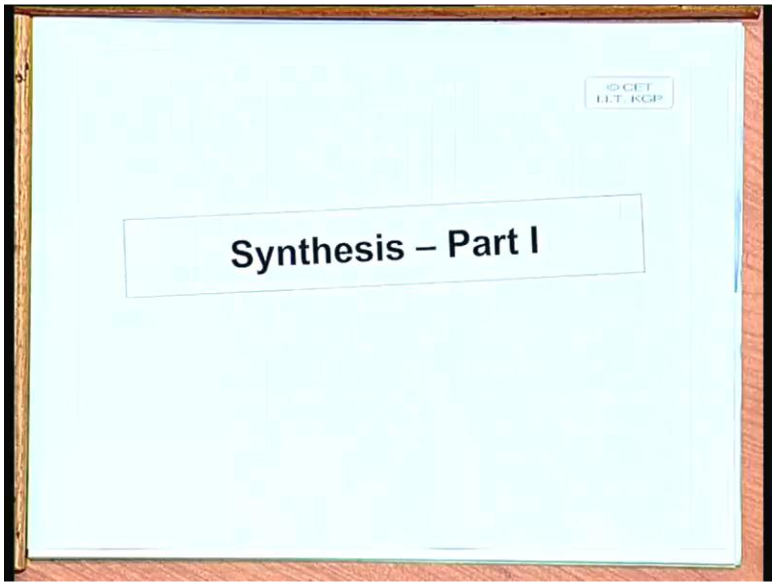
Electronic Design Automation

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No #08
Synthesis: Part 1

Today we will be talking in some more detail on synthesis. Now if you recall what we have discussed in our earlier classes we had looked at the language verilog as a tool for modeling, as a tool for specifying behavior, specifying structural aspects of a design, specify the test benches using which we can simulate our specification and we can and you can obtain a validation of our specification and design. Now after we have specified a behavior of a design the next natural step is to carry out a synthesis. Synthesis is essentially a detailing out of the design. So first let us try to understand what are the scopes of this synthesis process.

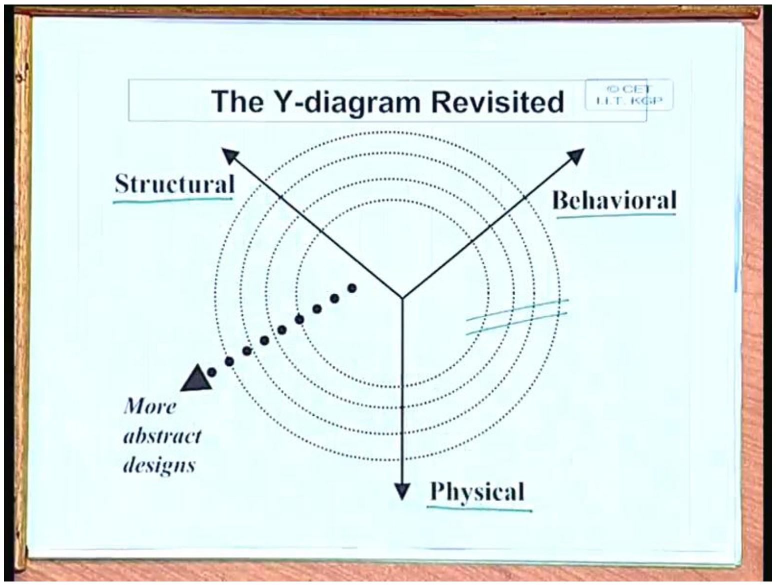
(Refer Slide Time: 02:00)


© CET
I.I.T. KGP

Synthesis – Part I

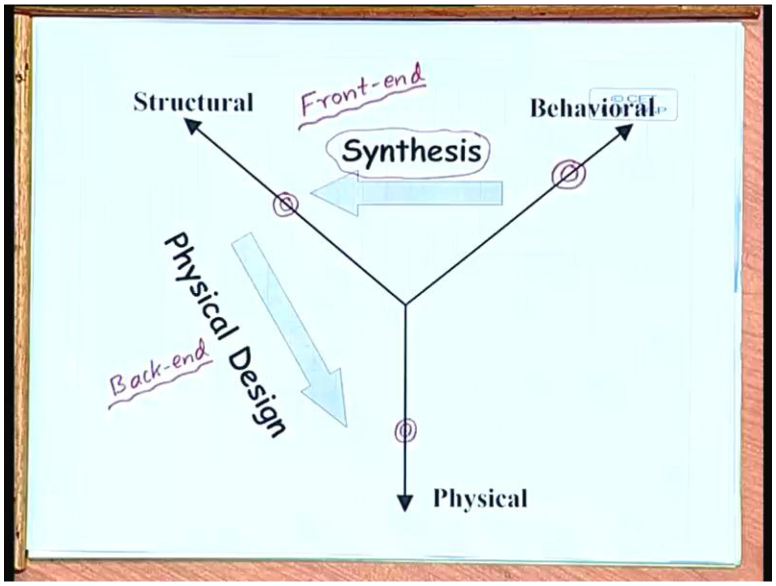
Now in order to do that, we first look at the so called y diagram again which we had seen earlier.

(Refer Slide Time: 02:11)



Now in this y diagram if you recall there are 3 dimensions or axis to a design, the same design we can specify with respect to its behavior, with respect to its structure where well at this level design would consist of a block diagram some cells and their interconnections and finally the implementation level or the physical level. Now in terms of the y diagram you can assume that there are several concentric circles as we have shown out here and as we move away from the centre of this y towards outside the design becomes more and more abstract. So when you are at the outer layer of the circles we are at a very abstract level may be behavior structure or physical. But as we go towards the inner side of this y the design becomes more and more decayed. Well we will just explain this with the help of an example. But before that let us try to see that how the transformation from behavioral to structural to physical takes place.

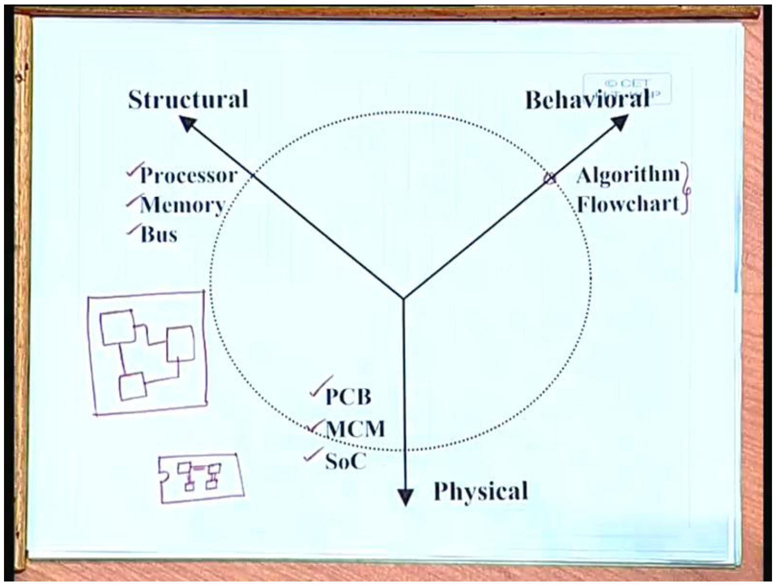
(Refer Slide Time: 03:32)



So let us look at a diagram like this. Well this is the same y diagram which shows behavioral structural and physical views. Now the behavioral level if you recall this specifies the behavior of the circuit or the system without explaining how this behavior has to be implemented or realized. Now here in comes the process of synthesis. So suppose we have given behavioral description out here, now when we carry out synthesis we translate this behavioral description into the corresponding structural description this is the process of synthesis. Now in the structural description all basic building blocks will be some components and the components may be cells from the libraries. It can be gates it can be transistors, it can be anything depending on the level we are looking at. Now this translation from behavior to structure is also sometimes called front end design. So when we talk about front end design or front end CAD tools we are essentially talking about the translation from behavior to structure.

