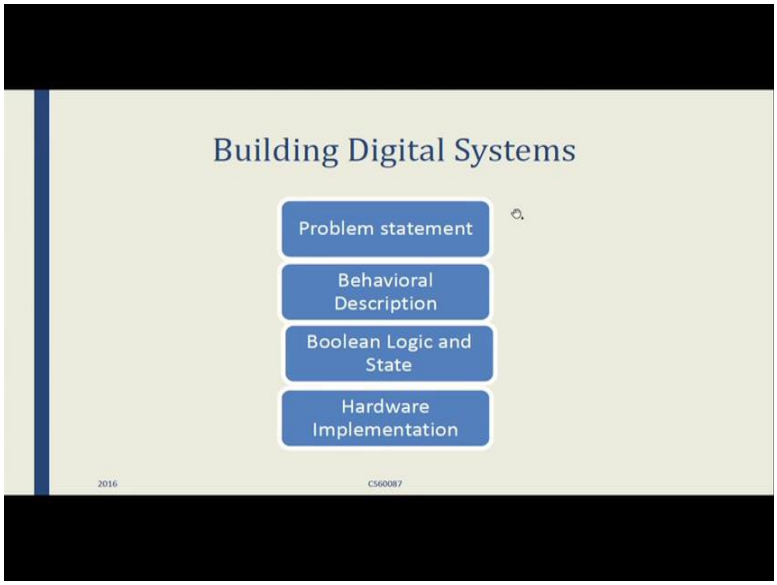
Embedded Systems Design
Prof. N. Vidya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 09
Tutorial – I

Hello everyone. I am Vidya, I will be one of the TAS for this course. And I will be taking 5 to 6 tutorial hour on HDLs and FPGAs. So, today's tutorial will be on HDL introduction will know what an HDL is, how we make an HDL design, why are they used what are their advantages and all.

(Refer Slide Time: 00:49)



So, let us start. Before we define what an HDL is I would like to; let us discuss how we build the digital system.

So, these are the overall steps that are involved in building a digital system. First of all we have a problem statement. So for example, we want to switch on a light if someone ring the doorbell; that is one of the problem statement. And the other problems would be like we have to do some computation of two numbers and get output, like we do some operation on two numbers and we get an output. So, that is one problem statement. So,

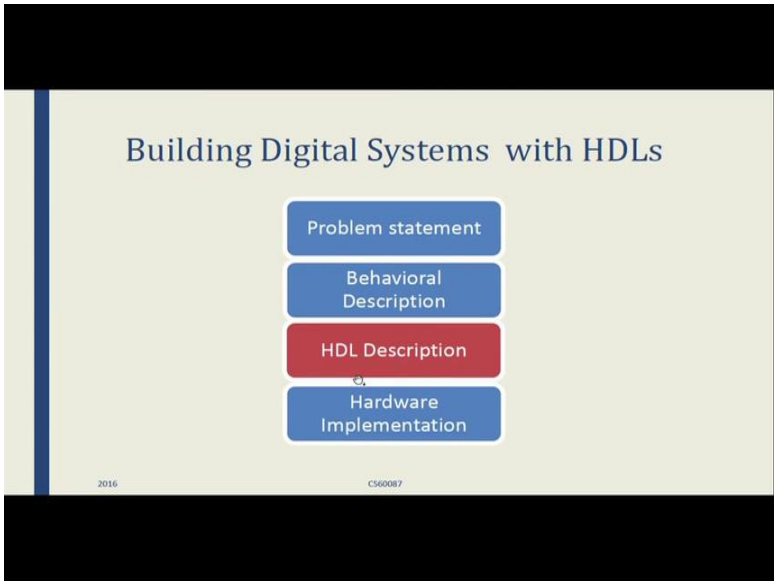
like this we have different kinds of problems statements. So, we convert the problem statement into lab or project specification. So, problem statements convert into to a project specification. So, once we document the problem into a specification. We move on to describe the behavior of the system. So, we go on to the behavioral description of the system.

So, in the behavioral description we have we define the operations functions processes etcetera that is required to convert the inputs of the specification into the outputs. So, that is what is done in behavioral specification. We generally design algorithms, we design, we put the functional operations etcetera in flowchart and RTL description will come to that later what is an RTL and all. So, the problem statement is converted into a behavioral description, and then once we have the algorithm the flowchart the state descriptions and all we convert it into a Boolean logic and state.

