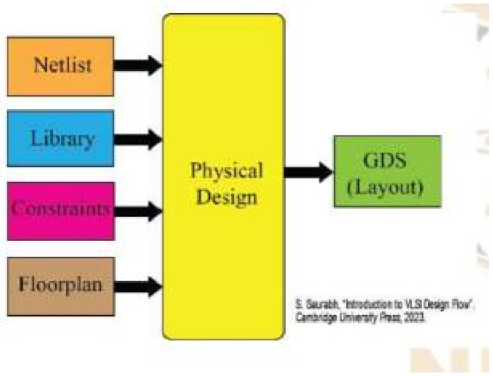
VLSI Design Flow: RTL to GDS

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

Lecture 7
Overview of VLSI Design Flow: IV

Hello everybody, welcome to the course VLSI Design Flow: RTL to GDS. This is lecture number 6. In this lecture, we will be continuing with the overview of VLSI design flow. In the previous lecture, we looked into RTL to GDS flow, and we said that we can divide the RTL to GDS flow into two major parts. The first part is the logic synthesis part, and the second part is the physical design part. In the previous lecture, we looked into the logic synthesis part, and in today's lecture, we will be looking into the physical design part.

Now, let us first understand what is a physical design. So, physical design is the process by which a design in the form of a netlist is converted to an equivalent design in the form of a layout, and this layout typically is represented in terms of GDS or the geometrical patterns that we want to have on the mask that will be further used in fabricating that IC. So, let us look into the framework or the inputs and outputs for a physical design. Let us see the major inputs for physical design. The first one is the netlist.


Netlist
Library
Constraints
Floorplan
Physical Design
GDS (Layout)
S. Saurabh, "Introduction to VLSI Design Flow", Cambridge University Press, 2023.

The netlist is basically the design that we have obtained after logic synthesis, the same netlist we give as input to the physical design tool. The cells that are in this netlist are standard cells or cells picked from the technology libraries, and therefore, to comprehend a netlist, we also need to give as input the technology library that we have used during logic synthesis, which is typically in the liberty format. Besides the technology library that was used during logic synthesis, during physical design, we also need some information regarding the physical aspect of the standard cells. For example, what is the bounding box for a cell? Where are the pins located for the cells? Why do we need this information? Because we want to place these standard cells on the layout, and we need to know their area, their aspect ratio, and the physical details. Also, we want to make connections between standard cells using interconnect layers. To make those kinds of design changes, the physical design tools need to know at least a little information about what a standard cell looks like in terms of physical dimensions and other things.

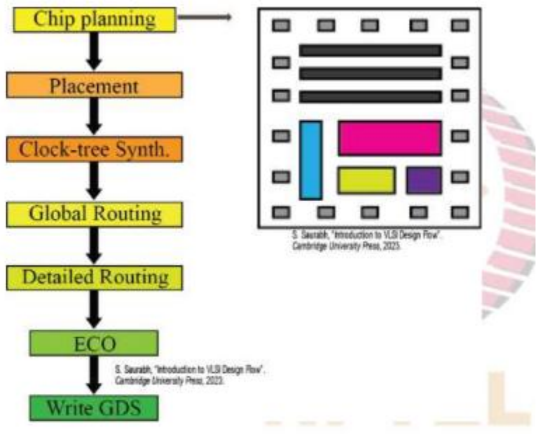
So, for that, we need to give the physical information of the standard cells in some abstract form, typically in a library exchange format or LEF files. So, this LEF file contains information like the bounding box for the standard cells, the pin locations, and also some information like sheet resistance and other information about the technology. So, using the information of the technology library and the physical library, the physical design tools can compute delays and other things, and also, they can understand design rules that need to be followed while making the layout and come up with the final solution or the layout in the physical design process.

So, besides the netlist and library, we also need to give the constraints. So, constraints define our design goals in terms of, say, expected timing behavior or the maximum operable frequency that we want for our design, the environment like how the signals are coming to our design, and what is the expected behavior for the signals generated by design, those are given in the constraints file which are typically in the SDC format. So, the constraint file that we used in the logic synthesis step we can use a same constraint file in the physical design, but with some modifications related to the wires and routing, etc., which do not exist in the logic synthesis step but are important in the physical design step. So, a modified version of the SDC file that we used in logic synthesis can be used in the physical design step. Besides netlist, library, constraints the information of the physical design or the intent of the physical design meaning what is the size of the layout that we are trying to make because finally, the layout that we will be making based on that the die will be fabricated and that die will be packaged and put in the chip. So, we should know what should be the die size or the maximum area that the layout can take and what should be the shape or aspect ratio of the die, etc.

