Level Synthesis

Module II: Scheduling, Allocation and Binding
Lecture I: Introduction to HLS: Scheduling, Allocation and Binding Problem
Lecture II and III: Scheduling Algorithms
Lecture IV: Binding and Allocation Algorithms

Module III: Logic Optimization and Synthesis
Lecture I, II and III: Two level Boolean Logic Synthesis
Lecture IV: Heuristic Minimization of Two-Level Circuits
Lecture V: Finite State Machine Synthesis
Lecture VI: Multilevel Implementation

NPTSA

(Refer Slide Time: 01:10:02)

Digital Design, Verification and Test

Test

Module VII: Introduction to Digital Testing
Lecture-I: Introduction to Digital VLSI Testing
Lecture-II: Functional and Structural Testing
Lecture-III: Fault Equivalence

Module VIII: Fault Simulation and Testability Measures
Lecture-I, II and III: Fault Simulation
Lecture-IV: Testability Measures (SCDAP)

Module IX: Combinational Circuit Test Pattern Generation
Lecture-I: Introduction to Automatic Test Pattern Generation (ATPG) and ATPG Algebras
Lecture-II and III: D-Algorithm

Module X: Sequential Circuit Testing and Scan Chains
Lecture-I: ATPG for Synchronous Sequential Circuits
Lecture-II and III: Scan Chain based Sequential Circuit Testing

Module XI: Built in Self test (BIST)
Lecture I and II: Built in Self Test
Lecture III and IV: Memory Testing

NPTSA

So, this is actually comprises the whole cad tools development process for VLSI design verification and testing for digital circuits. So, what we learnt this course will have 3 basic modules. So, first will be the design module we have already seen the design flow. Next lectures will be on high level synthesis following that will be have scheduling

allocation and binding for high level synthesis. Then we will have logic optimization and synthesis. Then we will have logic optimization that is actually logic synthesis how you form the RTL level how can you go to gate level stuff that will see. So, that will complete your design part.

Next we will have verification part how automatic tools can be developed to verify each translation or where you can go for one portion of circuit to other like from gate level RTL level to gate level or from specification level to RTL level. How can you verify them? So, in that verification part learned by binary decision diagram, temporal logic and model checking. And finally when you will come to testing you will have introduction to testing of some fault stimulation.

How can generate test patterns? For combination of circuits how can you generate sequences and how can you test memories? So, will have a very broad level or in depth discussion or how you can go for good test planning? Only one module which will not be covering in this course will be actually lay out part, backend part how cad tools can be developed from the backend part or you will not be able to go in depth in the course. Whenever you get time we will speak something about that because the course mainly covers 3 major topics in the front end part of the digital design into one course. And that is what we have tended or planned? So, thank you for the first lecture for attending the first lecture. And in the next class we will start going to depth in the cad tools for high level synthesis.

Thank you.