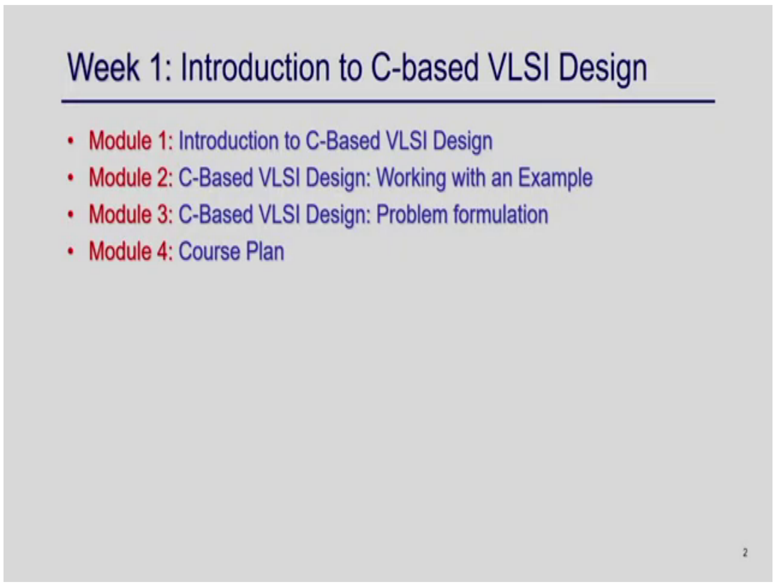
C-Based VLSI Design

Dr. Chandan Karfa
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 01
Introduction to C - based VLSI Design
Lecture - 04
Course overview

Welcome, everyone. In today's class, I am going to just discuss the overall course plan for this course C-based VLSI design.

(Refer Slide Time: 01:03)



Week 1: Introduction to C-based VLSI Design

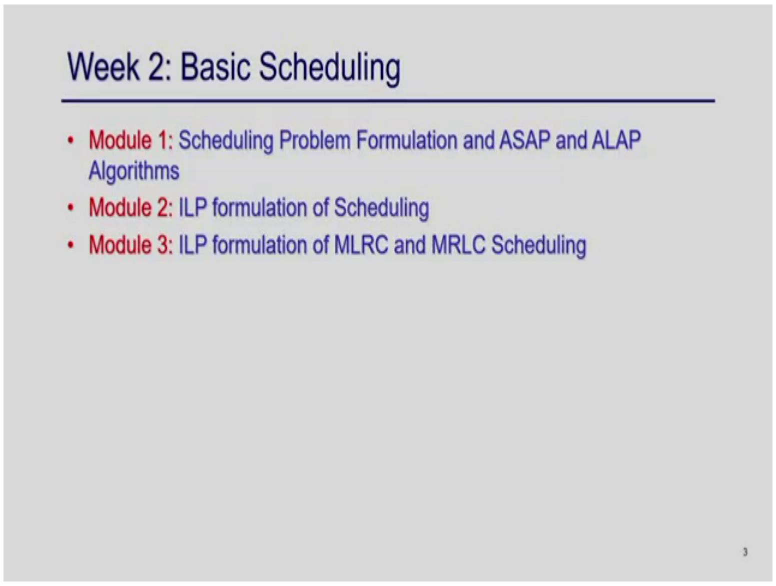
- Module 1: Introduction to C-Based VLSI Design
- Module 2: C-Based VLSI Design: Working with an Example
- Module 3: C-Based VLSI Design: Problem formulation
- Module 4: Course Plan

2

So, as you know this is a 12-week course. So, there will be 12 weeks of lectures, every week we will have 3 to 4 lectures and the total view duration of the video will be from 2.5 to 3 hours roughly. And, I am going to talk about today every week what are the topics I am going to cover in this course, ok.

So, this is week 1 and I have already covered what is a C-based VLSI design and also taken an example and explained how this C-based VLSI design happens and how C code is converted into RTL, and also, I have taken one class where I just talked about what are the internal problems inside the C-based VLSI design flow right. So, basically what are the steps to be followed to convert a C code into RTL, ok. So, this is this week's plan.

(Refer to Slide Time: 01:53)



Week 2: Basic Scheduling

- Module 1: Scheduling Problem Formulation and ASAP and ALAP Algorithms
- Module 2: ILP formulation of Scheduling
- Module 3: ILP formulation of MLRC and MRLC Scheduling

3

In the next week, I will go into the more theoretical part right. So, as you know the C-based VLSI design or high-level synthesis has three major steps; one is scheduling, the next is allocation binding and then data path and controller generation, right. So, the schedule as I mentioned earlier also is the most crucial step because based on the schedule the data path gets decided and from then the data path and the controller gets decided, right.

