Tool Setup and Tutorial

1. Introduction

In this document, we will set up our tools to run HDL simulation and synthesis with QuestaSim and Synopsys Design Compiler (respectively). You may complete these tasks in the lab (Light Engineering room 183) or from your own computer.

There is also an FAQ at the end of this document (Appendix A). If you have problems or questions, please start there. Then, check Piazza. If you cannot find the answer, then please post your question there. Often, several people will have the same problems, so discussing problems there will likely help everyone.

2. Grad Lab Departmental Computer Account

You will need a graduate lab computer account. If you have taken other classes that use these computers, you should already have an account that you can use. ***If you do not have an account, please email me at peter.milder@stonybrook.edu to request an account immediately.***

3. Working in the Graduate CAD Lab

If you are working from the graduate CAD lab (Light Engineering room 183), please follow the following instructions. ***If you are working remotely, please jump to Section 4 below.***

- Press any key to wake up the computer. Enter your username and your password.
- You will do most interaction with the computer through the terminal. Open the terminal by right-clicking on the desktop and selecting "Open Terminal."
- Now you can jump ahead to Section 5 to begin using our tools. Please note, when you are ready to log out, you can do so by clicking on the button on the top-right of the menu bar, selecting your name, and then clicking "Log Out."

4. Connecting to the server and forwarding X11 connections remotely

We will be using machines in the Graduate Lab in Light Engineering room 183. You may work directly on one of those machines or you may work remotely.

X11 is a protocol to run graphical user interface (GUI) applications remotely. The application runs on the remote computer, but you see the GUI on your own computer.

Peter Milder
peter.milder@stonybrook.edu

- ***If your Internet connection is too slow, you may find it difficult to use GUI applications remotely. In this case, I recommend that you either try using the "Remote SINC Site" (see Appendix B), or work in the graduate CAD lab when using QuestaSim's waveform view.***

- ***If you are working on a computer in the lab (Light Engineering 183), please skip the rest of this section and jump to Section 5.***

- ***The rest of Section 4 will help you configure your own computer to allow you to remotely access the CAD lab. If you have difficulties with these steps, I recommend that you try using the "Remote SINC Site" (see Appendix B), or work in the graduate CAD lab.***

- ***If you are working from your own personal computer, please continue with this section. Go to Section 4.1 if your computer runs Windows, Section 4.2 if your computer runs macOS, and Section 4.3 if your computer runs Linux***

4.1 Windows

Under Windows, I suggest using a program called MobaXterm. (However, if you are already familiar with tools like PuTTy and XWin32, you can feel free to use those.)

1. Download the latest release of MobaXterm from: http://mobaxterm.mobatek.net/. Click the "GET MOBAXTERM NOW" button and choose the Home version. I suggest using the "Portable" edition although the Installer edition will also work.

2. Unzip the file.
   - If you downloaded the "Installer" edition, run the installer.
   - Or if you used the "Portable" edition there is nothing to install.

3. Run MobaXTerm
   - If you have the "Installer" edition, you can run MobaXTerm from the start menu.
   - If you have the "Portable" edition, you can just run the .exe file from .zip file you downloaded

   If you get a Windows Security Alert, click "Allow Access."

4. Click "Session" in MobaXTerm's toolbar. Choose "SSH."
   - In the box labeled "Remote host," enter one of:
     - `eceap1.ece.stonybrook.edu`
     - `eceap2.ece.stonybrook.edu`
     - `eceap3.ece.stonybrook.edu`

   - Click the checkbox for "Specify username" and enter your username.

Peter Milder
peter.milder@stonybrook.edu

- Press OK. The terminal will prompt you for your password. Type it and press enter. Now, you should see the terminal prompt and something like the screenshot in Figure 1.