Well now once we have this structural view of a design the next step is to map it into the implementation level. For example if it is an interconnection of transistors so that will have to be mapped into a corresponding level which will means the layout on the floor of the silicon. This process from structural to physical design transmission is called physical design and sometimes this process of translation is also known as back end design or back end CAD tools okay. So this diagram will give you an overall idea that what we are trying to do first we will use the front end CAD tools for synthesis. Next we will be using the back end tools for physical design. So once this entire cycle is complete our physical realization or implementation is ready okay. Now let us see we have said that on this y diagram, we can superimpose the circles and as we go towards the outside of this y the design becomes more abstract. So let us try to understand what this means okay.

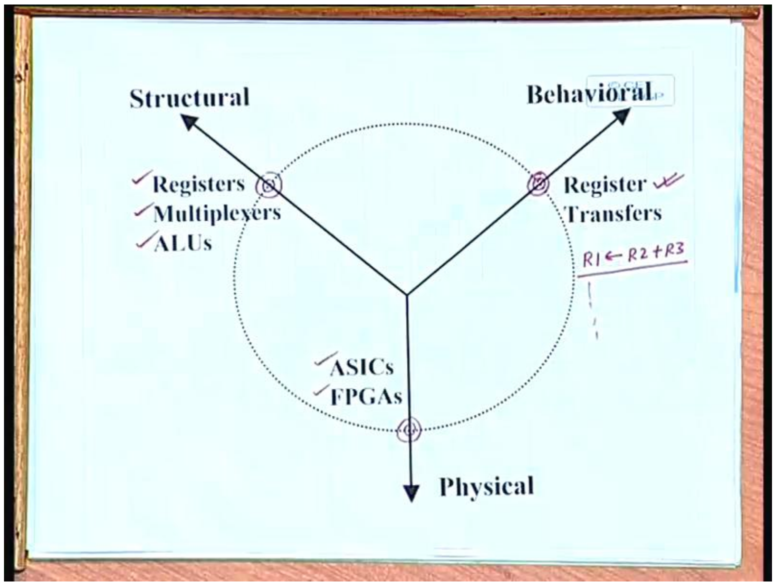
(Refer Slide Time: 06:19)



So we start with a circle which is the most abstract design that we can think of well in terms of the behavior at the most abstract level a design can be specified by simply specifying the algorithm or if it is a system label design you can also specify the flow chart. So algorithm or flowchart is a very high level kind of a design specification. So when we synthesize a algorithm or flow chart at the corresponding structural level we will have a some kind of a block diagram we will have some basic building blocks and their interconnections. Now these basic building blocks at the level of algorithm or flow chart will be something like processors, memories, buses, other IO interface units, etcetera. So as you can see that the basic building blocks are very high level modules. So at this abstract level we are more concerned with a block diagram kind of a design, how these high level blocks are interconnected rather than going into the details of it.

And similarly when we talk about the physical implementation of these. So now we are either talking about a printed circuit board where each of these modules or cells will correspond to 1 chip VLSI chip or we can talk of a multi chip module were inside the same integrated circuit we have several silicon wafers which are embedded on them and they are interconnected. So this PCB and MCM are very similar PCB the chips are put on a board MCM. They are put on the same package. Similarly we can have something called system 1 chip which is the trend today where very large systems can be put inside the same chip itself. So the physical implementation of this high level design can be mapped into either a PCB MCM or system 1 chip SOC. Well now after this as we go towards the centre the next natural level of detail that you can go into is the so called register transfer level designs at the level of the behavior.

(Refer Slide Time: 09:01)

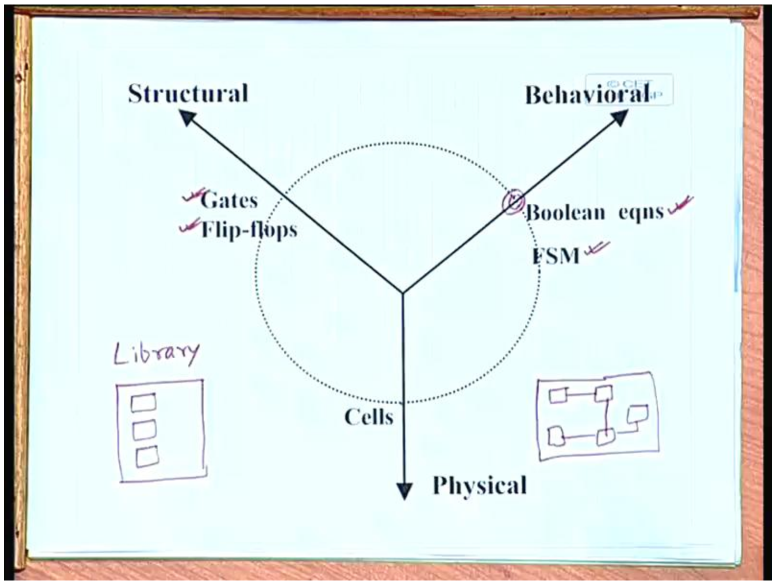


Now register transfer level design will look something like this. Suppose R1 assigned R2 plus R3, there can be a number of conditions. If then else kind of thing there can be a number of such register transfer operations you defined based on the inputs you give. So here we specify the behavior. But we do not say that how these register banks are implemented, how these adders, subtractors, are implemented. How many adders are there? So all those details we do not specify.

But as you can understand with respect to the algorithm description, as we had said earlier this is a much more detailed description okay. So with respect to a processor, suppose we are designing a processor. This may correspond to the so called microinstruction level. So at a very high level you can specify a processor with the external IO pins and the instruction set. But as we go into the design of the control unit of the processor, we will have to specify the different microinstructions.

The instructions execution cycle if there are any pipelines details of that what are the control signals etcetera so these we specify in terms of the register transfer operations. Okay. Now this register transfer level behavior when we translate into structure we get a design for the blocks are registers multiplexers ALU's and similar register transfer level components. Similarly when you map them into the physical implementation. So you can either have an application specified circuit or you can have a prototype on a field programmable gate array. So it depends on your target implementation what you are trying to implement and how you are trying to implement. So as you go into the next lower level or I mean more detailed level of the design we get a design which consist of register multiplexers, their interconnections, and their control and at this level possibly we are talking about mapping the design into a ASIC or FPGA okay. Fine.

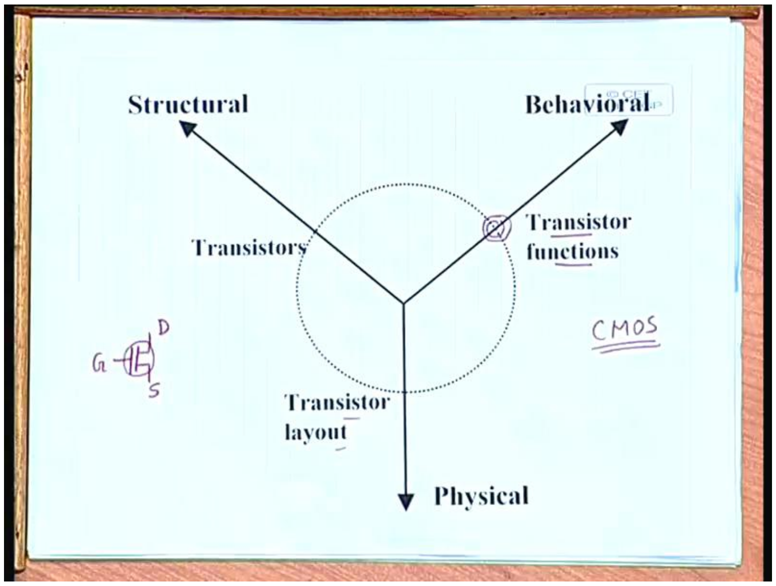
(Refer Slide Time: 11:22)



But as you decal the design further from the register transfer level the next natural jump will take is to have something like a logical level behavior specification where if it is a combination logic we will specify the behavior by a set of Boolean expressions some Boolean equations. If it is a sequential logic we will specify by finite state machines. These represent the behavior of the circuit of the system we try to design. Now as you know starting from the Boolean expression or FSM there are many methods which are available for synthesis where we get a circuit where the basic building blocks. Now become gates and flip flops, so we have now an interconnection of gates and flip flops starting from the behavior at this level.

