Tutorial 7

Objective: To gain a hands-on experience on static timing analysis using OpenSTA.

Requirement:

- In this tutorial, you need to use OpenSTA.
- The Verilog file (top.v), OpenSTA script file (test.tcl), constraint file (top.sdc) and technology library (toy.lib).
- All the essential files mentioned above can be accessed on the NPTEL website under the study material for Week 7 of the course.

Installation of OpenSTA:

OpenSTA is an open source static timing analyzer (STA) tool used in digital design. It is utilized to analyze and verify the timing performance of digital circuits at the gate level.

1. **Install Prerequisites:**
   OpenSTA relies on various dependencies and prerequisites. Please ensure that you have the necessary build dependencies mentioned below installed before proceeding with the installation. Other versions may work, but these are the versions used for development.

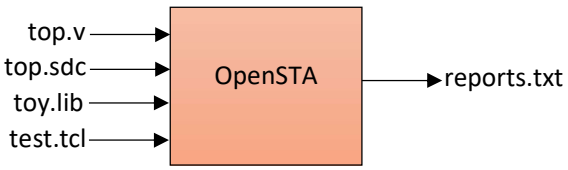
   cmake 3.10.2 3.24.2 3.16.2
   clang 9.1.0 14.0.3
   gcc 3.3.2 11.3.0
   tcl 8.4 8.6 8.6.6
   swig 1.3.28 4.1.0 4.0.1
   bison 1.35 3.0.2 3.8.2
   flex 2.5.4 2.6.4 2.6.4

2. **To install OpenSTA, follow these steps:**
   - Clone the OpenSTA repository by executing the following command:
     git clone https://github.com/The-OpenROAD-Project/OpenSTA.git
   - Move into the OpenSTA directory that was created during the cloning process using the following command:
     cd OpenSTA
   - Create a build directory using the following command:
     mkdir build
   - Move into the build directory using the following command:
     cd build
   - Configure the build by executing the following command:
     cmake ..
     This command configures the build process for OpenSTA, generating the necessary build files based on the project's configuration.
   - Build OpenSTA by running the following command:

make

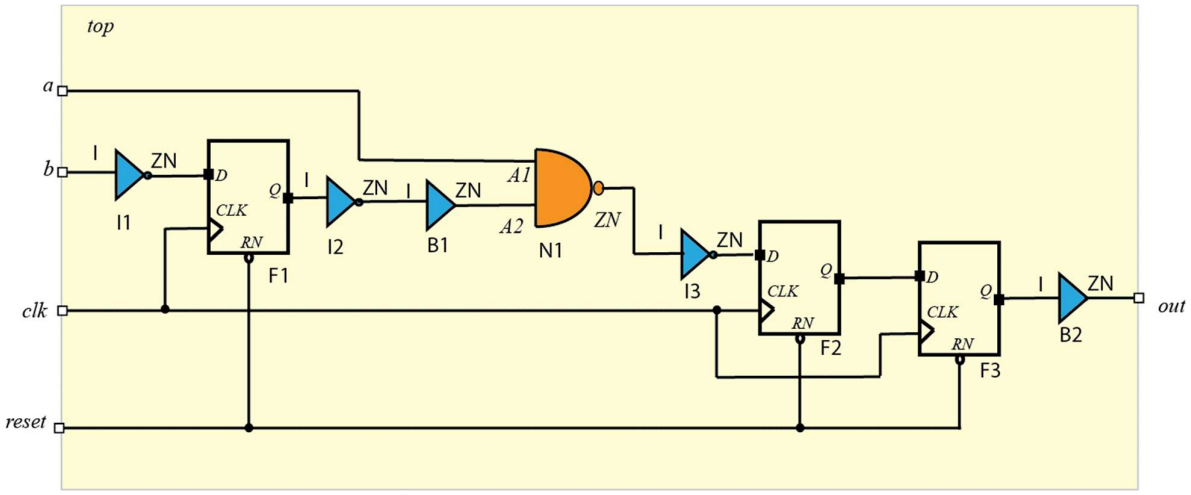
This command initiates the build process for OpenSTA. It compiles the source code and generates the executable files.

- Install OpenSTA by executing the following command:
  sudo make install



Steps to run openSTA:

1. **top.v:** This file contains the "netlist" of the design shown below.



2. **top.sdc:** This file includes timing constraints for the design.
   - *create_clock -name CLK -period 1000 [get_ports clk]*
     Defines a clock named "CLK" with a period of 1000 ps.
   - *set_input_delay 5 -clock CLK [get_ports a]*
     Sets an input delay of 5 ps for the input port "a"
     with respect to the clock "CLK."
   - *set_input_delay 5 -clock CLK [get_ports b]*
     Sets an input delay of 5 ps for the input port "b" with respect to the clock "CLK."
   - *set_output_delay 5 -clock CLK [get_ports out]*
     Sets an output delay of 5 ps for the output port "out" with respect to the clock "CLK."