So, the way the more efficient you schedule your design will be your hardware will be more efficient, right. So, I am going to take 3 weeks to discuss the scheduling because this scheduling is something that is a very interesting problem and there are many algorithms available ok. So, I more distribute this scheduling into 3 weeks – In the first week I am going to talk about the basics of scheduling; here there will be three modules.

In the first module, I am going to talk about the scheduling problem formulation, what is the problem right, in general, and then I will give you the two basic algorithms – what are called ASAP and ALP algorithm, and then I am going to formulate how I can formulate this scheduling problem as an integer linear programming, ok.

And, then I will give an ILP formulation of this scheduling of two types of scheduling in the third module which is called MLRC, and MRLC is Minimize Latency under Resource Constant and Minimize Resource under Latency Constant, right. So, there are two types of schedules that are possible. So, I am going to give this ILP formulation this is kind of all background thing in scheduling, right.

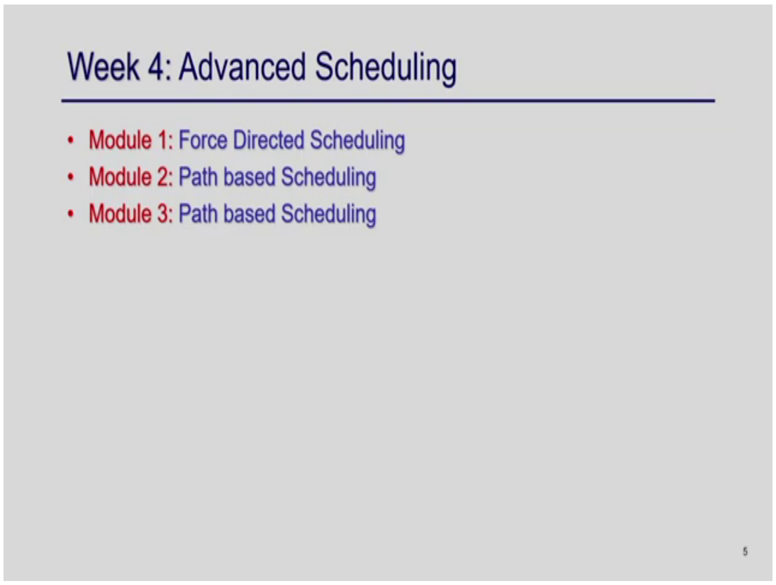
Next week I will move on to the heuristic-based scheduling I will discuss once I discuss this scheduling problem that this scheduling is in general NP-complete problem ok, it is a computationally-intensive problem. So, this ILP formulation the basic formulation sometimes takes exponential time to give you results.

So, for practical design probably we will go for a heuristic-based solution, where we basically will go for some heuristic which may not give you optimal results always, but keep decent results or good results in most the cases, right. So, in week 3 I am going to talk about I will have four modules.

In the first module, I am going to talk about what is multiprocessors scheduling and then I am going to give an algorithm for heuristic algorithm for multiprocessor scheduling in module 2 then I am going to talk about this least based scheduling which is heuristic-based scheduling for both the scheduling problem which is MRLC and MLRC, ok.

And, for this least based scheduling, this heuristic algorithm is the from where this particular this least based scheduling is evolved. So, that is why covered this part and then we will move into the least based scheduling, ok. This is the week for the third week.

(Refer Slide Time: 04:36)



Week 4: Advanced Scheduling

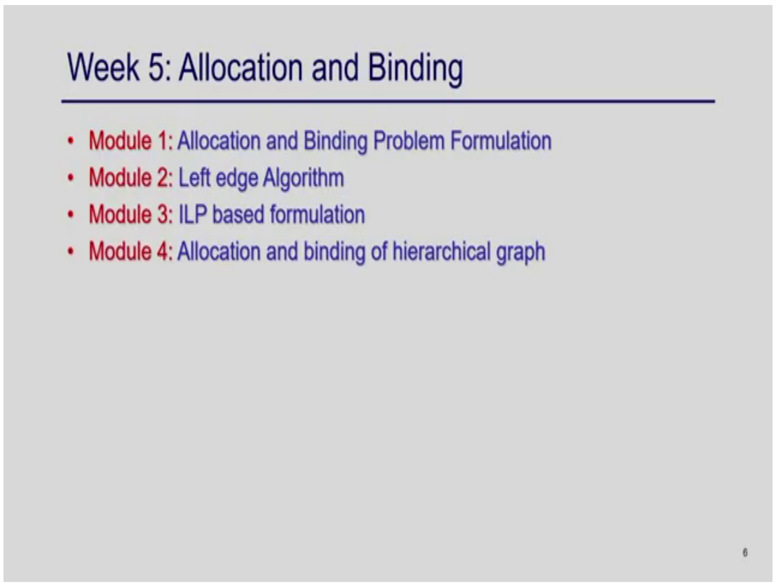
- Module 1: Force Directed Scheduling
- Module 2: Path based Scheduling
- Module 3: Path based Scheduling

5

And, in the 4th week, I am going to the more advanced level of scheduling specifically, I am going to cover two important scheduling algorithms – one is called force-directed scheduling, and one is called path-based scheduling, ok. Again, both kinds of heuristics and then the fourth force-directed scheduling will be discussed in module 1.

