

Optimization Techniques for Digital VLSI Design
Dr. Chandan Karfa
Dr. Santosh Biswas
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

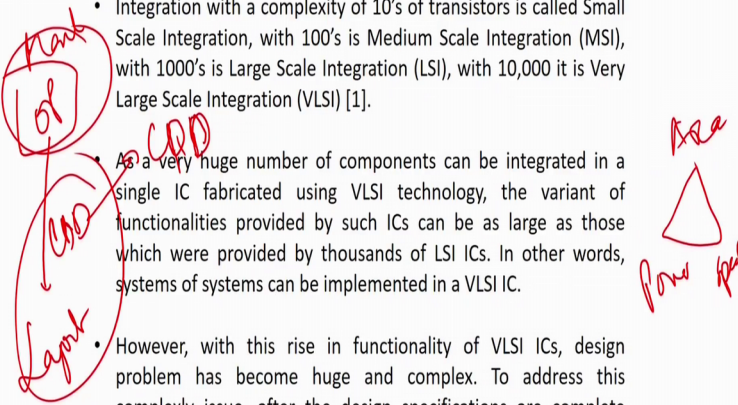
Lecture – 01
Introduction to Digital VLSI Design Flow

Hello everybody and welcome to the course on Optimization Techniques for Digital VLSI Design. This is the 8 weeks course and it will be taken by my colleague Chandan Karfa and myself I am Santosh Biswas. So, welcome to this course and this is the introductory lecture right.

(Refer Slide Time: 00:45)

Digital Circuit Integration

- Integration with a complexity of 10's of transistors is called Small Scale Integration, with 100's is Medium Scale Integration (MSI), with 1000's is Large Scale Integration (LSI), with 10,000 it is Very Large Scale Integration (VLSI) [1].
- As a very huge number of components can be integrated in a single IC fabricated using VLSI technology, the variant of functionalities provided by such ICs can be as large as those which were provided by thousands of LSI ICs. In other words, systems of systems can be implemented in a VLSI IC.
- However, with this rise in functionality of VLSI ICs, design problem has become huge and complex. To address this complexly issue, after the design specifications are complete almost all the other steps are automated using CAD tools.



So, basically what we are going to expect in this course; as you as you can see it from the title that this is saying about optimization techniques for VLSI design. So, what do you mean by optimization? Actually in a very raw language if I want to tell optimization means you want to get maximum benefits from minimum resources. That is minimum what maximum satisfaction which is actually a very idealistic and non-practical goal. So, in all domains of engineering origin, all domains of a life who always work for optimization, that we have some given set of resources and we want to get the max maximum benefit out of it. In this course we are you have two main terms optimization and digital design.

So, what digital design we have all done this course in a undergraduate level, this is basically a simple kind of implementation of hardware in which case we have some inputs in binary form and the outputs you will also get in sometimes of binary numbers. Basically you will be you will have to design some kind of circuits which can process given a digital input and we can give a digital output. So, in a quote unquote this is your digital design.

Now, basically what I want to optimize. So, there are three basic things you have to optimize. First thing I should be able to process this circuit input extremely fast rate. So, it should be have a gigahertz, mega megahertz, gigahertz or even very high extremely high frequency I want. So, that you can process the data in a click in a just in a click, we want very low area our rate why because more area over rate you have larger your circuit will be and you are more cost. So, you want a very less area over rate circuit to do the same purpose and as well as the same time nowadays you are talking about power.

So, basically you have three three part corner optimization problem. Area, power and speed as well why about power because nowadays most of the devices are handle. So, you want a no power for them because you are have a have a higher power requirement, then batteries will go down and you will have a lower battery backup which is one of the most critical requirements. Nowadays we all feel while using mobile phones. Because smart phones have lot of functionalities, but you see that every time in one day for multiple times you have to charge. Say for example, somebody came give you a mobile phone with lot of functionalities as well as seven days battery backup, I think that will be the taker of the market. So, basically these are three things we want to optimize. We want low area, low power and extremely high speed circuit and of course, put as much functionality as we can this is about optimization of the VLSI circuits.

Now, there are another way of looking at it how do you design it. So, you have a circuit, you have a specification rather I should you have specification now you have to design a circuit so, that it has low power, low speed and low area. So, if I talk about digital design there are two ways of looking at it. One way is to optimize the designs so that you have low power, low area, low speed and other ways that how we can design efficient or optimize VLSI circuits. So, in this course basically we are going to look at the other angle, that what are the tools or what are the angles or how we can theoretically automate this solving of digital circuit problems so, that they are actually optimize in terms of area

power and speed. So, one is the circuit itself that can be optimized and another is that as the circuits are quite large nowadays extremely complex as we will see in introductory lecture.

So, you cannot design the circuits manually, you can design a two bit counter you can design an adder you can design a multiplexer, you can design a simple lift elevator, controller as we have already done in the second year digital design course. That can be done manually, but if I say that you design a scientific calculator, the whole processor you design. So, it is extremely difficult all most impossible to design this by hand now think of complex circuits like a processor, multicore architecture, network on chip SOCs it is almost impossible to design any of this stuffs manually. So, basically only the functional requirements and basic top level decisions are made manually, after that there is the cad tool be call it actually you might have heard the name call cad for VLSI. So, there huge amount of cad tools which basically automate this flow.

So, now goal of this cad tool is to generate or optimize VLSI circuit, digital circuit which has low area, low power and high speed. But again that tool also has to be optimized, because if the tool takes 100 years to generate a circuit, that is not going to solve all purposes. So, larger the circuit becomes or more complicate the specification is there, larger is the amount of time the cad tools will take to solve or give the circuit output. So, those cad algorithms also actually optimized; so that you can handle very large and complex systems in a very small amount of time basically this course looks at this angle of optimization.

So, basically this course you can also call as optimization for cad for VLSI. So, cad for VLSI is the very well-known subject lot of courses has been available in the market, but they are generally looking at reasonably low complex circuits. If you look at the traditional cad for VLSI courses, which are available in the internet or the text books they handle circuits which are reasonably complex. But once you go for multicore architecture network on chip and SOCs those traditional cad algorithms will take very long time to solve or design circuits if they are very complex or they will lead to infeasible timing requirements in order to solve them.

So, in this course basically we are going to mainly look at the optimization techniques so, that we can design good cad architecture or good cad algorithm. So, that we can solve this circuit design more faster or architecture can be generated more faster. But as I have already told you that initial part because everything cannot be automated, everything automated there will be no requirement of engineers like us, but the initial specification, that what you want to do the basic hand architecture for the broad block level diagram of the circuit has to be done manually.

So, they have and those will be automatically synthesised to circuits. So, if you give a improper kind of inputs to the cad tools, they will generate improper output. So, on one angle we have to optimize the cad algorithm. So, that they can solve the problems in a faster manner and as well as same time, the first few stages of the design flow has to be done manually and there also we have to optimize. So, our tool will be something like. So, this is one part which is manual, which actually is how to design the circuit broad level specification and broad level block diagrams here you actually optimize, after that there is the cad flow, which automatically there is this broad level dia digit broad level diagrams or broad level circuit level diagrams, and automatically generate the layout of the circuits. So, this part of the cad tool basically is automated.

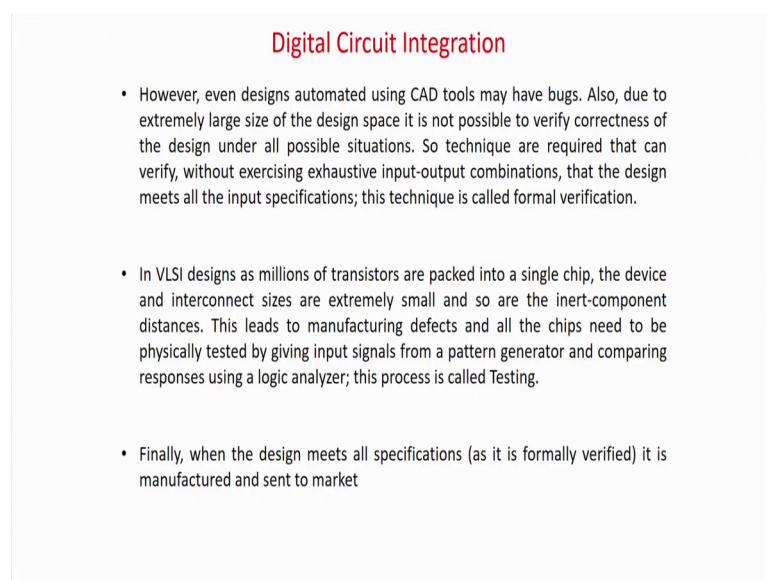
So, now so that they can generate optimize circuits with low area, low power and high speed circuits as much as possible. So, there are automatic tools which will do that. Now in this in this part of the design flow, we will try to optimize the cad algorithm. The cad algorithms we are going to try to optimize so, that we can solve given in manual specific manual specification and block level diagram of the circuits, we want to quickly come to the layout. So, that that solving is done automatically by with help of algorithms that part we want to have optimized algorithms to do that. So, our optimization course will be divided in this part. So, this introductory slide if you can see basically tells that nowadays how the circuit complexity have increased from small scale integration to medium scale integration, then to VLSI which is the current term right now, but. In fact, nowadays we are also calling the term call ultra large scale integration. So, if you are talking about NOCs and SOCs they are ultra large scale integration.

So, our cad for flow optimization which we will be mainly learning in this course will cater to design of ultra large scale circuits. So, very large number of components are now fabricated. So, we have system off system on chips network on chip and what not, but

any way all these things basically are automated by cad tools you cannot design in a NOC manually. So, you will have a larger diagram, you tell what are the core we require what are the functionalities in very broad level language like c and then automatically the cad tool will generate the circuit for you. But of course, if you are giving a but as a human being you have to tell what is this specification of the circuit, at the same time you will also tell about the broad level block diagram of the circuit like as I told you specification you have to write in terms of high level language like c system c system very log etcetera.

So, there are there is a manual part. So, there if you were they no no because that is an art no automation can design in art. So, art is basically the power or intelligence of the engineer. So, there also there are lot of optimization tools, optimization techniques rather. So, if you are given a system specification and if you do not follow the optimization techniques, then you may give a very wrong kind of input to the cad tools. So, even if your cad tools are optimized to solve the problem in a very fast manner, but still you are going to get a very bad quality circuit in terms of may be high area high power and low speed. So, if it is also has to be properly justify or properly optimized. So, our course we will mainly looking at this anyway.

(Refer Slide Time: 09:32)



Digital Circuit Integration

- However, even designs automated using CAD tools may have bugs. Also, due to extremely large size of the design space it is not possible to verify correctness of the design under all possible situations. So technique are required that can verify, without exercising exhaustive input-output combinations, that the design meets all the input specifications; this technique is called formal verification.
- In VLSI designs as millions of transistors are packed into a single chip, the device and interconnect sizes are extremely small and so are the inert-component distances. This leads to manufacturing defects and all the chips need to be physically tested by giving input signals from a pattern generator and comparing responses using a logic analyzer; this process is called Testing.
- Finally, when the design meets all specifications (as it is formally verified) it is manufactured and sent to market

So, I mean let us go ahead and then we will I will see.

So, now say for example, the circuit is there, there is the very you are you are very intelligent engineer you have given very nice block level diagrams and you are cad tool is the very much optimize. So, you get a very nice circuit with low area low power and high speed in a very quick processing time, you everything is optimized input and also the cad tool. But again how you do know the design is your design whatever you (Refer Time: 09:55) is correct because you will give inputs as some is high level language like system c or system verilog and finally, we are going to get something as a physical layout that is the transistor.