Now talking about the physical implementation now at the level of gates and flip flops well normally what happens when you are working through the back end CAD tool. There is something called a library it is sometimes also called technology library. Now in the library you have the standard components already available standard components like gates flip flops okay simple arithmetic logic units. So at this level once you have the gate level implementation you simply replace each gate or each flip flop by the corresponding cell available from the library. So now at this level you have actually a tentative layout on silicon where you have the cells these cells are fairly low level cells gates and flip flops and the way they are interconnected. Okay. So this is one level of the design.

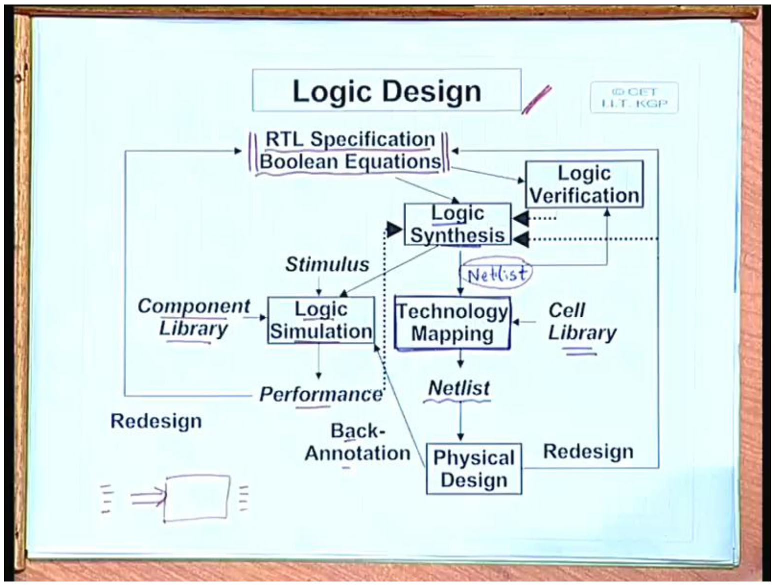
(Refer Slide Time: 13:25)



And as you go even further down you get a design at the lowest level. Well with respect to the behavior; see ultimately we want to implement a circuit typically in CMOS; using CMOS technologies. So you can also specify the behavior of a CMOS net list with respect to the transistor functions transistor functions typically for digital circuits a transistor or a MOS transistor is modeled in a very ideal way. We treat this as a simple switch. This is the gate, this is the source and this is the drain. So if it is a NMOS kind of a transistor depletion type. If we apply a high voltage to the gate then the switch is assumed to be turned on the drain and the sources are shorted. But if the gate is at a low voltage they are turned off. So each transistor is modeled as a switch this is what we mean by the transistor functions. So the whole circuit is modeled with respect to the behavior of the transistors.

Well with respect to some more simplified model, now from this simplified model when you go to the structural level. We get the actual net list of the transistors which is used to implement the behavior which we specify out here and finally at the physical level we get the actual layout of the transistors. Okay, so this y diagram and the way we proceed from the outer most circle to the inner most circle this tells you the different levels at which we can carry out a design and the 3 aspects of the design behavioral, structural and physical. Now one thing I would like to tell that for a particular design it is not necessary to go through all the different steps. For instance if your ultimate target is to build a prototype on FPGA. Then you need not go up to possibly the get level implementation or certainly the transistor level implementation. Slightly high level implementation can also suffice provided you have a way to map them into the corresponding FPGA cells or clb's what ever you call okay fine. So to start with we first talk about the logic design process as part of synthesis.

(Refer Slide Time: 16:12)



See we are talking about synthesis which means we are talking about translating the behavior into the structural specification. So we start with the level of logic design where the specification is given either in terms of set of Boolean expressions. It can be FSM's also or it can be slightly higher, it can be register transfer level specifications also. So somehow we are specifying the behavior of the circuit and from there we are trying to go through the different steps of synthesis. Now this diagram will show you what are the different steps that we typically go through okay. This block as you can see logic synthesis this is the block which accepts the specification as the input and generates some kind of a synthesized net list as the output net list means some basic components and their interconnection. Now if the specification is in the terms of Boolean expressions, then this net list will consist of gates possibly flip flops and then interconnections.

Well if it is actual level then this net list may be ALUs multiplexer buses and the way they are interconnecting okay. Fine. So once you carry out the logic synthesis you have or you get a net list where you have much more details about the design. But when you are trying to implement it this net list may not be sufficient. Why? See suppose you are trying to design using FPGA. So using FPGA you have some constraints you know that these are the functions I can implement using FPGA and these are the things I cannot. Moreover let us take another example. Suppose we are trying to design using CMOS. Now in CMOS again we will be having some kind of a library available with you where you will be getting a list of available gates or cells which you can use in your design. So the net list that we have obtained through the logic synthesis process, the basic cells out there, they will have to be mapped some how into the cells that belongs to that library. So that process is achieved or accomplished through a step called technology mapping.

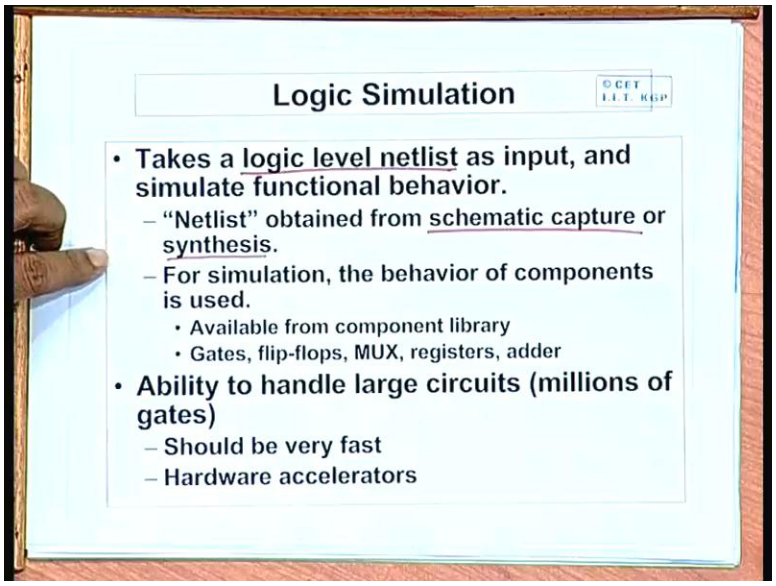
So technology mapping takes the net list which is generated by the logic synthesizer it also accepts the library this is a given cell library okay. And it tries to map the cells out here to the cells available in the library. And as the output it generates another net list where the basic cells are those which are available in the library. So this net list is something which can be implemented okay. So from this you can straight away go to the step of physical design. So if it is FPGA you can straight away map this design into the FPGA chip. If you are going for an ASIC design, so this physic design process will itself consists of number of steps like partitioning floor planning, routing clock and power out etcetera. This we will see later. So this is the essential process and after completing the physical design if you find that something is wrong the circuit is not working according to the specification, then you may have to redesign you may have to go back and change the original specification okay fine.

