

## VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering  
IIIT-Delhi

### Lecture 31 Static Timing Analysis using OpenSTA

Hello everyone, welcome to tutorial 7 of PLSI Design Flow RTL to GDS. My name is Pooja Beniwal and I am a PhD student at IIIT Delhi. I will be your TA for this course. Today's tutorial objective is to get hands-on experience on the open source tool OpenSTA. OpenSTA is used to do static timing analysis in digital designs at the gate level. Now let's move into the tutorial.

In today's tutorial, we will be needing the OpenSTA tool and the Verilog file top.v, test.tcl and the constraint file that is top.stc and the technology library that is toy.

lab. All these files are available at the NPTEL website under the strategic material for week 7. This tutorial is divided into two parts: installation and how to run the OpenSTA. First you have to make sure that you have all of these dependencies installed and the way to install them is already shown in the previous tutorial. To install OpenSTA, we have to follow these steps.

First we will use the git clone command to clone the OpenSTA repo from GitHub. Git clone command will copy the GitHub repo of OpenSTA into your local machine.

```
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta$ git clone https://github.com/The-OpenROAD-Project/OpenSTA.git
Cloning into 'OpenSTA'...
remote: Enumerating objects: 12193, done.
remote: Counting objects: 100% (12193/12193), done.
remote: Compressing objects: 100% (2049/2049), done.
remote: Total 12193 (delta 9939), reused 12009 (delta 9848), pack-reused 0
Receiving objects: 100% (12193/12193), 24.34 MiB | 2.10 MiB/s, done.
Resolving deltas: 100% (9939/9939), done.
Updating files: 100% (427/427), done.
```

Now we can see that OpenSTA is already copied. We will move into the OpenSTA directory and we will create a build folder inside it. Now we will move into the build file and start the configuration of the build process through the cmake command.

```

poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta$ ls
OpenSTA
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta$ cd OpenSTA
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta/OpenSTA$ ls
CMakeLists.txt          README.md   etc        liberty    sdc        util
Dockerfile               app        examples  messages.txt sdf        veri
Dockerfile.ubuntu_18.04 cmake      graph     network   search
Jenkinsfile              dcalc     include   parasitics tcl
LICENSE                  doc       jenkins  power     test
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta/OpenSTA$ mkdir build
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta/OpenSTA$ cd build
poojabe@DESKTOP-FGB2SN5:/mnt/c/Users/pooja/sta/OpenSTA/build$ cmake ..

```

We will build OpenSTA using the make command. The make command will compile the source code and will generate the executable file. It will take a few minutes to run this. Now we will finally install OpenSTA using sudo make install. OpenSTA has been installed.

Now we will invoke the sta and verify that installation is a success. As you can see here, the environment has been opened of sta means our OpenSTA has been installed. Exit command is used here to exit from this environment. Now we will move into the second part that is how to run OpenSTA. As you can see here, OpenSTA needed these four files that is Verilog Netlist, Constraint file, Library file and the Tcl file to provide you with the report.

This report can be in text format or you can get it on the terminal only. Here you can see that I have already stored this file in the folder experiment. All these four files are in my folder experiment. You can get these files on the Nptel website and store them in your folder. Now one by one I will show you the contents of these files.

First we will look into the netlist. This is the netlist of the digital design shown here. As you can see, ab and clock is the input and reset is also the input, ab clock reset and out is the output. This is the netlist of this design. You can have any other netlist also.

Now the next file is the Constraint file. Constraint file contains the constraints for the design. Here the first command creates a clock minus the name CLK. We will create a clock named CLK that is CLK and this is of period 1000 picoseconds. The unit picosecond we get from the library file.

```

1 module top(a, b, clk, reset, out);
2   input a, b, clk, reset;
3   output out;
4   wire y;
5   wire y1;
6   wire y2;
7   INV I1 (.I(b), .ZN(y1));
8   DFFRNQ F1 (.CLK(clk), .D(y1), .Q(y2), .RN(reset));
9   INV I2 (.I(y2), .ZN(y3));
L0  BUF B1 (.I(y3), .Z(y4));
L1  NAND2 N1(.A1(a), .A2(y4), .ZN(y5));
L2  INV I3 (.I(y5), .ZN(y6));
L3  DFFRNQ F2 (.CLK(clk), .D(y6), .Q(y7), .RN(reset));
L4  DFFRNQ F3 (.CLK(clk), .D(y7), .Q(y8), .RN(reset));
L5  BUF B2 (.I(y8), .Z(out));
L6 endmodule