And, this path-based scheduling is a completely new concept. So, which is something that needs kind of two modules to discuss; In the first module, I am going to discuss the basic concepts and the basic intuitions. And, then in module 3, I am going to go a little bit on the algorithmic side of this path-based scheduling. So, with these three weeks 2, 3, and 4, I am going to cover the basics scheduling part, ok.

(Refer Slide Time: 05:19)



Week 5: Allocation and Binding

- Module 1: Allocation and Binding Problem Formulation
- Module 2: Left edge Algorithm
- Module 3: ILP based formulation
- Module 4: Allocation and binding of hierarchical graph

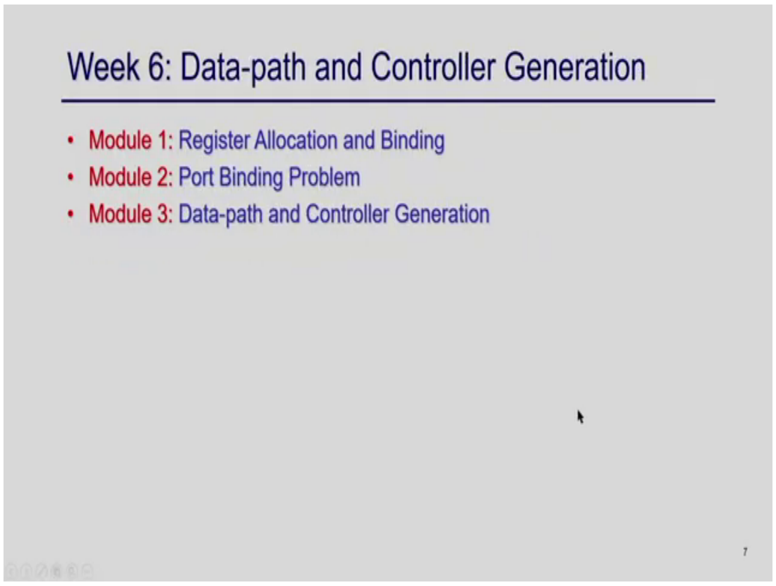
6

In 5th week I will move on to the allocation and binding problem. So, obviously, in module 1 I am going to discuss what is the allocation binding problem. Then I will go to discuss a very important algorithm called the left edge algorithm for this allocation and binding problem.

In module 3, again I am going to solve these allocation and binding problems in the ILP, integer linear programming-based solutions and then I am going to discuss in module 4 this allocation and binding of the hierarchical graph, ok.

So, basically in general our program has nested loops, nested if-else. So, that will create a hierarchy in the structure of input behavior right. So, how to do an efficient allocation and binding for this hierarchical graph that I am going to discuss in module 4.

(Refer Slide Time: 06:07)



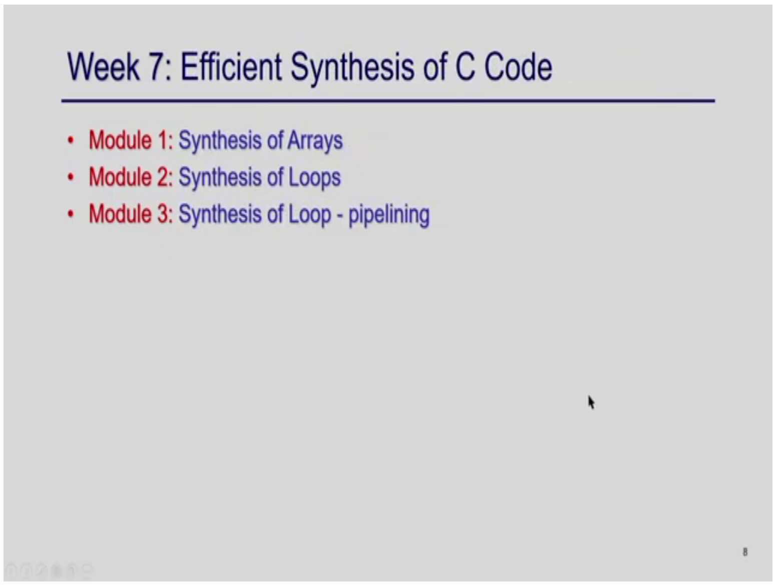
Week 6: Data-path and Controller Generation

- Module 1: Register Allocation and Binding
- Module 2: Port Binding Problem
- Module 3: Data-path and Controller Generation

Module 5, I initially covered so, in module 1 in this register allocation and binding problem which is basically allocation binding problem and then in next two modules module 2 and module 3, I will discuss the data path and controller generations, ok. So, which is is little bit straightforward method I mean approach if you have already decided about scheduling if you have decided about your allocation and binding of registers and functional units.