How do you know that everything has been operated properly, that your design from system specification to layout whether all the things are equivalently translated or not. See I said that I want to design a a plus b finally, adder finally, we have got a layout for the adder how do I know the transistor level implementation exactly map is equivalent to your adder implementation or not. Therefore, even it is said that automated tools like cad tools may have bugs; you see cad tools are huge large large amount of software which have been coded by engineers like you and me. So, they can also have bugs. So, how do I know that whatever I have designed or whatever is coming at the transistor level is same as what I intend to design. So, there is something called formal verification. Because if I try to set all possible input conditions it is almost impossible, because you can think that I have say 100 I have a 128 bit adder.

So, all possible input combinations will be 2 to the power 128 plus 128 plus 1 bit cad exponential problem, you can never solve that in a reasonable amount of time. So, you have to check it formally. Formally means we have to state base modelling and mathematically you have to verify that whatever input I have given as specification, the same thing has been implement in the transistor level or at the gate level are we intermediate translation stage we have to verify it.

So, that is called actually verification that is also very very important part in case of VLSI design. So, in this course one full module will be dedicated that how we can go for formal verification of reasonably complex circuits. Because if we look at the cad algorithms, cad textbook or which is all for cad for VLSI, you will find very primary topics like binary decision diagram, LTI, LCT is base model thing is etcetera, but in this course will be slightly jump above that and we see how we can formally verify circuits which are slightly at much higher level I will do it by abstraction. But anyway that is one

direction you have to solve finally, this something called testing. Because its its a very interesting thing about VLSI circuits that whenever we fabricate a circuit we are actually using a very complex technolo fabrication technology. So, the yield is actually not very high. So, yield is something around say may be 60 to 70percent that is out of 100 devices fabricated may be you are having 70 devices normal and 30 will have defects.

Now, what I will do, will I repair them no in VLSI we do not repair basically what we do? We throw the 30 devices out and sell the 70 device to the market and profit is made by basically increasing the cost and we always going to do that, because the higher technology you go for fabrication more functionalities you can get an we can give in the circuits and they will get more market more they will be more market capture you can do, but if you say I want to make a very nice table technology and I do not have any falls then you will be lad behind, many other companies or vendors will come before you and they will capture market. So, what is the idea? I will use a very sophisticated come fabricate technology no problem even if my yield is 60 to 70 percent, I will throw the bad chips in the bin and good chips, I will sell to the market and increase the profit by increasing the cost. So, there are takers.

But again if you reusing around 60to 70 percent we have to be very very careful all the devices, which has been fabricated be tested bad one we have to throw and good one you have to ship. But if the fabrications technology say the 99 per 9999 percent yield is there then you know that almost every device is normal just to the preliminary testing you can ship it to the market. Because you know that one in one million chip will be defected. So, that I can easily replace, but here you cannot do because around 30 percent customers will have a defective chip and you will be out of the market if you do that.

So, therefore, testing is also very very important paradigm of VLSI design. So, there is now again here one slight difference between integration be between verification and test is there, verification means you have to do it formally and these are software this based approach that you have a circuit design, which is you have designer specification level and you have the which I call actually layout of the transition level layout. So, which is GDS to file actually if will by the industrial term in called GDS file which will actually give you the configuration of the layout.

So, now you can verify whether this GDS file that is your transistor level implementation actually formally matches the specification or not (Refer Time: 14:07). Both are at the software level and you have to do it only once because after that things are going to be may be hardware. So, there are two versions of software one version you say that is the high level implementation of your circuit in terms of block diagram, one is the low level implementation in transition you just want to see whether they are formally matching or not.

So, verification has to be done once and in a software case, but in testing what happens? Now your fabrication has been done and one thousand million chips has been fabricated; now you want to find out whether there are any defects in the circuits due to fabrication because formal verification is proved that whatever you wanted to design in the high level is implemented in the transistor level. Now there can be defects while fabricating is not a design problem it is a fabrication problem or manufacturing problem, then what we have to verify. Then be 0 all the chip has to be tested that is different between testing and verification. Now testing means a physical device is there you have to apply inputs and all the devices has to be tested.

So, now you can understand that if I have to apply one extra test pattern, then one extra test pattern is we apply to all the one million devices. If I can reduce one test pattern then actually one million device is been have a lower test time. So, it is very very important to optimize, how I can optimize the amount of tests that how I can reduce test time or how can I have design can have good automated test pattern generation so that I can quickly generate good quality test patterns.

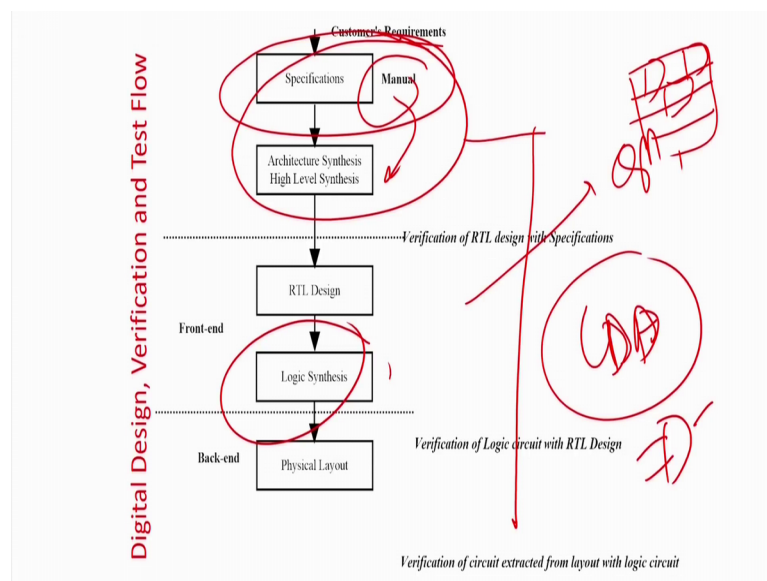
So, basically this, what is typical VLSI design flow. We have design intent, then we make the block diagrams, then basically there is some automatic tools which will give you the circuit gate level transition level implementation and then you fabricate and send it to the market. But now as the as this process extremely complicated will always put automatic cad tools to do that. So, finally, we have to verify it and finally, we have to test it so that we can give good chips to the market.

So, again repeating optimization are at two levels one level is you should have good level goods specification, optimize specification and optimize high level diagrams implementation so, that the tools can because the tools cannot doing magic the cad tools

cannot do any magic. If you give good quality inputs, it will give good quality circuits or optimize circuits as the output. So, you have to give optimal inputs to the cad tool and of course, the cad tool itself should be optimized.

So, that you are going to get very good tools very good circuits at a reasonable amount of time. So, there are also have to do some optimization and finally, when you are chips will be out in the market basically you have to test them physically so, that there do not you do not by chance ship some defective parts. So, here what can be the optimization techniques the of course, the cad tool which will generate test pattern for you should be extremely fast. So, that optimization also you should do. And finally, it test num the debut quality test pattern should also be such, that they do not take too much amount of time to test it so that you can test more number of physical devices in a lower amount of time.

(Refer Slide Time: 16:45)



Now, we will go in to more depth of the introductory part. So, we call it digital design verification and test flow in fact, you can also call it cad for VLSI. So, you can see the first phase is actually call specification that is manual and architecture level synthesis. So, this is a manual specification they are customer requirement and then you have to you know how to develop a specification. Because we vender or the customer will tell in a very non-technical language, they may say I need a processor so, that it can do image processing it can do video game playing this that and you design it for me.

Then as an engineer you have to model it using what can be synthesized as the hardware how do will be the block level implementation, say somebody say that I want to design a calculator. Here to tell that is the scientific calculator business calculator, then what will be adder, there will be multiplier, there will be a subtractor, what will be the keyboard looking looking like. So, all these specification you have to design manually and you have to give high level block diagrams and you can say that this will be an adder after that everybody knows how a adder is to be implemented and there automatic tools, which will do it for in.

But exactly you require serial adder or you look at carry look adder or you want to implement multiplier by using adders, that nobody can automatically do it that this there is actually call specification and high level design block diagram is the manual procedure. So, you have to give good quality inputs to the after that we can say the these are all cad tools.

So, you say that given the specification, you have very nicely designed the block level diagram how to the implemented. So, this RTL is done that is given to you. Then there is some called architecture synthesis, high level synthesis I will tell an example of the that will basically try to map your blocks to real hardware, but at the abstract level then basically this is called the front end.

So, they will convert this diagram, basically sometime people also call that people also say that there is some this part is some manual intervention is still required for this when required architecture synthesis or high level synthesis, there are some helping tools are there like high level compilers are there some tools are available, but some manual implementation is also required to design architecture synthesis one somewhat hardware level implement or hardware level details of the specification.

Because specification can be English, it can be do manual it can be a layman language, but when you are saying architecture synthesis or high level synthesis we are actually telling in terms of block level diagram, what will be the functionalities how will we implement. So, there are some tools should be as well as basically there are some manual intervention is also required because this is an art. But once you tell that this is the adder, this is the subtractor, this is the multiplier, I will do it in this fashion.

So, what are in the block diagrams are told then you can automatically have automatic as this is the digital design, because everybody knows how to design an adder there are automatic methods you know in digital design how to synthesize a counter. So, if you say you require a counter here I require a comparator here I require some block diagram here that can be just in one flow automatic in can be generated, because we know that by solving Karnaugh map Quine-McCluskey method how an adder can be implemented how a counter can be implemented. So, all these basic block diagrams can be easily synthesized.

So, basically what is this part is actually called the front end, which will take your block diagrams make a register transfer level design and from that they will make the gate level implementation these are full-fledged CAD flows. Because of course, we know that Karnaugh map can generally solve up to 5 variables, Quine-McCluskey still can extend to some more number of variables, but in case of circuit there may be more than 1000 variables or have been much much larger than that then their optimization tools, which are actually algorithms called as (Refer Time: 20:08) which are heuristic which actually solve this in a much faster manner, because if I this one let me know come to the example of optimization.

So, this part when I telling you above manual customer request to high level diagrams that you require an adder, you require a ripple carry or multiplier or subtractor is an art. So, given a problem to solve what is the minimum number of circuits required to solve it in a reasonable amount of time. So, for example, something for some operation I required 20 adders to do it, but say that I have a lot of time because I do not require that amount of speed. So, you can use 5 adders and then you can use it in two steps to solve the problem, but if you want a super-fast design then you have to add to 10 parallel adders. So, this part is fully manual or very little CAD comes in there. So, this is in this idea what is optimization here? That this is an engineers optimize engineers an engineers intuiting optimization that even the requirement of speed area and power how we can design the circuit in a block level.

Now, So, there is no CAD involved here. So, we will also see basic design styles, which can give you good quality inputs. So, if you give a very bad quality input to the CAD tool to design we are going to get a very inefficient circuit. So, here optimization will be how I can have very nice how I can in a very in a for my intelligence, how I can solve the

given problem in a more optimized manner because you know your design specification that, what this speed you have to achieve what is the power requirement and what is the area requirement you have, and what is the (Refer Time: 21:30) I am going to put in the market. Say that I can design a very super-fast adder I can do it in very low power may be and I can also have all other requirement. First then the cost of the chip may be very large you will not any customer.

So, you have to think based on the price and optimize and all the other parameters, what do the minimum amount of block size required to solve the problem. So, here there will be some design optimization which I have to do. Once that is done more or less all other part some full of algorithms like if I have a Karnaugh map given to you or Boolean function given to you how can you have the implement in term of minimum number of gates. You can use Karnaugh map, you can use Quine-McCluskey algorithm are for example, you want to implement a counter we know that we have present state is there next state is there then we go for a finite state efficient develop implementation, but all this is can be written as the code.

So, that is nothing, but your cad for VLSI. So, there are lot of cad tools to solve the problem, but only things that now the numbers input variables are not four or five they are hundreds and thousands. So, how to solve it actually they if you look at any standard cad for VLSI course, there are lot of algorithms tool like a espresso which actually solves Boolean function optimization in a very very less amount of time of course, the for optimization quality may not be exactly as the you have these are as are espresso means, what I am saying is espresso to the heuristic. So, depending on heuristic an exact the algorithm like Quine-McCluskey or Karnaugh map, we will always get the minimum size circuit mini or minimal implementation.