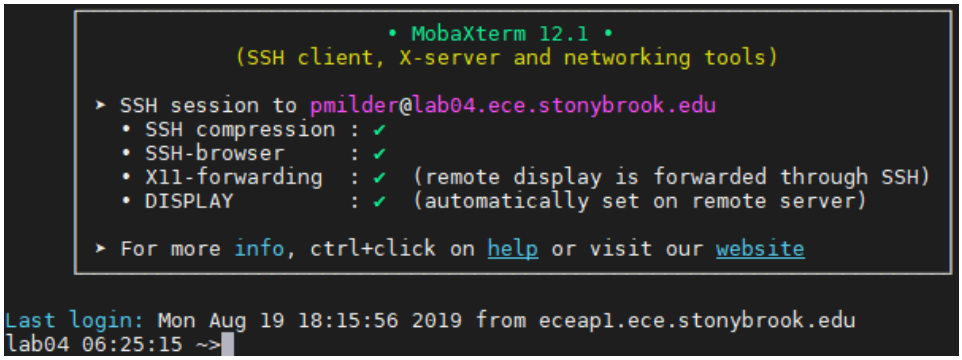


Figure 1. MobaXterm successfully connected.

5. Now, you should see a command prompt. Type the following command and press enter:

   ```
   ssh –Y [computername].ece.stonybrook.edu
   ```

   where for `[computername]` you can choose:
   - cadsrv0
   - cadsrv1
   - lab01 through lab16
   - lab21 through lab40

   For example, if you choose the first option, you would type:
   ```
   ssh –Y cadsrv0.ece.stonybrook.edu
   ```

   If you get an error at this step, try again with one of the other computer names listed above.

   If you are presented with a warning telling you “The authenticity of host … can’t be established.” You can answer “Yes.”

6. Enter your password one more time.

7. Lastly, you can test to make sure X11 forwarding is working by typing:

   ```
   gedit &
   ```

   This should cause a new editor window to open. (You may get some error messages printed to your terminal, but if the editor opens, you should be fine.) When the editor loads successfully, just close it and move to the next step.

8. While we are here, take note of the file browser that shows to on the left side of the terminal in MobaXterm. This file browser will let you interact with your files stored on the server. You can upload/download files, and even edit remote files locally. Please continue to Step 5.

Peter Milder
peter.milder@stonybrook.edu

4.2 Apple MacOS

1. If you have not done so previously, you will need to download and install XQuartz from http://xquartz.macosforge.org/landing/. XQuartz is a program that lets you remotely run a GUI from an X window system (which is commonly used in Linux systems such as our lab machines).

2. Next, you will open the Terminal application. You should find it located in `/System/Applications/Utilities/Terminal.app`.

3. Next, you will connect to our lab computer using SSH. In the terminal type:

   ```
   ssh –Y yourusername@eceap1.ece.stonybrook.edu
   ```

   Where you should replace `yourusername` with your actual username. You can also choose to replace with `eceap2` or `eceap3`

   Enter your username and password to connect to the server.

   Now, you should see a command prompt. Type the following command and press enter:

   ```
   ssh –Y labXX.ece.stonybrook.edu
   ssh –Y [computername].ece.stonybrook.edu
   ```

   where for `[computername]` you can choose:
   - cadsrv0
   - cadsrv1
   - lab01 through lab16
   - lab21 through lab40

   For example, if you choose the first option, you would type:
   ```
   ssh –Y cadsrv0.ece.stonybrook.edu
   ```

   If you get an error at this step, try again with one of the other computer names listed above.

   If you are presented with a warning telling you "The authenticity of host … can't be established." You can answer "Yes."

   Enter your password one more time.

4. Lastly, you can test to make sure X11 forwarding is working by typing

   ```
   gedit &
   ```

   This should cause a new editor window to open (through X11). When the editor loads successfully, just close it and move to Section 5 below. If this step fails, double-check step 3 above.

Peter Milder
peter.milder@stonybrook.edu

4.3 Linux

If you are running Linux on your computer, you should probably not need to install anything. Just open a terminal and then follow steps 3 and 4 of the macOS instructions above (section 4.2).