So, that information is basically related to the floorplan or the top-level description of the layout or how the layout looks like, the size, the shape of the die, and probably, we also need to give a predefined location for some entities. For example, we want the input pins and output pins or input pads to be located at a few fixed locations in the layout, or we want some blocks to be at a free predefined location. We know that those locations are good for a physical design. So, this information can be given in a floorplan and to the tool, and this floorplan can be given in terms of a proprietary format that the physical design tools that we are using understand. In the end, what the tool will do is that the physical design, taking these inputs it, will produce a layout in which all the entities of our design are placed at some location on the layout, and their connectivity is the same as it was in the given netlist. So, the connectivity should be the same as in the netlist, and the location of the entities, for example, standard cells or bigger entities like memories, macros, etc., are decided by the physical design tools.

So, the important information that is added by the physical design tools is first the placement of all the entities, and the second information that is added by the physical design tool is the connections between these entities, and those connections are defined in terms of wires running at different layers of metals. So, the step that defines the connections on the layout is known as routing. This routing should basically honor the connectivity that was originally given in the netlist. So, the major task of physical design tools is placement and routing, and sometimes, this physical design step is also referred to as place and route because the major steps are placement and routing. But understand that there are many other things that are done by physical design tools in addition to placement and routing, and those are also relevant.

So, now let us look into the major tasks that are performed by physical design tools. So these tasks are shown in this figure. The first step is the chip planning, the next step is placement, then clock tree synthesis, global routing, and detailed routing, and then this ECO is engineering change order, and finally, we write the GDS. So, these are the major steps that are done in physical design. So, we will be looking into these steps in subsequent slides and in much more detail later in this course.


Chip planning
Placement
Clock-tree Synth.
Global Routing
Detailed Routing
ECO
Write GDS
S. Saurabh, "Introduction to VLSI Design Flow", Cambridge University Press, 2023.
S. Saurabh, "Introduction to VLSI Design Flow", Cambridge University Press, 2023.

The top-level understanding of this flow is that with each step, more details are added to the design. So, details go on adding to the design, and what are the details that we add to our design in each of the physical design tasks that we have mentioned here? So, once we do the chip planning, we do a kind of floorplanning. Floorplanning means that we have decided the locations of bigger entities, and we also add PDN (power delivery network). This information is added by the chip planning step, including floorplanning the locations of bigger entities, etc., during chip planning, and the power delivery network is made during chip planning.

Then, in the placement step, what is done is the standard cells that are in our design those locations of those standard cells are defined or determined by the placement. Then, once the placement is done, the clock tree is made. The clock tree is basically delivering the clock signals to each of the sequential circuit elements or the circuit elements like flip flops, which take the clock as an input. So, the responsibility of clock tree synthesis is to make the structure in our circuit such that the clock signal is basically delivered to all the entities in the design that need the clock signal to operate. Then, this global routing adds global routes to the design. Now, what are the global routes we will see? A detailed routing basically makes the detailed wiring for each of the nets in our design.

Finally, the ECO step makes some incremental fixes or changes to our design, and at the end of ECO, once everything is clean, we get a layout that can be represented in terms of GDS, and this GDS can be basically sent for making masks and then fabrication. So, the top-level view is that as the design flow progresses from chip planning to, say, placement, clock tree synthesis, and to other next levels, more details are added to the design, and the abstraction level decreases. Now, let us look into each of these steps in some more detail.

So, the chip planning step is basically the first step in physical design, and this is a major step because chip planning is the planning stage for our chip or the layout. At this stage, we do not have much information about the layout, and therefore, we have a lot of flexibility, meaning that we can put the macros at different locations, try them out at different locations, and choose an optimal location.

