VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Week - 12

Lecture 57

Open-Source tool- YOSYS

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss about open source tool Yosys. The content of this lecture includes logic synthesis. Then we will discuss about Yosys introduction and installation. Then we will give a demo on Yosys. So, we have seen this slide earlier. One main point here is that this Yosys is meant for logic synthesis. This Yosys is meant for logic synthesis. So, basically it takes the RTL and converts into gate level netlist. This gate level netlist will be the output of the Yosys. So, if you can go into the tool, it is an open source tool for logic synthesis and it takes the dot clips or the timing library. Then it takes the RTL, then it takes the timing constraint files. Then we have a script which will be given to the logic synthesis tool and what is the output of this tool is basically gate level netlist and statistical reports, how many instances, what is the area, all these things. In this slide, we are describing from where we are taking the basically RTL. This RTL is taken from this location and timing libraries are available in open road installation path like the below path.

Anyone can download. Then you can use some basic Linux commands to locate the above input files in the Qflow and open road installed paths. So, this is the RTL and this is the dot lib. So, in this slide, we will discuss about how to run the Yosys tool. Basically we can run the commands into a different approach.

One first approach is basically first you type this Yosys-L run 01.log. So, what it will do? It will go to the Yosys prompt. So, in this case, what you can do? You can copy each of the lines of the command to the command prompt to know more detailed information about what each command is doing. So, this is run standalone command by command. So, this is one approach. The second approach is that we will run Yosys-L run 01.log. It will go to this Yosys prompt. There what you can do is that you can have a script file containing all the commands together in a single file. You can run that in the command prompt of the Yosys. So, here you will not know detail about each of the command, but you will get the final output of the script. So, in the second method, what we will do that? We will run the script flow.tcl file and in this one, all the commands are written inside that tcl file and whenever you run that tcl file, So, for example here map9v3 underscore run01.tcl whenever you are running that one, the final output will get at the completion of the tcl script.

Finally, we will get the final output. This is the statistics coming out of the Yosys tool. How many standard cells are used? What is the number of buffers? What is the number of cells inside the design? So, the output from the Yosys are basically .v file and .b file. These are basically the gate level netlist. So, this is basically the method to show or highlight some of the nets in different color to do the analysis. So, this is a script to show how the use of the graphically tracing the nets in Yosys. Welcome to the demo of logic synthesis tool Yosys. It is an open source tool.

So, anyone can use this one for doing the logic synthesis. So, logic synthesis is basically a process of converting the RTL to the gate level netlist. So, we will see here how we can convert RTL to the gate level netlist using this tool. So, first of all we will open this tool. So, this Yosys you can type Yosys-L, it will, Yosys-L stands for the log file. So, now once it is, if you type then it will open the Yosys prompt. So, you can do this logic synthesis in two different methods. In one method we will basically run the all the commands in a single command prompt. For example, if I type script space map9v3 underscore run dot tcl. So, all the commands will run together and at the end you will get the gate level netlist. It will show the gate level netlist in a graphical user interface. So, if you can go to this graphical user interface you can see the different cells what is there inside the gate level netlist. We can zoom it little bit more and you can see you have inverter AND gate. So, this is inverter, this is inverter, this is AND gate, this is D flip-flop. So, all these are combined to implement the RTL what is written inside that map dot v file.

So, this is the final gate level netlist. The final gate level netlist generated from the open source tool Yosys. This is very easy but think is that what is the issue of basically running in a all the commands together is that there might be some issue in some step you cannot correct that one. So, if all the steps are already corrected then you can run together in single line. So, next we will discuss how we can run this each of the step individually in a command prompt. So, first then we will exit from this tool then we will run this Yosys again. So, this is the command prompt for the Yosys. Now we will do one step at a time. So, this is the script for the logic synthesis. So, first of all we will read the dot lib file or the liberty file. So, this liberty file if you can see here I will copy and paste there and it has lot of information is there. If you can see we are using NAND gate 45 nanometer liberty file which has the timing information for the standard cells are given there. We discussed about the dot lib files in one of our lecture. So, here we are sourcing that liberty file first. So, now what it says that 134 standard cells in the library are imported.

Next we will go to the second step. So, it is basically reading the verilog file. So, what we are doing here is that we are reading the verilog file the map 9 v 3 dot v. So, here it has a RTL written in this file what we are basically reading in this Yosys platform. So, here this is an example verilog file but you can write your own verilog whatever needed for your design. So, this step this verilog is basically comes from the RTL designer. So, now this is successfully finished actually this step is successfully finished. This is the second step. The steps are written here. This is the first step. Then this is the second step. Now we will go to the third step. So, what is this third step is that check expand and clean up the design hierarchy. So, what it does is that it elaborate the design hierarchy. So, if there are multiple hierarchy is there or let us say some module is calling another module those hierarchy are resolved using this command. So, basically if you can see here we can this is the third step is we can do the hierarchy analysis or check in this step. So, it will analyze the design hierarchy analyze the design hierarchy in the top level module. So, this is the step related to hierarchy. This is the third step. Now we will have something called proc. So, it is basically related to the processes the internal representations of the behavioral verilog code into multiplexers and registers. So, what it does is that it converts your RTL the behavioral RTL in terms of multiplexers and the registers. Multiplexers are used for combinational logic implementations and the registers are used to implement the sequential logics. So, now this is the step we are doing. So, it has basically inserting the D flip flops, positive edge triggered you can see here positive edge triggered clock and flip flop it is inserted. So, all these are done in this step. It is very important step. Now after these processes then we can do some basic optimization. So, always we run this optimization to check that if any kind of optimization possible. So, basically this is the fifth step. So, it basically try to optimize the different intermediate formats.