Then this generation the data path and controller are kind of straightforward, right. So, that is why it will not take much time to explain things, but this is the plan for week 6. So, in the first 6 weeks I will cover the theoretical aspect of the background algorithms, and the background theory of this high-level synthesis process, ok. From the 7th week onwards, I am going to move on to other interesting topics.

(Refer Slide Time: 07:01)



Week 7: Efficient Synthesis of C Code

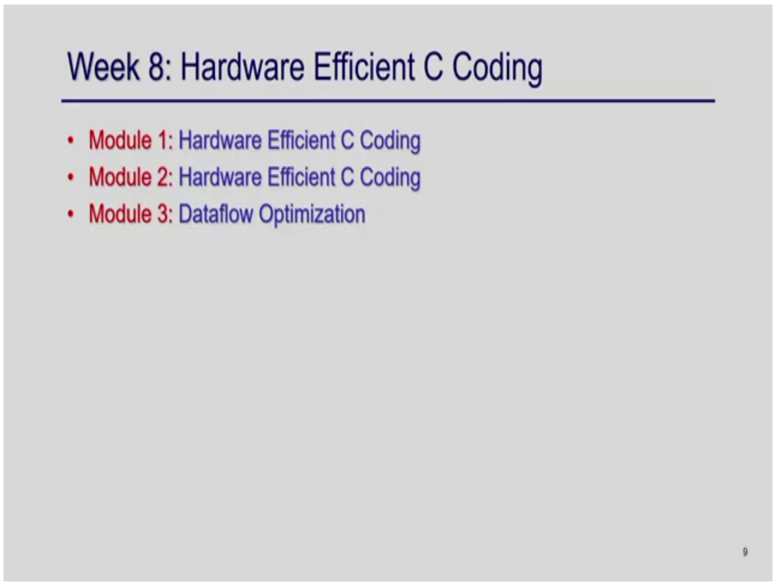
- Module 1: Synthesis of Arrays
- Module 2: Synthesis of Loops
- Module 3: Synthesis of Loop - pipelining

So, in the 7th week, I am going to cover the efficient synthesis of C code ok. So, this week I am going to discuss in a program we have arrays, we have loops, and how we can efficiently synthesize them into hardware, ok. There will be three modules. In module 1 I am going to talk about the efficient synthesis of the arrays because the arrays are I am going to discuss you will understand is a bottleneck for having efficient hardware or performance of any of the hardware, ok.

So, this managing array is very important and this module 1 will talk about that and then another important part of a C program loops and you can have a nested loop as well. So, and then again, the synthesis of the loop is also the most important thing in the whole high-level synthesis process because unless you synthesize the loops efficiently you will not get efficient hardware.

So, I am going to take two modules to discuss that. So, in module 1 I am going to talk about say unrolling and then partial unrolling of the loops and then another way that the loops get synthesized specifically in module 3 I am going to talk about the loop pipelining and how you can pipeline a loop in hardware.

(Refer Slide Time: 08:18)



Week 8: Hardware Efficient C Coding

- Module 1: Hardware Efficient C Coding
- Module 2: Hardware Efficient C Coding
- Module 3: Dataflow Optimization

9

Week 8 – I am going to cover another very important topic hardware efficiency coding. So, I mean as I already mentioned that if you just write any arbitrary code, it will not get synthesized into efficient hardware that you have to accept, right. You need to understand the basics of this philosophy. That is why I took 6 weeks to discuss the background theory.

Unless you know the theory will not, you will not able to appreciate how this whole process happens and you probably need to tweak your C code probably you have to modify C code to make it hardware efficient. So, two versions of the C code may be doing the same thing.

But if you just synthesize it into hardware, it may give very bad hardware in the sense of performance and just some small tuning of the program or manual modification of the program might give very efficient hardware. So, basically in these two modules, I am going to talk about how you can modify your input C code to make it hardware efficient ok this is a very interesting week to be discussed.

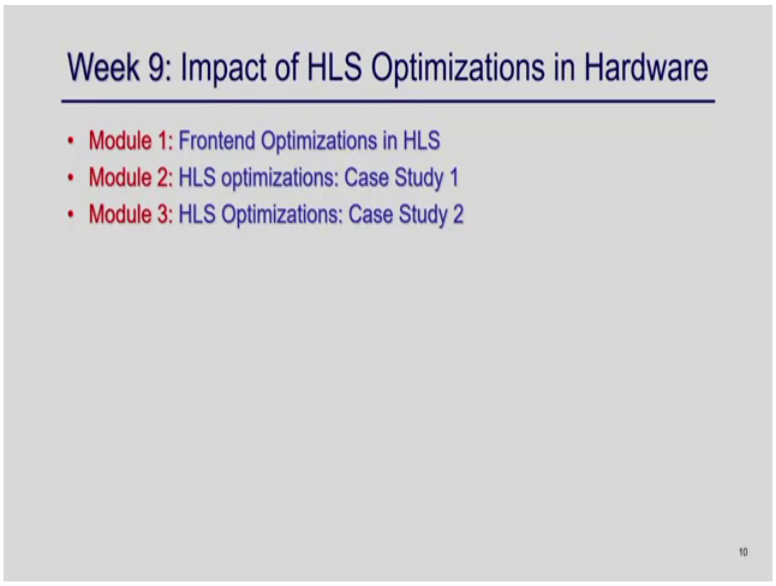
There is another very important and the I think that the most important optimization is called data flow optimizations in the context of high-level synthesis. Because if you think about this high-level synthesis here what happens? You have massive hardware parallel hardware, right.