But in case of espresso you will get minimal or straightly near the minimum value, but speed up of implementation is extremely fast right. If I you know 100 variable function Karnaugh map cannot of solve it, Quine-McCluskey he will take years to do it because in exponential algorithm, but espresso will solve it in a click because it is a heuristic and it is a very interesting we will also cover in this course. So, that is optimization of the cad flow that if you are taking a qm or Karnaugh map, it will take a long time to automate this flow because from RTL design or logic synthesis gate level design that nobody we will do it manually they will use an algorithm an a completed to solve the problem. But

Quine-McCluskey or karnaugh map is the unoptimized algorithm as I would say because they are exponential in nature and they take a huge time to solve it.

But espresso the heuristic which we can say either optimization of the cad so, that you can get a very quick result in terms of gate. So, here the optimization in terms of cad algorithm I can call it is the Espresso. So, Quine-McCluskey is the normal algorithm espresso is the optimization of them because it can solve the gate level design in an extremely fast manner similarly. So, this is also we discuss all parts optimizations will be discussed. But only thing is that after this part after the verify, after this second first dotted line we it is it is based on the optimization of the cad tool not on the input side is on the cad tool with actually automatically synthesis the circuit for you.

So, will be having good algorithm so, that you can synthesis the very good quality circuit and also the very fast amount of time. Because Quine-McCluskey Karnaugh map can also solve the problem, but will they will take years to design a circuit. But heuristics like espresso will do it in click in a very short amount of time then finally, there is something called physical layout, physical layout means you are we we gate level will be implemented in terms of hardware transistors.

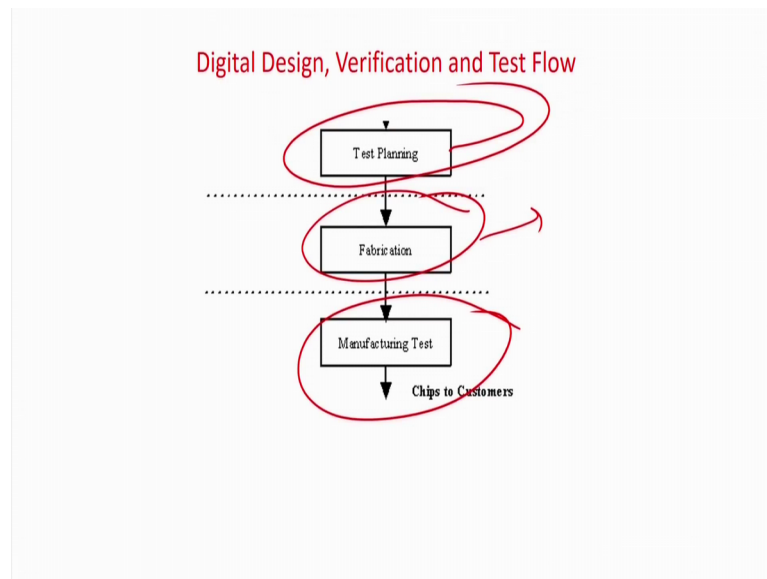
So, in that case also you should have very good type of placement and routing algorithm because what happens in placement and routing basically will have the circuit dining is there and you have to put your gates in all the places and you have to place the layout; layout you have to place this circuits and you have to make interconnection. Because I mean if when you implement a gate will not look like this a gate will look like a rectangle and there is lot of Q rows in a die, you have to put the gates in the proper positions in a minimal amount of area so, that you can to the interconnection.

So, when you this you can look at any standard back end architecture also will slightly cover this in this course and basically you are going to get a physical implementation of that, but know what is an optimization here. If you have if I get a very big die I can put lot of gates and lot of back end spaces will be there you will very easy to place and route there, but now these are very un-optimized synthesis.

So, if you want to bring it closer, there are very small circuit and I have to put all the gates, but if you make it too smaller then you may not able to put all the gates routing on may be not the feasible. So, you should have a reasonable amount of die size. So, there I

can put the gates reasonably and do not waste too much amount of space, still I can complete this circuit. So, you can nobody can do it manually that algorithms to do that. So, again optimization in this case means, how you can get a compact layout, but also that compact layout should not take years to be solved to the some gate level or should not take years to generate the compact layout. I should get it in a reasonable amount of time. So, I have to have good quality algorithms or optimize cad algorithm to solve it as well as the circuit implementation also to be optimize so that I can get a compact circuit.

(Refer Slide Time: 26:12)



Next is finally, the circuit will be fabricated and it has to be tested and send to the market. So, I told you fabrication means you can have lot of defects introduced in this stage. So, you have to finally, manufacturing, you have to test all the chips one by one because I told you each of the chip can have a fall because yield is generally 60 to 70 percent. So, all the chips have be tested. So, very important is something called the test plan. So, what is the test plan? So, test plan is basically so, how I can basically get good quality test patterns.

So, what do you mean by good quality test patterns, because all the chips has to be tested. So, if I give 100 test patterns to solve tester test a circuit nobody will take it. Either very good quality 2 or 3 test pattern because if I keep on applying test patterns there will be one million chip to be tested. So, it will take 10 days or 10 years to test the chips, but I required only one day many to test all the chips on separate in the market, but

on the other hand I cannot the very bad quality test pattern. So, that I do not detective falls at all.

So, here optimization means good quality test pattern. So, that you can ensure that all you testing is done, but also the testing should be size and minimal amount of test patterns are generated and of course, this is actually good quality test pattern on optimization in that direction. But again test patterns will not also be generated manually there will be also cad tools should do that, but if you generate test patterns is 10 years given a circuit I say the algorithm, you will be minimum number of test pattern. So, testing as well as very good quality test patterns I will give you, but good generate test pattern I will required 10 years nobody will take that.

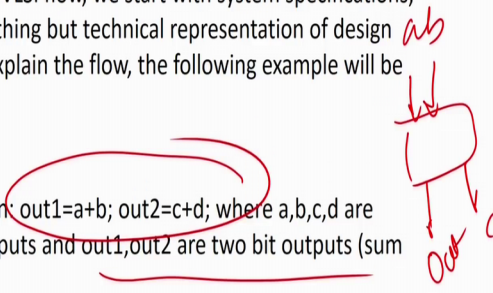
So, you test generation algorithm also has to be optimized and there is the cad flow, and what about output you are giving as there is also has to be optimize in terms of good quality test patterns as well as good quality, this one good quality and faster test patterns.

(Refer Slide Time: 27:49)

Digital Design Flow: Example

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

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



Now, in this entire test flow everything is out optimization how can you get the good quality circuit, optimize in terms of speed layout and speed area and power and also the wherever generating it are they algorithms your generating it has to be also very fast. So, that you get good quality optimize output as well as this process of doing. So, is has to be optimize. So, optimize generate all the level. Mainly in this course I am again repeating mainly in this course will mo mostly in this course and I say will try to focus on how we

can speed up the cad algorithms, but; obviously, the input part which is a manual part to be given as input.

So, that you can get good quality circuits out of the cad is a manual procedure. So, all how we can give good quality or optimized inputs to the circuit to the cad tools rather basically than also will be discussed. But again another very important part which will be also looking at this course, which is which will make it is separate from cad for VLSI course is how we can handle larger circuits like NOCs or SOCs standard cad flow will also talk about this same thing, how can you get good get good quality inputs how can you make the cad algorithm faster.

So, that you can get this circuit outs circuit designs faster and they should be optimize in terms of area speed and power. But they did not they only talk about a circuits generally in the level of gates because they one because they mainly such courses are focus to once circuit of reasonable complexity, but in this course we are actually trying to look beyond that we are trying to lead look at mainly circuits from NOCs, SOCs multicore architecture etcetera.

So, basically one needs to be done mainly the key word here is abstraction that you cannot go to gate level you have to do everything at abstract level like from the behavioural level we have to look at something for the register transfer level you cannot go to gate level. So, all the algorithms actually like test pattern generation, testing, verification, design all we have to do at a very abstract level. So, that at least you can handle very large circuit so that it can go to the system level.

So, another very important part which we learn in this course is optimization in the cad algorithms or designs to handle very very large system, which are which is generally not taught basically in standard cad for VLSI design. So, they cad for VLSI course anyway let us comeback we will now take an example, if you got of theory and open here statements I have mean there is taken example and quickly learn what is the optimization what we are going to do in this course.

So, basically very sim very simple example, we are taking two inputs out one out two it is a plus b and c plus d; 4 4 are this single bit inputs and out one are two bit outputs sum some specification has been given to you then there is some first step is called high level synthesis. Basically is somebody might have told that I have 2 2 bits coming you have to

add them very simple specification here. So, nothing must to thing about I can directly write this, but in the real case is speci general specification are written in text there is 100 and 100 man hours or 1000 man hours goes to converted in to implementable or technical specification. Then in next step is called something called high level synthesis.

(Refer Slide Time: 30:36)

Digital Design Flow: Example

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

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

High level synthesis or sometime called it is the arc architecture synthesis because in this case architecture synthesis not there basically you have to take two bits and basically you require an adder to do that as simple as that, but if it is more complicated then you have to think because in if I says that for a process are you can understand that they will be a full manual or a book, where this pos specification shall be written then you have to thing where have to put adders, for multipliers or require what will be the broad circuit diagram etcetera. And in this case it is very simple specification. So, you can just think an adder is required maybe you can also see a simple half adder will do. So, there is cad VLSI here, but then there is something called high level synthesis.

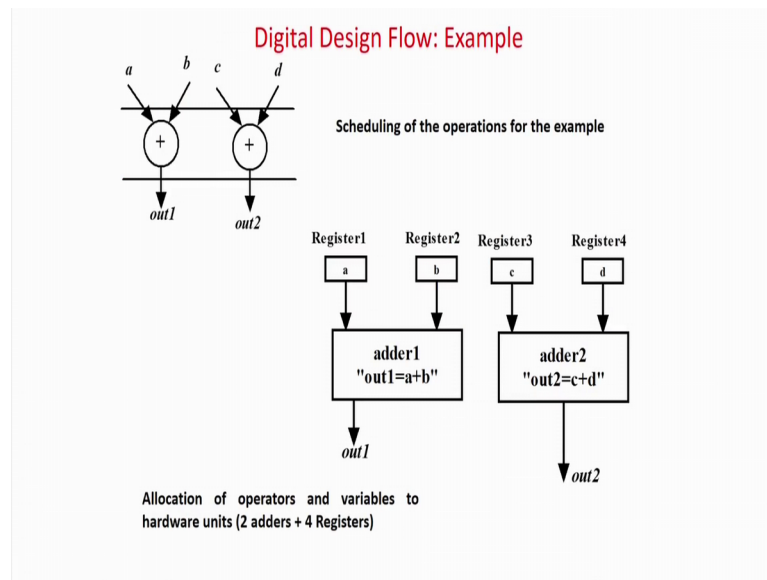
Now, from the specification and block level diagrams maybe I can just thing that you will require a block will have a b and there will be sum out sum out and carry sum something something will be there I do not require anything else. So, block level diagram you can thing for you might, but you will lot in think that what is you are maybe I mean if I say that and b are 32 bit, then at least some in from more (Refer Time: 31:32)

thought can go are that will the ripple carrier there are carry look at the depending on speed frequency etcetera ok.

So, more complexities comes more complexities or more sophistication will come over here, but as you can think this is the fully manual procedure for a reasonably for a many practical implementation because nobody can or no automation can generate from in this language to a sum, that is actually something is that you make an automatic singer are a person who can sing like Lata Mangeshkar there is automatic computer are any reputed singer are computer can do it, that is not at all possible it is my understanding there if everything can be automated that then no there will be no role of any engineers that is not possible, but the main (Refer Time: 32:11) of it is that one has to be design and how optimal it has to be translate from in English language to a technical specification cannot be automated.