This is a by this what I mean is the behavioral specification is converted into set of equation or a circuit schematic so that we can put it on to the real physical device an electronic device. So, we have in Boolean logic and state we have it logic creation and we have a circuit schematic. And once we get we have the logic creation and the schematic we convert it to a particular hardware; for example, and, or, etcetera. So, the equations will be converted into a physical entity.

So, this is a overall view of how digital system is built. In earlier days we used to have the designers used to build hardware by using paper and pen. So, they had to write the equations they had to solve Karnaugh map K map they had to solve everything and I have to like manually do everything as a complexity of the design increased people wanted to more automatic way of doing things.

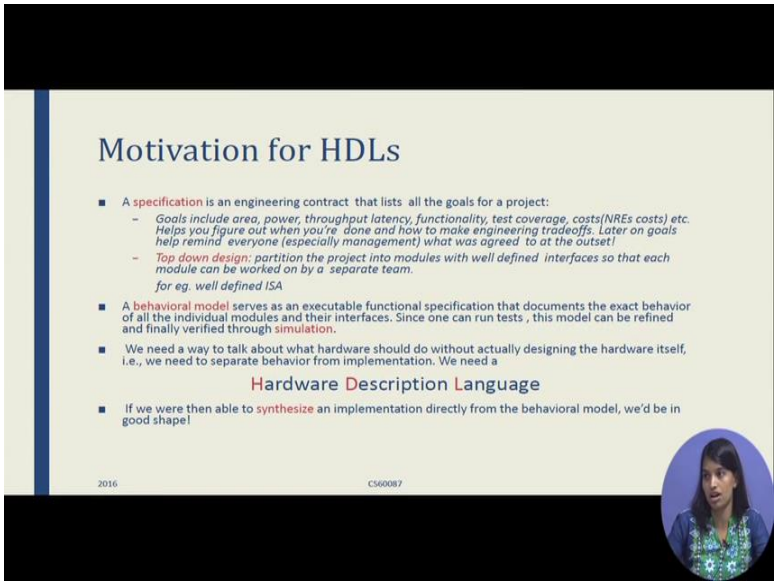
(Refer Slide Time: 03:57)



So next when the complex complexity increases we moved on to replacing the Boolean logic and state as a HDL description. So, this is where HDL comes into picture actually. So, HDL description is basically like helping us to overcome the complexity after designs that we are facing nowadays. So, main motivation behind HDL is like how I have told a specification. We have a problem statement we convert it into a project specification and their specification has some goals.

So, a specification maps into some goals and then a methodology is in which we design a system. So, there are two things: in goals we define how the system should be; what is the cause it is required to build a system, what are the power requirements, what are the latencies functionalities etcetera.

(Refer Slide Time: 04:49)



That uniquely define a system and then we can go back to that goals and check if that that are met or not. In the end and second thing is the design methodology.

So, there are two types of design methodology: top down design and bottom up design methodology. So, in a digital system for example, if we consider a very big system, if you take a Smartphone for example; we have lot of functionality that is integrated into it. So, in a top down approach we break it down into smaller modules and interfaces and then we approach each modules and interfaces individually, so that is the top down approach. And in the bottom up approach what we do is we first do the smaller modules first and then integrate it up to the higher level.

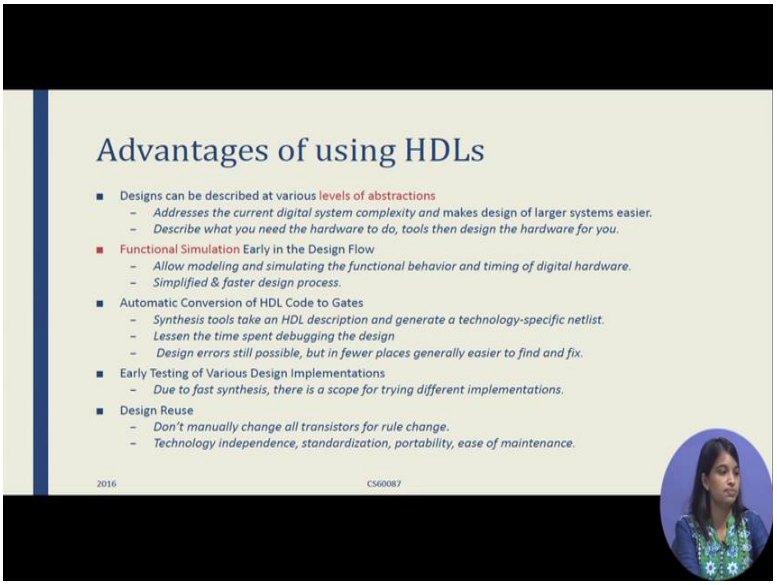
So, generally we follow top down approach because we usually start with the bigger problem and then break it down into smaller problems. For example, that I have mentioned in the slide that we have ISA; that is instruction set architecture of a processor. So, if we have a well defined ISA it will be, therefore for a longer time that we can use it for a longer time without actually facing any problem with the definition of the ISA. So, that is the example of a very good specification.