But as you can see that once you do the synthesis and physical design, well you have incurred a lot of you can say cost. If it is an ASIC design you will have to go through the entire steps of the ASIC design process which is expensive. So usually what people do before going for the actual physical design, so you take the net list which was obtained by the synthesizer and try to check whether this net list is working as per the specifications or not this you do through a process called logic simulation. So logic simulation will again take this net list as the input okay. So it will also take a stimulus which is just a set of inputs. Say I have a circuit I will say that well I want to test the circuit for these inputs. So we apply these inputs and the simulator will tell you further inputs what will be the outputs. So I can verify if the outputs are correct or not. So when you are doing logic simulation the behavior of all the cells of this net list must be known that is available in another library. This is called the component library.

Here all the basic components that can be there in this net list are present as well as their behavior is also present which helps in carrying out the logic simulation. So after simulation we get the performance and if we see the performance is not according to our requirement we again to do redesign go back and change the specs. Now sometimes even during the physical design we can do something called back annotation. Back annotation means form of very low level design specification we can re extract possibly the gate level design again. So we can go back to the logic simulation phase and again simulate with the same stimulus and check the performance.

If it again, see if you again find that it is not matching you can again go back and make changes. So mismatch and performance may need to changes out here may need to some changes in the parameters that we use during logic synthesis. Well of course there is another step involved which is slightly beyond the scope of the present discussion this is called logic verification. See logic verification is a process which accepts the specification and tries to formally verify whether the specification is correct or not. Okay but this is a block which we will not be discussing much during the course of the present class. Okay so now let us see some of the blocks out here what are the basic responsibilities and basic task of this is start with the block logic simulation.

(Refer Slide Time: 23:50)



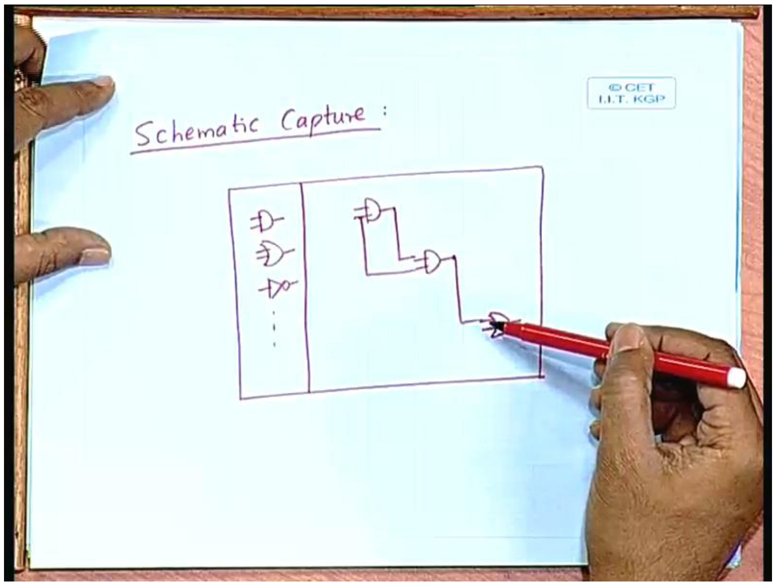
Logic Simulation

© CET I.I.T. KGP

- Takes a logic level netlist as input, and simulate functional behavior.
  - "Netlist" obtained from schematic capture or synthesis.
  - For simulation, the behavior of components is used.
    - Available from component library
    - Gates, flip-flops, MUX, registers, adder
- Ability to handle large circuits (millions of gates)
  - Should be very fast
  - Hardware accelerators

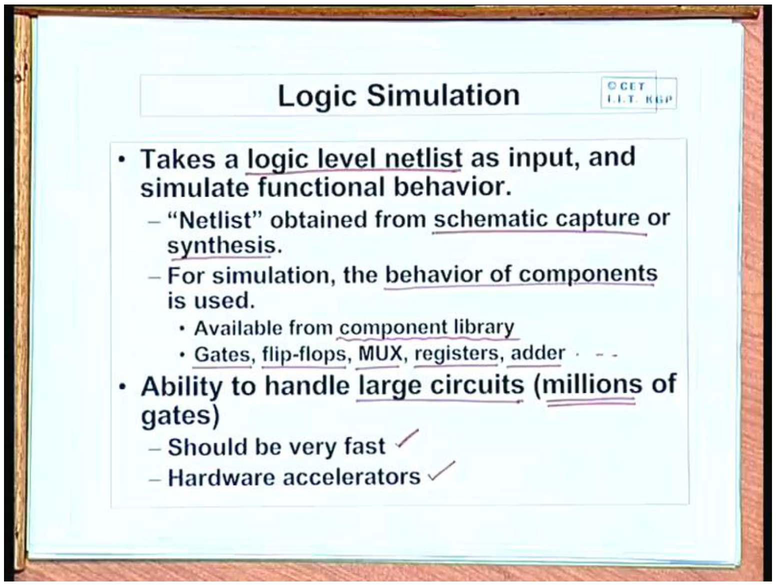
So as we had mentioned the logic simulation process takes as input a logic level net list. Logic level net list means net list comparison of gates comparison of flip flops and it simulates the functional behavior functional behavior. Means most commonly we would like to see whether the input output specifications are correct that if you give a particular input to the circuit what is the expected response or what is the response that we get and this logic. Simulation tool is a software program where the circuit you want to stimulate in terms of suitable data structure and through a systematic scan of the circuit of the data structure. We carry out or we evaluate the values of the gates and finally we obtain the values of the output and this net list which is taken as input. This if you recall this is a net list which is obtained through a process called synthesis. Well sometimes we can also get the net list through a process called schematic capture. Now schematic capture is an option which is available in many CAD tools.

(Refer Slide Time: 25:20)



Just let me try to explain what this schematic capture is schematic capture is an interactive way to specify a net list. See if you look at the typical window of your CAD tool, you will find typically on the left pane. You will be having a number of symbols available with you. These symbols may be the symbols of the gates okay etcetera. So what ever you use in mouse you can select this and you can drag and drop, you can draw a diagram here. For example you can put an AND gate here you can put an AND gate out here. You can put an OR gate out here okay and then using mouse again you can click here and click here these 2 points here will get interconnected okay. These 2 points will get interconnected these 2 points will get interconnected okay. So in this way interactively using a graphical user interface and mouse you can draw a schematic diagram. So when you complete the drawing of a schematic diagram this is nothing but a net list which you have created okay.

(Refer Slide Time: 26:42)



Logic Simulation

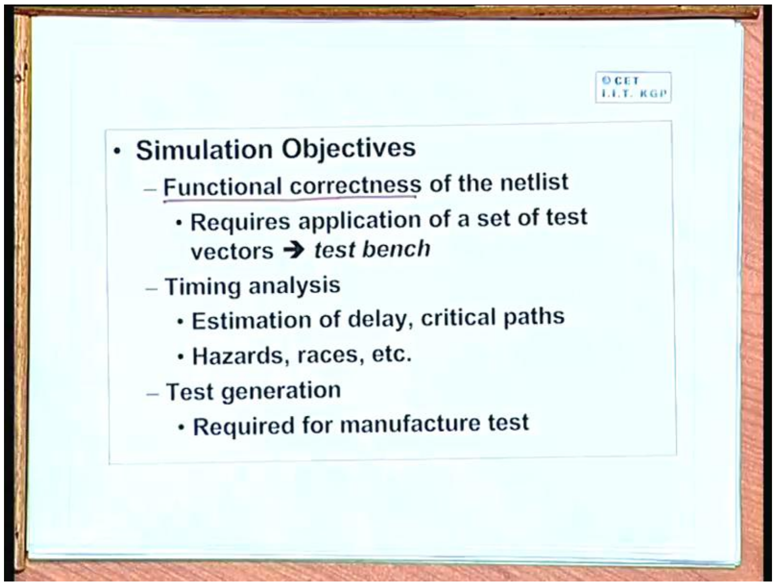
©CET I.I.T. KGP

- Takes a logic level netlist as input, and simulate functional behavior.
  - "Netlist" obtained from schematic capture or synthesis.
  - For simulation, the behavior of components is used.
    - Available from component library
    - Gates, flip-flops, MUX, registers, adder - - -
- Ability to handle large circuits (millions of gates)
  - Should be very fast ✓
  - Hardware accelerators ✓

So your input can also come from this schematic capture process. But when you are doing simulation you need the behavior of the components behavior of the components what does that mean? Suppose if the components are simple gates like AND, OR, NOT, then what you need is the input output behavior of the gates. You need either the Boolean function or the truth table in some form. So you need the behavior of the gate in order to evaluate the output of a gate when some particular input values are applied to it okay. So behavior of the components is assumed to be present in the so called component library. Now component library can be quite exhaustive you can have gates out there. You can have sequential elements like flip flops and registers you can have slightly more combinational more complex blocks like multiplexers, adders and other functions and models like decoders, etcetera.