So, at the chip planning stage, we have lots of flexibility, and therefore, the chip planning stage has a lot of impact on the final QoR that gets out of a design. So, if we make wrong decisions during the chip planning stage, all our following steps or tasks in the physical design flow will be impacted. So, at this point in time, at the chip planning stage, we make major decisions about the layout of our circuit. So, what are those major decisions? So, first of all, what we do is that if our design is big and our industrial designs are typically very big, so, in that case, we partition that design into subsystems or blocks.

We partition a design into subsystems or blocks, and then we carry out the placement, routing, and physical design for these subsystems individually and then combine the results at the top level. So, at the chip planning stage, what we do is partition our design, and once we have partitioned into major portions, we define the location of these bigger blocks or partitions on the die. So, during the chip planning stage, we decide the location of the bigger blocks that we have in our design. For example, there will be a CPU core in an SoC, a bigger macro or memory block, a controller block, etc. So, we define the location for these big blocks. So, those are arranged and put on the die.

So, at this level, a lot of manual intervention can be required by the tool. The tool may not be able to come up with an optimum solution, and in that case, we may be able to tweak the locations of these macros and probably get a better QoR. So, at this stage, what we also do is that we decide the location of standard cells on the layout. So, typically we will have numerous standard cells, it can be millions. So, for those standard cells, we need to specify a location on our layout where it should lie.

So, we allocate rows on the layout where we are allowed to or where the physical design tool will be allowed to put standard cells. But we do not decide the location of each individual standard cell; we just say that this is the location of standard cells. Later on, in later stages, standard cells will be placed in that area, but we decide on the location of bigger blocks at this stage. So, for bigger blocks, it is not only the location, but many a times the shape may also be variable. For example, we can have a rectilinear macro rather than a rectangular.

Rectilinear means that it can have a shape of, say, I or it can have a shape of L, and it can have a shape of U. Rather than rectangular, the shape of some of the macros that are flexible can be rectilinear. In the chip planning stage, we can decide the shapes of those macros also. Besides this, we also decide the location of the IO cells or the input-output cells on the layout. So, in this case, it is shown that all of them are at the periphery, but it need not be. So, there are other ways in which input-output cells can be placed in different kinds of packaging environments.

So, there may be a lot of flexibility where the IO cells should be placed on the layout. So, the chip planning stage is a stage in which we also plan our IO cell placement. So, this is a stage where we decide the locations where the IO cells will be there, and from there, the signals will be driven to the rest of the cell. Another major step of chip planning is the power planning stage.

So, we know that when we have a standard cell, for example, an inverter. Now, to operate this inverter, it needs a power supply, VDD, and ground. So, to operate any combinational or sequential circuit element, they need power to operate. So, each standard cell needs to have, say, a power connection and a ground connection. Now, we can have millions of standard cells. Now, for each standard cell, we need to deliver power to them.

Now, how to decide the network that delivers power, which typically comes from a few pins on the chip to the entire layout because the standard cells will be spread over the entire layout, and not only the standard cell, there will be macros and other entities that will need power. So, how we are delivering the power to each of the entities in our design on the layout is also done in this stage, in the chip planning stage, and that part is known as power planning. So, we make, in this case, a power delivery network, PDN. So, while designing this power delivery network or PDN, the critical requirement is that the voltage drop should be within some tolerance limit.

We know that if there is a voltage source and from that voltage source, the power is going to some standard cell, then on the way, there are resistances, and because of that resistance and current flowing through it, the VDD that we apply, the voltage that we apply at the pin, at the input port that will be dropping before reaching the standard cell. So, we should ensure that the voltage drop in the power lines should be within an acceptable limit. Another consideration that is important during this chip planning stage is the congestion. So, congestion will highly depend on how we place the blocks or bigger blocks on the layout. So, if we put a block in some way, then there may be a region where lots of nets are coming, or lots of wires are running through that region, and that region gets congested. This is what is known as congestion.

Congestion means that in a given region, lots of nets are running close together, and that leads to congestion and it can lead to timing problems, routability problems, etc. So, if it is highly congested, then the subsequent design steps like routing may fail altogether. So, congestion can also cause timing problems because if there is a congested region, then routing needs to avoid that region, and as a result, the length of the wire will be much longer, and there can be more delay in the wires, and timing problems can be created. So, during the chip planning stage, some consideration of congestion also needs to be taken into account. So, after we have done the chip planning, then comes the step of placement.