So, if you have three adders if you have to say four multipliers all are available in every clock. So, the way you need to map is you should try to able to utilize all those models in every clock then only your performance gets improved, right. And, if you look into this normal C port it is a sequential execution, right, so, line by line executions. So, this data flow optimization is trying to parallelize this execution of the program, right.

So, this particular optimization tries to break your code into saying producer-consumer relations, right. It tries to create multiple modules where these modules will communicate through some FIFO or say some communication buffer and then this module can run in parallel.

So, this is something very important optimizations, the recent context most many of the work going on how to make your C code when optimize or manually modify this your C code so that this can be mapped to parallel hardware, right so, and specifically in the context of hardware acceleration. So, this is a very important optimization. I am going to talk about this data flow optimization in module 3 of week 8, ok.

(Refer Slide Time: 10:49)



Week 9: Impact of HLS Optimizations in Hardware

- Module 1: Frontend Optimizations in HLS
- Module 2: HLS optimizations: Case Study 1
- Module 3: HLS Optimizations: Case Study 2

10

The next week is another interesting week where I am going to see the impact of high levels in this optimization in hardware. So, by these 8 weeks, we will understand there are many optimizations possible in the whole hardware flood I have already talked about these loops, arrays, and even synthesizing them and most of the high-level synthesis tools actually can optimize them.

There may be some problems or because of some dependency in the code, some optimization may not be applied. So, if you avoid that, if I just modify this then this optimization can tool can automatically apply, right. So, that is the thing very that is what I discuss in week 7 how to make your code hardware efficient.

So, once you make this there are a lot of optimizations that are already inbuilt into the high-level synthesis tool itself, right. So, I am going to take this week to discuss them. So, there are three modules this week. In the first module, I am going to talk about the fronted optimization high level in high-level synthesis is basically in the C level you can apply many optimizations, right.

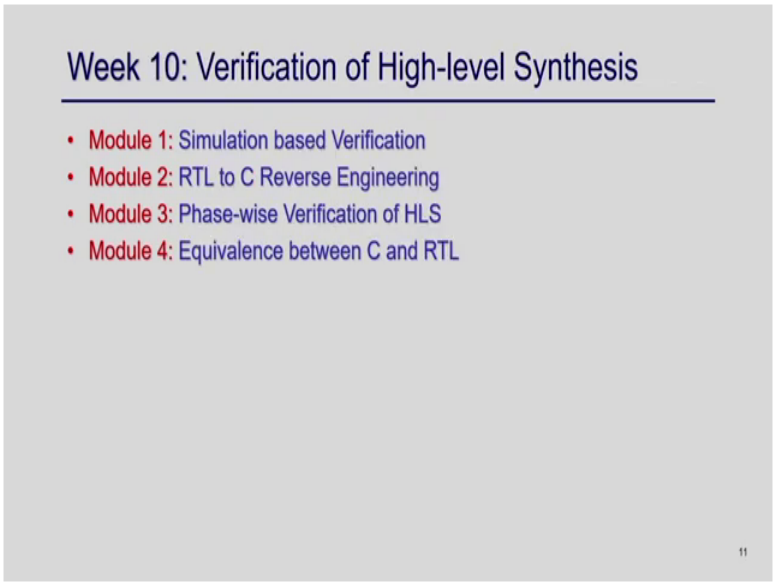
So, this week I am going to talk about what is the impact of these C-level optimizations on hardware, ok. So, this is module 1 then I create two interesting modules. The module 2 I do it case study. So, I am going to take an example and I say if I then I if I apply say some array optimizations what is the impact, right. I will take the example then if I say apply different kinds of loop optimizations what is the impact right.

So, this is how I am going to give various case studies to see the impact of this high-level synthesis. So, you will appreciate more about this optimization, and the importance of this high-level synthesis optimization. Then module 3 is a very important module that I am going to talk about I will take an example and specifically I am going to take merge sort, ok.

And then I will see how we can gradually modify this high-level merge sort step by step to make a C code or a merge sort which is the for which efficient hardware can be generated through high-level synthesis and you have to accept the fact that I mean as I mentioned earlier also that not an arbitrary code can be mapped to efficient hardware.

