

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATHI

NPTEL

**NPTEL ONLINE CERTIFICATION COURSE
An Initiative of MHRD**

VLSI Design, Verification & Test

Prof. Jatindra Kr. Deka

Department of CSE

IIT Guwahati

Module IV: Temporal Logic

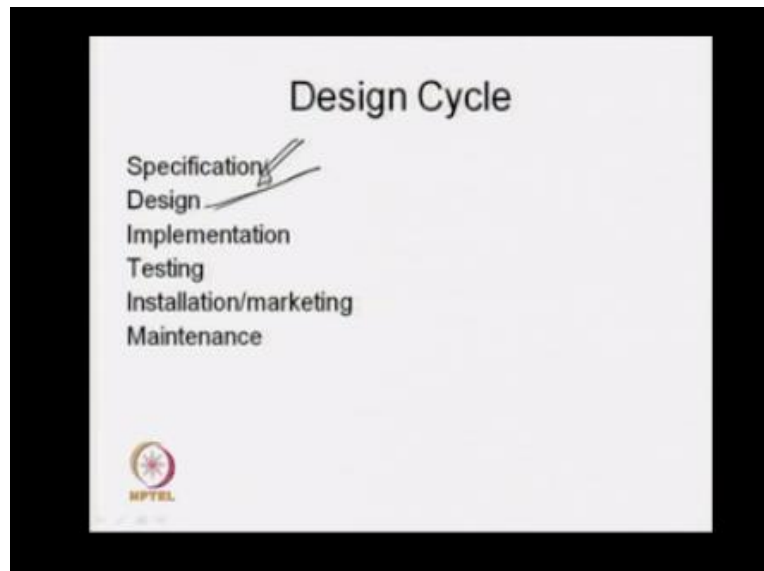
**Lecture 1: Introduction to formal methods for
design verification**

This course is about design verification and test of digital VLSI design. Basically we are having three component in this particular course one is design is of digital systems, second one is verification and third part is your testing about design. In our last part, in the first part already we have seen what are the design which was involved when we are going to design a digital system.

In second part it is the verification and third part is testing of our product. So in this verification part what basically we are going to do, what need to be verified and why we will go for verification without verification whether we can design our circuit, we can develop our circuit or not we are going to look all those issues.

So in this particular part we are going to look for the verification issues and one of the component is our is in our logic for verification. But before going for that particular logic basically we are going to cover about temporal logic, but before going for temporal logic we want to see what are the methods used for your verification and why it is need there. It is basically some sort of introduction I am going to give and I will say why it is needed, why we should go for verification.

(Refer Slide Time: 01:49)



So if you look into it in our design cycle basically we are going to get those following step specification, design, implementation, testing, installation marketing and maintenance. So in our first part basically we are talking about this particular design issues and when we go for design we need to know the specification of the system that means as a designer we are going to design a device and for that we are having some indents under this my designs and satisfy some specification.

First of all we have to keep this particular specification just to give a brief idea of what we are going to do in specification why from that specification will come just will give an example.


(Refer Slide Time: 02:36)

Design Cycle

Specification

Suppose we have to design the controller of a washing machine. There are certain aspects of washing clothes that the system has to take care of, like:

- The drier is activated after the wash not before it.
- Water is poured in before the detergent and it is drained before activating the drier.
- Cold water is to be used in soft wash where hot water in heavy wash, etc.



So we are going to design a controller for our washing machine. What we can say about washing machine, we say that our design or our controller of washing machine will satisfy some property and these property or specification can be mentioned as the drier is activated after the wash not before. So because first they are going to wash the cloth after that will activate the drier to dry out lot.

Second that we can give one specification something like that water is poured in before the detergent and it is drained before activating the drier. So these are basically our requirement, now we can say that these are the property that my controller should satisfy. And another specification we can say that cold water is to be used for your soft wash, because we are having some soft wash or sometimes we need to use hot water for our heavy wash.

So accordingly at appropriate time we have to activate our heater. So these are the several components that we have and according to our requirement we have to activate those particular component.

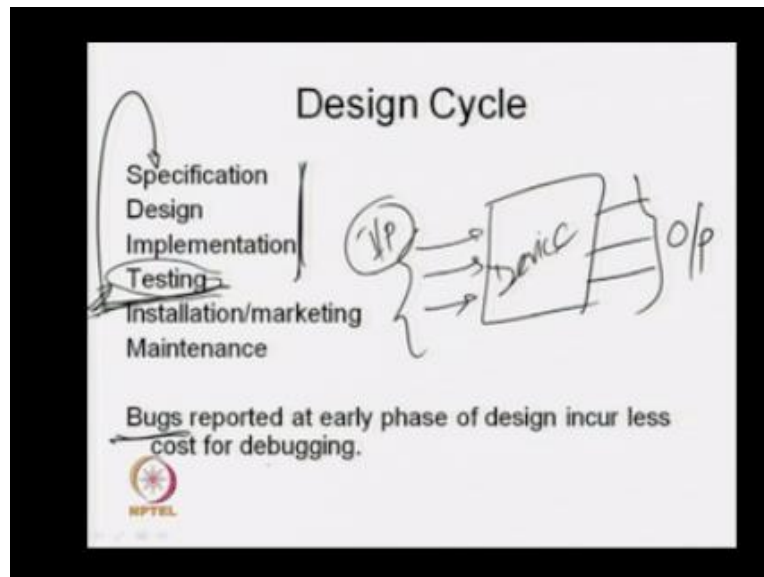
(Refer Slide Time: 03:47)



Now this is about your specification, design and one design is completed, then we will go for the next page is our implementation we are going to implement our controller. After controller is implemented then we need to test it, whether we are getting a correct result or not. But user is we are having this particular testing and when we are doing it, this testing we have done after implementation that means after completion of this particular tree fages.

So that means we are going to test about eh correctness of our product, then it will come for the installation and then maintenance. First we have to install or maybe we market our product and once it goes through market then what will happen, we need them, we need to have the maintenance test, we have to maintain it.

(Refer Slide Time: 04:35)



So here we have one issue that whenever we are going to design a circuit or going to design a system it may not be correct in the very first go. So it says that the bugs reported in the early phase of design incur less cost for debugging. So when we are coming to this particular testing phase we are testing it and how basically we test we will know our input pattern what are the inputs that we have to give along with that we know what are the desired behaviors.

So basically we consider my design I consider that I am designing a device and this is my device I can treat these things as a black box, now there are several inputs available for this particular device and according to this particular input wave we are going to get some output. We know the output behavior of this device according to this particular input combination, what are the input combinations we have.

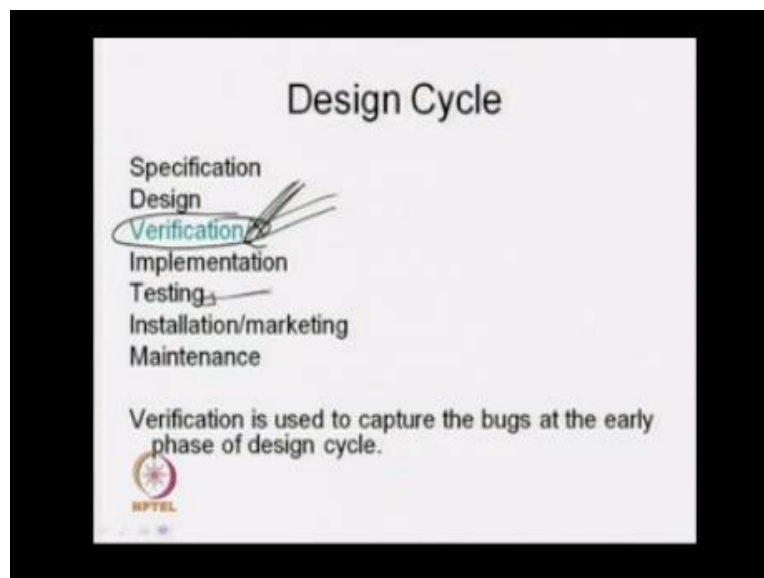
So during testing we are going to test for those particular input patterns and issues where we are getting the desired output or not. So in this particular case during testing we are looking for both testing actually, two types of testing one is your functional testing, second one is your manufacturing defect, because during fabrication, during manufacturing some fault may come into picture due to the fault manufacturing.

So in testing we are going to capture those particular issues also and if devices really your fault then we have to discard it, but if it is having some functional error then what will happen, we have to go back to our specification, we have to revisit our specification, we have to look for our design issues then we are going to redesign it or we will revisit our design. Once you fix the bug basically it is going back over here due to some bugs that has been reported.

So due to those bugs we are going to fix it and eventually again we follow this particular part we may have to do several time. So it says that since after implementation we are testing it and we can capture the bugs at that particular point only after this. So we are going a long way to get our test, so it says that if we can capture the design part your design fall in this particular early phase then what will happen it may reduce our cost.

Because another we are having time to market also. In some specific time we have to release our product to the market. So that is the hurting thing where that this particular test of thing can be done prior to this coming to the implementation or not.