So, this opt command is very useful sometimes to optimize your design. So, this is the fifth step. You can see here this is the fifth step. So, it has the fifth step itself has multiple sub steps like if you can see here you have nine sub steps are there in case of optimization. The 5.1 actually it executes the opt under expression pass which is the first step. And the last step is if you can see it is the finish of pass it is the final step. So, there are some intermediate step is there whenever you run this optimization or opt step. Then after you do this then we can analyze and optimize the finite state machines. So, using this fine FSM command So, you can write I will clean the screen before running the FSM. So, this is the finite state machine. So, it executes the FSM underscore map pass where we can analyze the finite state machines. This step consists of this FSM step consists of eight sub steps. After we do these eight sub steps then we can again optimize the design. So, again we will optimize the design. So, now after the optimization step, then we will analyze the memory and create circuits to implement them.

So, if you have any kind of sequential logic we can implement using this memory step. So, here if you can go and type this analyze memory and create circuits to implement them is the memory implementation. So, it consists of six sub steps actually from executing memory pass. So, executing memory deep flip flop pass. So, then you have opt clean pass then memory share pass like these all intermediate steps are there till it will get converting the memory cell into logic and flip flops is the final step. Now after this memory step again we need to run the opt step to do the optimization of the netlist generated. So, this optimization is done and after that if you can see here we have a take map which is very important step because here we are mapping our design to the generic library. If you can see here this is a technology mapping step what it does is that map coarse grained RTL cells like adder to fine grained logic gates. Let us say if you have a adder in the RTL what kind of logic gate should be used to implement that one is determined in this step and it maps the standard cells whatever we provided in the first step of the design. So, this is very important step and it will basically map the standard cells to the RTL. Now after this take map again we need to do the optimization and cleanup. So, this one is basically again do the optimization and cleanup. So, now what is the advantage of running this opt command here is that if you can see here this is the 11th step if you can see here it has many sub steps are there but here it starts actually whenever I run this one it starts here and if I will show you one very important point here that if you can see here this line basically you have 64 cells are removed to do optimization. 64 cells are removed to do the optimization. So, sometimes this opt step is very essential to optimize the design basically area power and timing efficient design.

Now after this one after this optimization step we have basically we have few more steps are left out. So, this is basically the register mapping phase. So, it maps the register to the available hardware flip-flop from the library. So, we have combinational gates which is mapped to the logic gates and similarly the register will be the map with the flip-flops available in the dot-lip file. So, this flow will do the mapping of the flip-flops because here you have written D flip-flop leave map. So, if you can see here we have 32 D flip-flops are inserted in the design. 32 D flip-flops are inserted in the design. This is alSo, very important step. After this again what you need to do is that you need to run the opt step. This basically useful for optimizations, but sometimes it does optimization sometimes it is not able to do anything. But if you can see here it removed 32 unused words doing this optimizations. So, this optimization thing should be run after every command to get even if you are not getting any benefit but some cases will get some benefits. So, it is the always advisable to run the opt statement or the command after each commands actually. Now, ABC optimizer is there then we will run this it will map the logic to available hardware logic gates in the NAND gate library dot-lip file.

ABC is the optimizer. Here we are used ABC for technology mapping. In this step only the combinational cells are mapped to the technology cells available in the standard cell library. So, if you can see here these are the cells finally this design is using and to X1, X1 means that it drive strength is 1. This is number of such cell is 3. Similarly if I go to the basically XOR 2 X1 we have 5 of them. So, now if I am interested for some complex gate like OAI 21 has 10 cells are there in this in this design. So, now after this ABC optimization we can do a flattened of the design overall design. So, this flattened command will basically do the flattened the complete design. So, in executing the pattern pass So, the design is flattened by this command.

So, this is a 15th step of this logic synthesis. Now after doing this we have basically replaced the undefined variable with the defined constant. So, if there is any kind of undefined variables there that should be replaced with the defined constant in this step which is very essential to avoid any kind of floating nets. So, this is very important. Now after doing all this then we can do on remove unused cells or wires. If there is any kind of unused cells and wires generated in the logic synthesis flow those can be removed or can be removed using this clean hyphen purge command.