So, you have to do this step. So, this particular module probably gives you the idea or intuition given a C code how can I gradually step by step modify it to make it harder efficient. So, this will be a very interesting module, this module 3.

(Refer Slide Time: 13:18)



Week 10: Verification of High-level Synthesis

- Module 1: Simulation based Verification
- Module 2: RTL to C Reverse Engineering
- Module 3: Phase-wise Verification of HLS
- Module 4: Equivalence between C and RTL

11

In week 10, I am going to talk about the verification of high-level synthesis. So, for any transformation or synthesis, you need to do, there may be there is always a possibility of bugs right their code is written by many software engineers across the globe. So, there may be some problems between the communications and the way both two-component are implemented there will be some cases that are not handled because this component is implemented by somebody else.

Or there may be some corner cases which is not implemented at all, right. So, that is why or may be implemented wrongly ok. So, verifying or ensuring the correctness of the high-level synthesis process that you are saying that yes, the RTL that I have generated is functionally equivalent to my C code is important, ok.

So, this week I am going to take four modules and in module one I will talk about the simulation-based verification of high-level synthesis which is the most common way nowadays even today the high-level synthesis process is verified. Then three modules are on a little bit advanced topic.

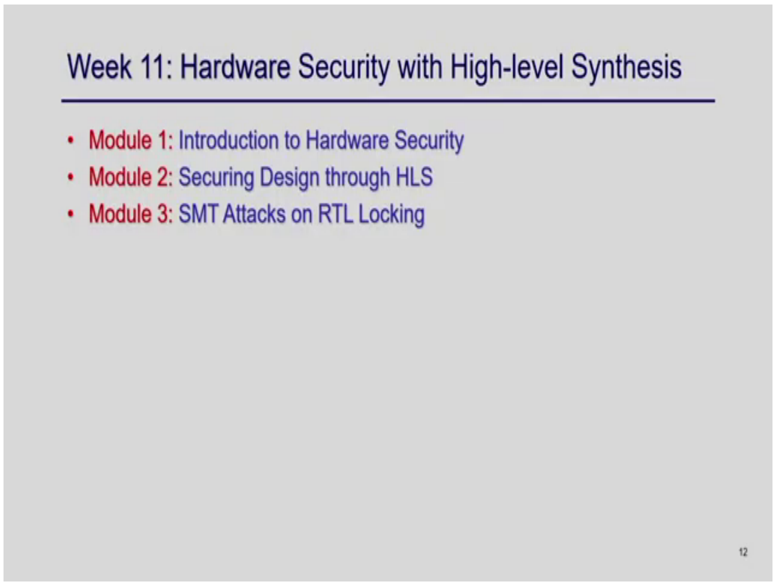
So, in one module I am going to talk about the work that our group has done on this RTL to see reverse engineering. So, I talked about how high-level synthesis is converting a C code to RTL. So, now, it is a reverse process that given RTL can you give me a C. So, remember that this C is not the input C I want to get a C which represents my hardware ok.

So, it is not the input C from where I get this order. So, then what is the advantage of this generating C? Because this represents my hardware, not the input C if I simulate this it simulates the hardware ok or the RTL. So, using this I can do a good simulation-based verification, right. So, this is why I going to put this module 2 in this week.

And, then I go into the two modules; module 3 and module 4 where I am going to talk about the formal verification of high-level synthesis. It is a very evolving area a lot of research is happening in this domain and there are two ways people are targeting now. One is phase-wise verification you try to verify the scheduling, you try to verify the allocation binding, and you try to verify the data path control generation phase.

So, that is one approach and another approach is that can I do an equivalence checking between C and RTL, right. So, I am going to cover both the things and some recent work in this particular area, ok.

(Refer Slide Time: 15:42)



Week 11: Hardware Security with High-level Synthesis

- Module 1: Introduction to Hardware Security
- Module 2: Securing Design through HLS
- Module 3: SMT Attacks on RTL Locking

12

So, this will be another interesting week and in week 11 I am going to cover the hardware securities through high-level synthesis ok. So, because of this third-party fabrication in the recent EDA industry in this VLSI industry, most of the time your IP has gone to some third party. So, and then as a result there may be a problem of IP piracy over the building, an IP threat, harder Trojans, and alright.

So, that is the harder security I am talking about in this context and then I am going to see can high-level synthesis give me help? Right. So, can the RTL that is going to be generated from the C code is something I can make I say that this is secure from this IP piracy or overbuilding, right. So, these are the things it is very recent research I am going to cover in three modules.

