VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Lecture 01

Introduction to VLSI Design

Welcome to all of you to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about various steps of the VLSI Physical Design. So, the main objective of the course is to discuss the various steps and flows of the VLSI design and VLSI physical design flow. Then we will also discuss about different very basics of the graph theory and algorithms that are applicable for VLSI physical design and we will explore the basic principle of static timing analysis and how we can apply those principle to the VLSI physical design. The various steps are involved in the VLSI physical design such as partitioning, chip planning, placement, clock tree synthesis and routing. All these steps will be discussed in detail. Then we will discuss about the open source tools available in the open domain for the VLSI physical design which will be useful such that everybody can use that tools for their to learn what is the basic principle involved in VLSI physical design. Then finally, we will discuss some of the advanced topic like statistical static timing analysis. After going through the objective of this course, then comes to the mind why we study this VLSI physical design.

So, what is the significance of VLSI physical design? So, if you can look into this slide, we have several processor from different companies and what is the transistor count inside that design. If you can see Pentium 4 from Intel has 42 billion transistors. In the other hand, in 2023 we have processor from AMD which is having 146 billion transistors. So, we have huge number of increase of transistor inside the chip. So how to design and how to handle those transistors such that it can meet the DRC, LVS and all the rules at the same time meet the specification of the processor.

It is a challenging task. So, if you can look into some of the layouts of the chip, it is so complex. For example, in 1974 Intel 8080 processor is the, this is the layout of that one. You see how complex it is. Then if you can see 2000 Intel Pentium 4 layout, it has huge number of transistor inside it. So if you integrate it using custom manner or manual method, it is impossible to design the chip and launch the product in the market. If you can go into the Intel Core i7 processor, it is the internal die photo. It has 4 cores, this 4 cores, 1, 2, 3, 4 and all of these are designed using physical design flow, physical synthesis flow. So this motivates to learn this course why it is important to learn this VLSI physical design and how it is useful for modern chip designs. So the content of this course includes the basically the VLSI design style, what are the different design styles are applicable for various types of designs and design abstraction, what are the different abstractions levels we need to go through in order to design a bigger design and synthesis which is basically converting a different levels of abstraction to the lower level details, we will discuss in this lecture.

So, the first we will start with different design style. The design styles is divided into two main category, one is the full custom and the semi-custom. These two are the main driving force behind our chip design. So the choice of the layout style depends upon various factors such as the type of the chip, cost, how much cost is involved in that product or the manufacturing of the chip and how much time available for the designer to make the design to the market. So, time to market is one of the driving force behind which design style we should adopt.

For example, we need to launch a product in a one year time frame or two years time frame, then the semi-custom design style is the most preferred one. Full custom design will require more detailed design and it will be more optimized however it takes more time to design. We will discuss these things in more detail in the subsequent slides. So the design styles is mainly divided into full custom, where we do the layout of the chip mostly manually. Most of the layouts are done manually and in case of semi-custom some prefabricated designs are available from third party companies and some cases some bigger companies they design their own prefabricated designs. Those are useful and those are used in the design flow.

So again the semi-custom is divided into two categories, one is cell based and one is array based. In case of cell based we have a particular types of cells are there which is used in the design process. In case of array based the same primitive is repeated multiple times inside the chip. We will see some of the examples. So the cell based is again divided into standard cells and compiled cells, then macro cells.

Then the array based is divided into pre-diffused that is called the gate arrays and pre-wired, which is FPGAs. Let us discuss the full custom design style. So the main idea behind a full custom design style is that we need to design majority of the major part of the design manually and in this case we have less constraint in placing the design in the chip area. We do not have any constraint in the size of the design. One thing the layout should be, layouts are drawn manually.

Second, no constraint in size, the no constraint in size of the design then third no placement constraint. So no placement constraint means that you can place it any part of the design based on the best routing resources available to that design. And if you can see here, we have a A to D converter which is placed close to the IO which is there, and its shape is determined based on how it can be optimized properly in the layout. So this one is done complete custom manner that is called full custom design style.

Similarly in this one if you can see majority of the blocks like data path, ROM, RAM, PLA, random logic all are of different shapes. There is no particular shapes involved here, but they are placed based on how we can optimize the design in the chip itself. And if you can see the there is some channels are available for routing or connecting those blocks in case of full custom design style. So if you go by the full custom design style you can design a compact chip and it has highly optimized electrical properties.

Your speed may be higher. However, nothing comes free means it is time consuming and it is highly laborious work and there may be chance of error while designing this. So in one end you will get a very high speed design, in other end it will take very long time to design. You may lose time to market in which cases it is useful. Basically if you can see in case of a processor let us say most of the part of the processor is designed using a semi custom manner using VLSI physical design flow, but some of the critical paths or data paths, which detects the speed of the processor need can be designed in a full custom manner to optimize the speed of the processor and improve the speed of the processor such that we can meet the specification of the processor.