So, this here in this case if you can see 140 unused wires are removed but there is no on use cells in the design. So, the wires are optimized by this cleanup step actually. Now we can map the IO pads to this design actually. So, here only we have given the output pads and we have added a buffer. So, output pins will be connected to the output pad through the buffer to the output pad. So, here we are doing the IO pad mapping actually. So, we have the output pins that will be connected to the output pad through the buffer using this command. So, this is the 17th step actually. So, this is the 17th step basically here we have not provided input ports but the output ports are mapped. Now after that again we will run the basically opt command to check and do some kind of optimization. So, after this optimization we will do that cleanup to remove any kind of unused cells and wire.

This is done. Then we will see what is the statistics of the cells used in the this RTL design after this gate level netlist is generated. So, if you can see here, this is a statistics actually. This is if you can see how many wires are there, how many basically memory cells are there, how many processes are there, how many total cell is there is 151. So, in out of 151 cells these are the different types of D flip flops are used which are the sequential cell and all are the combinational cells. And the total chip area based on the area provided in the dot lib it was calculated which is not the exact whenever you can do the chip level implementation because here we are just calculating the cell area based on the area of the standard cell provided in the dot lib. However whenever you will go to the physical synthesis to create the layout of the chip there we have more interconnect then the area will increase then the second one is the buffer will be inserted to satisfy the timing constraint then the area will be increased. So, those things will increase the area of the chip and this is whatever it is an estimate of the area for out coming out of the logic synthesis. Then after this step it finds out the statistics then if there is any kind of renaming of the nets are needed we can do that using rename hyphen enumerate command So, this step does that. Now after this we will write our design to the final

Verilog netlist. So, this one is the final Verilog netlist what we will get this Verilog is basically the gate level Verilog.

So, we gave the RTL as input we are generating a gate level Verilog that is map 9v3 underscore final dot v. So, this is the gate level Verilog. So, we can open alSo, to see this is a file you can open here. So, this is the gate level Verilog this is a big file with inverters NAND NOR all are basically logic gate level there is no order information order is now converted to logic gates there is no FSM information that is converted to logic gates everything is in the form of logic gates. So, this is very important file what we can use for physical synthesis. So, we can create the output file in various formats depending upon the various type of tools they accept it. So, here there is another format it generates is blif format. So, blif format we can generate the output file using this command actually. So, you can take it and So, this is the blif file. So, after this is the most important part we can see the schematic of our gate level netlist.

So, this is basically this command show hyphen stretch will show you the schematic of the gate level netlist schematic of the gate level netlist. So, hyphen stretch command will show the schematic of the gate level netlist in a graphical user interface will take some time to come up the window to come up rest. This is the screen actually it looks nothing but if you zoom in inside you can see you have logic gates like inverter here you have OAI21 underscore X1, X1 stands for drive strength is 1 this is another logic gate. So, here if you can see there will be if you can zoom it out you see here there are many D flip flops are there this is D flip flop which has clock D, Rn and clock D, Rn are the input Q and Qn are the output of the D flip flop. There is many D flip flops are there and these lines are basically the interconnection between the interconnection between the designs actually.

So, this is the overall complete netlist. So, this tool it is very interesting to design very complex design which is written in the RTL format to the gate level netlist using this open source tool which is called the Yosys which is a free tool anybody can download and install this tool and generate the gate level netlist for the given RTL. So, now we will discuss one of the feature of the Yosys. So, what it does is that we can trace the nets in the RTL or the gate level netlist using some simple commands in Yosys. So, first of all we will run the tool. Now the Yosys prompt is available. So, what we will do in this step the main idea is to trace the variables in a basically since the netlist. So, first of all we will read a small verilog file because it is easy to show that nets how it is traced. So, that is the reason we are taking a small netlist. So, here we are doing that proc command to do the synthesis and optimization.

So, first of all I will run these commands. So, after these steps are over then I can show the netlist in a graphical user format using the command show stretch. It takes some time to come up. Yes it is now available in the GUI format. If it is a very small netlist So, you have a few number of nets. What are the input nets clock A and S and some intermediate nets are there B C. So, we want to highlight all the nets in a different color to do some kind of debugging. So, this is very useful for debugging. So, what we did here is that we are creating a cone actually for the net A. So, we will run this command to see that how the A B and A B nets are highlighted. So, if I show this now you can see and see that this A net is highlighted and B net is alSo, highlighted. So, this if you can see here you have A B nets are highlighted using these commands actually and similarly A A and then here we have different basically colors assigned to that nets and blue for the B net and magenta for the A net and red for the cone A underscore B. Now if you can see here this A for magenta and this B for the blue and there is one more what is that A is basically B is basically your B nets blue and your A is So, this is the magenta actually corresponding to cone A magenta then you have blue for the state B then you have red for A B cone for A B cone red for A B cone like this. So, this is very useful for doing the debugging of the nets. If you have any kind of issues you can debug using graphical user interface. So, in this lecture we discussed about the open source tool YOSYS and its demonstration. Thank you for your attention. Thank you.