With this specification actually what it helps us to do is to model a functionality. So, we said that it is a behavioral model. And this behavioral model is used as an executable functional specification. So, we have problem statement, we have a project specification and specification is converted into a model using which serves as an executable functional specification. So, what hardware description language or HDLs mean is that we do not have to know exactly what the hardware is; we want to have a tool chain or something through which we can design the hardware without actually knowing what exactly the hardware is.

So, we want to know what the hardware functions, like we want to know what the mobile does, we want to make calls, you have to receive calls, if you want to send messages, but we do not want to know what exactly how exactly that is done. So, HDL as a way to help us do that, and map of functionality to hardware. So, if I will come to it latte word synthesis it. So, basically we synthesize an implementation of a behavioral model that is what hardware description language does.

And there are like in the current industry there are three HDLs that are being defined that are being used actually; so first is very Verilog, second is VHDL, and third is system Verilog. So, in our discussion we will mostly discuss Verilog and VHDL and in the later tutorials I will take some of the Verilog basics and some advanced concepts of Verilog. So, to Verilog we will try to learn how to write a Verilog code word; basically to build a digital system.

(Refer Slide Time: 08:17)



Since now we know what an HDL is we will look into what are the advantages that is HDL gave us to build a digital system. First are levels of abstraction. So, we abstracted the levels, like we do not have to know exactly what the hardware has to do. So, we have abstracted out the exact implementation from the behavior. So, we just have to know what the hardware does, so that is first that is the most important advantage of HDL, because nowadays we have so much complex systems that not everything can be like decided at the first design phase; so this helps a lot in that.

And second thing is functional simulation: functional simulation is basically we are testing the functionality of the design, whether whatever we are thinking what the design does it is it is actually being; the design does that is for functional simulation does it verifies whether the functionality of the system is right. So, we can do functional simulation using a HDL very early, like before the hardware actually comes before it is build we can do that simulation of that design. So, that is how it helps in functional simulation.

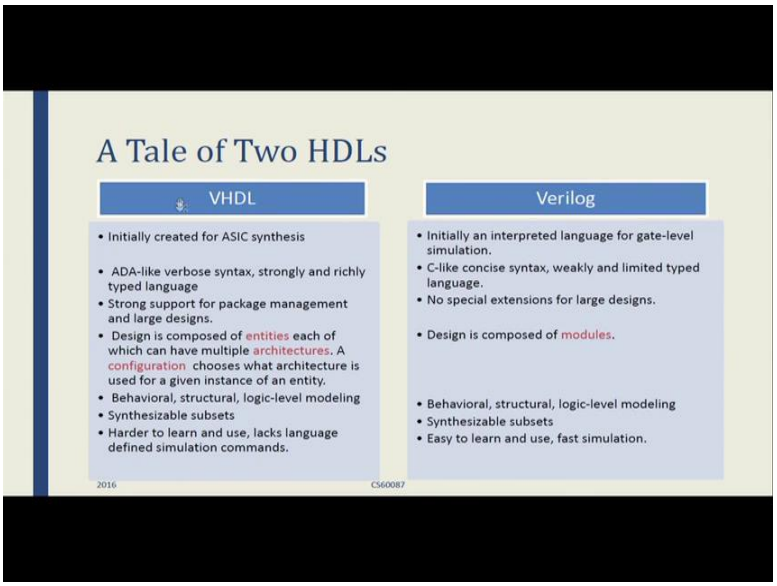
Third thing is the HDL code directly gets converted into gate using some tools that I will discuss later. So, we do not have to like draw the schematic and then convert it into hardware gates and then like it takes time and it we can fix any errors easily if you are

using HDLs. If you are doing it by hand and my paper it is very difficult. And there something called technology specific net list. This also I will come to you later; basically an HDL code gets converted into gates using a process called synthesis and generally we have standard tools to do that.