So not the complete processor is designed full custom manner, only the critical paths or the data paths which is detecting the speed of the processor that can be designed in a full custom manner to improve the speed. Similarly in case of FPGAs the switch matrix, the switch matrix then look up tables and CLVs those are designed in a full custom manner. After it is designed it will be repeated to design the FPGA. Similarly analog blocks like ADC, PLL all these will be designed in a full custom manner.

ADC, PLL all these are designed in a full custom manner. If a design where which is cannot be done using semi custom design style then you can adopt full custom design style to optimize the design. So, what is standard cell? Standard cell is basically a type of cell it is the basic building block for any digital design. So if you can see one of the basic structure of this one it has fixed height all the cells of a standard cell have a same height or multiple of the same height either same height or multiple of that height. Then second thing is that the width is not also random it is multiple of height of let us say inverter, height of NAND gate both are same, but width will be different. Width is the multiple of unit tile what is defined in your technology file. So height is same for both the cases here it is H here it is also H and the widths are multiple of the unit tile defined in the technology library and height is also not comes randomly it has some rules it depends upon how many metal tracks it is allowed. If it is a 9 tracks library so you have 9 tracks in the your standard cells. So, it is pre-designed by some company, and you can utilize that for your designs. They are rigorously tested and characterized library they provide library files, timing files, power characterization files those files are rigorously tested and provided to the designer RTL designer or PD designer the physical design engineers.

So then they have basically let us say feed through cells. So let us look into this these are the if you can see all cell A, cell B, cell C, cell D all heights are same, but widths are different, and heights are determined by the number of metal and tracks which is used in the standard cell and width of the cell will be multiple of the unit tile. Then if you can see here this is a very older design where you have basically less number of metal tracks are there so that so they introduce some something called channels between the placement of the standard cells to these are the standard cells actually this is 1, 2 these are standard cells and these are the channels actually. So, between the channels these are the channels basically between the channels they do the routing and since number of layers is metal lines is smaller they have something called feed through cells in earlier designs so that the metal connections can come from one channel to the other channel. But nowadays these empty resources are filled with the standard cells also because we use high level metals to connect them.

So this is the background behind your standard cell design and if you can see here you have a VDD line one top line is your VDD and you have a ground line is your this is your ground. Let us say this is your ground and let us say this is your VDD. So, this is standard cell design style, so you have a RTL then you target this standard cell library to do the synthesis process. So it is basically less time consuming simpler than full custom design style and it occupies more chip area compared to obviously more chip area because it is not a full custom design but as the technology scales into lower technology nodes it the full custom way of designing and standard cell way of designing will area wise there will be not lot of penalty is there in terms of area or performance. So, area penalty is very very small but only advantage of the standard cell based design style is that it is very fast full custom will take more time to design and if we have more time to design means we will lose the time to market.

So standard cell-based design style is more popular compared to full custom design style in case of digital design flow. So now you have a cell-based design so there are the second category which is the macro cell. Macro cells are little bigger and in case of standard cell there is some constraint in placing the cells those will be placed row wise all the row wise it will be placed. But in case of macro cell those constraint is a little bit relaxed based on the size of the cell we need to find out an area to place that one. For example, basically you have a memory compiler which generates a memory block we need to find out area inside the chip to put it there such that the distance from other memory access units all the distance would be optimized.

So, you have a larger logic component with reusable functionality, and it can be simple, or it can be highly complex, it can be used as embedded processor also and memory blocks. Let us say here you see you have a RAM random access memory and you have something called programmable logic array all these can be generated using some kind of means they can be provided by third party to be used directly and which is there is no constraint in placing the cells so it is very useful and that IP will if you do by cost full custom method it will take more time but these cells you can let us say if you have a memory compiler you can specify the size of your memory it will generate the memory in very very less time so that you can use that memory in your design. So, whatever I told it can be placed anywhere in the layout to optimize the routing distance and electrical property basically we need to optimize everything we need to optimize the power performance everything so we need to optimize the interconnect distance so you need to place the macro cells in such a place so that our delay will be minimum. So, some cases we can use some pre-existing macro to optimize our design to complete our design so then we have a gate array so there are two types of array based design style one is called gate array one of them is gate array so which basically we have identical cells gate arrays means some identical cells are there which is placed in array fashion like a matrix like format. So, throughout the chip those cells are repeated and whenever we are doing a RTL or something we need to convert that to the gate array to connect the cells actually.