(Refer Slide Time: 07:15)



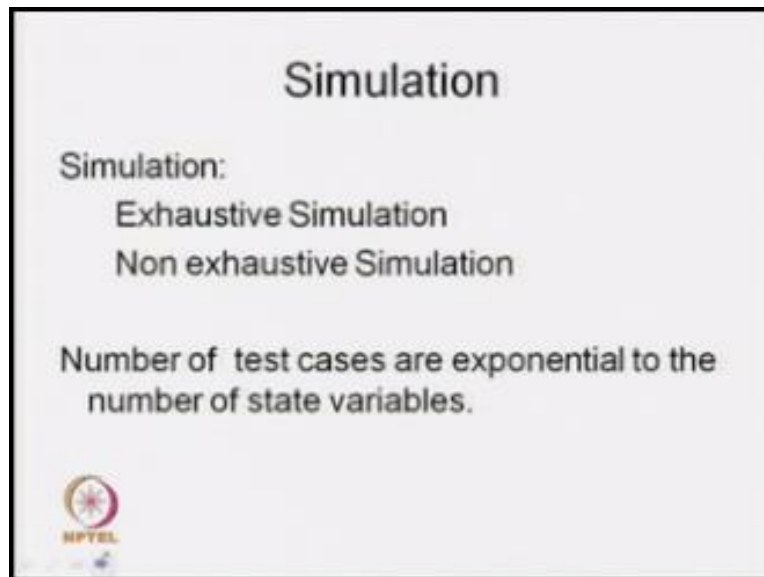
So third is in our design cycle we are incorporating one more phase over here which is we are talking about the verification. And in this particular part of our course we are going to look for those particular verification issues or ways that we are going to verify, what we are going to verify and upto some extent I will say why verification is needed.

And secondly there are several way of doing the verification, but here we are going to look for thermal verification there. So here what we can say that verification is used to capture the bugs at the early phase of design, we just say that we have placed the component verification just after design.

By looking into our specification we have come up with a design and after the design we know that for a specific purpose we are going to design a device that design, that device which satisfy some of the property or some of the specification in this verification page we are going to check for this particular correctness of those property. And we consider this is some sort of your property verification or we can say this is a functional verification we are going to verify the functional property of our device.

Once we satisfy that whatever design we are doing it is going to satisfy our requirement then we will go for implementation and after implementation we will go for testing. So when we are coming to this particular testing point then basically it is remaining left with your manufacturing defect always. So we are trying to capture most of the design issues or design bugs in this particular verification test.

(Refer Slide Time: 08:59)



Now we are talking about here that verification, now what are the different ways that we can do it, whether it was here or not in the earlier phase. So in most of the cases we are having that particular motion of having simulation. We do the simulation of our device and we check whether our design is correct or not.

So basically in simulation once again we think that this is my device, I have designed it and we know that this device is having several inputs and we know depending on the input pattern we should get this as output. So we are going to check those particular desired output with respect to this input pattern. So this is a basic simulation, so we are going to have a simulation method of our device and we are going to check it and if you find that it is correct then we go for fabrication.

But whether it is possible to go for simulation for all type of devices, if you look into that issue you will find that now the system or device being more and more complex and we are having complex design for the devices because now due to the advancement about semiconductor technology you can put more and more component in a silicon wire force so we can go for complex design and depending on our design the number of input is also increasing, so the

number of possible input cases may be we can say it is your 2^n if we are having n number of input signals, okay.

So it is exponentially growing so for that it is not possible to similar for all possible combinations so people all saying that we are going for a selective simulation so that is why it is talking about this non-exhaustive simulation in case of our exhaustive simulation we try to test it for all possible combination but due to the large number the input combination because that input pattern is your exponential in the number being input signal it is not possible to test the enter circuit for all the possible combination, so for that we for selective simulation and we said thia is non-exhaustive simulation.

So what is that is a number of test cases are exponential already I have mentioned so in that particular case we are going to test it for some fixed number of input combination.

(Refer Slide Time: 11:24)


Non Exhaustive Simulation

Instead of using all possible combinations, simulation is done for some selected input combinations.

To find the appropriate subset is a complex problem.

Test case generation

We may not cover all possible error cases.

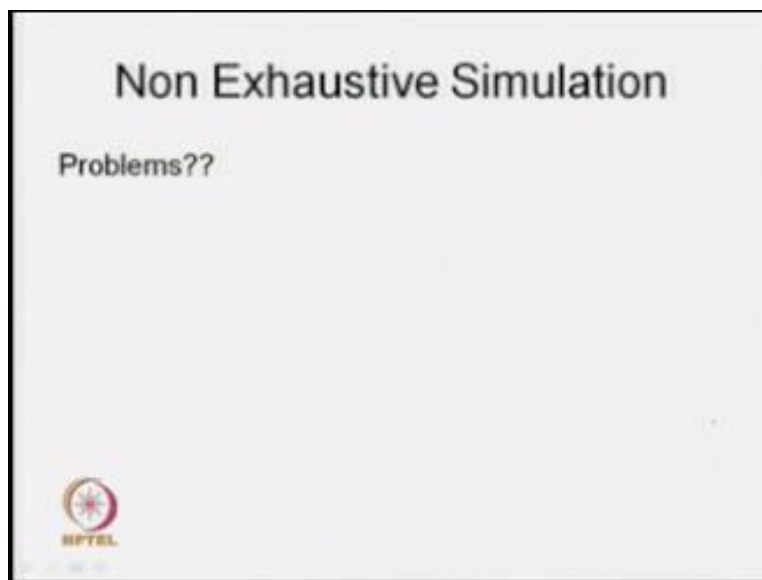
 NPTEL

So here instead of going for your all possible combination we will go for some selective combination so as just I am submerging it to find the appropriate sunset it is again a complex problem so we are having another resource problem over a how to find out the appropriate subset

of those particular input pattern this is come under this particular test case generation. Okay, since this is another domain we are not going to look into this is over here but just I am mentioning at this is another appropriate more we can say that ATPG automatic test pattern generation with the help of those particular techniques we can find out what is the appropriate subset to test our design.

So since we are going for non exhaustive simulation so what happens we may not cover all possible error cases, it is quite obvious. Because it may happens that some of the combination may not be captured by the test cases that we have given.

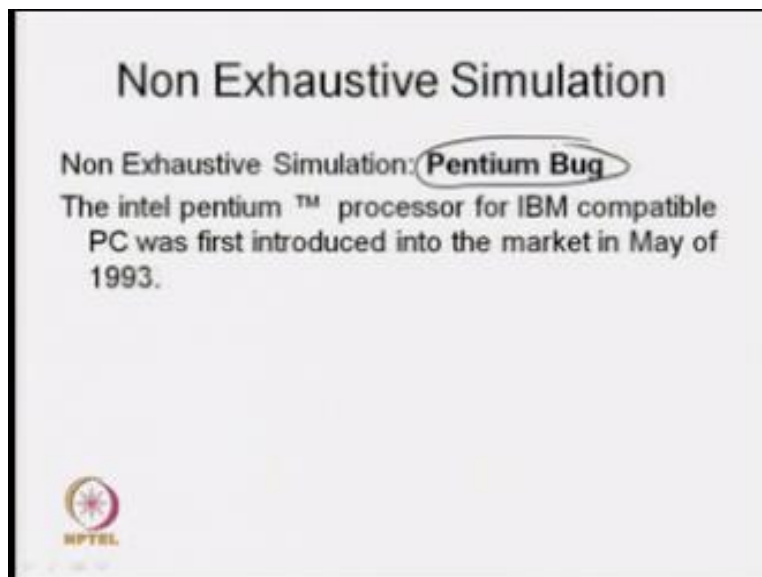
(Refer Slide Time: 12:25)



Now from exhaustive simulation we are coming to non exhaustive simulation so in this case use just see that what may be the problems that we may face, you just think we are not considering all possible input cases and we are giving subset of it, whether that particular subset is capturing all possible error combination it is already I have mentioned that is another problem another complex problem it is difficult to find it out, but still we are having some heuristic with those particular heuristics we try to get a reasonable good or some set of test cases.

So people are working with it, it is going fine but what problems are coming over here why we need to go from your non exhaustive simulation to some other domain that in this class the I am going to cover about your formal specification why we need to go for the formal specification necessarily we have some problems in non exhaustive simulation and it is quite obvious you just sit up here not considering all possible cases.

