VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh
Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 2
Basic Concepts of Integrated Circuit: II

Hello everybody, welcome to the course VLSI Design Flow: RTL to GDS. This is the second lecture. In this lecture, we will continue with the basic concepts of integrated circuit. In this lecture, we will cover the types of integrated circuits, design styles, economics, and figures of merit for an integrated circuit.

First, let us look into a very top-level understanding of what is a VLSI design flow. So, VLSI design flow is a methodology to design an IC such that it delivers the required functionality or behavior. So, it is a methodology of designing an IC, and the goal is to get the required behavior from the IC. Now, on what factors does a VLSI design flow depend on? What decides the VLSI design flow? The VLSI design flow depends on the type of integrated circuit. And what are various types of integrated circuits? We can divide or categorize integrated circuits based on the application scope or design styles.

Let us first look into the various types of integrated circuits based on the scope of application. So, there are two types of integrated circuits based on the scope. The first is application-specific integrated circuits or ASICs, and the second is general-purpose integrated circuits. Let us understand the differences between these two types of integrated circuits. In terms of functionality, an ASIC or application-specific integrated circuit is targeted to perform a specific task or a particular end system. It is designed for a particular application, which is why it is known as an application-specific integrated circuit, while general-purpose integrated circuits are designed to perform a wide range of applications.

Let us look at an example, and then it will become clear. So, an example of an application-specific integrated circuit is an integrated circuit for a digital camera or an integrated circuit for audio/video processing or security chips. Here, the application is very well-defined. For example, the chip digital camera is used just for the digital camera.

While, a few examples of general-purpose integrated circuits are microprocessors. So, we know there are many types of microprocessors, and we can use microprocessors for many different applications. Similarly, memory can also be used for many applications. Then, if using an FPGA, we can realize many different functions or functionalities. So,

microprocessors, memories, and FPGAs are examples of general-purpose integrated circuits.

Regarding programmability, the ASICs or application-specific integrated circuits are not that much programmable. Why? Because they are not targeted for those kinds of applications. Because they are not targeted for general use, they are targeted for specific purposes, and therefore, the programmability will cater to that particular purpose only. While general-purpose integrated circuits such as microprocessors are highly programmable, even FPGA can be programmed, and you can get different functionality out of it. So, general-purpose integrated circuits are usually software-programmable and perform a wide variety of different tasks.

And regarding the production volume, the volume of application-specific integrated circuits is less. Why? Because they are targeted for a particular application, and the requirement for those targeted applications may not be for a high-volume product. While general-purpose integrated circuits like in microprocessors, FPGAs are general in nature. There are wide applications for those chips, and therefore, they will be catering to a bigger market. Therefore, the volume of production for general-purpose integrated circuits will be typically more than for ASICs.

Now, let us look into the various types of design styles. So, based on the types of design styles, we can categorize integrated circuits into the full-custom design, standard-cell based design, gate-array based design, or FPGA-based design. FPGA is a short form for field programmable gate arrays. Now, we will look into these four types in more detail.

Now, what is a full-custom design? In the full-custom design, we design the circuit at the transistor level, i.e., at the lowest level, and we decide the layout of the transistors, the geometric aspect like W by L of the transistors, and how those transistors are connected, etc. So, the layout of the transistors and the interconnections are design-specific. We are designing at the transistor level in full-custom design, and as a result, since we are designing at a very low level, we need to make a lot of decisions manually. And therefore, a huge design effort will be required in full-custom design. There are only a few designs that are full-custom. A few examples of full-custom designs would be, say, analog circuit designs or mixed signal designs where the design produced by manually designing at the transistor level typically achieves better results or better performance and other measures for analog circuits compared to the automated flow. In those cases, the full-custom design is done. Or the case in which there is a high volume product, for example, a microprocessor, which is a very critical product, and in that microprocessor, there is some critical path, and in that path, we will be trying to design at transistor level because we want to improve the performance of that particular path to a much much much higher level. Therefore, the full-custom design may be undertaken for those particular paths in microprocessors or high-volume products. But in general, full-custom designs are less

undertaken than other types of design styles. And what is the merit of full-custom design? The merit is, of course, the optimization. We design at the transistor level; therefore, as a designer, we have a lot of flexibility for a full-custom design, and if we have that flexibility, we can utilize it to improve the quality of our design and, as such, the quality of full-custom design can be much better than other types of design styles.