And four things is we can decide like if for example, we have a design it can be implemented in different ways. So, using HDL we can do that implementation in different ways in a very small time. So we can test, we can design the implementation we can test the implementation. So, that is another advantage of HDL.

Last thing is design reuse that if you have heard I do not know we have IP blocks. So, intellectual property blocks, which have being used reused again and again for different purposes like if a company has designed an IP it can be used in some other companies if that the other source company allows it to be used. So, design is reversible and it is portable across different technologies if we are using HDL. So, these are the main advantages of HDL.

(Refer Slide Time: 11:15)



Next will discuss the two main HDL set there are: there is currently VHDL and Verilog. So, VHDL was initially developed for creating ASIC synthesis as it is Application

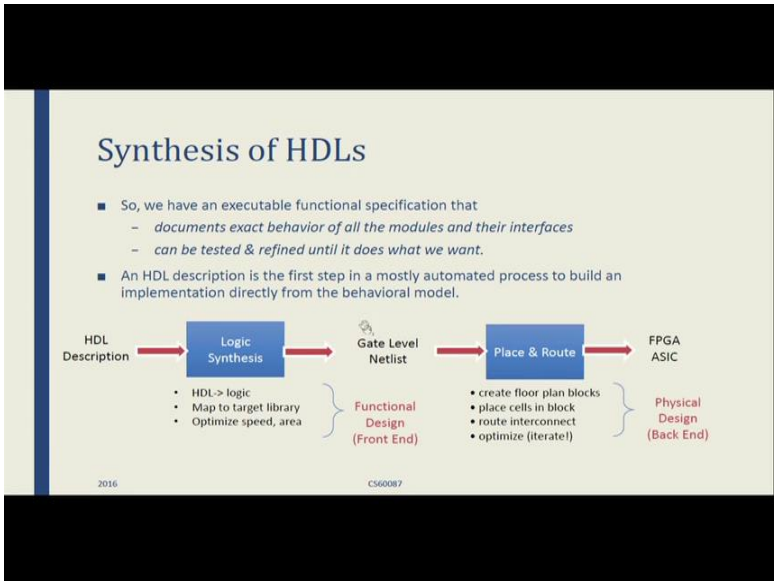
Specific Integrated Circuits. So, for synthesis of ASIC systems VHDL was used and it was developed by us defense of department. And if you see Verilog, Verilog was used for gate level simulation; simulation of the gates basically. And it was developed by a company called gateway design automation and that was acquired by tardiness. And VHDL is very verbose kind of language and Verilog is C like language.

So, it is very easy to understand Verilog and it is comparatively harder to understand VHDL. VHDL has some package management structure and it is support package management for larger design, but Verilog does not have such extensions for larger design. And if you see the way the code is written in VHDL and Verilog: in VHDL we have entities and the will have architectures inside them and act architectures have configuration. So, this is how a basic structure of a VHDL code looks like. And in Verilog we have only module. So, from this only you can see how different it is like VHDL has it is verbose from this only can understand how text it is very like richly typed language.

So, there is something call modeling, there are three different types of modeling: behavioral, structure and logic. Both VHDL and Verilog support those. We will study in detail what these are with respect to Verilog. And there is something called such synthesizable subsets which allow us to port the Verilog description into hardware. So, Verilog and VHDL can also be used for simulation purposes which does not map into hardware. So, there are specific synthesizable subsets that can be map to hardware; like in the last thing is what do I told like VHDL is hard to learn and use and Verilog is easy to learn because it is very similar to C.

So, these are the main difference between VHDL and Verilog. So, in our tutorial sessions we will be mainly dealing with Verilog.

(Refer Slide Time: 14:03)



I have been talking about synthesis a lot in the previous slides. So, we will see what is synthesis is. We talked about how is HDL serves as an executable functional specification which define the digital system using a design methodology, where that the designs are divided into modules and interfaces. So, the executable functional specification document the behavior of modules and interfaces and it can be tested and refined. So, that is what executable functional specification is.