(Refer Slide Time: 13:28)

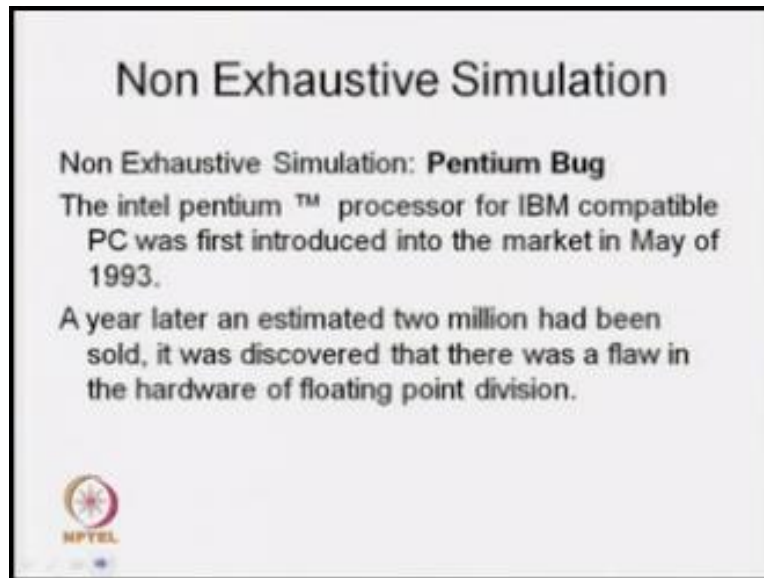


Now is this say now we are going to see what is the problem here is a Pentium bug I think you might have heard this particular term what is Pentium bug and this is a measure saliencies that Intel has receipt in early 90s, so what happen Intel has released their first Pentium series of their product in 1993 so you now said that Intel is a big house is your processor design and if you look into their histoline or time line you will find that they have started with Intel 8086 is the basic processor then they have go with 1H6, 2H6, 2H6 and up to 4H6 they have gone and you know that this is your upward compatibility, upward compatibility means what about we can do in 1H6 17 can be done is 2H6 but with enhance species.

So this is their time line, so after 4H6 they have released the product Pentium series again it is upward competitive whatever we can do in 4H6 same thing can be done in your 5H6 or the

Pentium series. Now say it is upward compactable whatever we are doing in 4H6 same thing can be done in your 5H6, but where is from this bug is coming.

(Refer Slide Time: 14:48)



You just see that A year later an estimated two million has been sold it was discovered that there is a flaw in the hardware of floating point division, you just see what is happening basically said they have released the Pentium it is upward compactable whatever we can do with 4H6 same thing can be done in Pentium along with that we have some of the additional features now Intel design team has design it properly and they have tested it and they have released the product.

But after one year a report has been report error has been reported for efficiently 14 point division which was they are in 4H6 also. So while from this particular error has occurred now you see that in their time line or during the design they have sends they devices on algorithms so earlier they have used some other algorithm but in Pentiums it is they are looking for a first the algorithm and they have used this particular SRT 14 point division algorithm.

(Refer Slide Time: 15:46)

Non Exhaustive Simulation

Non Exhaustive Simulation: Pentium Bug

The intel pentium™ processor for IBM compatible PC was first introduced into the market in May of 1993.

A year later an estimated two million had been sold, it was discovered that there was a flaw in the hardware of floating point division.

It uses the **SRT** floating point division.
(Sweeney, Robertson and Tocher)

INTEL

It is basically a SRT is coming from three scientist algorithm Sweeney, Robertson and Tocher they have divide this particular division algorithm. Since it is a first the algorithm so Intel has decided to use this particular division algorithm in their new Pentium processor. So what is the difference so basically what happens in this SRT division we have to use a lookup table to find out the some input pattern looking into some input combination.

So they have implemented this lookup table in their RAM and due to an error in this particular one end error in one entry in this particular lookup table for some input combination it is giving error, so this was the problem basically this is the Pentium bug so we are having a lookup table we had some entries in the lookup table but there is an error in one entry and whenever we are going to use that particular entry it is giving as a wrong result.

But see Intel people have tested it how they have tested it they have gone for non exhaustive simulation okay, so with non exhaustive simulation already I have mentioned that all it is not possible to capture all error because you have very much selective about our input test pattern so some design from on error is slept over so this is the case what this particular designer has slept.

So in for depth after this particular Pentium bugs so it is an intelligent big business house so they can sustain such type of your loss because what it will depends.

(Refer Slide Time: 17:31)

Non Exhaustive Simulation

Non Exhaustive Simulation: Pentium Bug

The intel pentium™ processor for IBM compatible PC was first introduced into the market in May of 1993.

A year later an estimated two million had been sold, it was discovered that there was a flaw in the hardware of floating point division.

It uses the SRT floating point division.
(Sweeney, Robertson and Tocher)

Since about two million shapes they have released faulty what they did they call back all those particular shapes and replace it by a new one and for that they have in a loss of some about 500 million dollars. Since Intel is a bigger they can cop up with this particular error, so from this particular error all idiot components lined and ensured and they founded or deputing that non exhaustive simulation is not the solution, you are coming for exhaustive simulation to non exhaustive simulation but the non exhaustive simulation is not the right one to do it is not going to give an error free design. At that one people are thinking now what to do.

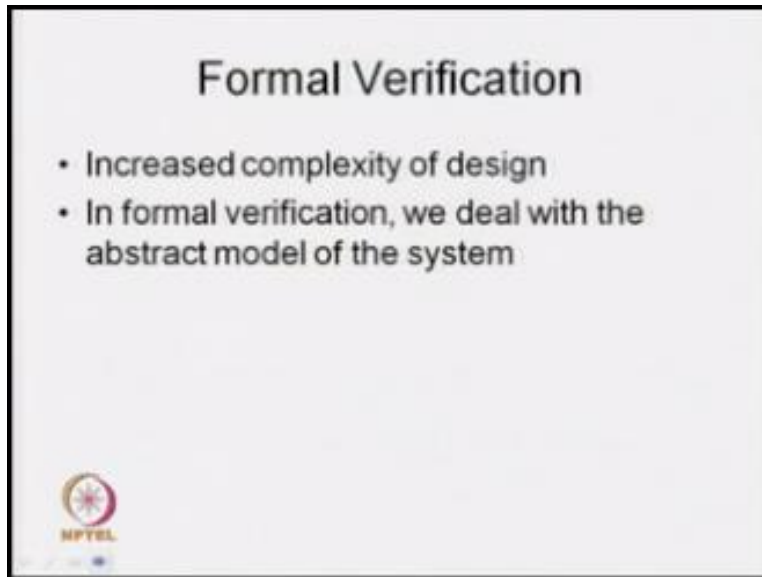
(Refer Slide Time: 18:14)



So in that case this motion of formal verification is coming into big show which is not like that we are encounter is formal verification after Pentium bug it was their which is a walking with this particular matter for how to formally capture a design how to form a specify out property and how to check that the specification or the property or sew in the particular design so this is basically formally your time to capture it was there resources were working over there. But industry people are slightly reluctant to using, but after Pentium bug all people has think about it now you should look for the alternative.

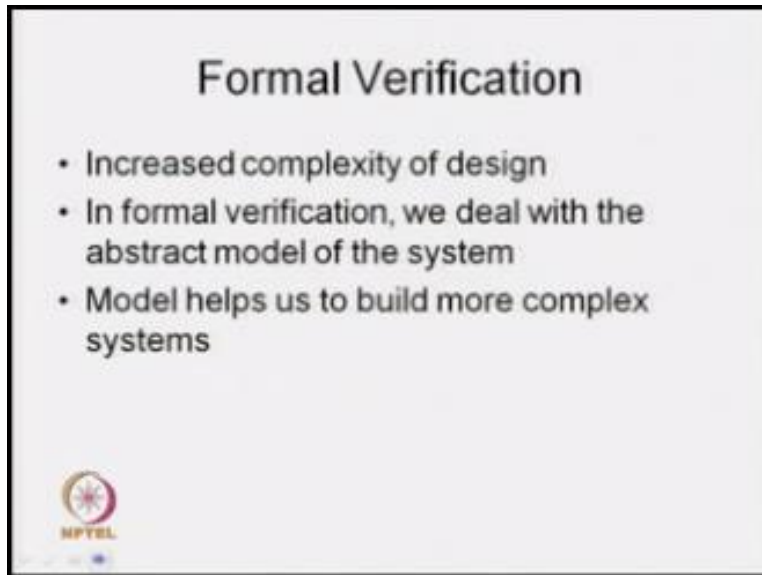
So people are going what is particular formal verification and what are the issues that we can let that is over here already I have said that it is the design complexities increasing thereby there because in a small silicon area now we can put more and more components and indirectly it means that we can press more and more devices in same or small silicon area that means we can place a complex circuit of complex device in a small silicon area that is why the complexity of the design is increasing.

(Refer Slide Time: 19:21)




So what we do in formal verification basically we start from our abstract model so we know our design we know our objective but instead going to the direct product or direct design what happens we try to abstract out the relevant information and here coming within an abstract model and generally we deal with this particular abstract model, so how to come up with an abstract model for that we need formal methods we use some form methods formal mechanism and after we apply some verification techniques and that is why is analyzer this is the formal verification.