Now logic simulation is used before we are actually going to the physical design. So one of the requirements of logic simulation is that it should be possible to handle large circuits. So when we talk of large circuits in the present day scenario, we are actually talking about millions of gates okay. So the logic simulation process should be very efficient it should be able to handle large circuits of the order of millions. It should be relatively fast and in order to handle so big circuits fast typically some special hardware called hardware accelerators are used whose sole objective is to speed up the process of simulation. It acts just like a helper or a co processor to the software simulator using the hardware accelerator. You can actually speed up the rate at which you can process and simulate the gates okay fine. Now this is what simulation is.

(Refer Slide Time: 29:11)

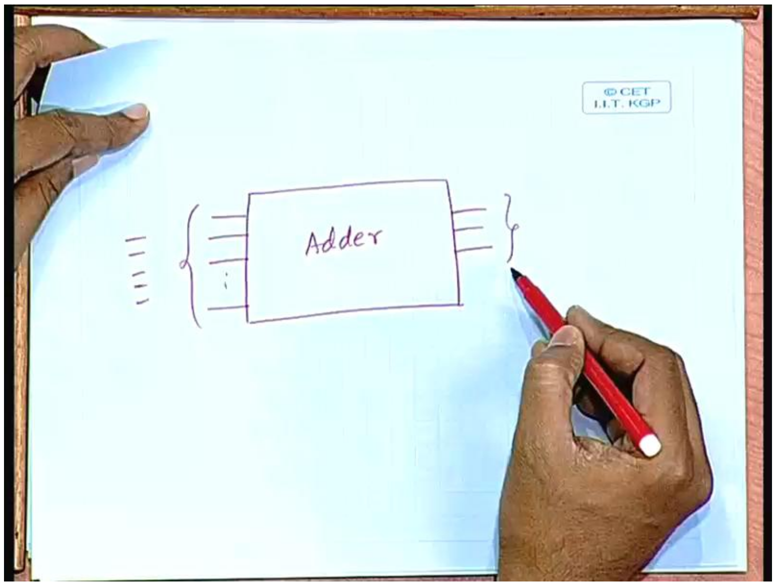


© CET
I.I.T. KGP

- **Simulation Objectives**
  - Functional correctness of the netlist
    - Requires application of a set of test vectors → *test bench*
  - Timing analysis
    - Estimation of delay, critical paths
    - Hazards, races, etc.
  - Test generation
    - Required for manufacture test

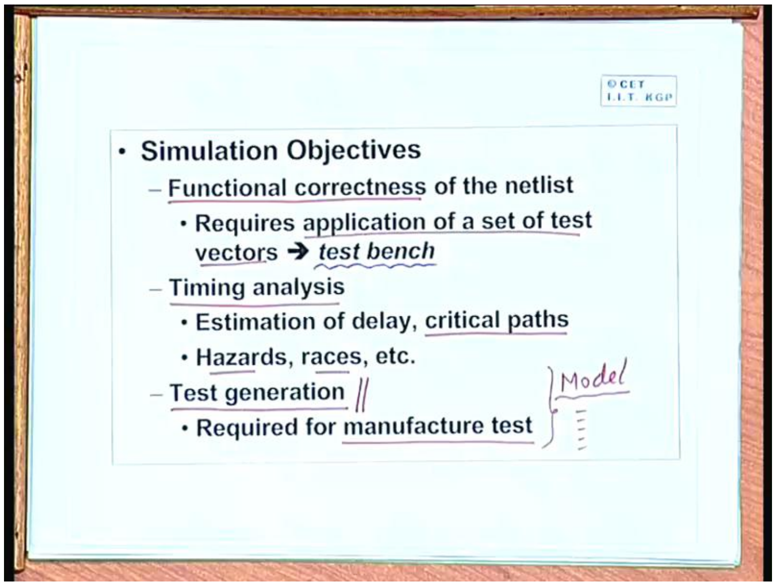
Now when you talk about the simulation objectives. So the main objective is of course to verify the functional correctness. Functional correctness how do we verify we verify? It like this.

(Refer Slide Time: 29:31)



Given a circuit so I am treating it as a block diagram this is some kind of a net list. There are some inputs there are some outputs. So through simulation I can apply a set of inputs. Suppose this is an error okay. So I specify a set of input data and I check whether the output result is coming as per the specification. If I have added I know what expected output is okay. For given inputs I can verify what outputs are obtained and what output should have been obtained.

(Refer Slide Time: 30:10)


©CET I.I.T. KGP

- Simulation Objectives
  - Functional correctness of the netlist
    - Requires application of a set of test vectors ➔ *test bench*
  - Timing analysis
    - Estimation of delay, critical paths
    - Hazards, races, etc.
  - Test generation //
    - Required for manufacture test

Model

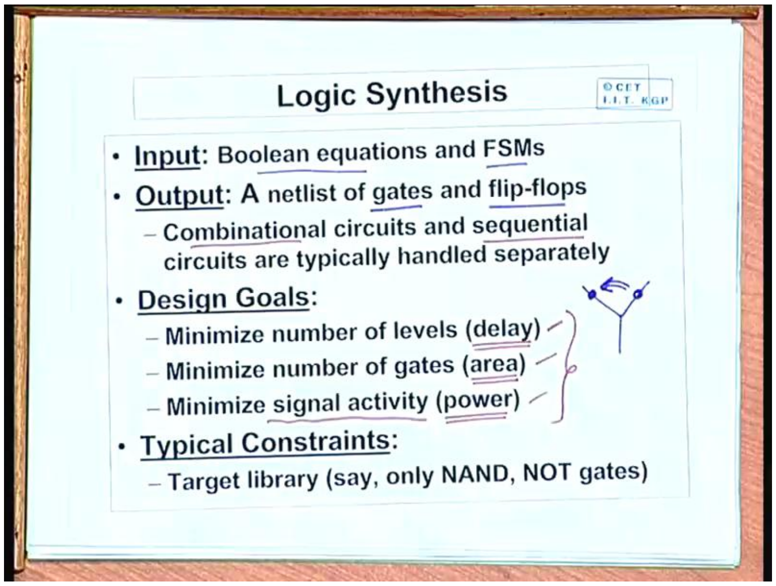
So this requires as I said application of a set of test vectors. Now we have already seen dealing on the discussion of language verilog that we can specify test benches using verilog which is used for the purpose of simulation for specifying the test vectors that we apply to a given module. So functional correctness is of course one of the main objectives. But in addition to functional correctness we need something more. We need something called timing analysis like if we have some estimate of the delays of the basic components say gates. Then you can estimate that what is the total delay of the circuit. What is the maximum speed with which our circuit can work we can identify the critical paths. In the circuit critical paths are those paths which actually affect directly the total delay. So if you can optimize the critical path the total delay of the circuit become less.

So through timing we can analyze potential problems like hazards races and if we see that there are some problems you can go back and redesign and again do a timing analysis to find out that whether this problems get eliminated or not. So timing analysis although it may look to be a slightly more complex process. Yes, it is a more complex process. But for big and complex designs it is an essential part of the design process. See you do functions simulation. You see that input output behavior is correct. But that does not tell you anything that any practical design will have the design requirements. You will have some constraint on the speed with which the circuit is supposed to work. So through timing simulation only you can get an estimate about the actual speed with which your circuit will be able to work or operate okay fine.