The other design style is standard-cell based design. So, in standard-cell based design, what is done is that first, we design standard cells. Now, what are standard cells? Standard cells are simple cells such as AND gate, OR gate, flip flops, simple adders, etc. So these are gates, logic gates, or simple cells; these can be macros and complex cells such as full adders, multipliers, and memories also. But what we do is that we first design these small cells at a transistor level: AND gate, OR gate, and other standard cells and characterize them and put that information in a kind of library. Once we have designed those AND gates, OR gates, and those kinds of standard cells, we do not touch the transistor; we treat the complete standard cell as a unit and use that in our design.

So, there are two levels of designing done in standard-cell based design. First, the designing of the standard cell which is done at the transistor level, and we do it very optimally; we design each of the cells and then put all that information in a kind of library, and in the next step, what we do is that use those standard cells, which are in the library, in our design. So, for a designer who is doing a general design using standard cell design, in that case, the level of freedom is only which standard cell to use, how to connect them, and how to put them on the layout. The internal details of the standard cell are not allowed to be changed by the designer who is making the design. So, when we make these standard cells, we typically keep the height of the standard cell fixed, and as such, the properties of these standard cells are well defined in terms of geometry and other things, and therefore, it allows a high degree of automation. For example, we can put the standard cells in a row kind of structure.

Why we can put it in a row because the height of all the standard cells are the same; therefore, we can put all of them in one row and then make the connections. So, these are standard cells, and in a standard cell design, we can also use macros, which are bigger cells, for example, multipliers, memories, etc., and we can also have IO pads. Now, rows of standard cells are used in standard-cell based design with interconnections between them. The custom blocks can also be embedded in a standard-cell based design. The type, location, and interconnections of standard cells are design-specific. So, since we are allowed to change the location of the standard cell (and the standard cell internally contains the transistor), it means that when we make a design, all the layers, the masks related to the device layers, and the interconnect layers will be design-specific. So, the designer only knows where the devices are on the layout. Why? Because where to place the standard cell exactly, that freedom is still with the designer. We are not changing the

location of a transistor within the standard cell, but one standard cell can be placed over the layout anywhere as per the wish of the designer.

The third type of design style is gate-array based design. So, in gate-array based design, what happens is that the transistors are predefined on an IC in the form of a gate array. So, the transistors are predefined. So, the designer has no control over where the transistor will be on the layout as that is predefined. In a gate-array based design, the smallest element that is repeated to form a gate-array that is known as a base cell or primitive cell. So, this base cell or primitive cells are pre-placed on the layout. The designer can change the interconnections between these transistors. Since the devices are already at a fixed location, the device layer will not change, but the designer can change the metal layers or the connections between the primitive cells. Therefore, the top layers of the gate-array based design are design-specific. So, for gate-array based design, realizing some functionality, for example, memory, can be very difficult. Why? Because we have only a primitive cell and, we have to design everything in terms of primitive cells. However, the vendor of gate-array based design may have products in which these memories, micro-controllers, etc., are pre-embedded in the layout of the gate-array based design, and those can be reused for obtaining the functionality.

The fourth design style is FPGA-based design. Now, in an FPGA-based design, what happens is that the hardware is fixed. The designer does not have any control over the hardware, meaning the structure that is there is already there. What the designer can do is they can program the connections between the hardware. A designer can obtain the desired functionality by programming, and programming makes changes in the interconnections between the elements of the circuits. So, what are the elements of the circuit inside an FPGA board? So, FPGA consists of an array of logic blocks. For example, there are logic blocks, and there will be IO blocks, which are at the periphery through which the signal can enter and leave, and there can be routing channels, which are basically used for making connections between different logic blocks. So, logic blocks can be programmed to perform different functions such as AND, OR, adder, etc. So, the functionality of each of the logic blocks can be changed as per requirement and also their interconnections. However, understand that the hardware is fixed. Once you get hardware from an FPGA vendor, hardware cannot be changed.