So anyway, this second step is called high level synthesis. So, in high level synthesis basically such implementation like block diagrams I have has to be written in a abstract language like we like in system verilog or C and some higher level languages we write it and so, that basically the basically they are high level the actually convert this block diagram specifications in to register transfer level circuits. Register transfer level means you have some registers, some combination circuits, some registers and that fashion. So, that automatically the combination circuits will be fabricate of will be synthesis in gate level by automatic tools.

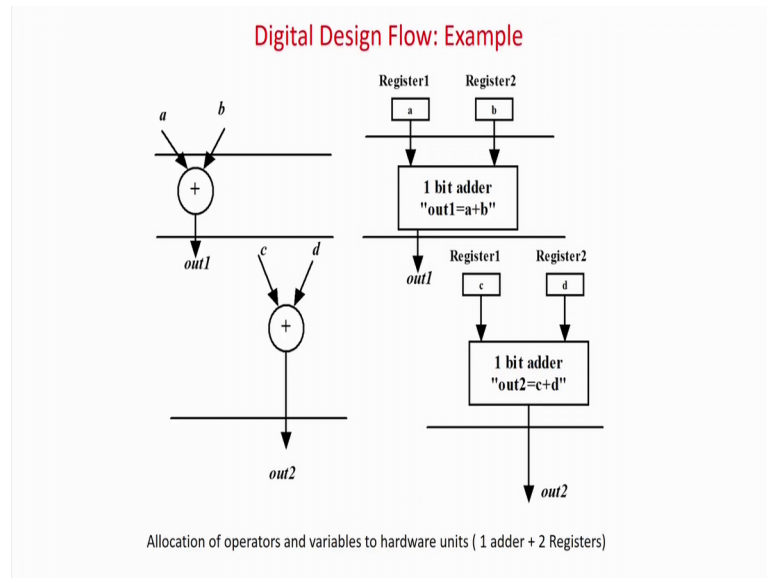
(Refer Slide Time: 32:56)



Anyway this slide you can read it but basically an example will make that example make the more things more clear then again we can come back to the slide. Say for example, we require something like this, this is the actually called scheduling the high level synthesis means first is this scheduling process, that you have to do a plus b c plus d. Do I need the answer in one step or do I need the answer in two step? If I need the answer in one step then over the adders has to be scheduled in step 1 because they there is no dependency register the generated.

Now, fine this is one schedule. Now you now thing is that how many hardware I have how many blocks I have. Say for example, somebody tells you that I have two adders and four registers to do that simple one adder will be there one adder will be there are simple implementation will be there.

(Refer Slide Time: 33:36)



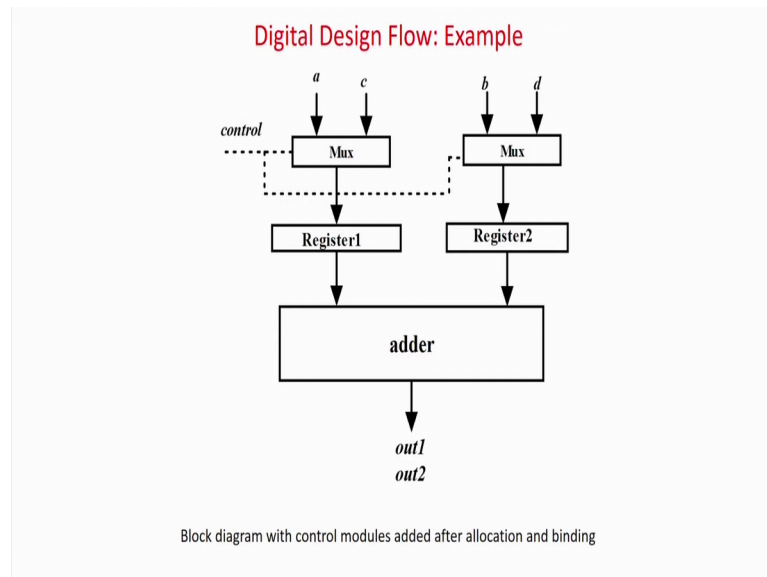
Somebody tells no, I do not have that many amount of adders I have basically only one adder and two registers I do not have that my budget then how do you do it then of course, your schedule will be changed. So, you have first step you do it, second step actually you reuse the same adder and the registers first step you put a one a and b register add it, second step actually you use the same registers put c and d and get it.

Now, you will have to I get the answer in two time step this is a very simple example, but of course, if there is in dependency like if you say that instead of c it is out 1 plus d is equal to out 2, then even if you have multiple resources because of this dependency you have to wait for two stages. So, this is called say scheduling. So, of based on you are basically requirement itself dependencies you are do get an optimized schedule. There are all say lot of textbooks with the algorithms which given the requirement of speed, power, time etcetera they can generate good quality schedules for you.

So, there are lot of optimization can be done to generate good quality schedules depending on the resources available and your timing requirement speed power etcetera like this is example if you see, we will take more number of adders, but will give you solution faster. Low area average or the time of generation is slow. So, this is the first stage of the high level synthesis. So, you are going to get from schedule you are going to get the RTL circuit.

Now, of course, if I implement something like this, there should be lot of multiplexing arrangement this just showed you the block diagram. Finally, you are going more radial towards circuit this one is the more simple your two independent circuits are there uses speed the input you are going to get it, there is no more circuit required.

(Refer Slide Time: 35:10)

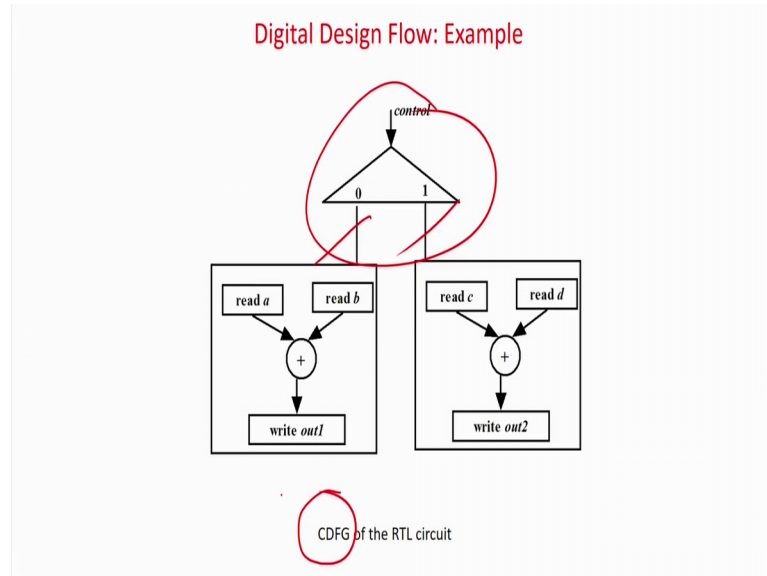


But if there is this if you are going for this implementation then in the problem. So, first time a will go b will go you will added, second time c will go d will go are then you are going to add a 2. So, you require a multiplexing arrangement to do that. So, first the control will be 0. So, this one this one next stage control is going to be 1; so this stage and this stage. So, you require a controller implementation with this.

Now, this part basically you can is an automatic flow, there are lot of as I told you, but still these some manual intervention, but once this is done how to generate the multiplexer, how to generate the adder and how to generate the register how to generate the adder is extremely state forward there is no manual intervention required. But whether I required this schedule or whether I required this schedule there are lot scheduling tools available which can automatic, but still I I generally call semi automatic because there are lot of schedules available will be coming out of your tools and then you can select any one of them to decide, which one is the best suiting for you. But once you get a register transfer level diagram like this there is no more manual intervention

required because everything can be automated, because they are just some block level diagrams to for gate level synthesis.

(Refer Slide Time: 36:18)



Now, of course, this is you are controller, I have to generate the circuit for this. So, first stage it will be 0 and second stage it will be 1 and this done. So, we call it as a CDFG control and data path diagram for your RTL circuit after that everything is automatic au automatic via cad tool. So, in the first stage what we are going under look at this slide whatever I have told you is written in this slide you can read through it. So, the input to the high level synthesis tools is some high level language like c system c system verilog that is the top level you will write like this then there are lot of schedule may be a high level tool for schedules the computation at different control steps. So, there are lot of good quality scheduling algorithms available.

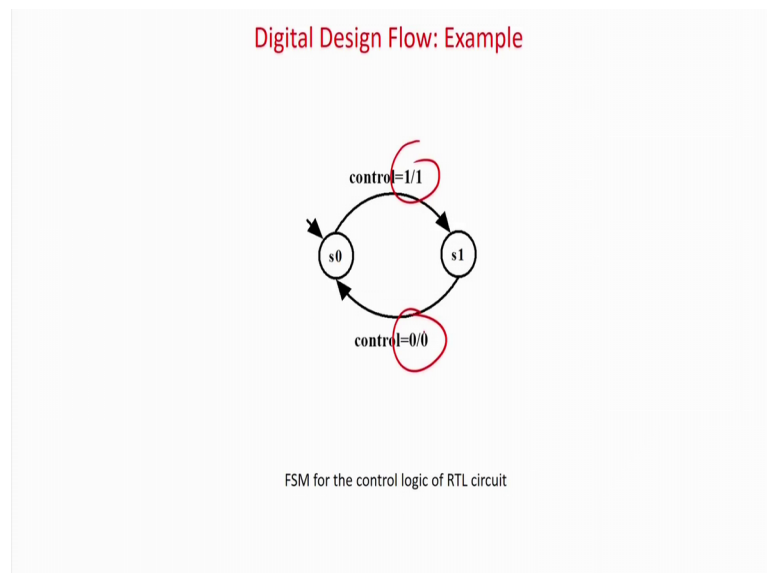
So, they will give you lot of schedules, you can take anyone of them or even if you just want to take by the tool you can take any of the readymade schedules available for that, but anyway in this step for some amount of user intervention is also required. Following that depending on the availability of hardware units and constraints schedules computation that is you are scheduled output are allocated and binded to hardware unit like adders, multipliers, subtractors that is this part is scheduling and after that based on the availability you will bind them to exact hardware in this case this is possible to go for a single state scheduling because there is no dependency. As I told you if I would have c

equal to out 1 that a plus b is equal to out 1 and out 2 is equal to out 1 plus d. So, in that case; obviously, this one will this block will come down.

So, depend based on dependency and similar the factors you will have some schedules generated and based on the availability hardware, you allocate and bind them to the operations to the hardware. So, once that is done that is actually called the high level synthesis, which is fairly automatic, but still some manual intervention still required, but after the RTL circuit is gone it is fully optimized. So, what we will study optimization here. We will first see that how can you write good quality high level system C system verilog or high level languages to give a good quality of inputs like.

In this case it is very simple a plus b c plus d, it is not much you can do over here, but will see examples today when you can write slightly more how given input, how you can slightly tune it so that you are going to get a good quality circuit output. So, given this specification, the input to the high level synthesis that is the some block level diagram how you can write in a more optimize manner so, that the circuit output you are going to get is much better. So, that optimization we are going to look at. But as I told you after this everything is very much automate automated.

(Refer Slide Time: 38:41)



So, how do you implement this controller this is nothing, but control 1, control 0, that 1 0 1 0 means first it may be a 0 then you can get a 1. So, it is up to you how do design it because you may be it should be something like this should be a 0 and this should be a 1

sorry it should be something like I just made a mistake. So, you can say that in the 0 and you can say is the 1 that be in some the first you will generate a 0 and then after sometime you will come over here and the stage we generate a one.