Last thirdly there is another simulation objective which is also a very important process. This is something called test generation. See test generation is something which is required for manufactured test. Well what is manufactured test see you have a design and you have done some kind of synthesis on it. So what we obtained is a net list okay. So this net list is available to you. But this is not something which you have already fabricated. This net list is still on your computer okay. So through simulation you try to verify whether the net list is working as per the specification okay. This is done on the design. But suppose the design process is through you have completed the physical design process also and now the chips are getting fabricated. Suppose you have fabricated 1000 chips for your design. Now during fabrication there can be a large number of faults which can creep in obviously. So once you have manufactured the circuits in terms of the chips you will have to test each and every chip by applying some input stimulus input test vector and checking the output. This is the so called manufacture test that I am talking about. So in this step well again you will have to apply some input test patterns.

But here again we do some kind of a modeling we assume that what kind of faults can potentially occur in the circuit with respect to that assumption we try to obtain a set of test generation. Because see suppose I have designed a processor and I ask you what are the inputs I should apply to this chip in order to verify that this chip is working correctly before actually I can send this chip into the market for sale. Now it is a very difficult question obviously we will have to tell that well these are the testing or these are the tests that I need to apply to the chip. So in order to do that you will have to identify or understand that what kind of defect or fault modeling you are doing and whether the test vectors or the test set you are trying to apply will be able to detect all the modeled so called faults under your consideration okay. So this is also a complex problem okay. So after simulation comes the process of synthesis. So now let us see what this logic synthesis process is all about.

(Refer Slide Time: 35:54)



Logic Synthesis

© CET I.I.T. KGP

- Input: Boolean equations and FSMs
- Output: A netlist of gates and flip-flops
  - Combinational circuits and sequential circuits are typically handled separately
- Design Goals:
  - Minimize number of levels (delay)
  - Minimize number of gates (area)
  - Minimize signal activity (power)
- Typical Constraints:
  - Target library (say, only NAND, NOT gates)

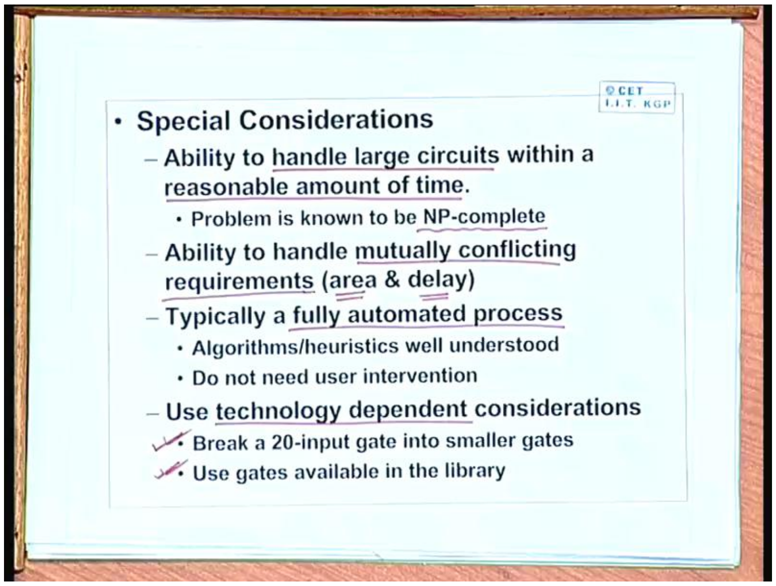
Now logic synthesis process with respect to the y diagram. I have already mentioned well with respect to the behavior you have Boolean equations and finite state machines from there you want to carry out a translation into the structural domain where your basic components will be gates and flip flops okay. Now you must be familiar with the number of methods that are used to translate from a Boolean equation an FSM into a net list of gates and flip flops you have (( )) (36:43) methods you have (( )) (36:45) methods you have formal ways formal tabular ways to synthesize an FSM. And when you are synthesizing an FSM you know that typically you separate out the combinational part and the sequential part and you synthesize them separately okay. So there are standard methods of doing them but we will see shortly that these standard methods have some problems. So we need to do something about it. So as I said typically for sequential circuits the combinational parts and the sequential parts are handled separately okay.

Now when you are carrying out logic synthesis depending on your requirement depending on your ultimate goal your objective may vary. Suppose you are trying to design a circuit for a very critical application where delay is most important to you. You want to design a circuit that will work as fast as possible. So here if it is combination circuit your designed goal will be to minimize the number of gate levels. So as you reduce the number of levels the total delay of the circuit will also reduce. Well another alternate design goal may be that well I am not that well interested in delay but I am more worried about area. So I want to design my circuit within the smallest area possible. So there well our requirement translates into minimization of the number of gates. Well we are not worried if the number of levels goes up.

But we are more worried of the number of gates. Thirdly that is another very important design goal which has kept up in recent years with the proliferation of you can say mobile and handle devices which operates on mostly on not an electric means power but on back up battery power. So one objective with the design of those circuits is that if the circuits involved in the design consume less power. We can run them for longer periods of time and that is really very desirable. So we want to minimize signal active signal activity means as the switching activity in the circuits get reduced the total power consumption also get reduced. So we want to ensure that the number of signal transitions that are going on signal transition means 1 to 0, 0 to 1 they are going on per unit time in the circuit is as less as possible. So these are the 3 conflicting design requirements.

See well you typically these 3 design goals are mutually called conflicting. So if you try to minimize the number of levels you may see that the area is going up and if you want to minimize the area the delay will go up power again the same thing holds for power. So it is very difficult to consider all these 3 things at the same time and try to do the synthesis. So typically people will give priority to one of these and go ahead with this synthesis process. Not only that when you are doing synthesis you have some other external constraints that will guide you like some typical constraints may be. Well you will have to synthesize only using NAND and NOT gates or may be using NOR. This is the so called target library. So you will have to complete the synthesis where the components will belong only to the target library. These are the typical constraints that you will have to satisfy.

(Refer Slide Time: 41:14)

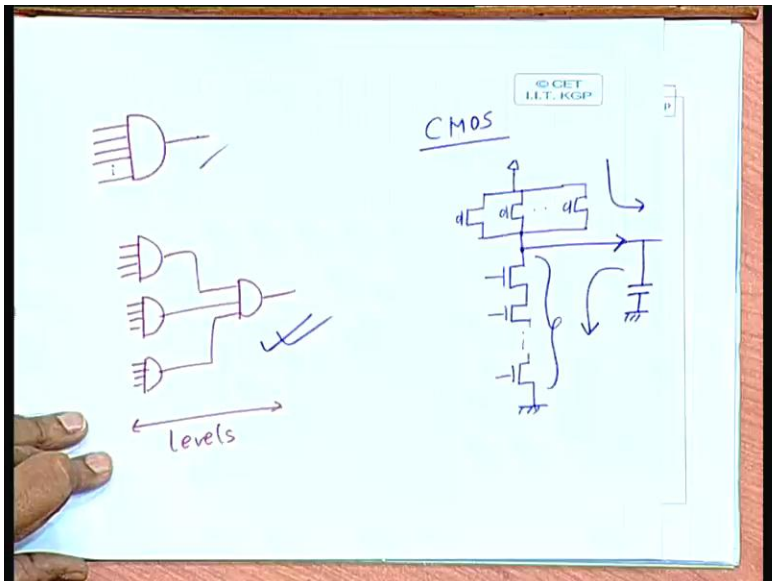

© CET I.I.T. KGP

- Special Considerations
  - Ability to handle large circuits within a reasonable amount of time.
    - Problem is known to be NP-complete
  - Ability to handle mutually conflicting requirements (area & delay)
  - Typically a fully automated process
    - Algorithms/heuristics well understood
    - Do not need user intervention
  - Use technology dependent considerations
    - Break a 20-input gate into smaller gates
    - Use gates available in the library

Now there are some other considerations that when you are doing synthesis obviously today we want to handle very large circuits. So we would like that we should be able to handle fairly large circuits within a reasonable amount of time. But unfortunately the problem of getting the best solution has been proved to be an np complete work which means that it is computationally infeasible or computationally very expensive to ensure that the design you have synthesized is the best possible one. So obtaining the best possible design is not practical for large circuit. This is something which we need to understand. Moreover as we have just mentioned that we may have mutually conflicting requirements. Some designers may say that we want to concentrate both on reduction of area and delay. But these 2 may be mutually conflicting how to do this.