So, HDL is a first step to automate all the process of digital design. So, this executable functional specification we can take it as HDL description which is given to a logic synthesis tool and we get a gate level netlist. So, what I mean by logic synthesis is we have an HDL description of a system to map it on to hardware on like and gate or gate or counters or some higher level ALUs example like that. So, to get a netlist we feed the HDL description and feed target libraries into the tool which has the basic primitive logic gates and digital construct basically and which is a technology specific.

So, we give the HDL description and we give the target library to the tool and the tool synthesizers a netlist. Netlist is generally file in the format EDIF which is electronic data interchange format. So, that is a technology independent like any vendor can take a netlist and they can use it to do further processing of the system. I will come to this

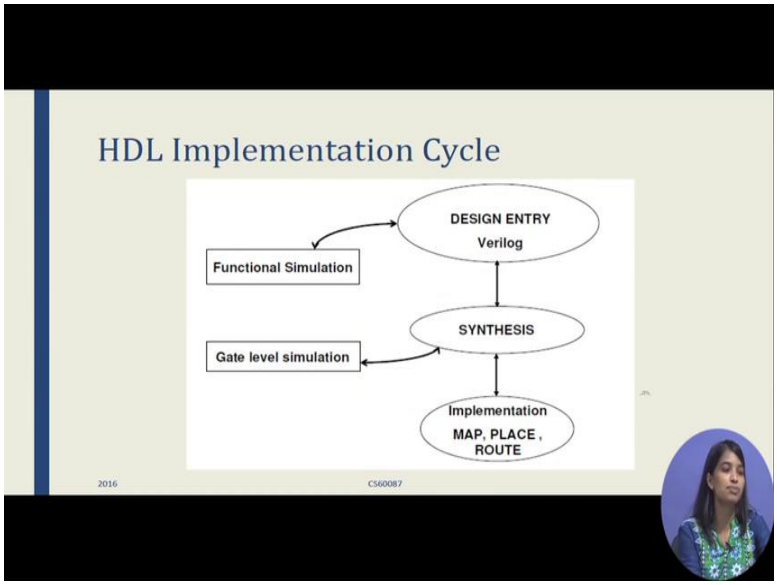
later. So, what we have understood is HDL description maps to a gate level netlist using target library. And this is iterated over and over again for a higher speed and lesser area perspective.

So, we want the system to work very fast, and we wanted to use smallest area possible. So, we tried to like optimize the design for the same. So now this whole process is called functional design and it is a part of front end design. And once we get the netlist we have to map that on to a physical system. The physical system can be a FPGA or an ASIC. So, the gate level netlist it is placed and routed to an FPGA to a particular device.

So, how we do that is for example, if you are making a big building we have fluid pan architecturally using like that, once we have the small blocks that are that have to be made for the judicial system we make a floor plan of the whole system. And we place those we place particular modules and particular places in on the hardware with which is the basically our space where we will put the design. So, we place the design and then we connect different modules to get the whole system. So, this same thing we again optimizer thing we place it here and then and we check which is the optimize path which will give us the high speed, less area, less power, all those things are done.

So, now we have a design on the FPGA, sorry on the FPGA. So, this process is called physical design or back end design. So, this is how we will convert HDL description into FPGA or ASIC; this is how we build. So, in this tutorial we will mostly considered FPGA and I will be teaching how to design a system or how to develop a small basic embedded system on FPGA.

(Refer Slide Time: 18:30)



So, the HDL implementation cycle is in over all like this. So, we talked about product specification, we talked about behavioral description etcetera so that is what is called design entry which we specify using the Verilog code. And then we check the functionality of the design by doing functional simulation. So, we check if the computation is being performed properly or not. So, that is the functional simulation and from Verilog we do synthesis of the Verilog code and we get a great level netlist. So, in that level also after the synthesis we do gate level simulation, we check if after the logic has been ported into a netlist form whether still the functionality is same or not. So, we do gate level simulation.

Once we are done with the synthesis and the gate level simulation we implement the design map it place it and route it on the hardware. So, this is a basic HDL design flow, and hope you understood what where Verilog comes into picture and how we will use Verilog to build a digital system or an embedded system as a. So, that is all for today and you can have the reference of slides if you want to know more about it, you can refer to this links and share the PPTs.

Thank you.