Circuit is divided into identical blocks each of the each equivalent to the cell on the gate array so this basically your design is mapped and placed on the prefabricated cells during partitioning. Blocks are mapped or placed onto the prefabricated cell during partitioning basically what we need to do is that we need to map we need to have a tool which will map our RTL or the main design to these gate array designs automatically such that our actual design is implemented in the gate array. So, it is of two category one is uncommitted gate array one is committed gate array. What is the difference between uncommitted and committed? In the uncommitted gate array, we need the help of a foundry facility because those cells are placed but there is no routing was done. So on the top of it we need to do the routing to complete our design so these are prefabricated chip where the routing layers are added on the top or to in the route fabrication facility so it is expensive because we have to send that we need to buy the gate array and we need to define the routing layers send that design to the fab then the fab will do the connections. So, it is expensive, but it obviously will get more performance out of it but more cheaper one is basically committed gate array.

Gate array designed with full custom routing layer now routing layers are there already in case of the committed gate array but you need to do the programming to connect the routings based on our requirement. Here if you can see these are called uncommitted gate array means that all the cells are repeated like an array fashion this is one of the basic building block which is repeated multiple times. So your actual design will be mapped to this one and we need to create the routing layer above it such that our actual design will be created which requires the help of a foundry. This is an example of committed gate array. What is happening here is that you have pins like A, B, C here so here your A this is the pin A is connected to this pad B is connected to this part and the C is connected to this pad and if you can see here how many NAND gates are there 1, 2, 1, 2, 3, 4 and 5. So here 5 cell will be used each one of them is programmed for a NAND operation and those connections are made. So here if you can see 5 cells are used this is one this is one this is one this is one this is one so here if you can see 5 cells are used finally you will get the output.

So, these connections are done using some programming. Connecting through some kind of programming. So this one is cost effective less cost because you do not need to send it to the foundry and it is easier to produce than full cost term standard shell design it is but it is less flexible you cannot optimize like full cost term or a standard shell based design style and routing in the gate array is simpler it is basically very very simpler compared to any design style and it is also basically the focus on routing rather than we need to implement the design such a way that it can be implemented in less time it takes less time to implement the whole design. So now we have a field programmable gate array the field programmable gate array it is basically more popular these days. What is the main idea behind it is programmable interconnect, programmable IO, and it has programmable lookup tables. It has three things programmable IO, a programmable interconnect, programmable logic blocks. All the things are programmable in case of FPGA, and it is highly popular because it has the reconfigurability facility programmability is there and offers large scale integration lots of very new things are now incorporated inside the FPGA and it is very popular architecture nowadays. So, logic blocks in FPGA are like memory blocks basically if you program the memory blocks then it will program your lookup tables. So it consists of logic blocks separated by routing channels these LUTs and programmable switches are programmed in using that tools actually you have logic element this is logic element this is called switch box the switch box are basically interconnects which is programmable in the runtime and logic elements consist of CLB configurable logic block logic elements are consist of configurable logic block then slices then LUTs.

So, you can implement both combinational and sequential design using these CLBs, and it has first basically carry chain logic to implement fast adders. So here if you have a gate label design which is mapped to the your lookup table using the tools actually. This P1 is mapped to this the table then P2 is mapped to this one P3 is mapped to this one then the P4 is mapped to this one and it will finally implement your design. So this is one of the flow plan of FPGA where your design is implemented interconnected so you have three things whatever I told you one is programmable IO programmable interconnect and programmable logic block or configurable logic block all the things are programmable in case of FPGA. Now we will discuss about basically the design abstraction level there are different levels of abstraction. So, whenever you talk about a system or a processor, we are looking for a chip as a processor, but if you go inside the detail of what is inside the processor, we have modules.

Modules are basically different blocks inside the processor or a system, how they are connected with each other and each of the individual blocks how it is implemented. If you go inside one of the module how the gates are there different logic gates are there that is called gate level. Then if you go inside a gate, you have transistors NMOS and PMOS transistors. So those are how they are connected with each other to create the logic gates. So that is your circuits.

If I look into each of the transistor like NMOS or PMOS how they are implemented in the silicon this is called the device. So, these are the different levels of abstraction whenever you we go from a system to the device. In the top you look for a processor, in the below we look for the very detail how the transistor is implemented inside the chip. So, here this is very top-level view, and this is very bottom level view. So, we need to understand each one of them we need to optimize at each level of abstraction to optimize the complete system.

So, we need to optimize in the device level, we need to optimize in the circuit level, we need to optimize in the gate level, we need to optimize in the module level and finally, we need to optimize how the modules are connected such that my more important targets like performance, power and area. These are the three targets one is the performance or speed, then area, then the power. Minimum these three things should be satisfied. So, in case of abstraction the basically what is happening inside abstraction. It is showing some of the features without disclosing the more detailed what is inside it.