(Refer Slide Time: 19:59)



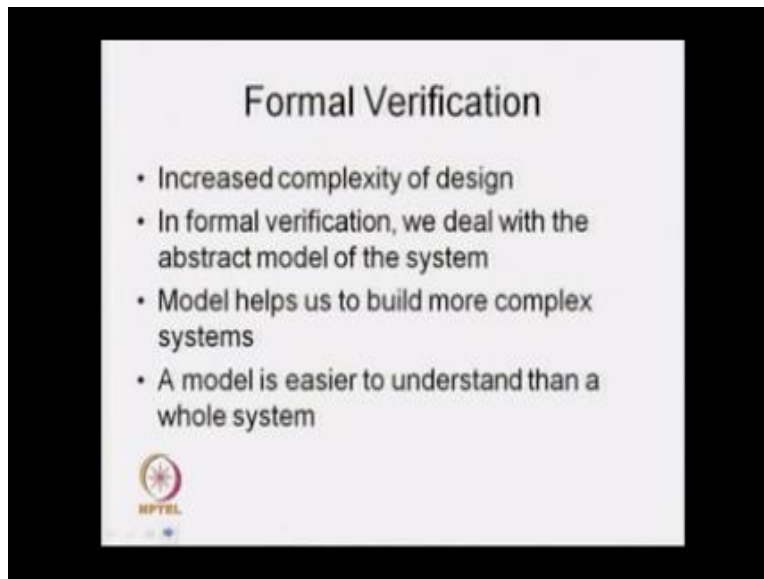
Formal Verification

- Increased complexity of design
- In formal verification, we deal with the abstract model of the system
- Model helps us to build more complex systems

 MPTEL

So model help us to build more complex system already I have mentioned that whatever is given whatever is required we tried to abstract it out we have a design out of it so some sort of model design we can think about it and for applications we can have small model we tried to capture the design behaviors and we know what it is properties or what are the special natures satisfied we are going to look for the satisfaction of those particular properties.

(Refer Slide Time: 20:33)



The slide is titled "Formal Verification" and contains a bulleted list of four points. The NPTEL logo is located in the bottom left corner of the slide content area.

Formal Verification

- Increased complexity of design
- In formal verification, we deal with the abstract model of the system
- Model helps us to build more complex systems
- A model is easier to understand than a whole system

NPTEL

So again on the other hand the model is easy to understand whole system because we are coming down to a smaller piece now of our whole system.

(Refer Slide Time: 20:39)



So in this particular we are going to construct a model in which we can demonstrate a certain property holds so we need to test on system model and second one is the specification of a property as I said already I said that I am going to look for models somehow we are going to give our design and we said that this is our model and another one we are going to have specification or the properties we know that I am going to design a new device.

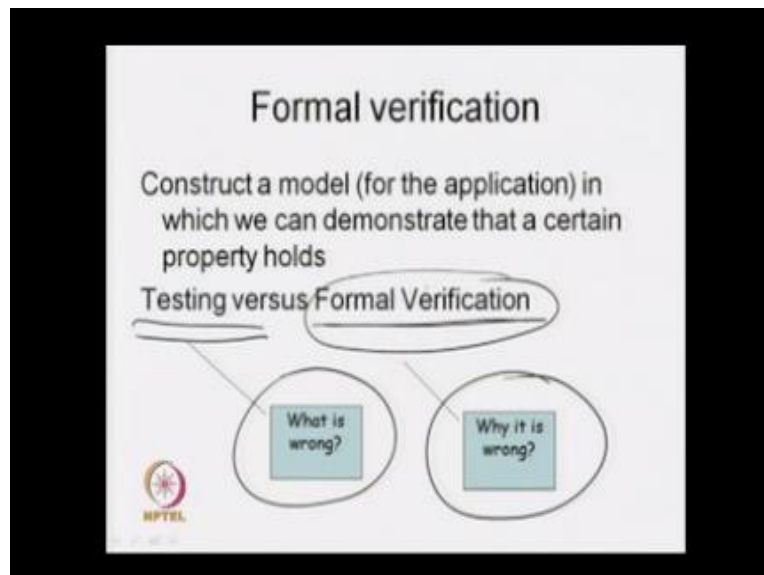
And that device should satisfy some of the properties already I have given one example given that you are going to design for your washing machine and you said that it needs to satisfy some of the properties secondly who look into formal verification of formal errors it is not like that you can apply on your designing of circuit or hardware circuit it can be applicable in any design okay one simple example I can give say you are walking in say network in computer network and you are designing some new computer protocols transform protocols transport protocols.

Now in most of the protocols what will happen one is to transfer message form source to destination what happens generally that source as it get back the acknowledgement then only the sender will be knowing that the message has been delivered properly in your destination so what is my requirement for the I can set up one sender sends the message eventually get back the

acknowledgement so this is we set up this is the property this is the specification of the protocol that I am going to give.

So I am telling you what is the specification but somehow formally we can specify or capture this particular property similarly that about your system model we have that also formal we have to device we need to design eventually we are going to check whether this property is satisfied by the particular model this is formalism or formal mechanism and we said that is a formal verification.

(Refer Slide Time: 22:42)



So again now already we have the testing and now we are coming to the formal verification now what is the difference between the two things so what is testing and what is verification basically in testing we talk about what is wrong so this is basically testing so we that is we have the design we have the product we tested and we said that this is the testing result and what are wrong in the particular case but in case of verification when we are going to do it formally then we can just try to address this particular issue why it is wrong

So just after you design face we will do the verification and in verification we will give the feedback why something is going wrong why it is wrong so when design gets this particular feedback what they can do they can revisit design they can fix up the bar they can fix up the error and they can rectify the errors and they can come with the new rectified design so with this new rectified design again we apply this particular verification and we are satisfied that it can now satisfy all my properties.

All my requirement and then we proceed further then we will go to the next next is implementation of verification.

(Refer Slide Time: 23:58)



So how we are going to this formal verification so we have several approaches to go for formal verification designing the system formally and specifying all their properties and looking for the correctness of the property of design since there are several mechanism and several approaches are there but in our course in this particular course we are going to look for the logical formality we will see how the logical is used for formal verification we may not go for the other issues so if you look into this particular issues when you come for the logic we know that we are having

three different type of logic one is propositional logic which is basic one and then we look for the first order logic and higher order logic.

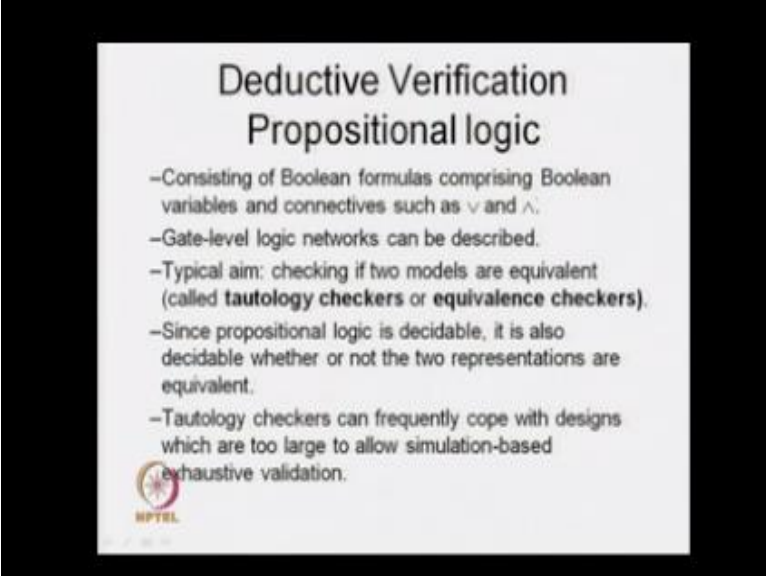
These are basically predicate logic we define predicates and we try to reason about these particular predicates so in propositional logic these are basically we are dealing with the declared step mans and we are going to check whether something can be derived from some other given declared statements are not okay.

So like that we go for first order logic it is predicate logic we go for the higher order logic so in this way we can go up for higher order logic and we will see now what can be done in propositional logic or when we need to go for the higher order logic and what are the difficulties we are going to face if you go for the higher order logic since if you look into the this particular order this first I am showing propositional then first order then higher order so that means we are going to for basic logic for more and more complex logic.

When we are going for a complex logic basically what we can say simply I can I say we are having more aggressive power basically what we can express in the propositional logic something can be expressed in our of first order logic secondly again if something cannot be expressed in first order logic that thing can be expressed in the higher order logic so the expressive power is more means we are having more expressive power so logic becomes more complex.


And reasoning on this logic again become more complex so if we need going for more expressive power then we have to go for a complex logic and our the design that logical inference will be your complex one.

(Refer Slide Time: 26:28)



Deductive Verification
Propositional logic

- Consisting of Boolean formulas comprising Boolean variables and connectives such as \vee and \wedge .
- Gate-level logic networks can be described.
- Typical aim: checking if two models are equivalent (called **tautology checkers** or **equivalence checkers**).
- Since propositional logic is decidable, it is also decidable whether or not the two representations are equivalent.
- Tautology checkers can frequently cope with designs which are too large to allow simulation-based exhaustive validation.

 NPTEL

So propositional logic what we can do is simply I am just going to give an idea that what can be captured in propositional logic and what we can use this is some introductory things I am going to tell you and after we will go for how we are going to do the verification so in propositional logic we are going for a deductive verification what we have here is a setup consisting of Boolean problems comprising Boolean variables and connectives such as AND, OR etc.