However, the internal connections within the hardware can be programmed, and the vendor typically gives some tools to program that hardware through high-level languages. So, FPGA boards may also have embedded microprocessors and analog components and blocks for performing special functions such as DSP block, and the most popular FPGAs are Xilinx, AMD FPGA boards, Altera, or Intel boards, which are used typically in the semiconductor industry. Now, to summarize, what are the differences in the design styles? We should understand the design styles carefully because our design flow depends on the design style used. So, in this course, we will typically look into the standard-cell based design. This is the area in which this course is working, though the concepts that we will be using will be applicable at least partially for FPGA-based designs, and a lot of them will also be valid for custom-based designs.

Let us summarize the differences between these four design styles. So, the full-custom design style is a design style in which the customization is done at the level of transistors and layout. In the standard-cell based design, customization is done at the level of interconnections between standard cells and also the instances that will be used inside our design that is customized by the designer. In gate-array based design, only the top-level inter connections are design-specific; the transistor is predefined. In the FPGA board, the complete hardware is predefined. We can only change the connections by programming, and the functionality of individual logic blocks can be changed.

Now, in terms of design effort, the full-custom design requires the highest effort. Standard-cell based design will also require a high effort, but less than full-custom design, and gate-array based design and FPGA-based design require lower design effort. Then, there are the mask layers. In full-custom design, the mask layer for all the layers, the device layers, and the interconnect layers will be custom or design-specific. In standard-cell based design also, the mask will be design-specific for the device layer and the interconnections. For gate-array based design, only the top-level mask will be design-specific. The device layer is predefined, there is no control of the designer over them. And in FPGA-based design, the hardware is given by the vendor, and the designer does not have control over the hardware.

Then, in terms of power, performance, and area, i.e., the quality of the design that we get, the full-custom design is the best because we are designing at the transistor level, and we have the full flexibility to make our own design, and therefore, full-custom design can achieve the best power, performance, and area measures. The standard-cell based design is also very good in terms of power, performance, and area. And comparatively, the gate-array based design and FPGA-based design are inferior in terms of quality. However, the strong point about FPGA-based design is that the design effort is less, and if there are any bugs, it can be easily fixed in an FPGA-based design by reprogramming. Now, there are different design styles. When we start a project, how do we choose which design style we need to use in our project: i.e., FPGA-based, full-custom, or standard-cell based design? One of the major factors in making this decision is the economics of the integrated circuit.

So, what we mean by the economics of integrated circuits let us understand that. If we consider the cost of an integrated circuit, there are various components, and we can divide it into two types: the first one is fixed cost, and the other is variable cost. So, fixed costs are those cost components that do not depend on the volume of the product, meaning that if we produce, say, ten items or 1 million items, the cost will remain the same. Those costs are known as fixed costs. For example, the cost of designing a circuit. As the designing will be done once, the design effort is a fixed cost; either we produce, say, 100 chips or we make 1 million chips, the design effort will be the same, and that is why it is a fixed cost.

And then, there are costs related to software tools that we use in designing again; those are one-time costs. The hardware that we require for designing, for example, the CPU, the computers that we use to make the designs, etc., are fixed costs that do not depend on

the volume of the product or the chip that we manufacture. And the cost of the mask. So, the mask is also made once. The same mask is used multiple times, and that is why that makes the copying of the integrated circuit much easier because the mask is prepared once at a very high level of accuracy. That mask is replicated on the integrated circuit. So, the cost of developing that mask is a fixed quantity, which does not depend on the volume. Whether we make 100 chips or 1 million chips, the mask will still be fabricated, and the cost of the mask will depend on the number of layers.