5. Tool path and license file setup

First, make sure you are logged into one of the `cadsrv` or `lab` computers as described above. **The CAD tools will not run on the `eceap` machines.**

Next, we will configure your shell to correctly "source" the license files we will need for the tools and add their locations to your path.

1. Log in to a `cadsrv*` or `lab*` server as shown in Section 4 above.

2. Determine which shell you are running by typing:

   ```
   echo $SHELL
   ```

   If the system returns `/bin/csh` or `/bin/tcsh`, you are running C shell. If you see `/bin/bash`, you are running bash. If you are running a different shell, you will see that. (Unless you have changed it, you will probably see `/bin/csh`).

3. I have provided scripts to set up the tools for you. If you are running C shell, type:

   ```
   cp /home/home4/pmilder/ese507/ese507setup-csh ~/
   ```

   If you are running bash, type:

   ```
   cp /home/home4/pmilder/ese507/ese507setup-bash ~/
   ```

   This command will copy the setup script into your home directory. You will only need to perform this copying operation once.

4. Now, to setup the tools, type the following into your terminal:

   ```
   source ese507setup-csh
   ```

   (or `source ese507setup-bash` if you are running bash).

   This command sets up your environment to run the tools. **You will run this each time you want to use our tools.** (Or you can also set up your `.cshrc` or `.bashrc` file to do this automatically if you know how to do that.)

5. Lastly, you can test this worked correctly by typing:

   ```
   which vlib
   ```


Peter Milder
peter.milder@stonybrook.edu

into your terminal. If you see:

```
/usr/local/mgc/questasim/questasim/bin/vlib
```

then the tools are set up correctly. If instead you see an error message, something went wrong. Double check the prior steps, especially steps 1 and 4.

6. Setting up a Private Directory and Getting Familiar

If you are unfamiliar with Linux terminals, now is a good time to get a little bit familiar with navigating the file system. I have included some explanation and helpful hints in this tutorial—this will be enough to complete HW2. However, I highly recommend doing some more reading to familiarize yourself with the terminal. A good starting point is the "Learning the Shell" section of linuxcommand.org.

In this section, we will also set up a private directory that you will use for storing your work. It is very important to make sure that you are using a private directory. This is so you can be sure that others are not able to take your work. The following steps will show you how to create a private directory you can use for all of your work this semester.

1. At the terminal prompt, type

   ```
   pwd
   ```

   This stands for "print working directory." This will show you what directory you currently are in.

   Each user has a personal "home directory." On our systems, your home directory is:

   ```
   /home/home5/[yourusername]
   ```

   (where `[yourusername]` is replaced with your actual username). If `pwd` does not show that you are in your home directory, enter command `cd` to go there.

   The ~ symbol is a shortcut to your home directory in the terminal. If you ever need to refer to your whole home directory in a command, you can type ~ instead.

2. Next, create a new directory by typing:

   ```
   mkdir ese507work
   ```

   into the terminal. Here `mkdir` stands for "make directory." (Of course, you can replace `ese507work` with whatever you want your directory to be called but the following steps will assume you chose `ese507work`.)

3. Now, change the permissions on this directory by typing:

Peter Milder
peter.milder@stonybrook.edu

```
chmod 700 ese507work
```

This will set the permissions so that you have read, write, and execute privileges, but all other users will have no access. Any subdirectory you create inside of `ese507work` will inherit its permissions. (You can read more about file permissions by searching online if you are interested.)

4. The cd command will allow you to change directories. Type `cd ese507work` to enter the directory you just created. Then, you can go back to your home directory by typing any of the following:

   ```
   cd /home/home5/[yourusername]
   cd ~
   cd
   ```

   You can see that all four of those commands do the same thing. The first one explicitly tells it to go to your home directory. The second one uses the ~ shortcut to avoid typing your home directory. The third one shows that if you type cd without anything else, the terminal will go directly to your home directory.