So we are going to work out Boolean formulas and Boolean formulas will be constructed with Boolean variables and with connectives all we know is that WFF in propositional logic which is basically well formed formulas we have to just rules to construct well formed formulas and we are going to work with the particular WFF and you know that whatever we are going to look for a digital device or digital circuit it can be always expressed with the help of Boolean formulas.

So it is a set of Boolean formulas which is going to give the behavior of our systems so this Boolean formulas can be again treated as a subset in our propositional logic so that is why I am saying that gate level logic network can be described with the help of this Boolean formulas plus propositional logic and in typical I mean how we are deduced by because what basically we said were the two models are equivalent or not called tautology checkers because in our design

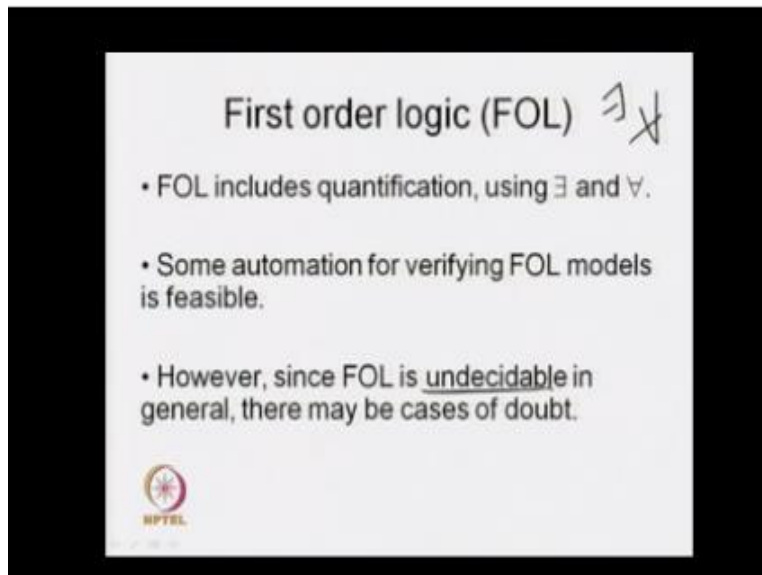
face we are having an abstraction first we come with a model then we and we will make it more refine we will go for another level.

Now after that we have to check we are going to get equivalent translation or not so we can look for such type of equivalent checker so that these are basically sometimes you can say tautology checkers or equivalent checker again we said that since propositional logic is decidable now we have to see what is the mathematical theory of the since it is desirable we can say that it is decidable whether to represent the equivalent or not again tautology checker can be frequently coupled up or not with the design which are too large to allow simulation based exhaustive validation.

So we have problem with the exhaustive simulation so now tautology checker can couple up with those particular problem we faced in the exhaustion so in a large simulation what we can do so basically digital system of the digital system can be a there is a combination of your billion formulas and these billion formulas can be mapped to an analogical step man can use the propositional logic to reason about those particular formulas.

So this is on a but we know that the expressed power is less so whatever we can do is also restricted then next level we are going to talk about first order logic.

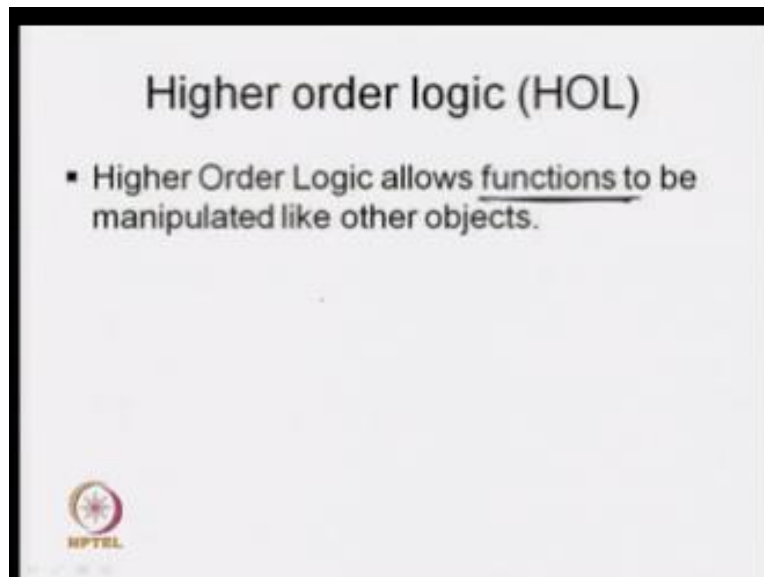
(Refer Slide Time: 29:14)



Or first order predicate logic we know that this is the basically what happens we are going to use to quantify our basically they are exist and power on and with this particular quantify we tried to capture a state of step mans with power all is something is true or the there is no existence something is true or not so with the help of this thing that means this is having slightly more expressions power secondly we are having one since it is a FOL is undecidable say in general so there may be cases of doubt.

So when we use first order logic for your reasoning system after doing it we may have some doubt in some cases because it is undecidable for the manual in the reason is required and in this some automation for verifying every model is feasible but always not true so automation is not possible for all case. But some time when we can automation also so that means manual intervention is always required when you go for fast order logic or higher order logic.

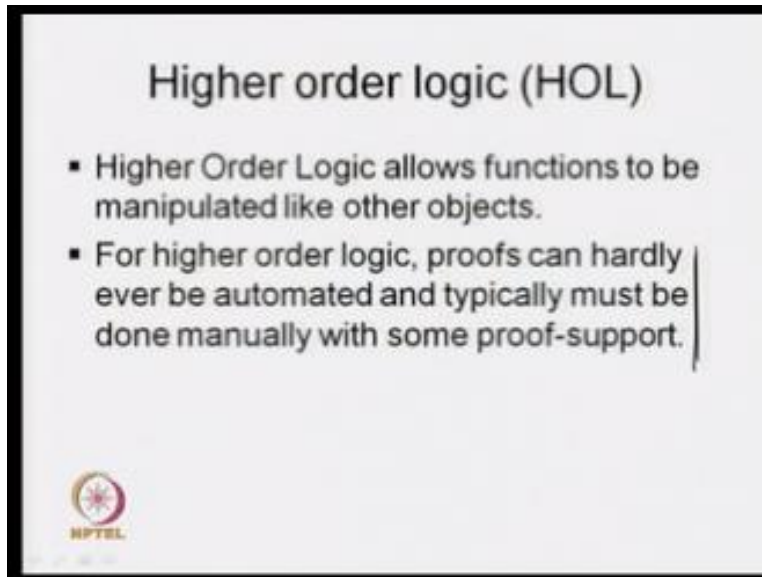
(Refer Slide Time: 30:18)



So next is higher order logic again I say that this more expressive and so dough logical it is become complex so in that particular spot that advantage we are getting what is the expression that we are getting it says that it allows function or function to be manipulated like other object so that function can be treated as an object that thing cannot be captured in your first order logic but higher order logic we can capture that means now it becomes more expressive so expressive power is more.

So that means now we can express many more thing it higher order logic but since it is complex now lessening with higher order logic is also complex.


(Refer Slide Time: 31:00)



The slide is titled "Higher order logic (HOL)" and contains two bullet points. The first bullet point states that Higher Order Logic allows functions to be manipulated like other objects. The second bullet point states that for higher order logic, proofs can hardly ever be automated and typically must be done manually with some proof-support. In the bottom left corner, there is a logo for NPTEL, which consists of a circular emblem with a star and the text "NPTEL" below it.

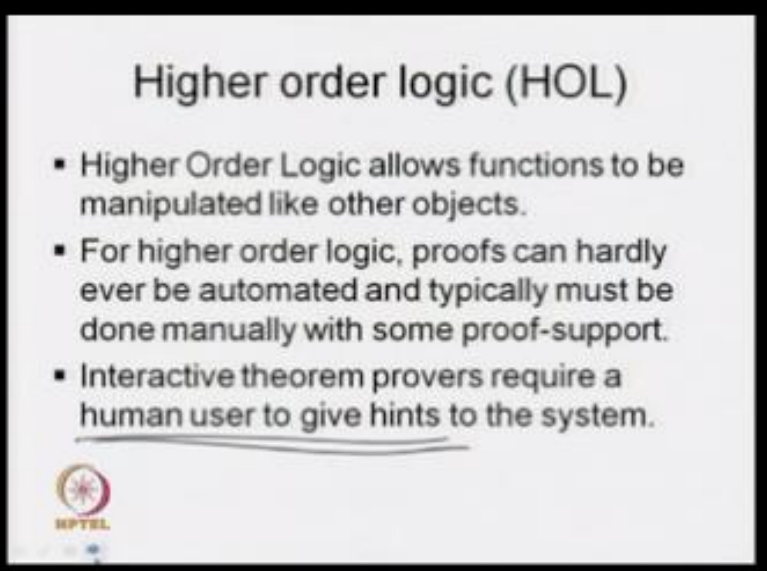
Higher order logic (HOL)

- Higher Order Logic allows functions to be manipulated like other objects.
- For higher order logic, proofs can hardly ever be automated and typically must be done manually with some proof-support.