So this logic synthesis process is typically a fully automated process. This is a well understood well studied process and we do not need or we do not require any user intervention between okay. There are other things that you will also have to look at some considerations which are dependent on target technology. First one of course I have just mentioned that you will have to use gates only available in the library okay. So may be, you are using an exclusive NOR gate which is not in the library. But ultimately you will have to break up that exclusive NOR gate into a net list of NAND gates. That will really not help your cause okay. May be your delay will go up. So it is better if you know what are the gates that are available in the library and you carry out the design based on that okay. Now another example which we mentioned here is that if you have a gate with large number of inputs. You better break it up into smaller gates.

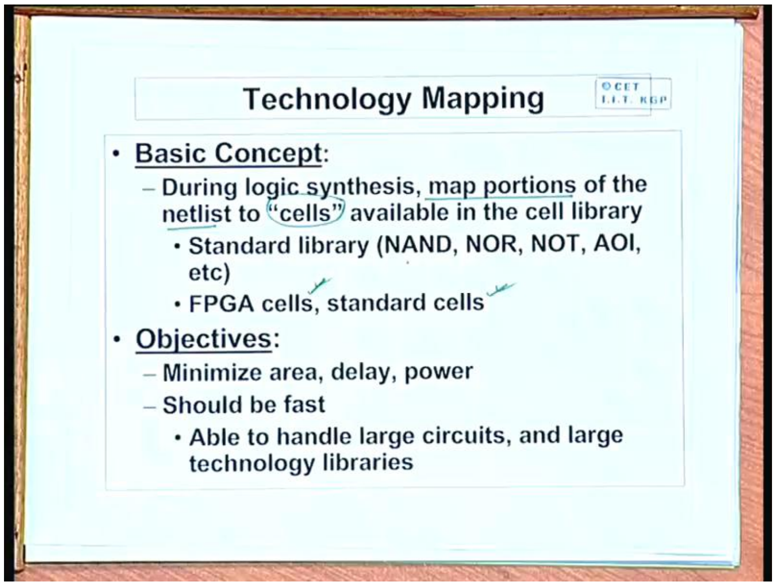
(Refer Slide Time: 43:50)



Well what I am saying is suppose you have say an AND gate. Let me call it an AND gate so you have a AND gate with a large number of inputs. So what I say instead of having 1 AND gate with a large number of inputs you may be possibly break it up into several smaller AND gates. But you may argue that if I do this I am increasing the number of levels okay. So apparently I am also increasing the delay because the number of gate delays is more. But well this is not true this is not true. Why because, we try to understand that gates are typically implemented today on chips using MOS technology, typically CMOS. See if you look at a NAND gate the way the NAND gate is constructed in CMOS. See in the pull up you have a set of p type transistors and in the pull down you have a set of n type transistors in series.

Well now as you increase the number of transistors in series the total impedance of this chain of transistor increases. So when the output of this gate tries to switch from 0 to 1 actually you are discharging a current along this path or you are charging a current along this path because the load can be approximated by a capacitor. So either you are charging the capacitor or discharging the capacitor. Now if you have many transistors in series the time it will take to discharge the capacitor will be much larger. So larger the number of inputs slower will be the gate. So it has been found out that rather if you break it up into smaller input gates with a larger number of levels may be this will be a faster solution as compared to this. Typically we do not use gates because inputs are more than 4 or 5 that is a roll of thumb okay fine.

(Refer Slide Time: 46:52)



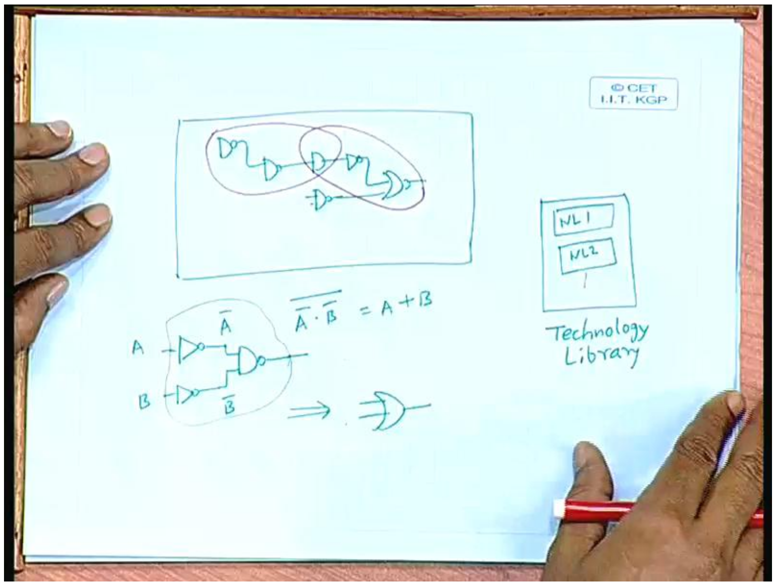
Technology Mapping

©CET I.I.T. KGP

- **Basic Concept**:
  - During logic synthesis, map portions of the netlist to "cells" available in the cell library
    - Standard library (NAND, NOR, NOT, AOI, etc)
    - FPGA cells, standard cells
- **Objectives**:
  - Minimize area, delay, power
  - Should be fast
    - Able to handle large circuits, and large technology libraries

So there is another step called technology mapping that I had mentioned that technology mapping is a process where we have the net list with us. We have a technology library with us we will have to map our net list from some modules which are available in the technology library. So the basic concept is that during synthesis we will have to map portions of the net list to so called cells which are available in the cell library. See cells can be these standard gates NAND, NOR, NOT, AND, OR, invert or depending on the target where you want to map the design on if it is FPGA. So you can have FPGA cells you can have standard cells where you have standard basic building blocks like adders, multiplexers, decoders, etcetera. So actually what I am saying is that is something like this.

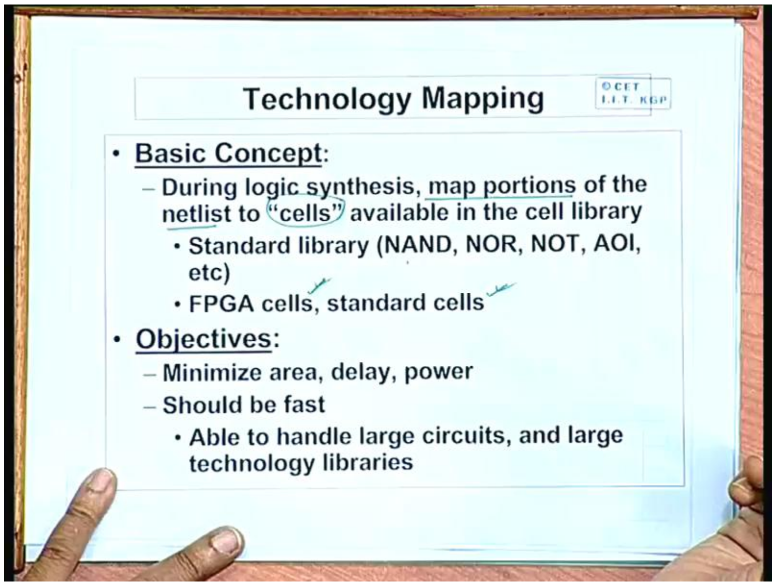
(Refer Slide Time: 48:01)



Say you have a given net list to start with this can be a net list of gates any kind of big net list you have obtained okay. It is a big net list you have obtained. Now we are showing a part of it. Now you will have you have with you a technology library. Now your technology library you will be having small net lists. You will be having net list 1, net list 2 etcetera. Net list 1 what this net list will contain? Say it will contain something like this that if we have something like this, this is equivalent to what? This is a, and b, this is a bar this is b bar, the output will be this is a or b okay. So if we have a sub circuit like this, this is equivalent to a OR gate. So you will have to find out this kind of subsets in the original circuit and we have to identify that which small net list out here matches with that.