Now, in the chip planning stage, we have defined the location of the bigger entities, only the macros or the subsystems or blocks of the design, but for the standard cells, we have said that we have defined a region, we have not yet placed those standard cells in the given rows on the layout. So, that step of placement of the standard cells is done during the placement stage. So, the placement basically decides the location of standard cells in the design. Now, what is the difference between the placement of, say, macros and the placement of standard cells? The problem is in terms of scales because there will be, say, tens of macros or hundreds of macros, maximum. So, we will not have too many macros on the layout.

So, in that case, when the area of macros is much larger than standard cells and the number of macros that need to be placed, those are comparatively less. Now, what about the standard cells? Typically, when we are trying to find placement for the standard cells on the layout, the number of standard cells can be millions. And therefore, in the placement of standard cells, manual interventions are very, very difficult. So, the placement step is a highly automated step. So, we may tweak a little bit of placement when we find some violation, but typically, the location of the standard cells is handled by the EDA tools or the physical design tools.

While for the placement of macros, a lot of human intervention can be made because of the many things a human eye can see by experience that, probably this is not a very good location and, therefore, perturb that location of macro, and maybe we can come up with a better solution than machine-generated floorplanning or placement of macros. But in the placement stage, automation is the only way because we have a very large number of entities that need to be handled. So, when we do the placement, what are the important objectives that the tool must consider? The most important objective is wirelength minimization. So, that is what the most important objective is for the placement to minimize the total wirelength in our design. So, what it does is that it tries to place the cells, the standard cells, which are connected together very close together.

So, if there is an inverter and there is another inverter that is connected to it, then probably these two inverters should be placed together. And if there is one inverter and another inverter that are totally independent, they are not sharing signals, and they are not connected. Then, probably, they can be moved further apart on the layout, and we may not impact the PPA. So, the wirelength minimization is a tricky task at the placement level because, at the placement level, we have not yet done the routing in the design. Therefore, we do not know exactly how the wires will run on the layout. So, the placement tool must rely on some estimate of the wirelength, and based on that, it will do some optimization.

So, the placement tool needs to estimate the wirelength, and based on that, it will try to minimize it, and that is a challenging problem because the wires are not yet laid out. The other objective of the placement tool is to ensure that the timing is met. So, it needs to reduce the delay of the critical path on which the frequency of the circuit that is being designed depends. So, an important consideration for the placement tool is to ensure that the timing is met or the delays on the critical path are minimized. To do this, the tool puts the cells close together that are on the critical path.

So the wire delay can be minimized, and also, the capacitors that need to be driven by a given gate are minimized, and as a result, the delay of a given gate can reduce and timing can be easily met. The condition can also be considered by the placement tool because a bad placement of some standard cells can lead to overcrowding of nets, and as a result, it can create timing problems, routing problems, or routability problems later in the design.

So, after the chip planning and placement are done, the next step is clock tree synthesis. Clock tree synthesis basically decides the topology of the clock network, meaning how the clock network will be structured on the layout and how the clock signal will reach from the clock source, there is a clock source, then there are the endpoints, and there are the clock pins of the flip flops at these endpoints. How will the clock signal reach from the clock source to the sinks? The flip flops where the clock signal needs to be reached are known as the sinks, that will be using the clock signal. Now, we require to have connections such that the clock signal behavior is more like an ideal clock signal. So, clock tree synthesis, also known as CTS in short, also performs the wiring of the clock network, and it avoids detours since the majority of routing resources are still unused. So, note that in a synchronous design, a clock is a very, very critical signal because all the operations are synchronized by the clock signal. A clock, whenever the clock signal comes, flip flops can toggle state, and then that signal propagates and reaches another flip flop, and so on. So, basically, a clock signal orchestrates all the operations in a synchronous design, and therefore, the clock signal is a very, very critical signal in a synchronous design, and therefore, we give the highest priority to the clock signal when doing routing.