So for higher order logic proofs can hardly ever be automated and typically must be done manually with some proof support so this is the findings that we have so it is hardly can be automated because now it is a complex automation is not possible but partial automation may be possible.

(Refer Slide Time: 31:19)



The slide is titled "Higher order logic (HOL)" and contains three bullet points. The third bullet point is underlined. In the bottom left corner, there is a logo for IPTTEL, which consists of a circular emblem with a star-like pattern and the text "IPTTEL" below it.

Higher order logic (HOL)

- Higher Order Logic allows functions to be manipulated like other objects.
- For higher order logic, proofs can hardly ever be automated and typically must be done manually with some proof-support.
- Interactive theorem provers require a human user to give hints to the system.

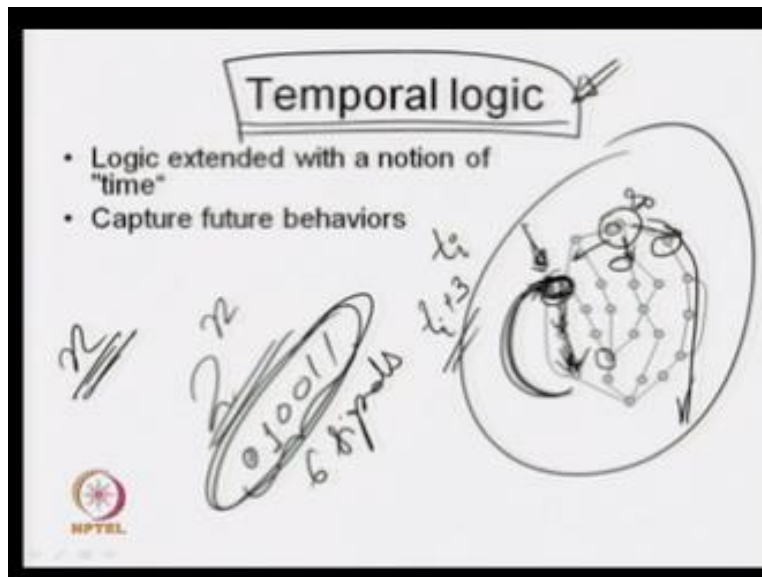
IPTTEL

So we consider interactive theorem provers requires a human user to give hints to the system now we are having interactive theorem provers interactive we are going to check from one sentence to another sentence or one statement to the another statement from some step it may go in a automated word that means we are having some formal way of doing it but after it human interaction is required.

That means we have to give some input so that my proof directs to the correct direction so that why you are saying that we have to human user most give some hits that means we have to give some inputs to go into the proper direction when we look up the reason with this particular higher order logic so this is basically the you know about these things proposal logic you know first order predicate logic you know second order predicate logic on words.

We said is higher order logic the simply I am telling you what we can capture with this particular logic and in formal verification when we are going to apply those things what we are going to use in this particular lecture I am going to tell about it and what we will be leading for death we need some more things but that will be based on this three logic only that is why I have mention about these logic.

(Refer Slide Time: 32:34)



So here we need one more information this is about time so you know that we are walking with our digital signals and we are going to have controller to walk with our digital signals and we know that this system is going to be here defined to a at different point of time that means timing will come to picture so you example you think about that washing machines you have spoiled a machine initially it as to keep pore the water after that it should give you the detergent then it will go into the wash mode.

So after sometimes around say you can say the time like 10 minutes or 15 depending on watch type what happens you have to drain out the water so user see that after sometimes we have to drain out the water I can put paste water into this particular washing machine just nay body have lead that timings coming to picture so somehow we have to capture this particular timing users we need some fomentation we need some mechanism to specify those particular timing issues.

So power depth we cannot do with this particular logics like propositional logic or predicate logic for that we have to go for and this is basically temporal logics that we are going to talk about it so this temporal logics and the methods proposal method that we are going to discuss in

this lecture based on this particular temporal logics so what is the temporal logic already I have mentioned that I can set a accidents.

And no some to time actually we can capture the time and we can capture the further behavior now you just say that what is a system already I have talk at somehow I have to capture the design resources we have some up with model in most of the cases we are going to get a finite model all way if you say slightly remove that timing issues because in case of timing it may be turn up to reactive system which will repeat the something from several times.

So in general you can say that we are going to walk with our signals we have are having fixed number of system and most of the cases we are going to finite step machine so the step is always finite and what will be the test it we know that I am working with n variable or n control signals then we do not know the number of different combinations that we are going to get is you to 2^n that means what about week may be your n always we are going to get a finite steps.

And average steps are going to talk about some configuration that signals value either 0 or 1 we are going to talk about digital systems so that is why I can think about such type of your model so we can say that initially my system is your say this is say m state this is the state s_0 then depending on my input behavior or depending on input signal or the system where it can take go in three different way so I can this depending on some condition either I am coming to this particular state are we are going to this particular.

If I am coming over that means by excuses will follow this particular path so this is the way we can capture over this particular path so this is the way we can capture over this and here user say that this step I am in a particular time now here the time is different I get user see that here I am going back to this particular state so this is the execution twist and going back this now say this particular step basically this is this side or this is going to give it an binary encoding or might this particular n inputs involved because this step I am going to define with this particular signals only.

Out of this signal is high or signal is low basically if I am using signal may be high or low so this may be one particular pattern so if am going to have 6 signals if I am going to say that this is a step where the pattern of my signals is this one now say I am going to follow this particular path and when I am reaching this particular step depending on situation I am going back to this particular configuration here again.

This my input signal pattern in same that is why I am coming to sample here but when look in to a timing this is then what will happen value passed encounter this particular issue step where depend that set timing set t_i after this three steps I say consider that every state is having one unit so 1, 2, 3 next time when I have gone back to this value as set that my time is your $t_i + 3$ note this time is different and in our case what will happen or issue is to get this particular timing notion and we can do this thing help of this temporal logic.

So that is why I am going to the brief either about temporal logic and this is about this module where basically talk about the temporal logic and we are going to have series of lecture and I am going to give the idea about this particular temporal logic.

(Refer Slide Time: 37: 50)

The slide is titled "Temporal logic" and compares two models of time. On the left, "Linear time" is described as modeling physical time where only one future behavior is considered at each instant. On the right, "Branching time" is described as considering all possible future behaviors at each instant, which models different computational sequences and allows for nondeterministic path selection. A diagram on the right shows a single arrow for linear time and multiple branching arrows for branching time. The NPTEL logo is in the bottom left corner.

Temporal logic

- Branching vs. linear time:
 - Linear time
 - Models physical time
 - At each time instant, only one of the future behaviors is considered.
 - Branching time (at each time instant, all possible future behaviors are considered).
 - Models different computational sequences of a system.
 - Nondeterministic selection of the path taken.



NPTEL

So what basically we are having so I have said that we having two more timing behavior we are going to capture we are going to sue temporal logic for that we are in two way of doing it one we are going to talk about a branching ways of time and second one is your linear measure of time so in case of a branching measure then what happens basically we set up we just think the time progress in one direction only but in case of your branching so this is liner measures sorry I talk so in linear case time is progress in one direction only. But in case of your branching time can branch out in several direction okay.

(Refer Slide Time: 38:29)

Temporal logic

- **Branching vs. linear time:**
 - Linear time
Models physical time
At each time instant, only one of the future behaviors is considered.
 - Branching time (at each time instant, all possible future behaviors are considered).
 - Models different computational sequences of a system.
 - Nondeterministic selection of the path taken.



Now just come back to this particular model so if I am going to loop for distance step from this particular step I can go into three different direction okay so depending on my input pattern or depending on my system configuration it can follow one of this three particular path so that means I can say that the measure of timing having a branching measure at that particular point so it branch out in three different step but if you concern that on a particular part over here.

And thus said that we have in duster for this particular exposition path then we may not look for other issues so in that particular case we say that this the liner time and we are going to capture it by this particular linear motion so this is a basically about timing issues one is your linear and

your branching talking going to listen about this particular system then we think about the non deterministic issues of this particular branch timing we can think any one of this three possible combination.

(Refer Slide Time: 39:31)

Temporal logic

- Discrete vs. continuous time
 - Discrete time
Used by most temporal logics, mostly using natural numbers to model time.
 - Continuous time
Using real numbers

N
1 2 3 4 5 ...