And you can replace it with the corresponding gate in the technology library. So actually what I will have is that you will be dividing the total net list into smaller ordinary. This should be one module there can be overlap also this can be one module. So ultimately this segment will get mapped into one particular component of the technology library. Similarly this segment will get mapped into another component from the technology library. So in this way you can map the different subsets of course. This is a graph theoretic problem you can understand. You will have to match the sub circuits in the graph and you have to find out.

(Refer Slide Time: 50:21)

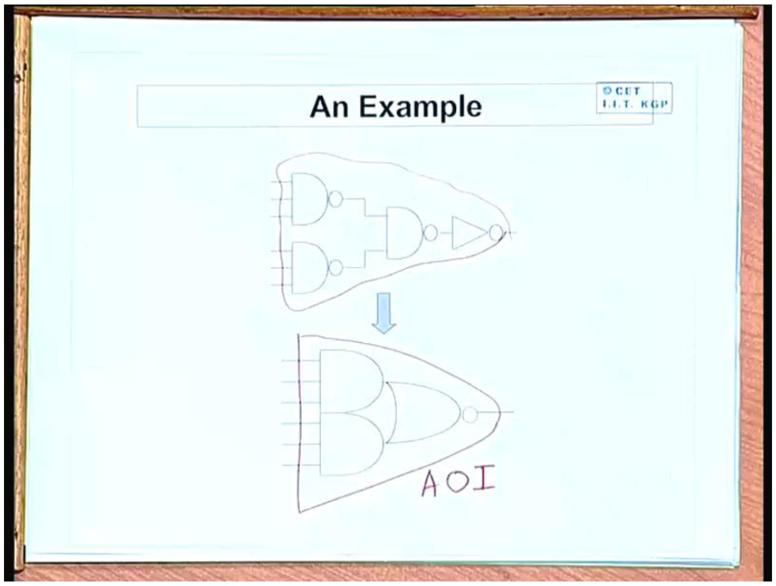


Technology Mapping

- **Basic Concept**:
  - During logic synthesis, map portions of the netlist to "cells" available in the cell library
    - Standard library (NAND, NOR, NOT, AOI, etc)
    - FPGA cells, standard cells
- **Objectives**:
  - Minimize area, delay, power
  - Should be fast
    - Able to handle large circuits, and large technology libraries

So objective is obviously to minimize area delay power there can be conflicting requirements should be fast, it should be able to handle large circuits and typically technology libraries are large. So I have done the example to show you.

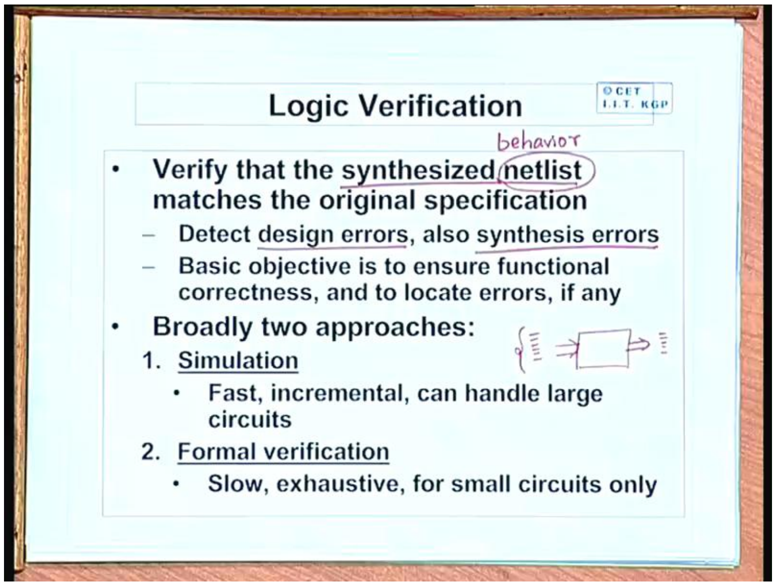
(Refer Slide Time: 50:38)



An Example

So suppose you have obtained a sub circuit like this as part of the bigger circuit. Now we search the technology library and find out that well there you have a information that if we have a sub circuit like this, this is equivalent to component, this is AND OR invert, this is AND OR invert gate. So AND OR invert gate is available as a component in the technology library okay. So the idea is simple you have a big net list of gates you try to find out the small sub circuits from there which matches with some net list available in your technology library. And if you find a match replace it by the corresponding component of the technology library. So if you do this what you have is that your entire net list which was there at the beginning will now get transformed into a net list of cells which have been picked up from the technology library. Now as I had said there can be overlap among the cells, the objective is to have minimum number of cells if area is important or to reduce the number of levels of the cell net list if delay is more important.

(Refer Slide Time: 52:10)



Logic Verification

©CET I.I.T. KGP

- Verify that the synthesized $\overset{\text{behavior}}{\text{netlist}}$ matches the original specification
  - Detect design errors, also synthesis errors
  - Basic objective is to ensure functional correctness, and to locate errors, if any
- Broadly two approaches:
  1. Simulation
     - Fast, incremental, can handle large circuits
  2. Formal verification
     - Slow, exhaustive, for small circuits only

So on and with respect to the diagram. So verification which had said that well you do this logic verification in order to check whether the given specification is correct or not to verify that the synthesized net list. Well it is not always the net list sometimes this can also be the behavior okay. So it is not necessary that you are verifying always on a given net list. You can also verify on the behavior the objective of doing. This is to detect design errors if I have met some error while coding the design and if we are trying to verify the net list. Then we are also trying to detect if there are any synthesis errors may be there are any problem with the synthesis tool. So I want to check whether the synthesized net list is also correct basic objective is to ensure functional correctness in logic verification. We are not talking about delay or power. We are basically talking about the input output behavioral specification whether they work correctly. Well we talked about simulation.

Now actually simulation is also a way of doing logic verification because there also we have a given net list. We apply some inputs and we check the outputs. So logic verification broadly falls in the 2 categories. One is simulation which is done on a computer typically by modeling the circuit in some way. This is typically very fast where you can simulate circuits which are very large say up to the order of million gates. Also it can handle very large circuits but in simulation the only problem is that you have to clearly specify that these are the inputs I need to apply and I want to check the output for this. Now the question arises that how many inputs you need to apply before you can say with confidence that well I have verified my design. Now it is naturally a very difficult question particularly if your design is complex like a professor say you are designing a CPU.

But in contrast the other method which is called formal verification this does not apply any input vector rather it tries to prove that your circuit behavior is correct through formal techniques through mathematical techniques. And since it uses mathematical techniques, it is much slow, but it is an exhaustive method exhaustive in the sense that it will take care of all kinds of errors which are there. So it is not just dependent on what are the inputs that we are applying. And since it is slow you can apply it only for smaller circuits. So today actually we had looked at the different aspects of logic synthesis logic. Synthesis logic, simulation logic, technology logic, mapping logic, verification. Now in the next class we will be looking at some of this steps in more detail may be looking particularly at the step of at the process of synthesis both at the level of logic. And subsequently we will see that when we have a design at a higher level what do you do about it? How do you synthesize the design at the behavioral level or so?