So, in this case it is 0 0 1 1 that is same way of implement this is another way of writing in this case first is gen in this case is first it generates basically your c plus d and the generates a plus b, and if you write it as 0 and one then you can will generate a plus b first and then c plus d same way. But this is the final stage machine implementation basically for this controller, but as and as I know you know the final stage machine can be easily very easily directly converted to gate level. So, that part is also fairly automatic now what to optimize over here? That given this diagram finally, I will get a gate level diagram like I need to gen synthesis, mux, register adder then this controller from there I need to add a gate level circuit. So, this I want to do it very very fast.

(Refer Slide Time: 39:45)

Digital Design Flow: Example

- After the RTL is verified to be equivalent to system specification, logic synthesis is performed by CAD tools. In logic synthesis all blocks of the RTL circuit is transformed into logic gates and flip-flops.
- After the equations for the circuit specifications are obtained they need to be minimized so that the circuit can be implemented using minimal number of gates. Karnaugh map, Quine-McCluskey algorithm etc. are some standard techniques to minimize Boolean functions.
- Again equivalence of logic synthesis output should be established with RTL design.

So, here the cad tool has to be generate has to be optimized in such a fashion. So, given the RTL circuit as I include as the block diagram and the input, the gate level implementation can be very very fast. So, there are. So, when we will be looking at such problems such type of issues routing. As I told you that if you are going to use Karnaugh map and Quine-McCluskey optimization is better that you just go for a heuristic way like espresso which can give you the circuit in a more faster manner ok.

So, this what I have written, after the RTL is verified equivalent system specification logic synthesis performed by cad tools logic synthesis means from RTL to gate level diagram. So, they are transformed into logic gates and flops. So, the cad tool has been very fast and you have to do it in this manner. After the equations for the circuit specifications are obtained they need to be minimized. So, that the circuit can be implemented in minimum gates that this is logic synthesis that circuit specification is there in terms of adder block multiplier block for this final state machine as I have shown you Karnaugh map Quine-McCluskey are going to must standard ways to minimize Boolean functions, but again there very very slow. So, you have to they have they have to made faster the optimization using heuristic like espresso.

So, in the second part basically in the from one words, if you look converts from r t RTL to gate level or transition level implementation, that is your gate level synthesis and your back end the optimizations will be mainly how to make the cad tools operate faster so that they can handle very big circuits.

So, in those part will be looking at optimization in that algorithm. Now again impartment important is that, everywhere you have to find out the logic synthesis should be equivalent with the RTL design everywhere you have to see the equivalents that is actually the verification. If you look at it you have to verify that whatever architecture synthesis you have done should be equivalent to RTL. Whatever gate level design you have done to the physical layout, let us should be equivalent to the logic gates.

So, in all the stages you have to formally verify, that what was our translation from stage a to stage b equivalent should be maintain that is it should not be something like I had some x here and while going to the next stage I have implemented something called x prime or y which is different that should not happen they are implementation in a different versions or in different ways like specification, block diagram, RTL gates layout, but implementing the same functionality, only the way of representation is different.

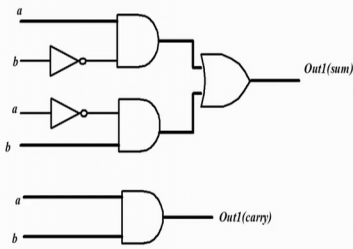
So, equivalents have to be maintained and as I told you in this case you have to do it formally because all possible inputs cannot be given. So, here optimization should be in such a manner so that I can do the verification extremely fast and more complete.

(Refer Slide Time: 42:08)

Digital Design Flow: Example
Input-output of the adder module

<i>a</i>	<i>b</i>	<i>Out1(sum)</i>	<i>Out1(carry)</i>
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the table we have Boolean equations for
 $Out1(sum) = \bar{a}b + a\bar{b}$ and $Out1(carry) = a.b$

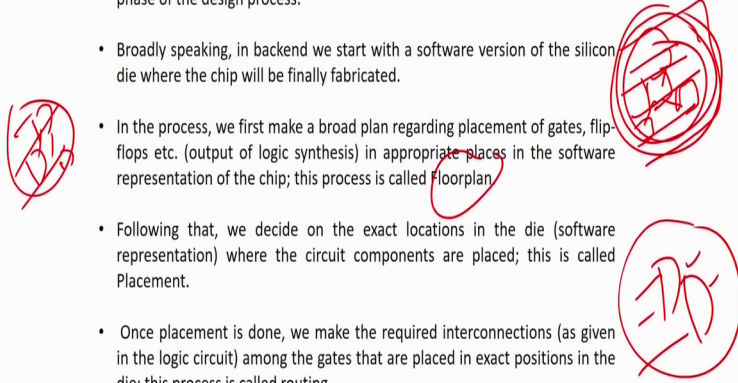


So, this is the example of a gate level synthesis. So, the simple adder as you know that this is a b full carry uses require Karnaugh map to get the circuit as the output, but if it is very large Karnaugh map Quine-McCluskey will not able to it you have to use some heuristics called other told you espresso.

(Refer Slide Time: 42:27)

Backend

- Once the logic level output of the circuit is obtained we move to backend phase of the design process.
- Broadly speaking, in backend we start with a software version of the silicon die where the chip will be finally fabricated.
- In the process, we first make a broad plan regarding placement of gates, flip-flops etc. (output of logic synthesis) in appropriate places in the software representation of the chip; this process is called Floorplan.
- Following that, we decide on the exact locations in the die (software representation) where the circuit components are placed; this is called Placement.
- Once placement is done, we make the required interconnections (as given in the logic circuit) among the gates that are placed in exact positions in the die; this process is called routing.



Then once the gate level is done basically you have to as I told you you have to fabricate this circuit in terms of layout. So, there would be a die. So, you will have some rows and columns. So, the gates has to be placed and finally, they have that is actually called the

floorplan, if wherever it is gates will put to will be the memory block. So, broad level placement and placement we do above the gates which is called the floorplan and then basically finally, the all the gates will be place which is called the placement and finally, based on the requirements of the circuit they has to be properly connected which is called routing.

So, now again tell me what to the optimization. So, what here? Simple I require a very compact design because if I have very large design, things were cluttered up and basically you will; obviously, get an implementation, but will be high cost. So, I want to put everything together, but if not is should not be so compares that I cannot put everything. So, it has to I have to get an optimize size die and of course, the optimality cannot be coming in years. So, I have to have a very fast algorithm so that I can get a good quality backend design in a reasonable amount of time.

So, here we going trying to see in this course will have a one one one lecture will be dedicated for this, which will discussed that what are the placement and routing how it is basically done and what it is what it optimizes. So, that you have a reasonably good quality circuit implemented in a; because this finally, what is going to be fabricated in a loop as simples very nice gate level circuit like this or finally, it have fabrication diagram loop something like this.

So, one figure we have already shown you in the introductory lecture, which is the fifteen minutes lecture we have posted on this site as the introductory part you can just see how a layout rows. So, layout basically has some rows, columns and blocks are over there. So, finally, I should get a front end this it is not as nice as this, it is a die with some rows and some gives a put here and there and some inter connections are done.

So finally, that should also be the reasonable space or circuit has to be are gates has to be put and final it will be placed and routed and again transfer to be done in a reasonable amount of time, because placement routing should not it huge amount of time so that your design speed is high. So, this again backend is fully automated by cad tools and are the optimization mainly will be time to look at in this course is, how they algorithm what are the algorithms to go for this backend design and basically how why how they can be made faster. So, that you can solve the backend problem or physical layout in a faster manner and you can get a compact layout.

(Refer Slide Time: 44:40)

Testing

- In VLSI designs millions of transistors are packed into a single chip, thereby leading to manufacturing defects. So all chips need to be physically tested by providing input signals from a pattern generator and comparing responses using a logic analyzer.
- Testing by applying all possible input combinations is called exhaustive functional testing, which is avoided because of prohibitive time requirements.
- Testing is therefore done based on "structure" of the circuit and is called structural testing.
- In structural testing we first decide on set of faults that can occur, called Fault Models; stuck-at, bridging etc. are some well known fault models. Then we apply only those inputs which are required to validate that faults (as per fault model) are not present.

Now testing; so already I have told you in in VLSI design millions of transistors packed into a single chip. So, they have lot of manufacturing defects. So, as the yield is not very high, not like 99 percent plus, which is the generally case for traditional engineering devices like electric fan and iron. So, we all know how to design a fan and a iron. So, once you design it 99.9 present surety is they will do there functionality.

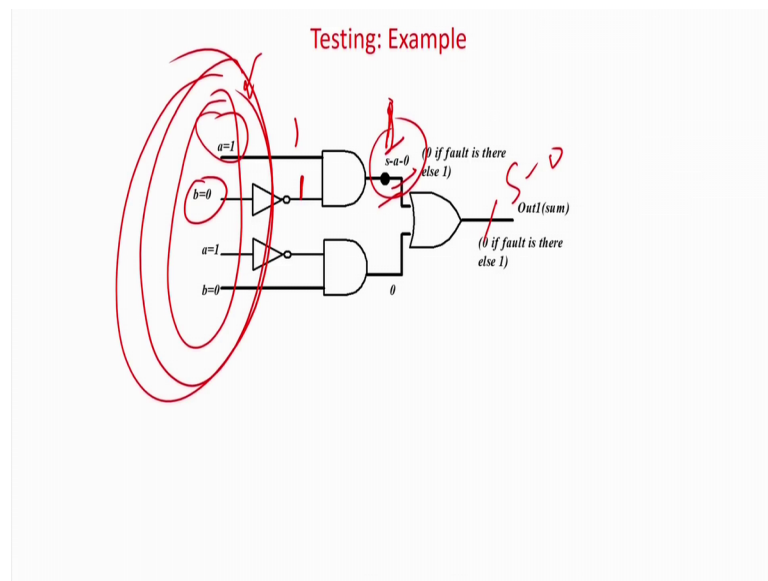
So, there testing is not that important, but in this case and in this high. So, you have to test all the circuits. So, testing by so; obviously, testing by all possible combination which is the exhaustive impossible because N inputs 2 to the power N and all the devices has to be apply verification is still simpler, because you have to just take one version of the implementation, one version of the other version of implementation and just see whether then equivalent in the software level and if it is equivalent you are done. But it testing each chip has to be tested. So, of course, it is out of question to put of even think about going for exhaustive test.

So, there are lot of optimization which people I have done which will be looking at in this course, that if will do testing on something called the structure of the circuit, which is called structural test and if it is also something use something called fault model that where are then verify the functionality they take a circuit and they have very well known fault model. At least will just need to verify or just need to test that none of the faults some that fault model in this circuit. People (Refer Time: 45:57) because people will

have already verify statistically and over years, that the somebody can find out that these circuit does not have any of this model faults of form a fault set, it is the very very small subset of the exponential complexity of inputs, then you can be reasonably ensure that is circuit is working and then you can sell in the market.

So, here what will be the optimization?

(Refer Slide Time: 46:15)



So, I just giving that test for this adder, basically we will started for model is the very well known fault model in which you ensure that none of the inputs are stacker 0 and stacker 1. So, may be the finding out whether this times is stacket 0 or not. So, if you going to verify that this time is or test is time is not stacker 0. So, you have to put a 1. So, one in this and get me this a has to be 1 b has to be 0. So, this is one right this is 1 0 means this is I am putting has 1, this is I am going to see how it is happening. So, 1 means it will be a 0 over here 0. So, this is the output of the and gate is 0.

So, now you can see this is 1 normal case this is 0 normal case. So, output is going to be 0 if they if fault is there, because if there is the fault this one will be convert into stacker 0 because this line is stacker 0. So, in the case will be 0, but in the circuit is normal there will be 1 1 or 0 is 1. So, they if I find out one and the output circuit is normal otherwise is design in a stacker 0 1.