5. Now, let's make a private directory for this assignment inside of your `ese507work` directory. If you aren't already there, type

   ```
   cd ~/ese507work
   ```

   to go to the work directory we just created. Then, type

   ```
   mkdir hw2
   ```

   to create a new directory inside of ese507work for this assignment. The full path for this directory will be

   ```
   /home/home5/[yourusername]/ese507work/hw2
   ```

   or, you can think of it as:

   ```
   ~/ese507work/hw2
   ```

   Now, navigate to this new directory by typing `cd hw2` from within the ese507work directory.

6. You can also navigate directories relatively. For example, from your hw2 directory, you can run:

   ```
   cd ..
   ```

   To go "up" one level to ese507work. Or if you ran:

   ```
   cd ../..
   ```

   then you would go up two levels.

Peter Milder
peter.milder@stonybrook.edu

When you are done with this, go back to `~/ese507work/hw2` and move to the next section.

7. Text-Based Simulation with QuestaSim

Now, we will use QuestaSim from the command line to do text-based simulation. (Later, we will use QuestaSim's GUI to get a graphical waveform view of the design's signals.)

1. Use `cd` to navigate to the new directory we created above and copy our example design into your directory by typing

```
cp /home/home4/pmilder/ese507/demo/* .
```

when you are in the correct directory (probably `~/ese507work/hw2`). This says to copy (`cp`) all of the files located in the specified directory to your current local directory. Don't forget the space and the dot at the end of that line—the "dot" says that the destination is the current directory.

Then open up `test.sv` in your favorite text editor[1] and examine it. Here this is one module that contains a simple adder. Next, examine the `test_tb.sv` file, which has a basic testbench. The testbench provides a clock signal, basic input stimulus, and uses a `$monitor` statement to print the system's signal values as they change.

[1] If you don't have a favorite text editor, you can always use `nano` or `gedit` which are very basic, or Visual Studio Code. To use `nano`, just type:

```
nano nameoffiletoedit
```

To use gedit, type:

```
gedit nameoffiletoedit &
```

To use Visual Studio Code, type:

```
code nameoffiletoedit &
```

If you are familiar with them (or want to learn), you can also use `emacs` or `vim`. Plenty of information about these editors can be found online.

If you are using MobaXterm on Windows, you can also open this file on your local Windows computer using whatever editor you want by using the file browser on the left side of the terminal. Then, any edits you make to the file will be automatically saved on the server.

You can also use SSH capabilities built into modern text editors like VSCode, Sublime Text, and others to directly edit files. (If you use VSCode on your local computer, I suggest trying the sshfs plugin.)

2. Now, we will use QuestaSim to compile your design. Type:

```
vlog *.sv
```

   (This says run the `vlog` command on every file in the current directory that ends in `.sv`. If you wanted, you would equivalently run `vlog test.sv test_tb.sv`)

   At the end of the output, you will see a list of the design's top-level modules. Here, you should only see one: the testbench module named `tbench`.

3. Now, direct QuestaSim to simulate your code at the command line. You can do this in one line by typing:

```
vsim tbench -c -do "run -all"
```

   Note, here you specify `tbench`, the name of your top-level module. Then, the `-c` flag tells QuestaSim that you want to run in command line mode. Then, the `-do` flag tells it to execute a specific command, which you provide in quotation marks. Be careful with `run -all`; if your testbench does not contain a `$stop` or `$finish` it will run until you break with `control-c`.

   Please note: because of the way characters get embedded into PDF files, you will likely run into trouble if you try to copy/paste commands directly from this PDF into your terminal. For example, often the quotation marks or dashes get converted to other characters in the PDF. If you have problems with command, make sure you try typing them by hand to see if this is the case.