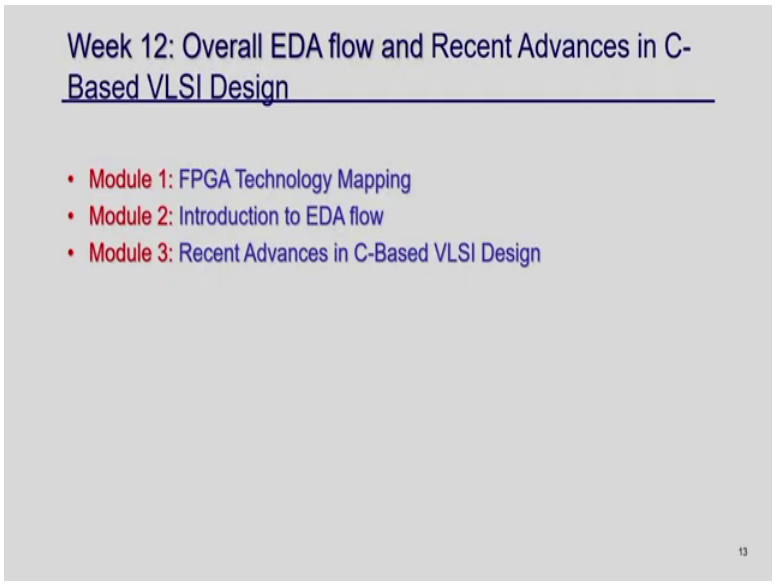
In module 1, I am going to talk about the issues of this hardware security what is IP piracy what is this building and why it has happened. And, then I am going to talk about what are the kind of countermeasures or the measures has been taken to overcome this security threat, right. So, that is the basic background of this hardware security.

And, in the next module, I am going to talk about how can high-level synthesis do the same thing whatever the measure is already taken to make your higher. So, this your IP secure can the same thing can be done through high-level synthesis, right. So, in module 2 I am going to talk about securing design through high-level synthesis, ok.

And, in so, whenever you do something secure obviously, the attacker will come into the picture they try to break your security, right and that is what is going to happen in this domain also and there are some recent works. So, am trying to try to see that whatever the high-level synthesis locked circuit I can break it, right.

So, there is a decent work by our group which is called say SMT attack and we show that whatever the securing design through high-level synthesis is breakable. So, there is no end to it there is a lot of research happening here, but I am going to talk about these three things which are kind of the state of the art till today on this hardware security through high-level synthesis, ok.

(Refer Slide Time: 17:50)



Week 12: Overall EDA flow and Recent Advances in C-Based VLSI Design

- Module 1: FPGA Technology Mapping
- Module 2: Introduction to EDA flow
- Module 3: Recent Advances in C-Based VLSI Design

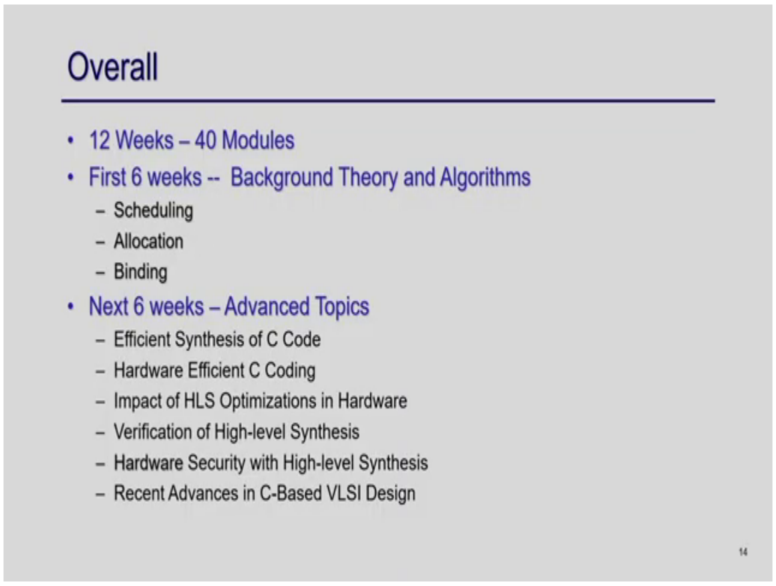
13

And, in the last week, week 12 I am going to give the overall EDA flow. So, for 11 weeks I invested to learn high-level synthesis right. So, as I mean as you are already aware that this whole process is the high-level synthesis, then logic synthesis and physical synthesis right then finally, your chips get fabricated.

Just I am going to give a very overview on this if you try to map the RTL that is generated by high-level synthesis to the FPGA board, what are the steps to be followed. So, FPGA technology mapping or if you try to generate ASIC chip from that from the RTL so, you will go through this logic synthesis and physical synthesis. So, I am going to talk about this logic synthesis and physical synthesis probably in some modules.

So, module 2 might will have two sub-modules, one is logic synthesis and physical synthesis and then I am going to talk about this FPGA mapping also. And, then I am going to conclude this course with some recent advances in C-based VLSI design – what is the current research trend in high-level synthesis, and what are the things to be addressed in the future to make your high-level synthesis tool more useful. So, I am going to talk about that and conclude this course.