Now, in testing you have to verify that none of the line is having a stacker 0 fault. So, when will be dealing in express because express we are full module with 6 lectures on testing. So, there will see how basically this is the more optimize implementation cause core in compressive exotic amount of testing in this case if there 4 input lines. So, all possible will be 2 to the power 4, but it very large circuit with hundred inputs or possible input patterns will be 2 to the power 100 not known not even any question or doing it, but it will be later shown in this course that how this stacker fault model there will there very efficient manner, in which case we required a very very less number of patterns and still we can ensure that none of the stacker faults are there and your circuit is (Refer Time: 47:44) fine.

Now, what to optimize here? Lot of things has to be optimized over here, given a circuit and a fault like this how can you automatically find out what is the test pattern to do that and that also the very fast because as I told you I have done something like 0. So, I have to put a 1, if I requiring a one in the output of the, and gate. So, I have I require a 1 and here I require a 1. So, this is the, and gate I require a 0. So, this back working I am doing and is the stacker 0. So, I require a one this is go synthesis faults sensitization.

So, but you can clearly get an idea that this can be very easily automated using a graph traversal algorithm, but again it should not take years and years to generate the test pattern for that we should not do that, we should be quite fast to do it. So, one optimization is given in the circuit is very very large how I can generate such test pattern extremely fast. Secondly, test pattern should be a good quality what you mean by good quality? Like for example, this pattern can test a stacker 0 fault even here you can easily verify that even this stacket 0 fault here also can be verified. Same pattern this same pattern all can also verify or stacket fault on the output. So, this is the good test pattern because it can test multiple faults.

So, e are there some good quality test pattern which can test multiple faults, I would rather take them rather than taking a fault an only dedicate test pattern for that that I will not do it is the test pattern which can test 100faults I will take that. So, how can we generate good quality test patterns and also very less set of test pattern. So, that I can go for faster testing, but again the test patterns which had be generated of course, this is by adder automatic test pattern generation cad tools, they should also operate fast. So, that I can get good quality test pattern and also the faster it.

(Refer Slide Time: 49:18).

Verification

- In digital VLSI design process, after specifications are complete almost all the other steps are automated using CAD tools.
- However, even designs automated using CAD tools may have bugs.
- Also, due to extremely large size of the design space it is not possible to verify correctness of the design under all possible situations.
- So techniques are required that can verify, without exercising exhaustive input-output combinations, that the design meets all the input specifications; this technique is called formal verification
- So we need to have formal verification methods which verify equivalence of
 - RTL with input specifications.
 - Gate level designs with RTL
 - Gate level design with layout

SPL

|

Last

Last, but not the least is verification. That at every stage you have to find equivalents right like you have to find equivalents RTL with input specification gate level design with RTL, gate level design with layout. So, every stage you have to find equivalents because finally, you are going from the layout you are going to implement basically implement these specifications specifications. And finally, you have layout. So, they are implementing the same thing, but there lot of translation that has happened in this step. So, you have to find out whether they are equivalent.

This is a pure computer science mathematical problem because you cannot apply all possible input should do it, you have to mathematically verify that whether it happens are not. Basically and in that case it is a very difficult problem to be solved. All the phases till now I have been discussed like design of a circuit with the very very large circuit like NOC SOCs testing an NOC and SOC is more or less it is possible with tools and optimization techniques available we have very very very high size circuits, but verification is a exception here. Even with extreme amount of research we have done tools we have high power servers we have, we have not been able to formally verify this in reasonably complex circuits like entire processor in one go.

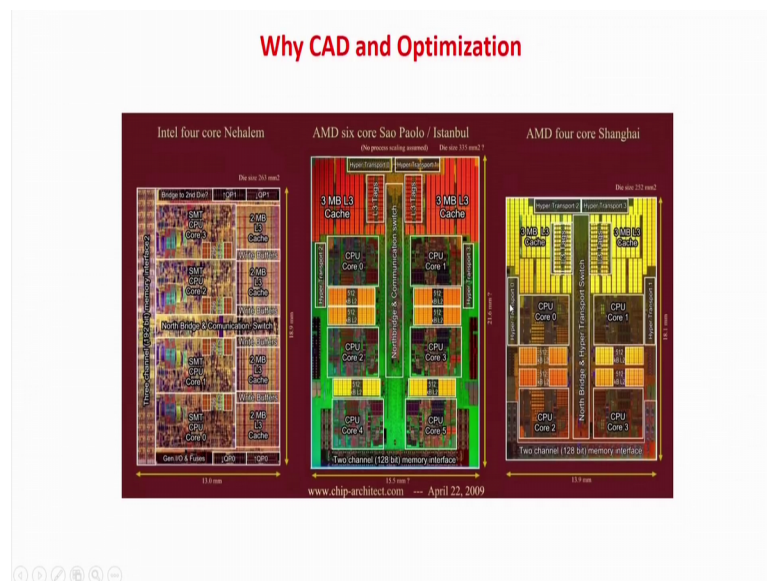
Entire processor say with four course or eight course, we have did not a very techniques or algorithms to verify need one go because the more complex problem they all others. So, which will actually also have a full dedicated module on verification. In most of the

cad flow we just see verification of very very small circuits not even the range of to mo above 2 to the power 20.

Basically we have some state, state machine implementation and then we go for some kind of model checking on that. In this course basically we will try to go ahead of that, like we using we will be using binary decision diagram. So, that we can slightly move ahead then we will see very abstract level modelling and we will try to verify or that level, then instead of verifying the entire system can I give something reasonable in a certain amount of steps because generally the bugs if there are some bugs the generally happened very earlier in the design.

So, in case of verification, we will try to see how fast we can get the solutions or we able to verify mathematically, but still with all this even here not even be able to go to very larger complex circuits is the very challenging problem.

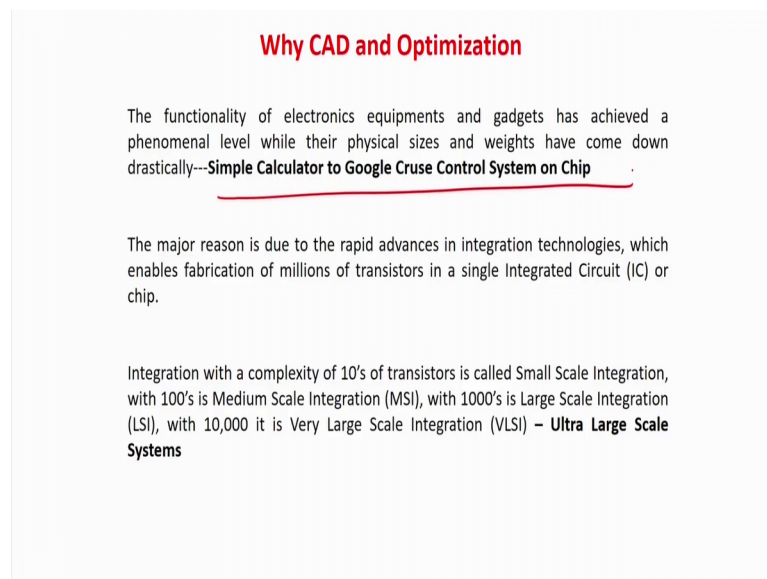
(Refer Slide Time: 51:24)



Now, the question, why cad and optimization? Again just repeating in two words 90 percent of the digital design flow is automated. So, there are cad tools we cannot read manually. Secondly, we want good quality solutions and also in a very fast manner. So, cad for VLSI actually all this standard courses you will tell you some of all the cad algorithms which one for reasonably complex circuits. But now this course will try to cover all this thing basic and then we are going to see optimize the cad algorithm, so that they can handle very very large system like NOC and SOC.

Also we will talk about how the solution quality also can be improved, if I very fast algorithm to solve it, but the solution quality is very bad, the circuit we generate is very bad no one will take it. So, our optimization will be makes in the making the cad algorithms work fast to give a circuit solution quickly as well as this circuit test pattern, we also be the having the quality. So, therefore, cad and optimization as very very required require mean are very intelligent terms when you are looking for large level circuits like NOC and SOCs.

(Refer Slide Time: 52:27)



Why CAD and Optimization

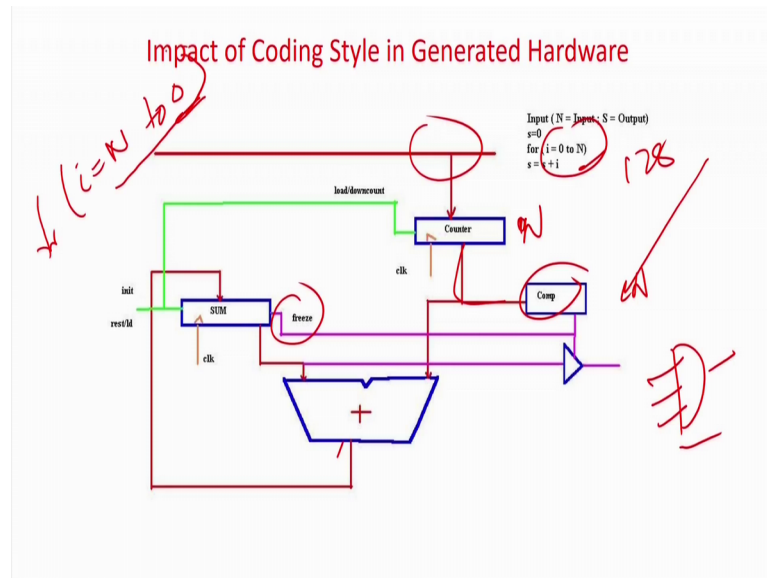
The functionality of electronics equipments and gadgets has achieved a phenomenal level while their physical sizes and weights have come down drastically---**Simple Calculator to Google Cruise Control System on Chip**

The major reason is due to the rapid advances in integration technologies, which enables fabrication of millions of transistors in a single Integrated Circuit (IC) or chip.

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

So, basically I have just are we tell I will just the modification from a single simple calculator to Google cruise control. So, that is this jump we have made and all our algorithm should jump from a calculator to a cruise control system.

(Refer Slide Time: 52:37)



Now, some specific examples which will tell what are the optimizations will look about. So, this is the very in simple circuit, I want to design for input N output S. S equal to 0 simple loop for I equal to 0 to N s equal to s plus I. Basically this is your block level circuit for this there is an adder. So, basically is the counter the counter will come count from 0, 1, 2, 3, 4 up to N this is one input and basically these are a and an this some basically nothing, but s which is the temporary register storing the value. So, initially it is reset.

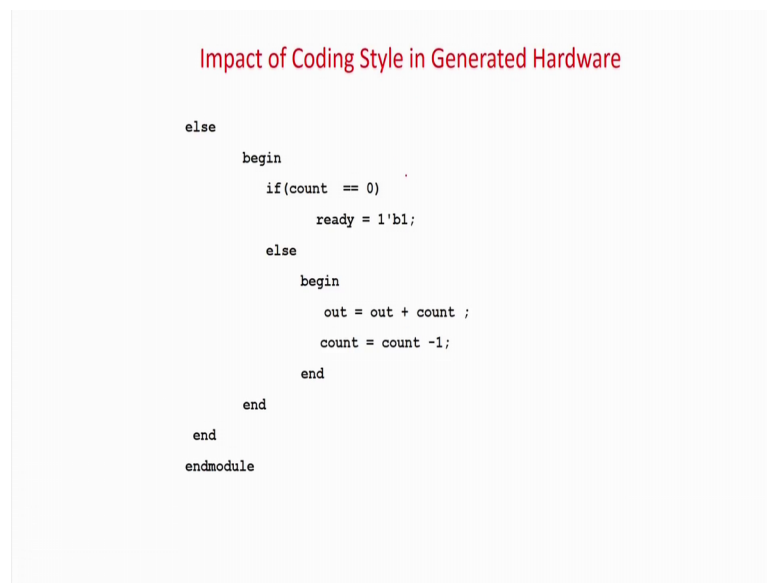
So, I am not going to the details the very standard simple digital implementation. So, initially sum is equal to 0 the counter is also basically equal to 0 or 1 whatever you want to set it every time it will add s equal to s plus i, this S is this will be equal to the value of the counter, it will be stored in the sum. It will keep on doing in till basic sorry it will keep on doing this architecture, it will (Refer Time: 53:31) ok.