3. **toy.lib:** The ".lib" file contains timing models for each standard cell. These models provide information about the cell's delay and transition time under different input conditions. In the next tutorial, we'll learn more about it.

4. **test.tcl:**
   The script file "test.tcl" contains a series of commands that will be executed by OpenSTA to perform various tasks related to static timing analysis.

Here's a breakdown of the commands commonly found in the "test.tcl" script:

- *read_liberty toy.lib*
  Instructs OpenSTA to read and load the Liberty file "toy.lib".
- *read_verilog top.v*
  Intructs OpenSTA to read and load the Verilog file (gate level verilog netlist) "top.v
- *link_design top*
  Using "top," which stands for the main module, links the Verilog code with the Liberty timing cells.
- *read_sdc top.sdc*
  Reads and loads the Synopsys Design Constraints (SDC) file "top.sdc".
- *report_checks -path_delay max -format full*
  Report of the timing checks for the design (setup)
  or
  report_checks -path_delay max > reports.txt
  Store the report of the timing checks for the design (setup) in the reports.txt file.
- *report_checks -path_delay min -format full*
  Report of the timing checks for the design (hold)
  or
  report_checks -path_delay min > reports.txt
  Store the report of the timing checks for the design (hold) in the reports.txt file.

5. How to run:

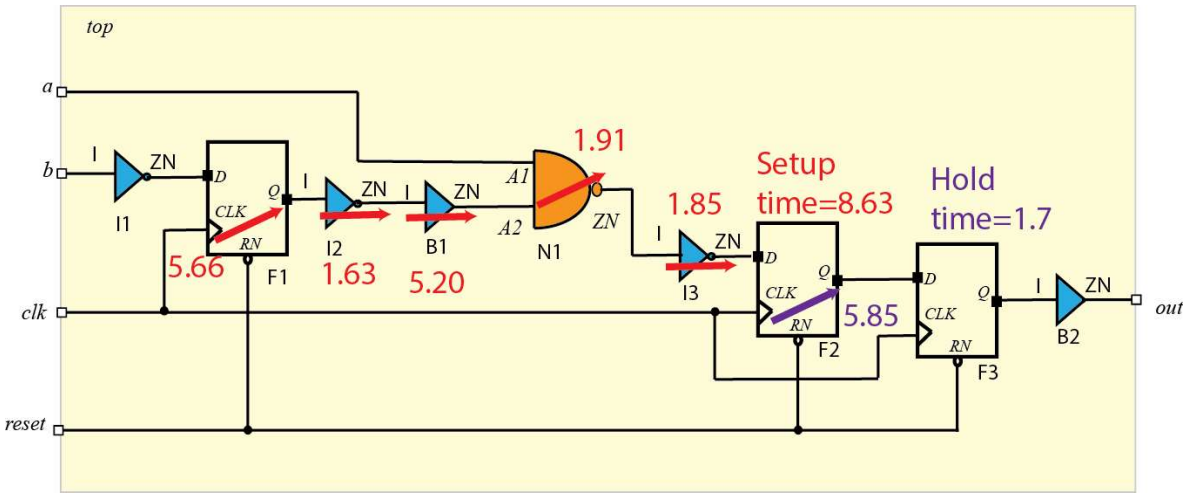
sta

The “sta” command is used to invoke OpenSTA and launch the tool. By running this command, you can check if OpenSTA is properly installed. If the installation was successful, the OpenSTA tool interface should launch without any errors or issues

source test.tcl

This command executes the commands specified in the "test.tcl" script file.

6. Analyzing report outcomes:



For more information:

Github:

https://github.com/The-OpenROAD-Project/OpenSTA

Manual:

https://github.com/The-OpenROAD-Project/OpenSTA/blob/master/doc/OpenSTA.pdf

Common error:

tcl.h: No such file or directory

To resolve the error you can follow these steps:

- Locate the file named FindTCL.cmake in the openSTA/cmake/ directory of your OpenSTA installation.
- Open the FindTCL.cmake file in a text editor.
- In the file, you will find the following code snippet:

```
# Locate tcl.h

if (NOT TCL_HEADER)
 find_file(TCL_HEADER tcl.h
   PATHS ${TCL_LIB_PARENT1} ${TCL_LIB_PARENT2}
   PATH_SUFFIXES include include/tcl
   NO_DEFAULT_PATH
   )
endif()
```

- Locate the path where the tcl.h file is located. For example, if tcl.h is in the /usr/include/tcl8.6/ directory, note down this path.
- Modify the code snippet as follows, replacing include/tcl with the correct path:

```
# Locate tcl.h
if (NOT TCL_HEADER)
 find_file(TCL_HEADER tcl.h
   PATHS ${TCL_LIB_PARENT1} ${TCL_LIB_PARENT2}
   PATH_SUFFIXES include include/tcl8.6
   NO_DEFAULT_PATH
   )
endif()
```

By making this modification, you are updating the search path for the “tcl.h” file to the correct location where it is installed, which should resolve the error.

Please note that the exact path and version number (tcl8.6) may vary based on your system and installed version of “tcl”.