So, let us say if I look into a processor, I do not look into what type of transistor is implemented. Whenever I look into an adder, I do not know what kind of adder is implemented in silicon. So, the processor of different architecture whether it is a RISC architecture or a CISC architecture. So, it has different abstraction level like architectural level, one is logic level and geometrical level.

There are three levels of abstractions are there. So, each one of them can be represented using architectural model into more detailed models like geometrical models. So, we are refining the models from architectural model to the geometrical model such that it can be suitable for manufacturing in by particular technology node and finally, we will get a chip back which can be used in different applications. It can be used in a mobile phone; it can be used in a general-purpose processor like a laptop.

So, here if you can see you have architectural level. So, here if you see here some level example is given here PC equal to PC plus 1 fetch PC decode instruction. What does it mean that we are incrementing the program counter and after the program counter is incremented, you are fetching the instruction from the memory, then you are decoding the instruction using the decoder. So, it is written in a architectural level and we can represent that one using flow diagram or RTL model also or you can write it in some different language also like high level language. So, the processor basically described by an HDL, HDL stands for hardware description language. So, we can reprint that one is also in terms of HDL.

So, here if you can see we have logic level. What is that we are doing here? We are dealing with the logic gates. We have NAND, NOR, XOR, AOI, plenty of different gates are there. Those gates are used to implement that RTL. So, you have a gate level schematic whatever it is given here that is called logic level. So, here your design is represented in terms of schematic level.

So, here we are going for geometrical level. Geometrical level means that we are doing the layouts. So, whatever the logic what is represented in terms of schematic, now we are converting that schematic to layouts because the layout is the thing which is basically understood by the foundry that the foundry can understand. They cannot understand your logic gates, they cannot understand your RTL, they only understand the layouts. So, this is the two-dimensional geometrical picture which will translate into the mask layout to be printed in the silicon.

So, there are different types of views are there, behavioral, structural, and physical. So, basically in case of behavioral circuit is explained regardless of its implementation. How it will be implemented does not matter. I need an adder. So, it is let us say for example, your PC equal to PC plus 1.

The program counter will be incremented by 1. How it is implemented does not matter for me. I just need the increment of the program counter, but in case of structural view it will be more detail where it will say about the interconnection of the components, how these components are connected with each other. In case of physical view, we are dealing with the transistor, the physical object, how they are represented in terms of transistors. This is a Y chart which is basically proposed by Gajski and Kuhn, and if you can see here, we have a behavioral domain, we have a structural domain, we have a geometrical or physical domain. Three of them are here. So, we describe our behavior in terms of an algorithm, then we translate that to using some form of a processor which is in a structural domain, then we can go to the chip floor plan, then we will go to the finite state machine, then we go to the register and ALU arithmetic logic unit, then we will go internal and finally, we implement everything in terms of transistor placement, then we go to the mask and implement our design. So, this is most popular y chart which is used

to design from system level specification to the layout which will send to the foundry to design your system completely and implement and send it back the chip to you. So, now we will talk about synthesis. Synthesis is basically transformation from one of the view to the other view. So, how we are moving from one view to the other view that is called synthesis.
So, synthesis is basically movement or transformation from one view to the other view. There are three types of synthesis, one is called architectural level synthesis, one is called logic level synthesis, one is called geometrical level synthesis. So, architectural level synthesis, logic level synthesis and geometrical level synthesis. So, in case of architectural level synthesis, we basically have a bigger picture of our design that PC equal to PC plus 1, FETCH PC decode instruction, all these are written in very high level language. Then we will move to how the blocks are connected with each other, we need an adder, we need a multiplier, we need a memory, we need a controller, all these how they are connected with each other.

This is called architectural level synthesis. And in case of a logic level synthesis, we finally get a logic gate level implementation. Let us say it is a RTL, it is a data flow statements because we are using data flow operators like plus and basically this RTL can be converted to gate level netlist. So, that is called your logic level synthesis. In this case, we are moving from RTL to the gate level netlist.
So, this is RTL, this is gate level. In case of geometrical level synthesis, we have gate level to the layout level or geometrical level. So, we are moving from your gate level to the layout level that is called geometrical level synthesis or physical synthesis. So, basically in this lecture, we discussed about different design style and there are several design styles like full custom, semi custom and standard cell based design style and macro cell based design style. Then we discussed about array based design style. We also highlight the significance of different levels of abstraction using architectural level and logic level and geometrical level, and this we discuss about different types of synthesis flow. All these are very very important in case of system level design. Thank you very much.