And then there are variable product costs, for example, the cost of the wafer. Now, when we take in a wafer and make a circuit on the silicon wafer, we need to use chemicals, for example, for photolithography and other stuff. Now, those chemicals will be dependent on the volume. If we are making lots of products, then more chemicals will be required, and therefore, that is a variable cost depending on the volume of manufacturing. The cost of the die depends on the area or the size of the die. If you have a small die, you can have many of those small chips on a given wafer. But if you have a larger area, then on the same wafer, you will have a less number of dies on that wafer. Therefore, the size of the die will impact the cost of the die, so that is a variable cost. Depending on the volume of dies or chips produced, you will incur more cost or less cost.

And then we have yield; yield means that out of, say, 100 chips that we manufacture, how many chips are defect-free? So, manufacturing is a complicated process, and when we manufacture a chip, all of them cannot be defect-free. Typically, if more than 95 percent of the chips are defect-free, we say the yield is good. So, yield defines that out of all the manufactured chips, how much percentage is good? So, the variable cost will depend on the yield. If the yield is high, the variable cost will come down.

So, these are the major divisions of cost for integrated circuits. So, we can write that as:

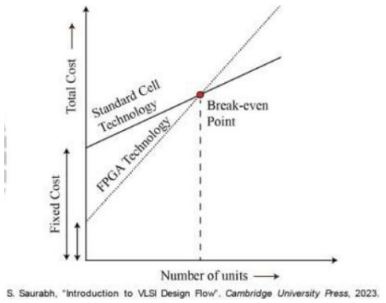
Total product cost = Fixed product cost + Variable product cost × Number of Product

So, as the number of products increases, the variable product cost per piece remains the same, but the number of products will increase. The fixed cost will remain the same. So, now let us compare two types of design styles.

Let us compare standard-cell based design and FPGA-based design and compare the fixed cost and variable cost. Now, for standard-cell based design, the fixed cost is high because it requires more design efforts and more costly tools, i.e., EDA tools for designing, it will require all the masks customized for a design, etc. So, the fixed cost will be higher for standard-cell based design, but it will be lower for FPGA-based design because to get a design, we just need to do the programming, and therefore, the design effort will be low, and therefore, the fixed cost will be low. And what about the variable cost? Now, for standard-cell based design, the variable cost per unit product will be lower. This is because the cost of the die will be lower as we can optimize our design to occupy less area, and the die area can be reduced. As such, you can get more dies on a given wafer, and therefore, the variable cost can be reduced. For FPGA-based design, we get hardware from the vendor, and we do not utilize the complete hardware for our

functionality. Therefore, the cost of the die will be higher for FPGA-based design because the area will be larger for FPGA-based design. Because of the larger area, the yield will also be lower (we will see how), and in effect, what happens is that the variable cost of FPGA-based design will be higher compared to the standard-cell based design.

Now, to give you a very crude estimate of what the cost will be like for the FPGA-based technology and standard-cell based technology. This is a very crude model in which we assume that the variable cost is a linear function of the number of units. So, for the standard cell technology, the fixed cost will be higher compared to FPGA-based technology, but it will be rising at a slower rate because the variable cost is lower for standard-cell technology. For FPGA technology, the fixed cost will be lower, but it will be rising at a steeper rate with the number of units compared to standard-cell based technology. Therefore, there will be a break-even point, and below this break-even, the FPGA technology will have less cost for a given volume, and above this break-even point (in the right region), the total cost for a given number of units that will be lower for standard-cell based technology. So, this analysis means that typically, for small volume, FPGA is better, and for large volume standard-cell based design would be better. The cost versus the number of units may not be a straight line as shown in the figure. This is just for the illustrative purpose.



S. Saurabh, "Introduction to VLSI Design Flow", *Cambridge University Press*, 2023.

Now, in addition to economics, another important factor that decides which design style to use is the figures of merit for an integrated circuit. Now, what do we mean by figures of merit? So, when we make an integrated circuit, the fundamental question is how to measure the goodness of that integrated circuit, meaning how good it is compared to some other design type of integrated circuit. So, to measure or assess the goodness of the integrated circuit, we use some figures of merit. So, three important factors that are considered important figures of merit are: power, performance, and area.Together, they are known as PPA.