So, it will keep on taking the values from the counter and it will do it till when there is the comparator, which compares the counter output with N. So, when the counter value is N it will actually freeze this sum. So, that the sum value will be retain and S will have the value of S plus i. Now the based on this you have to design, this architecture this no tool can do it idealistic is the very simple example. So, there may be possibility for just think that if the slightly complex specification, you have to tell this to be that I will have a sum, I will have counter, I will have comparator, this feedback after that basically we

know how to design (Refer Time: 54:10) automatically then some register automatically counter automatically comparator automatically, thus everybody can do it.

Now this is a design which you have to give how can I optimize it. Just let me just change is slightly, if I say that because we know that what is the count comparator with N will required there lot of S (Refer Time: 54:29) gates required slightly if I change it for i equal to N to 0, just i it was the counter.

(Refer Slide Time: 55:52)



So, what I will do is there initially I will load the counter with N instead of 0, I will load the counter with N directly I can load it from this the backend then what is the comparator level to? The compare has to compare with 0. So, what is compare with 0 involve? A simple and the gate. So, if I just reverse the loop like from N equal to from i equal to N to 0 instead of 0 to N, I am going to get the same implementation, but if I have to compare something with N, the comparator will be lot of (Refer Time: 55:07) because I have to compare with the normal value of N n can be say a 120 then you required some (Refer Time: 55:10) to do it.

But if I say that from N to 0, then you are going to just compare with 0 terms it will be a down counter cost of up counter and down counter is similar, there were down counter and whenever you get an N you have to started. So, compared with 0 is simple here end this is one optimization technique which no tool can automate, that is based on the designers intuition how we can write good quality architecture.

So, from 0 to N or N to 0, we can have a most simpler design if you are having (Refer Time: 55:42) implementation from N to 0. This is the I am just some verilog level diagram for this hardware and basically you can see that if you are using a counter. So, basically if count is equal to equal to 0 this is the down counter. So, basically here I am freezing it. So, basically this verilog implementation will be cheaper because I if count equal to 0 instead of n. So, if I look for N checking basically this will be a more larger circuit because count checking with N is a lot of (Refer Time: 56:10) gates involved, but count with 0 is the simple adder and get simple adder means simple example is there.

(Refer Slide Time: 56:18)

Impact of Coding Style in Generated Hardware

- **Impact of**
 - function
 - Data path width
 - Loops
 - Arrays
 - Data types

Only three new data is needed.
Can we re-write the loop to incorporate this?

```

Input A[N][N]
Void par_access ( data_t A[N][N])
{
  for(i=1 to N){
    for(j=1 to N){
      n = ( A[i-1][j-1] + A[i-1][j] + A[i-1][j+1] +
            A[i][j-1] + A[i][j+1] + A[i+1][j-1] + A[i+1][j]
            + A[i+1][j+1] ) / 8 ;
      out (n);
    }
  }
}
  
```

They want to generate the hardware, in which case basically you have to take this central block and 1, 2, 3, 4, 5, 6, 7, 8 this 8 block average it is taking. So, if we just look at this loop this we can zoom and look it they are taking from i to N, j to N that they are taking all the periphery numbers and to dividing it by yield. So, there we are going to generate the average. So, if somebody we will tell you this point we have you generate this average.

Now, say other I want to doing a moving window. First I will do it for this point then I will do this. So, this is a moving in the concept the again if I simply give this as an input this one row implementation if I give as the input in a high level language like this, you tool be generate the hardware tool. But you see there lot of redundant arithmetic is

involved right for example, if I am taking this point, then you have to add only this three bits extra because all this bits and this bits basically have already been added.

So, if you are using a rudimentary high level language to specify a design like this, your hardware implementation will be extremely compromise. But if I sprightly take the algorithm that basically if I want to go for this part, only this new bits has to be added and because this part is already added you just implement and do it you will get a faster solution. So, this is the impact of coding style in hardware generated.

(Refer Slide Time: 57:31)

Impact of Compiler Optimizations on HLS Results

- Compiler transformations can have significant impact on HLS results
- Can we rewrite this code so that we do not need any multiplier?

```
c = 7;
for (i = 0; i < N; i++)
{
  y[i] = c * i;
}
```

→

```
c = 7;
k = 0;
for (i = 0; i < N; i++)
{
  y[i] = k;
  k = k + c;
}
```

- Data flow based optimizations
- Control flow based optimizations

So, therefore, the first part that is from the specification, you are going to generate from high level language and high level specifications, you are going to generate in terms of c or system c you should have intuition. So, that I can write good quality high levels codes so that the ultimate hardware very cheaper like the two examples that I have given.

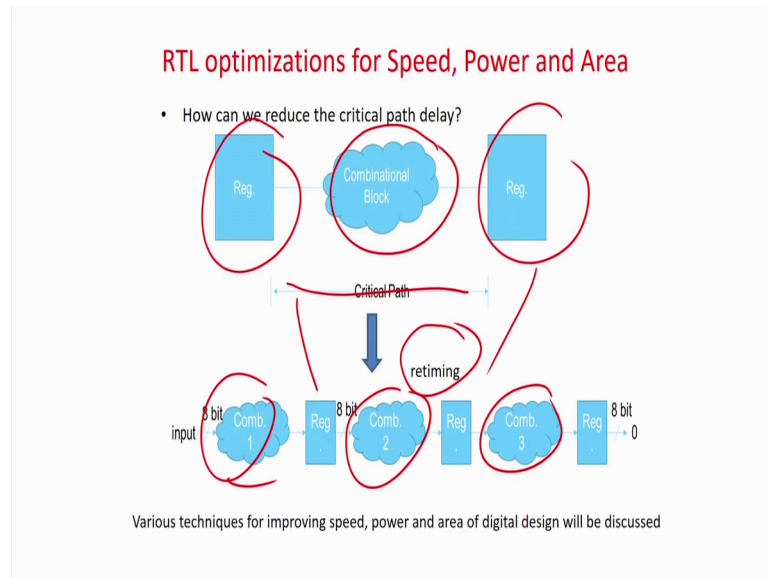
So, in this course first we will see optimizing techniques at that level so, that I can get the final implementation nice. As I told you, you give garbage to the cad tool you will get garbage as the output. If you we give a good quality input optimized circuit level implementation the input of the cad tool, the cad tool also give you good quality circuits. So, first level prove the design optimization and the high level or abstract the specification level or the abstract level how good quality codes can be return so that you can have a good quality solution at the end.

So, next is whenever you are writing some high level tools in c language, basically there will be lot of compilation involved there will be lot of compilers will be involved will generate the verilog code because you have to finally, get the RTL design. That part is as I told you semi automated or more average less automated. We will write some c codes like this and then or simple like this finally, you will going to get the gate level implementation by automatic tools.

So, there will be lot of as you are going to higher level of specification or or representing the circuits there will be lot of compilation involved. So, when there lot of compilers involved also there will be lot of compiler optimization coming to picture like for example, this simple c code. $C = 7$ for $i = 0$ this, $y = c * i$, but again this requires a multiplier. So, multiplier is larger in circuit if I can slightly compiler optimization which is called strength reduction. So, this same thing I have written with slight modification with any one extra variable, I can implement the same thing using an adder you can just check it. This is simple standard comp compiler optimization fundamentals that is one basically this called strength reduction dead code elimination. So, whenever we are designing circuits of the gate level, all this things should not come in to picture.

Now, basically what is happening is that you are writing the c codes at very high level the are circuits specification high, you are the do it because there are your specifying a processer you are specifying NOC, you cannot do it at the than well level you have to get a much higher level. So, there is no question of doing at the bit level, but whenever going at the higher level of the design abstraction, codes has to be compile and lot of compiler optimization will come in to picture like one example have given you in this strength reduction. You strength reduction means you will require less hardware to implement in basically.

(Refer Slide Time: 59:59)



So, one part few lectures will be dedicated on how you can have one of the compiler modifications you have to understand. So, one optimizations you have to understand. So, that you can get good quality input specification to the cad so that the final design is also good. Secondly, very large circuits as you already told you there will be lot of problems in speed area and power.

So, you also see how can we actually go for low speed circuits and low area circuits and how we can lower area, lower power and how speed can be input there is this something called basically part clock partition in. So, if you see the register and one register is there this is the combination circuit, it is say very big block this is the critical path and your clock frequency will be less. So, various techniques are available in one case actually break it up which called retiming.

So, your breaking the combinational circuit in to smaller parts and putting registers in between. So, if the combinational circuit is smaller in size, higher speed can be achieved is something similar to pipeline architecture. So, some modules or some lectures will be dedicated, that how we can write this circuits my there all the parts with the design is being optimized. Because as human we are specifying the design requirements at the top level of the specification high level specification, they are not automated. So, at this state level we also look at optimization for speed power and area in terms of how we can have retiming this circuits, how we can actually you have low power low switching. So, there

is power is reduce what we can do like in this case. So, that the area as optimized like in this case as I have shown the area will be optimized; this parts cannot be automated in that in code un code automated they have to the because there all engineering problems and there are. So, basically this is has to be done by the engineers.

So, the first part of the lectures in this course will be looking easily optimization from the design level, which is given as input by the engineers or designers at the high level specification. Next once the RTL is given, there is the gate level synthesis from now cad starts there is very little intervention of the engineers.

(Refer Slide Time: 61:49)

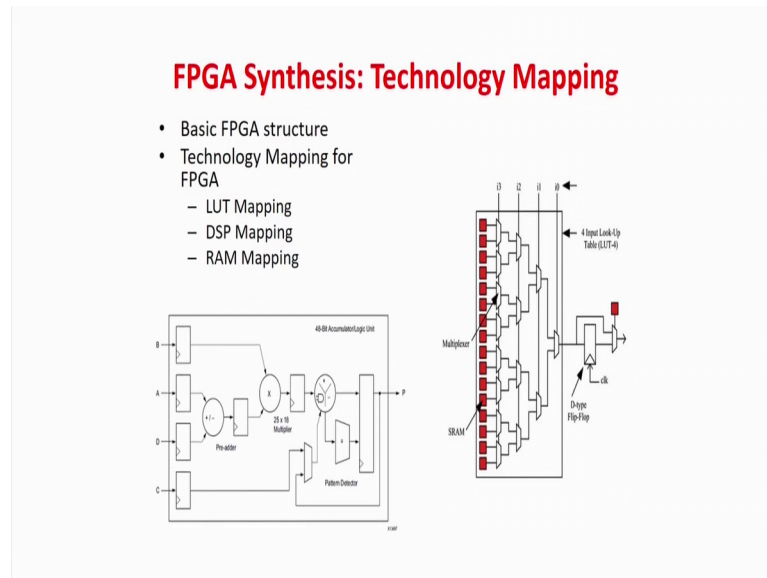
Gate level synthesis optimization

Karnaugh map and Quine–McCluskey techniques work well if the number of inputs is less. However, in case of practical VLSI circuits the number of inputs are in orders of hundreds, so minimization is carried out using heuristics techniques

So, from a circuit like this circuit like this, this way, this way, this way a circuit has to be designed. So, from the RTL you have to get a gate level specification gate level output. Is very simple, there automatic tools should do it, because as I told you Quine-McCluskey solve this problem, Karnaugh map can solve the problem, but again if the number is very very high there should be heuristic, in this case will study a heuristic called espresso, which actually solves this problem in a very fast manner, because everywhere you require optimization.