As the simulation runs, your testbench will print a line whenever any of the `a`, `b`, or `z` signals change. Note that it also prints the simulation's time along the left. Due to how we construct the clock signal, the positive clock edges occur every 10 ns on the fives (5, 15, 25, ...). Note that at this point, this 10 ns clock period corresponds to a 100MHz clock signal, but this is an arbitrary decision—our design does not simulate logic delays.

Look at the output and observe that the adder is working correctly. For example:

```
#                    6 a =   1, b =  -8, z =   7
#                   15 a =   1, b =  -8, z =  -7
#                   16 a =   2, b =  -7, z =  -7
#                   25 a =   2, b =  -7, z =  -5
```

So, at time 6, we see that the inputs are 1 and –8. Then, on the following positive clock edge (time 15), we see that the output `z` is equal to –7. Just after that clock edge (time 16), we see that the inputs change. Then on the following positive clock edge (time 25), the output changes to –5, their sum.

To simulate Verilog (instead of SystemVerilog), all steps are exactly the same. If instead you wanted to simulate VHDL, you would use `vcom` instead of `vlog`.

Peter Milder
peter.milder@stonybrook.edu

8. Using the QuestaSim GUI

Typically, the sort of console-based simulation we used above is very useful to quickly verify that your design produces a small number of correct outputs. However, this type of simulation is not very helpful for debugging or analyzing complicated designs. Imagine that you have designed a large complex system, but it isn't working correctly. To debug it, you would want a tool that allows you to see waveforms of all of the design's internal signals, which would allow you to trace through them over time. This is exactly what the QuestaSim GUI provides: a graphical waveform view of your signals, which can allow you to observe the top-level signals as well as the internal signals that are inside of your modules.

Using the instructions from Section 4 above, we can use X11 forwarding to access the GUI from any place where you can SSH into the server. However, the usability of the GUI will depend on the quality of your network connection—particularly the latency when you are trying to use the GUI's buttons and other interface elements. If you find that your personal Internet connection is too slow to use the GUI reasonably, then you can either:

- Try using the Virtual SINC Site. Please see directions and an explanation in Appendix B at the end of this document.

- Work directly in the Graduate CAD Lab, where all software will run quickly and responsively.

QuestaSim is a large and complex tool; there are many ways to do most tasks. Here, we will try some basic things, but you are encouraged to play with the tool and find other resources if you want to use other features.

1. If you haven't already done so, compile your design and testbench with QuestaSim by typing:

   ```
   vlog +acc test.sv test_tb.sv
   ```

   Note that here we have added `+acc` which tells the simulator that we will want to observe internal signals.

2. Now, we will launch QuestaSim in GUI mode. Type:

   ```
   vsim tbench &
   ```

   Now, the GUI will launch via the X11 connection we forwarded. It may take a bit of time for this to load. If you get an error here, review the steps in Section 4 and make sure you didn't miss anything.

3. When QuestaSim loads, you will see a window as shown in Figure 2 (on the following page). In the top-left, you will see the "sim" window, which shows the hierarchy of your design (the modules `tbench` and `dut`). To the right of that, you

will see the "Objects" window, which shows the signals located within the module selected on the left. Notice that when you select the `dut` module, the signals change; the internal signal `z_delay` appears (which is not visible to the `tbench` module). Lastly, at the bottom of the workspace is the "Transcript" pane. You can use this pane to interact with the system via typed commands.