So, what is power? Power is the sum of static and dynamic power consumed by an IC. So, an IC can be doing active computation. So, the power consumed by an IC when it is doing an active computation is known as dynamic power. Power consumed when the IC is not doing active computation is known as static power dissipation. These two numbers define the power numbers for an integrated circuit.

The other figure of merit is the performance, meaning how fast the chip works. So, typically, it is measured by the maximum clock frequency at which an IC can work. For

example, if the IC is working at a higher clock frequency, then we say it is operating faster. The third figure of merit is the area of the die for an IC because that determines the variable cost. So, typically, for a chip, we will say that for this particular integrated circuit, these are the PPA numbers. For example, we can quote that (1 W, 2.0 GHz, 1 mm^2). So, this represents the power, performance, and area numbers, respectively. So, typically, when we design an integrated circuit, we quote the PPA or figures of merit for that integrated circuit.

Now, in addition to PPA measures, there are other figures of merit. What are other figures of merit? Those are testability, reliability, and schedule. What do we mean by testability? Testability means how well we can test an integrated circuit. As we discussed, when we manufacture an integrated circuit or fabricate an integrated circuit, all are not defect-free. Some may have defects, and most of them will be defect-free. But we want only a good product or an integrated circuit without any defects to reach the customer. So, we want the chip, the integrated circuit that we manufacture, to be tested. Before shipping it to the customer, we want to test it and then only give it to the customer so that the customer does not get any faulty products. So, how well we can measure or test the product after manufacturing that is known as testability. Some integrated circuits may be very good testable, meaning that if it is defective, we can know there is a defect. Some integrated circuits may not be that easy to test. The testability of that product will be lower, and a bad integrated circuit for that particular badly manufactured integrated circuit, for that case, can reach the customer and can be troublesome for the reputation of the company.

The other figure of merit is reliability, meaning how the performance of a chip or the quality of a chip or an integrated circuit deviates from the rated behavior with time. We want that, with aging, the performance of our chip should not degrade below the acceptable limits for at least the rated age of the chip. So, that measure is the reliability. Another figure of merit, i.e., schedule, is very important. Schedule means how much time we have taken to make a design, or this is also related to time to market. Given a specification, how much time did we take to design that chip? Typically, the success or failure of a product highly depends on the schedule. If we can design a product or an integrated circuit very quickly, we can bring it to the market quickly and capture the market share. Therefore, the schedule or the time spent in design is very important and is one of the important figures of merit for an integrated circuit.

These figures of merit are also known as quality of results, especially when these results are obtained using EDA tools, and we get a measure of the performance, power, area, and other metrics, then we call these figures of merit as quality of results. Moreover, an important characteristic of these figures of merit or quality of results is that when we improve one of these characteristics, the other characteristics or other measures can come down. For example, in a circuit, we want to improve the speed of the circuit. We can improve the speed of the circuit by using a bigger transistor or a transistor that drives the load very quickly or use a wider transistor. But as a result of improving the performance or speed, we are increasing the area of our design. Therefore, when we improve the performance, the area also goes up, which is a bad thing, and typically, when we improve

the performance, the power can also go up. So, when we improve one metric, the other metrics can actually degrade, and therefore, designing is always a step or a process in which we have to choose a trade-off. So, we always have to trade-off some measures to improve others. So, when we design, we need to understand these aspects very carefully, that when we are improving one metric, what other metrics are becoming bad, and whether that is acceptable or not.

The problem becomes more complicated because the dependency of these figures of merit on the design parameters is very complicated, and finding a mathematical optimum for these figures of merit is rarely known or difficult to achieve. That's why the goal of a design flow is not to go for an optimum solution because we usually do not know the mathematical optimum. The goal of the design flow is to find one of the feasible solutions with acceptable figures of merit. We design such that we get a feasible solution and the figures of merit are acceptable as per our requirement. So, this is the design goal, and in the following lecture, we will be looking into how we can achieve these design goals in a design flow or what is the role of design flow in achieving these figures of merit in some better or optimal way.

So, this brings us to the end of this lecture. These are some of the important references that I have used in this lecture. Thank you very much.