So, in this case the optimization is in terms of the cad flow and not in the terms of inputs or outputs because you have to get a smaller circuit in a very fast manner and this processes automated. So, that automated process of cad has to be optimize so that we will study in the gate level optimization, gate level synthesis optimization.

(Refer Slide Time: 62:37)



Then again nowadays there are lot of variation. So, that either you can fabricate your circuit in axis and nowadays FPGA is are coming out.

So, now we are very very large size FPGA. So, even processes can be implemented and valid in FPGA. So, few modules will be dedicated where we will try to see basically how you can have a optimize design if you have a FPGA in place. Because FPGA is not a gate level block there are something called LUTs, readymade blocks like digital signal processing block RAMs some dedicated blocks are available. So, say that I require a multiplier you need not synthesise the multiplier should be available. You have some digital signal processing block already there available.

So, you can directly map with there, even those fix blocks how the other parts can be optimized there is some technology which is recently coming and not too much discussed in normal cad for VLSI books. So, in this course will there will be also dedicated lectures on FPGA synthesis optimization, that given it because in acic or normal cad everything is this fabricated multiplier means you have to have a gate level multiplier DSP means you have to implement the DSP, but is FPGA instead of basic gates there something called LUT because FPGA is a some kind of generate architecture are available there in terms of LUT is which was to need the do the ra interconnection so that you can get a circuit.

So, basically this is called the field program, we will get everything is same thing can be use for adder, multiplier, tomorrow you can implement a person day after tomorrow you can many many processing block, because you would not require any fabrication for that. So, that is now very very popular because it to get a fab hold to a fab and fabricated circuit a very expressive procedure.

So, as complicated are very sophisticated FPGA is are available, which can fabricate very very large circuits there. So, a lot of market is coming into picture. If you talk about academic institutes more than 99 percent of the academic institutes been in case of digital design FPGA because students can fabricate they make error, they can erase it even if it is industry level you are allowed to make mistakes, you can check you can prototype and verify. So, therefore, FPGA is also very important apart from (Refer Time: 64:29). So, a few lectures will be dedicated that what things will change your what optimizations are require if you are thinking about a FPGA based design.

(Refer Slide Time: 64:37)

Test Optimization

- **Speedup** the Sensitize propagate justify approach to deterministic TPs by
 - Determining redundant faults
 - Better paths for fault propagation and justification
- **Quality TPs**
 - Patterns covering multiple faults
 - Patterns covering multiple fault types
 - Selection and sequencing of faults for which TPs are to be determined
- **TP compression**
- **Test at abstract level**
 - RTL
 - High level fault modeling

Now, coming into test, what to we need in test? We require good quality test patterns that is multiple faults and which can a minimum number of test faults etcetera and also you should better you should able to speed up the sensitized propagate justify approach basically sensitized propagate justify approach means test patterns generation time test patterns generation time. So, to speed it up given circuit the cad tool will tell you, what are the test patterns for that.

So, I need to optimize the cad algorithms. So, that I given a circuit I get the test patterns very quickly. But again in terms of solution generated by the cad tool like for example, if the circuit is there we call it cut this is your cad flow, this is your test cad, which is the sensitized propagate justify you will give you lot of basically your test patterns. But given a circuit this has to be very fast that is one way optimization that we will see that how I can go for doing it very fast.

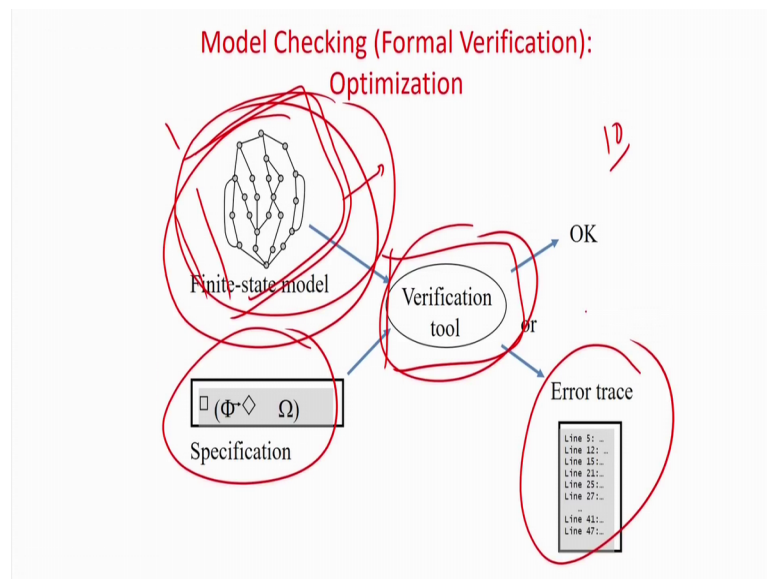
Secondly, this test patterns should be generated should be quality that there should be some patterns should cover multiple faults from different type of formally should can covered sacred for your discuss their other faults will be discuss in the lecture like bridging fault, delay fault. So, can there be a minimal setup patterns discussed do it for do all the testing in one rule because as I told you one fault one pattern I am not bothered to take that pattern, but one fault 10, one pattern 10 faults I am very happy with it, but if I apply one pattern 10 faults will be detected.

So, I go for I need better quality test patterns finally, there other way of optimization because I want to compress test patterns and put it. So, that I do not you have to use too much power of the testers then again as I told you if we do everything at the gate level things are not going to scale up, if I when you are going for high level implementation like NOCs and SOCs.

So, will talk everything at the higher level of abstraction, that we not even got one gate level we will talk about register transfer level, high level decision diagrams and try to solve the testing problem at that level there is will not even give the input in terms of gates, rather will give very high level inputs of a circuit we will try to generate the test pattern from that levels.

So, it will be very fast this algorithm will be very fast, but of course, there will be slight compromise in the test pattern quality. Because everything is very much well verified at the gate level in case of testing, but if we do not get the circuit in the terms of gate level as input, you have to go at the higher level then you will gate test patterns, but of course, there will be slight compromise in the test pattern quality, but it time equal to generate the test pattern will be very very high.

(Refer Slide Time: 66:56)



So, you have to the optimization is that how can we do it will also be covered in this in this course there will be one dedicated module one test.

Finally as I want a optim verification model checking of formal verification. There is some specification in terms of map your design intent has to be there, there has to be of formal model of this system if it will be of finite state machine book here automata, this is the extremely as I again thing are the other parts are mainly related to VLSI design circuit, gates all those things, but whenever you are talking about model checking this more of a theoretical computer science perspective.

So, whatever circuit are you have you have design has to be first representation of formal module and like a finite state machine in a very broad language and there will be some specification that is want you wanted to design that also to be determine in terms of some CTLR LTL formula for mathematical formula. Then there is the verification tool, which will tell you whether your model matches this specification or if not they will give you a error trace or a counter example. So, that you can go and debug it look very nice, but in fact, it is an extremely complex problem because finite state machine system with 20 variables even you cannot model this machine it will block.

Then verification tool there lot of complications like you have to raise the (Refer Time: 68:10) point etcetera. So, even if a model size is very very large you cannot do it. So, here will in this course we will try the whole course on model checking will be seen how we can actually optimize this model representation. We require to expressively do it is

state space like final stage machine which is exponential complexity or can you use some type of intelligence methods like binary decision diagrams, symbolic model thing lower.

So, here they optimization will be mainly will try to see, but how we can actually bring down is model because if you have a 10 bit counter. So, you will required 10 flip flops state basis 2 to the power 10 so, but 10 bit counter as you know is maybe only very very very minute part of a circuit design. So, very very large system means main giving part will be the model itself, but when we go you should have a formal model because it is a formal model checking mathematically. So, you should have a graph and all those things in nature.

So, how can we represent this in a more optimized fashion so that we can get the answer yes or no in a very quick fashion? So, that optimization in representation and model checking will be fully dealing so, that basically we can reach to reasonably complex circuits. But again as I am telling you compare to design and test optimization is still as a (Refer Time: 69:18) stage compared to the complexity. It can handle because you can really design NOCs SOCs you can really test NOCs and SOCs, but if you take the entire NOC together no model checking tool even with optimization will be able to handle it. So, basically we will partition it and then try to very some. So, lot of work has to be done in case of optimization.

Basically is the course plane. So, first will go for introduction and basically which is lecture one we have already done, then there will be high level synthesis will be told to you then there will be RTL optimizations that it how can you give a good quality high level input. So, that you get good quality circuits, then logic synthesis and physical synthesis. So, logic synthesis means from this part actually you can say your cad tool comes up this is how to write good quality codes.

(Refer Slide Time: 69:58)

Course Plan

Module 1: Introduction and High-level Synthesis
Lecture 1: Introduction to Digital VLSI Design Flow
Lecture 2: High-level Synthesis (HLS) flow with an example
Lecture 3: Automation of High-level Synthesis Steps
Lecture 4: Impact of Coding Style on HLS Results
Lecture 5: Impact of Compiler Optimizations on HLS Results

Module 2: RTL Optimizations
Lecture 6: RTL Optimizations for Timing
Lecture 7: Retiming
Lecture 8: RTL Optimizations for Area
Lecture 9: RTL Optimizations for Power

Module 3: Logic Synthesis and Physical Synthesis
Lecture 10: Introduction to Logic Synthesis
Lecture 11: Overview of FPGA Technology Mapping
Lecture 12: Introduction to Physical Synthesis

Handwritten notes: "good circuit quality" (written vertically on the right side of the slide) and "CAD" (written below the horizontal line).

Or good quality circuit should be given to inputs as to the cad as that is good quality circuits optimized or good circuit circuit quality, input you have to give that is high level description of circuit from here cad starts. So, how can you take a circuit and generate gate level in very fast manner.

(Refer Slide Time: 70:13)

Course Plan

Module 4: VLSI Testing
Module 4: VLSI Testing
Lecture 1 and 2: Introduction to Digital VLSI Testing, Automatic Test Pattern Generation (ATPG), Design for Testability
Lecture 3 and 4: Optimization Techniques for ATPG
Lecture 5: Optimization Techniques for Design for Testability
Lecture 6: High-level fault modeling and RTL level Testing

Module 5: Verification
Lecture 1: LTL/CTL based Verification
Lecture 2 and 3: Verification of Large Scale Systems
Lecture 4, 5: Symbolic Model Checking
Lecture 6: Bounded Model Checking

Handwritten notes: "CAD" (written vertically on the right side of the slide) and "CAD" (circled in red below the horizontal line).

Then there will be entire module on testing. So, given a circuit how can we generate optimal test patterns and in a very fast way this is also more or less by fully by cad tools this is verification. That is also by mainly by cad tools just the model you have to write

(Refer Time: 70:27) you have to design and this specification to be test tool design after that this is the model checker will tell whether this specification is met or not. But again at every point we will be looking a optimization in case of verification as I told you optimization the how we can have a minimal representation of the system and do verification. Test means you have to generate basically faster solutions faster test patterns solution has to be generated for circuit and also good quality test pattern has to be done and again as I have already told you.

So, basically this brings us to the introductory part of this course and I think I have I am I am able to make clear to you that what we will be expecting out of the course, what is the optimization and what has been automated and what you will be optimizing in the automated cad tool and what will be automatic in the design part. So, that just I am trying to give you gave you I have try to give you in the lecture in this introductory lecture, and I wish well welcome you all to this course from next lecture onwards we will be looking into more depth of the concept we have discussed about assignments and basically your notes.

So, every week we will be finding out some assignments you have to solve them in the part portal and basically also some notes and ppt's will be uploaded to you so that you can also get the reading matter from them. And as will find later this course is slightly towards oriented towards research because many topics are very emerging and we cannot cover everything in this 20 hours course. So, basically we will give you some references and papers which will not be exactly required for your exam purpose, but will be required for you to carry research in this advance cad flow. So, all this things will be given in proper time.

Thank you.