So, we see during the CTS stage, we have not yet routed other signals. So, we are routing first these clock signals because the routing resources are fully available. Till now, we have used only the routing resources for power delivery networks, which defines how the power goes from the power source to each of the standard cells in the chip planning stage or power planning stage. The rest of the routing resources are still available while we are doing clock tree synthesis. So, the clock tree synthesis can easily avoid the detour on the clock signals, and it is very critical that we avoid these detours and reduce the delay of the clock signals from the clock source to the sinks.

Now, what are the objectives that the clock tree synthesis tool should consider? So, the most important objective for a clock tree synthesis tool is to minimize the skew. Now, what is a skew? Skew is basically the difference in the arrival time of the signals at two different points. Suppose the clock signal was generated at this point at, say, 0 picoseconds, at the clock source.

Now, as the clock signal propagates through the clock network, it will undergo a delay, and it will reach the sink at some point in time. Suppose a clock signal reaches a sink at time, say 10 picoseconds for a flip flop. And for the other flip flop, suppose there was a delay difference in this path, maybe because of some buffer with a different delay that the clock reached here at, say, 15 picoseconds. Then what is the skew between the arrival time of the clock at the two points is 15ps-10ps=5ps, this is the clock skew. Ideally, what we want is that for an ideal clock network, the clock skew should be 0 because we want all the flip-flops to get triggered at the same time, and that is what the assumption was made in the logic synthesis tool and based on that, the standard cells and other things were decided during technology mapping and generating the netlist. So, during physical design, we try to make our clock network as ideal as possible, and therefore, we need to minimize this clock skew ideally to 0, but at least the clock tree synthesis tool should try to reduce the skew to as low as possible. And to do that, what does the clock tree synthesis tool do? The clock tree synthesis tool basically tries to minimize the clock skew by choosing a topology such that it is symmetric. If the architecture of the clock tree is symmetric, then the chances of minimizing the clock skew are very, very high. Ideally, if it is totally symmetric, then the clock skew should be 0. Nevertheless, the skew can be something non-zero value because of process-induced variations and other things. But during the design stage, clock tree synthesis tries to ensure or make the clock tree as symmetric as possible to minimize the clock skew.

The other objective of clock tree synthesis is to minimize power dissipation. So, the activity on the clock signal is very high. So, for example, if I take a clock signal. So, in one clock cycle, it will have one rise and one fall edge. Whenever a clock signal changes from 0 to 1, the capacitances associated with the clock tree will be charged from 0 to 1, and when it goes from 1 to 0, then those will be discharged.

So, there is a continuous charging and discharging of the capacitances on the clock network, and that leads to power dissipation. So, the clock tree synthesis can consume power in the range of, say, 40 percent of the total power dissipation in a circuit. So, the clock network contributes highly to the power dissipation or significantly to the power dissipation of the entire circuit. Therefore, the CTS tool must consider minimizing power dissipation using some techniques. Typically, what these CTS tools do is insert clock gaters, meaning that they do not allow the clock to propagate based on some condition because maybe that clock signal is not required for a few clock cycles or whatever may be the reason.

So, the clock tree synthesis tool will try to minimize power in the clock network using some techniques. Then, after we have done the chip planning, placement, and clock tree synthesis then, we go for the routing. So, what is routing? Routing basically creates a wire layout for all the nets other than the clock and power supply. Why, other than the clock and power supply? As for the clock, we have already routed that during the clock tree synthesis step and power because the routing of the power nets was already done during the chip plan. So, for the other nets that are there in our design, those will be a huge number of nets because in a multi-million gate design, there are lots of signals, there are lots of logic gates, and those will be connected.

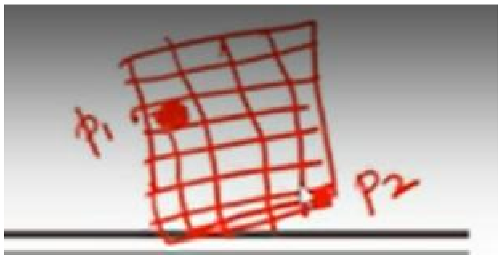
So, there will be many nets that are still unrouted, and those will be routed during the routing phase. While routing, it needs to satisfy some constraints. The constraints may be related to delay, wirelength, or some other measure. So, the objective is to minimize wirelength. A typical objective of a routing algorithm is to minimize wirelength or routing area or via count. So, these can be the objectives. The other objectives can be minimizing delay or improving the maximum operable frequency for a given circuit.