R

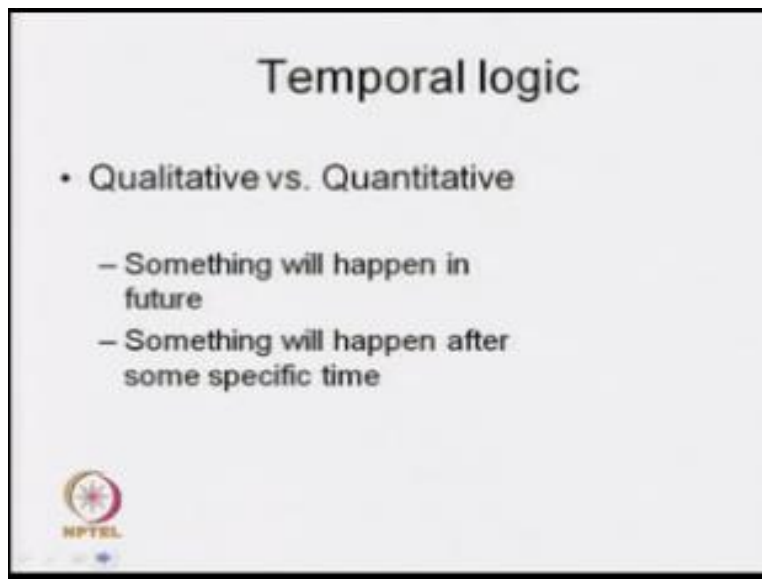
MPTEL

Again we are giving another issues which is called time how we are going to capture the time one is your discrete in measure and second one is your continues time so basically if I want to talk about out digital system we basically walked in discrete domain so time can be captured in a discrete way voice and we can use the measured number system to give all times so I can say that the arctic 12345 like that in discrete way I can define of time.

And we can say that this is discrete in measure and in case of continuous timing we are going to walk it a real number system well every timing instant is possible and may try to design about every timing instants. Now by just I am saying that either we can use the disked time or we can use the continuous time you know about a properties of real number and you know about the integer on nested numbers.

From here itself but you can physicals that designing about the continuous time will be your complex one yes indeed it is really a complex issue but designing about discrete system slightly easier okay but some system we need to design with your real number also or continuous time coming also.

(Refer Slide Time: 40:44)



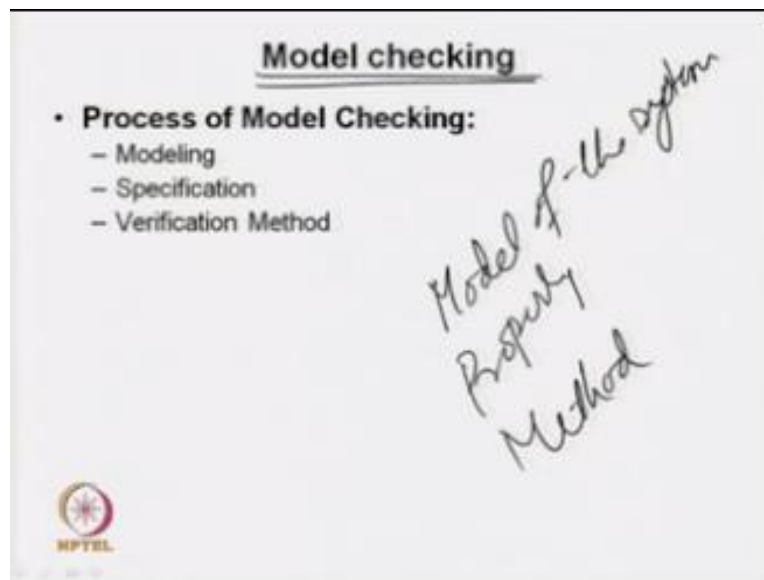
Another issue we have going to capture timing and the issue we have is about qualitative free designing and quantitative designing time qualitative way and quantitative way so in case of qualitative way we just talk about a behavior of the time we say that whether something is happening now or something is going to happen in future okay just say that I am talking about that particular issue about that nacho protocol when send a message eventually it should get back the acknowledgement I am talking about eventually it should get back the acknowledgement that means in future sender is accepting the acknowledgement okay.

So this is some of the qualitative timing wave I am saying that I send the message now in future I should get it but in case of quantitative designing we specified a quantum optima also in quantitative designing what we said at now if send the message now whether after five unit of time senders will get back the acknowledgement so this is a quantitative we have quantifies a

future behavior future behavior say that after five unit of time whether send our get the acknowledgement or not.

So these are the issues that we have to look in to the in term of logic we are going to look for those issues.

(Refer Slide Time: 42:09)



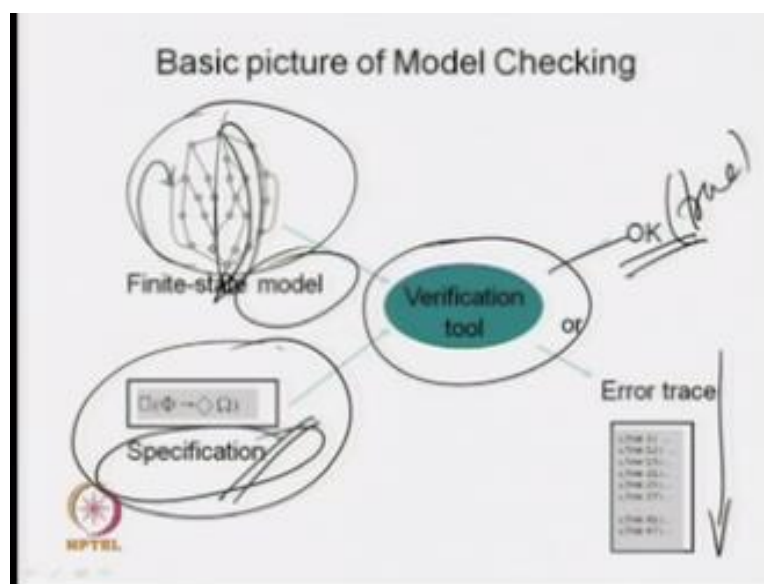
Now we have going to loop for already I have mention that in this particular lecture we have going to look for a particular mechanism formal mechanisms to prepare what is them and we are going to look for this particular module seeking up prose okay so now just I am going to talk about the this particular model second so what is the recoating you know model checking so we are having basically three components one is your modeling one is your specification and third one is your verification method.

That means we need some formalism two give the model of the system okay so you have to give the model of the system then we have to some of specify the property and we need the method to say whether this properties are through operating this system or not, so basically in model second

we having this three components. Since you are going to look for the facility of the properties are specification it is also term there is you property verification.

We are going to verify the properties about system so it is a property verify system. So basically we having these three components you know model checking now you just see that to what we are going to do in this model checking.

(Refer Slide Time: 43:34)



So what are the basic components about model checking so we are having a verification tool so this is our model checking tool and we are having two different inputs to this particular model checking tool one is your model; and second one is you specification and I am talking about this model as your final touch because already I have mention that generally you are going to get a finite number of steps our design.

Because it depends on the number of variable that we have if we are an n number of variables the total possible combination is 2^n and we are going to get 2^n depends that. Now when we talk about the time now present instants I am instate but in future I can come back to that particular state again. But still excluding that particular time we are having the fix number of states but

with time whatever you can say that now we are in more number of times, but again we are going to represent this whole system with half of this finite step model.

And this is basically going back issues going to say this is the general time instants now this is the model our system say that we are going to design some controller or we are going to design some circuit first of all we have to capture this design with the help of this models. Some models we see in what we are going to capture ion this after that secondly what will happen I have little model checking is nothing but the property verification somehow we have to specify how property or we have to give the profile.

So this is the sum in grit notation we have written something and w have said that this is the specification we have given to here, now when both this component will be even is a input to my verification tools and after that now this verification tool is going to check whether they given specification is to in this model or not. So what is the output of this particular model of the tool or the verification tool or the model seek of, if this property is true in this model then it will just give you the output okay or it will say that the two that means the given specification is true in my model.

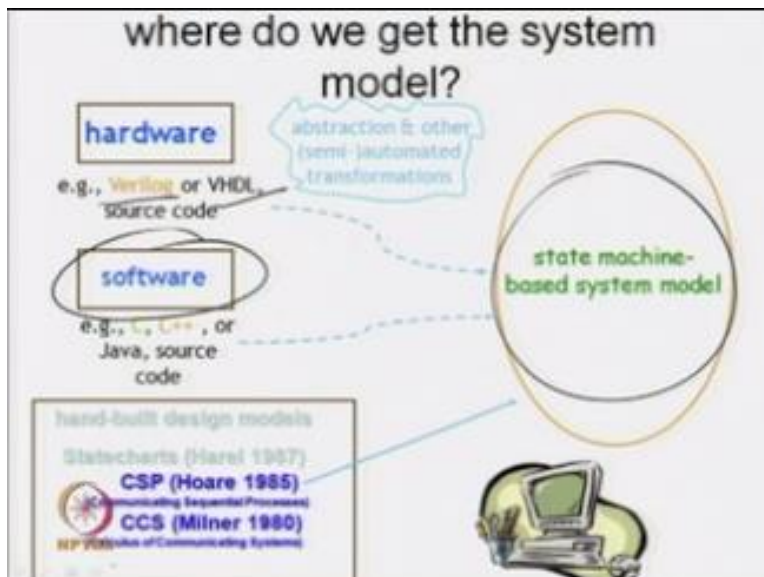
But if it is not true just say that we have come up with the design and you are trying to check whether this particular verification is true or not so the verification tool try to check all possible combination and it finds that this specification is now true over here so that my verification tool is simply it will not simply say that it is falls along with that it will give me some information this is basically it will give this particular error trace.