```

And the port clock means the source for this clock is CLK. The second command here we set the delay constraints for the input port A with respect to the clock CLK that is 5 picoseconds. Indicating that the arrival time for input A will be taken as 5 picoseconds. Same will be for the third command that will set the constraints for port B of 5 picoseconds with respect to the clock. Set output delay will set the output constraints for the port out.

Now we will move to the toy.lib. Toy.lib is the toy library created for learning purposes only. So the library file contains the timing model for each standard cell.

For example here we can see this NAND gate for the slew here at A2 and the load capacitance at ZN for this arc the delay will be modeled in this library. And we take the delay from here and STA analysis is done. More details about this toy.library will be shown in the next tutorial. Now we will move to the test.

tcl. Test.tcl contains the series of commands which at open STA will run. These are the four commands I have kept in test.tcl. You can also use report commands also in this file only. Completely britchy toy.

lib this command will read and load the library file. Read a very long top.v it will load the netlist. Link design top it will link the design with the timing cell. Here top is the main module of the design.

Read sdc top.sdc will read the constraint file. Now we will see the invoke command of STA that is STA and source test.tcl is used to run the commands which I have already shown you. Now I will show you how to execute this command.

The test.tcl has already been executed. Now I want to get the reports of the timing checks. So for that I will be using these commands. Report check part delay max minus format full and report check minus part delay min for minus format full. You can also use these commands that will generate the report in the text file.

Now I will explain what kind of results we are getting in this report. The command report check minus part delay max will check the delay in the maximum delay path and setup check is done here. Here the starting point is your flip flop f1 that is here and the end point is your flip flop f2 this is here. The path group is a clock.

There is only one clock clk. So for other things other reports also the path group will be clock only and part time is max the maximum delay path. Now we will see what we are getting in the report. For the first f1 clock there is no delay between f1 clock then f1 to q here we are getting 5.66 delay for i2 1.

63 for i2 it is 1.63 for b1 5.20 5.20 and for n1 n1 we are getting 1.91 second 1.91 second and for i3 1.

85 picosecond 1.85 picosecond. So what will be your data arrival time? It will be the sum of all these delays and we can see that it is here 16.25 picoseconds. Here the clock is 1000 picosecond which we have already given in the constraint because setup check is done on the next clock cycle so clock period here will be considered. Library setup time is taken from the library where it is 8.

3 picosecond 8.36 picosecond so we have subtracted this from the clock and we got the data required time that is 991.64 picosecond. So what will be the slack the slack will be the data required time minus data arrival time so this is our slack here we can say that our slack is positive means no setup violation is happening in the design. Now we will check about the hold violations. Hold violation check is done on the minimum delay path.

So this is the command for set hold check. The reports after running the check for hold violation is this. Here the starting point is F2 which is this flip flop and end point is F3 which is this flip flop. So the hold check is done between this flip flop and this flip flop which has the minimum delay present.

Here F2 relays 5.85 picoseconds from clock to queue 5.85 picosecond and therefore data arrival time will be 5.85 picosecond. There is no combination logic between those flip flops, therefore there is no delay for other things, therefore our arrival time here is 5.

85 picosecond. Hold check is done on the same clock edge therefore we do not consider clock period here so there is no clock period. Library hold time we get library hold time from the.left file that is the 1.70 picosecond for the F3 flip flop. For hold check we subtract data required time from the data arrival time.

After doing this we get 4.15 of slack which is also positive means also there is no hold violation. For more information on the OpenSTA you can go to this link. For more information on the commands of OpenSTA you can use this link and if anyone is getting this error he can get it resolved by looking into this. Now I will exit from this tool. Thank you.