Now, routing is a very complicated task because there are too many nets that need to be handled by the routing tool.

And it takes a lot of run time, maybe hours or even days, to complete a routing for a complicated chip. So, since this is a very complicated step, we decompose this routing step or break this routing step into two steps. So, we decompose the total routing step into two steps: global routing and detailed routing. Let us look into each of them, what global routing is, and detailed routing is.

So, global routing is basically the planning stage of routing. So, in this stage, what we do is that we do not create an actual wire layout; we are just creating a routing plan for each net in our design. So, to create the routing plan, we divide the entire layout into routing regions. For example, for the layout of our design, the routing tool will divide it into rectangular regions. Each rectangular region is known as the global bin, and then the routing tool will decide that if I need to route a net, then how will the routing be done in terms of global bins or these rectangular regions that we have set?

Suppose we need to route a net that consists of two pins, P1 and P2, which are lying at two global bins. Then the global routing tool will say, for this particular net, follow this path in terms of global bins, meaning that from P1, it will go to which bin next and so on upto P2. So, actual routing is not done. It is just planned that for routing this particular net, I will go through these global bins. So, this is a kind of planning stage for the routing step. So, we can take an analogy that suppose we need to decide that we have to go from, say, Delhi to Calcutta.



Now, for going from Delhi to Calcutta, we may decide that from Delhi first we go to say Lucknow, then to Patna and then from there, I go to Calcutta. So, this is just one way in which a person can go. So, this is a very top-level plan. So, that is what global routing is, and now I have to do the planning in a more detailed manner. Then we say that from, say, Delhi to Lucknow, which highway I will take, for which cities we will take the bypass, and so on.

So, we go into the details of the path, and that is what detailed routing is. So, the global routing is just planning that I will go through these regions and detailed routing, then go into deciding the actual wires or the layout of the wires. So, detailed routing decides the actual layout of each net in the pre-assigned global bins, and these global bins were decided by the global routing step. So, the detailed router decides the actual physical interconnections of nets by allocating wires on each metal layer. So, we have discussed earlier that interconnects are in layers, and we can go from one layer to another layer using vias, and then the wires can run on the metal layers, and then those can be used to make complicated connections.

So, these complicated connections through various layers of metals and through using vias are decided by the detailed router. So, once the routing is done, basically, our implementation is done. Once we have done chip planning, placement, clock tree synthesis, global routing, and detailed routing, our implementation of the design is done. Then, if there are some problems or some changes need to be made that we find in the verification step that this thing is not working as per requirement, or some new requirement emerges, then we can make changes in our design, and those are very controlled changes and those changes are introduced using a method which is known as engineering change order (ECO). So, this is the method through which we make controlled changes in our design. We want to make controlled changes because, at the last step, we do not want to introduce any new bugs, and that is why we have to be very careful.

So, at the last stage, small fixes are made to the design. So, once we are done with those fixes, then our design is completely ready and to be sent out to the foundry for fabrication. So, then, we can get a GDS out of the layout. The GDS basically defines the features on the mask that will be taken through photolithography, and then that will be used for fabricating the chip. So, once we have the final GDS, we take that GDS and ship it to the foundry for fabrication. So, this task of dumping the final GDS and sending it to the foundry is typically known as tapeout, and this is an occasion for celebration for the design team because it ends an arduous design implementation phase, and then it goes to the fabrication part.

So, it completes the design part. So, a few things that need to be noted regarding physical design are that between the major steps of the physical design task, we can have optimization steps. So, between each physical design task, there are optimization steps for small changes in the design to improve the PPA. For example, we can insert buffers on long wires to improve the delay, or make a change in the location of the buffers or the size of the buffers for nets and so on to improve some PPA measures, or we can change the size of a given cell. So, these are a few examples of what optimization steps can do: they can do buffering, they can resize the cell, it can increase the size of the cell if you want to improve the performance or maybe to meet some slew criteria or some design constraint, or it can reduce the size of a cell to improve the power or meet some design goal, or it can change some placement location to optimize the timing.