(Refer to Slide Time: 19:05)



Overall

- 12 Weeks – 40 Modules
- First 6 weeks -- Background Theory and Algorithms
  - Scheduling
  - Allocation
  - Binding
- Next 6 weeks – Advanced Topics
  - Efficient Synthesis of C Code
  - Hardware Efficient C Coding
  - Impact of HLS Optimizations in Hardware
  - Verification of High-level Synthesis
  - Hardware Security with High-level Synthesis
  - Recent Advances in C-Based VLSI Design

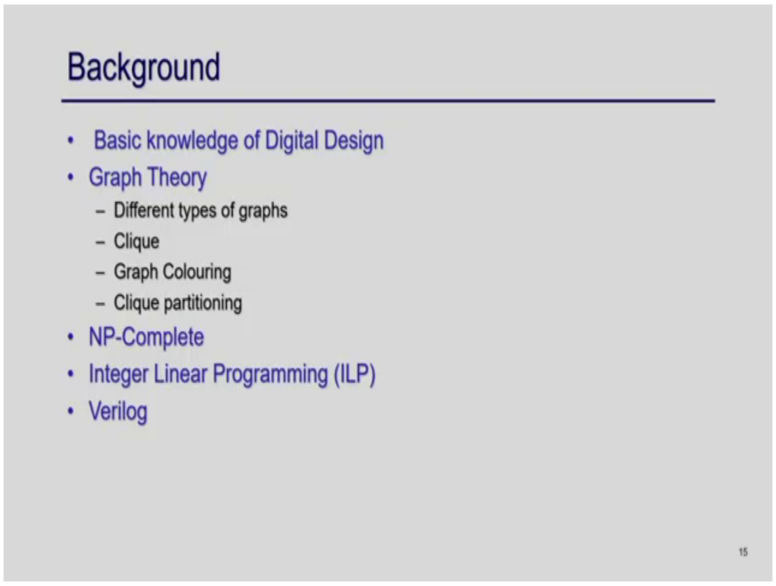
14

So, overall, we have 12 weeks, 40 modules and the way I just break this course is two-part – one is the first 6 weeks is the background theory and algorithm so, where I am going to talk about this is the theory of the algorithm involved in this high-level synthesis process – specifically in scheduling, allocation, binding and then also data path control generation, right. So, these are the first weeks of the basic theory.

And, the next 6 weeks as I discussed are the advanced topics specifically like the efficient synthesis of C code, hardware efficient C coding, HLS optimizations, verification of the high level of synthesis, security issues in high-level synthesis, and this recent advance, right. So, if you notice here, I purposefully skipped all the background knowledge needed here, right.

So, because trying to make this course more complete in the sense that if I spend more time discussing the background then I am going to lose important time which it could have been invested to cover the recent works that are happening in high-level synthesis topics. So, that is what I did in this course.

(Refer to Slide Time: 20:27)



Background

- Basic knowledge of Digital Design
- Graph Theory
  - Different types of graphs
  - Clique
  - Graph Colouring
  - Clique partitioning
- NP-Complete
- Integer Linear Programming (ILP)
- Verilog

15

So, I will expect that you will have this background and that you should have basic knowledge of digital design. So, you should be able to understand what is a clock, and what it says in the digital design of an adder, subtractor, and multiplexers these are the secondary digital design things right, and also this says what is a sequential element and how they work and all, right. So, how when they get updated, what is edge-triggered, what is level triggered, those things you should understand a little bit, ok.

And, also since the first 6 weeks, I am going to cover about this the background theory, and most of the time it involves certain most of the time I convert that problem into some graph coloring graph problems, right. So, you should understand different types of graphs – what is a directed graph, what is an undirected graph, what is a clique in a graph, or say what is a perfect graph, and what is not a perfect graph. So, those things you should understand, and in the overview level that is fine, no need to go into detail.

And, also you should understand what is graph coloring problem, what is clique partitioning problem, why this problem is difficult, right, and how this problem is getting solved. So, those are the things you should understand at a high level I am not going to cover this. I will expect that you will already do some Googling on that and you should have some basic knowledge.

Also, what is an NP-complete problem that should be understood? What is this ILP or Integer Linear Programming and how does it help to solve the optimization problem and how any problem can be formulated as an ILP. So, that knowledge also you should able to do in as background knowledge, right. So, I am not going to cover, but to make this more comprehensive in the sense have more subject on high-level synthesis rather the background knowledge and also at the end I will expect that you know what the basics in terms of Verilog ok.

So, again I do not want you to be an expert in Verilog, but at least if I give a Verilog code should be able to understand ok this is going to have work like this right what is the behavior of that. So, at least you should be able to understand that, ok. So, I am not going to cover this part, I will expect that you will at least be familiar with this terminology before going to the next week, right. So, that will help you ok.

So, with this, today's I mean I conclude this course plan session. I hope this course will be very useful for you and it will be a good learning experience for you.

Thank you.