So in this particular error trace what happen it give me the error trace in such error it says that if you follow this particular part then the given property is not true that means the design that may getting some feedback on this particular verification method now taken concentrate the design is fall this particular execution race on this so basically it may happen on the correct our path are concentrate over here.

So some setup focusing the path and design think can now look in to issues related to this but it does not over rule dept error are not present in the order set only it may present pressures also because some designers going to fix this particular path then next time we will say that may have this sin summer of part of it, but anyway with this particular model checking application by apply a verification when should I can capture most of the error and you can fix it.

So it is some of giving the not only saying that it is falls but along with the give some indication also where does possible error is.

(Refer Slide Time: 47:20)



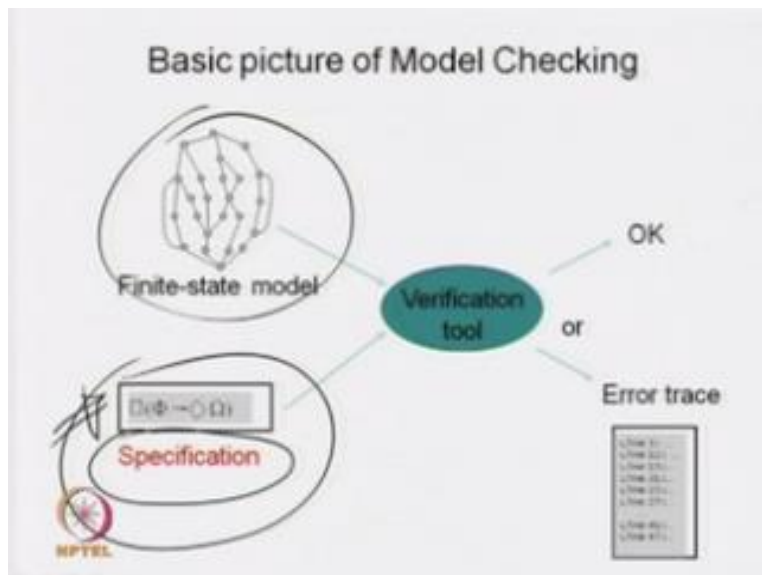
So now where from we are going to get this particular state based machine I am saying that we need a model so in previous slide I said that these are the two inputs one is a specification second one is you model where from we are going to give this particular model so you must see that we all looking for this particular model so define people are designing the system indamines it may be hard ware or it may be software also.

I am already I have mention that since I am going to talk about one method verification method it can be applied for all design cases it may be hardware or soft we are so in case of hardware case

generally what happen we describe our system in article level with the help of some HDL lingers hydro lingers like Verilog or VHDL so we can have those particular Verilog and VHDL if it is a software then we can have C C+ and like that or some other design can be done with the help of that states are which is define by your Hoare CSP CCS.

So these are depend you are having a presentation the forms are capturing awards get that machine okay so we need to have promolisim to get the essential get this particular states in machine for our model checkout.

(Refer Slide Time: 48:37)



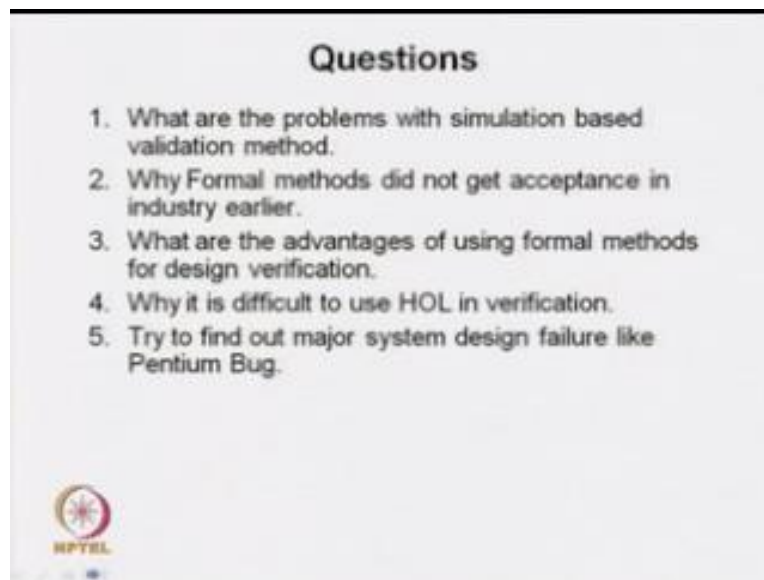
After that this is the things now we have to look for this specification how we are going to give it so this is basically we are going to use tempo logic this specify or property or this involve box diamonds are having the own notation own meaning in this particular model of tempo logic we are going to basically talk about or going to discus about this particular notation what those particular notation means and what we can specify.

So basically now in this case what happens you see the what you have seen that we are going to apply for a particular verification that make which is model setting it next two component one is

our model and one is our specification we will going to look ion it how you are going to give the model and how we are going to give the specification.

So next class we are going to discuss about the tempo logic and it is going to say what is index and symmetric of temp logic and what w can specify what is the acetic power of tempo logic okay so next class we are going to look in to the this is soon.

(Refer Slide Time: 49:45)



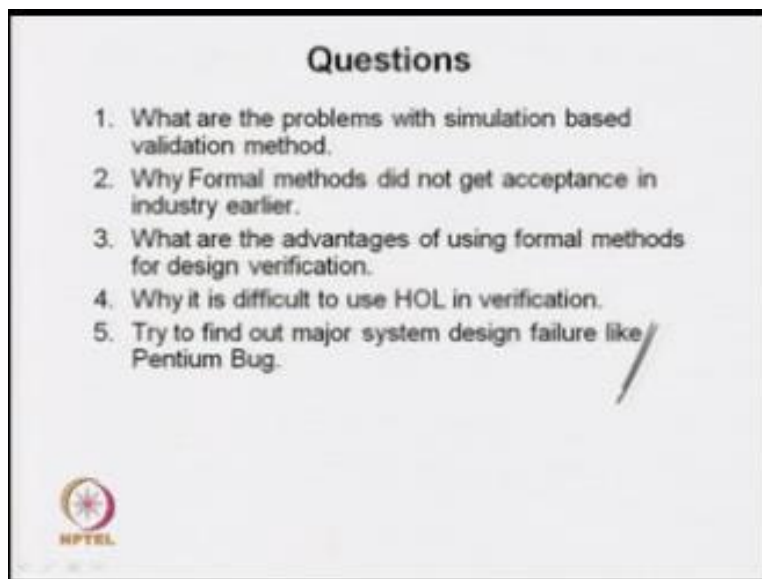
So after going through this particular lecture you just see that some simple problems that you can think about simple question that first question that I am putting it like that what are the problem which immolation where simply does not matter already I have given you some idea about what happens in simulation what happens is you like just simulation on non just simulation why cannot go for the just simulation always.

So this is something like that you can get some more information in your book in your net just see that what are the things why it is not visible at pressure you have to advantage of the technology also it is not possible because it is not in a small silicon place you can put more and

more devices second just now we have already mention in m lecture also why formal methods did not get acceptance in industry.

Earlier because it cannot be auto methods secondly human iterance is required that means we need people formed at particular doming from auto method theorem proving doming because we have guide the theorem provide while doing this things so that is the direct slightly detected and they try do it your simulation baser method but when theorems bogus forced the people to go for this particular formal methods.

(Refer Slide Time: 51:02)



Third question as I am saying that what are the advantages of using formal methods for design verification I have not mentioned probably but in the dual course we are going to talk about it but you just think or you would just expect to explode the information in net and like that one basic thing is that we are going to use formal mechanism to specify our system or to model our system we use some formal mechanism to give the property.

The formulation that we are going to use are having a predefine index and predefine semantics so ambiguity will be remove when will pass to a design team so this is a form design name it is going to the next verification team or it will go to the implementation team then it will remove

the ambiguity basically formally we are defining it so all are having predefine syntax and semantics. Why it is difficult to use HOL in verification already I have mention you can look into it and some your HOL verification methods and you can find some more idea, try to find out measure system design failure like Pentium bugs.

So with this Pentium is not only the bug that has been reported after going to the public summary issues also do you have, some are the devices are also they have which has reported bug after design after it is going to public you will get some information and try to compile this particular information and make a report and see what are the failures that we have due to the design error and what problems that human being are facing due to those particular area.

So that problem by you can think about like that you are going to write a report about it and try to collect those particular information we having several design error fault in our history, so with this I am winding up my lecture today so next class we are going talk about or we are going discuss about the temporal logic about its syntax and semantics, bye, bye.

**Centre For Educational Technology
IIT Guwahati
Production**

**Head CET
Prof. Sunil Khijwania**

CET Production Team

Bikask Jyoti Nath

CS Bhaskar Bora

Dibyajyoti Lahkar

Kallal Barua

Kaushik Kr. Sarma

Queen Barman

Rekha Hazarika

CET Administrative Team

Susanta Sarma

Swapam Debnath