It can also make some minor changes to the routing to improve some QoR measures like its timing, etc. So, these optimization steps can be introduced in between these major physical design tasks. So, these changes that are made in or these optimization steps that are undertaken in the physical design flow are incremental refinements to the design. We try to make only incremental changes. We do not want to make very large changes in this optimization step because if we make large changes, then it can create large disruptions in the design. Maybe we are improving PPA measures for one portion of our design, while in the other portion of our design, the PPA can go bad. So, it may be that the improvement that we introduce creates a very large disruption in the design, and if we keep on making large changes, we may not be finally able to converge on a solution.

Therefore, in these optimization steps, typically, small changes are made so that only a targeted optimization is done in a targeted region of the layout and not in a large region of the layout, or big optimization tasks are not introduced in between these tasks. The other thing that we need to consider is that when we are doing physical design implementation, we also need to do verification for timing, power, signal integrity, etc., and once we do this verification, we may find problems. Those problem needs to be fixed by making changes in our design and achieving design closure. Design closure means that we have achieved that state in which everything is fine in terms of the properties that the design should satisfy. Then, we say that the design closure has been achieved. Now, achieving design closure is challenging because design tasks that are done, for example, chip planning, placement, and clock tree synthesis, are working at a higher abstraction level, and based on that, they are making some decisions.

For example, when we are doing placement, the placement tool is trying to optimize the wirelength based on the estimates of the wirelength. It has not yet laid out the wire, so, those estimates can go wrong because whatever the estimate was made by the placement tool, later, when the detailed routing was done probably, that wire got detoured, and the wirelength increased by a lot, and as a result, timing and other things can get screwed up. So, if that happens, then we need to fix those problems, and to fix those problems, we might need to go into the previous steps. For example, what we have done is that we have done the placement, we have done the clock tree synthesis, we have done global routing. During detailed routing, we found that one particular net could not be routed because of some reason, and on analysis, we found that the location of a few cells was creating a lot of congestion and that the routing resource was all utilized in that region and as a result of that routing, the detailed router could not find a route for this particular net. Now, if we find such kind of scenario, then we might probably need to go into the placement step and then fix it.

So, in one design task, we might need to go into the previous design task to fix some problems, and this introduces feedback in the loop. We should understand that the flow that we have shown from, chip planning to the final writing of the GDS is not completely linear in one direction. There can be loops and iterations of these steps, and these iterations are required because we might need to retract some of the decisions made by the previous design task. Typically, in industrial design, it is very unlikely that in one flow, one push button kind of flow, we are able to close the design or get the required results on the layout. There are inevitable iterations in the design flow introduced because of the retracting of decisions of the earlier design processes. Therefore, the goal of a good VLSI design flow is to basically try to reduce the number of iterations that we might need to take to close this loop and to get the final layout that is desired. So, we try to reduce the number of iterations in this physical design flow, and there are techniques to do that, for example, if one of them could be a better estimate.

If we do a better estimate at a higher level of abstraction, then probably the decisions will be more likely to be correct, and therefore, it may not be required to retract. And these days, a lot of machine learning capabilities are used in design flows to reduce these iterations in the physical design flow. Not only in the physical design flow, the iteration can actually go back to logic synthesis. So, a kind of predictive analysis can be done and has value in logic synthesis. So, these are some of the references that you can look into to go deeper into these topics. Finally, to summarize in the last lecture and in this lecture, what we have done is that we have looked into the RTL to GDS implementation flow.

So, the implementation means that we are making changes to our design, say from the RTL, we have a higher level of abstraction, then to the netlist, which has a lower level of abstraction, and then to the layout. So, we have actually made modifications to the design, and we have implemented our design, but this is only half of the story. The other half is to ensure that the design that we have made is actually delivering the required functionality and also satisfying the requirements of timing, signal integrity, design rules, etc., and these need to be verified. So, we need to verify that the implemented design is fulfilling our purpose. Also, once we have designed it when that design goes to the foundry, is the foundry able to, or has the foundry made that particular chip defect-free, meaning that the way we have designed it, is a similar replica of that design on the chip or not and that needs to be tested. So, we need to account for testing during the design phase also, and therefore, there are two major steps that we need to carry out during the designing step apart from the implementation, and those are the verification step or group of tasks which are known as verification task and the design for testing task. So, these two tasks will be discussed in the next lecture. Thank you very much.