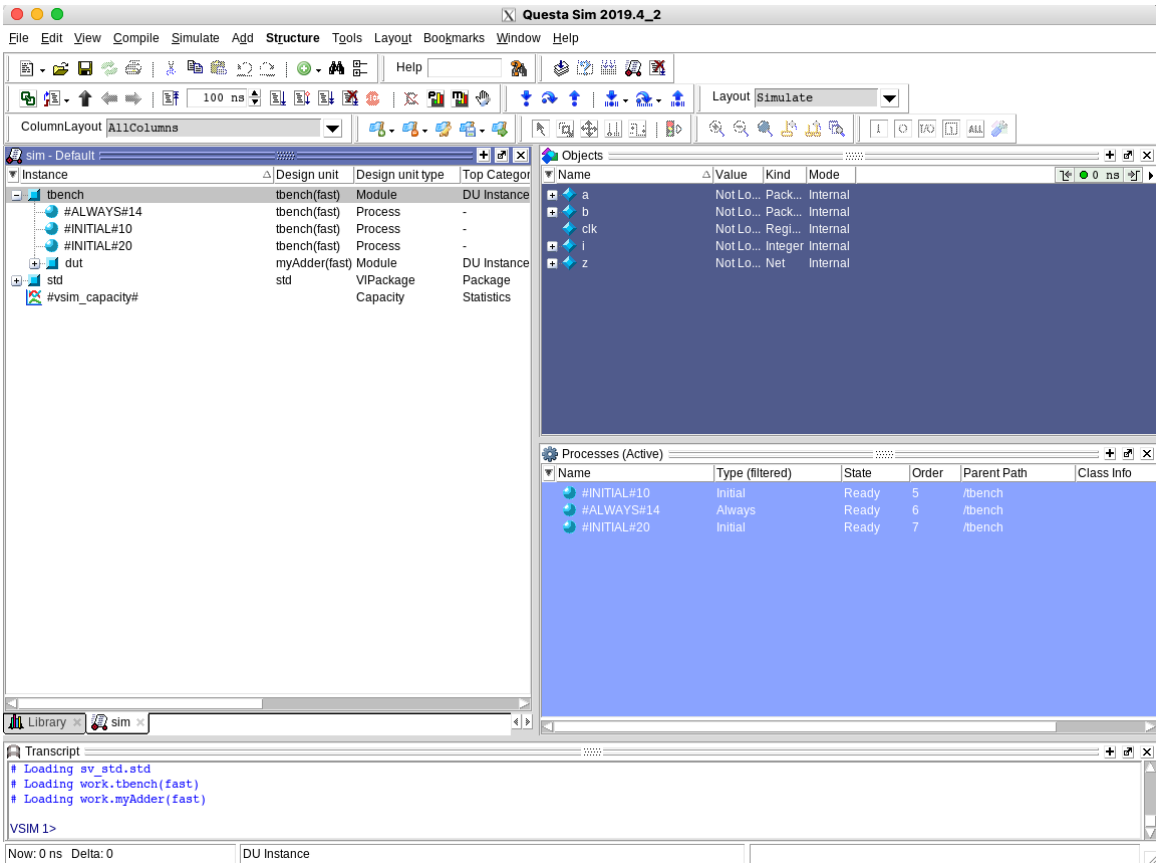


Figure 2. The initial QuestaSim window.

4. Now, let's create a waveform window. Select all of the signals from the dut module, right click, and select `Add Wave`. The waveform window appears on the right with those signals listed. (Or it may appear as a new window, depending on your settings.) However, the signal values are not shown because you haven't run the simulation yet.

5. Next, we will run some of the simulation. In the Transcript window, type:

```
run 100ns
```

   This will run 10 clock cycles of our simulation (remember, our testbench set the clock to have a 10ns period). As it runs, the waveforms begin to fill in on the right. You will most likely need to zoom in to view the data—to enable the "zoom" icons in the toolbar (a magnifying glass with a plus symbol), you may first need to click in the waveform window. Then, you can navigate around this

Peter Milder
peter.milder@stonybrook.edu

window. Try a few things such as changing the "Radix" of the signals (you can show values in decimal, binary, unsigned, hexadecimal, etc.). You can also do things like save the waveform format or save certain signals.

Note also that you can navigate to any module in your design and observe any signal. Here you are also looking at the `z_delay` signal, which is an internal signal to the adder module—it is not visible in the testbench module. This is a very important ability because it lets you view internal signals in your design.

6. Lastly, run the rest of the simulation by typing `run -all` in the Transcript window. One important note: when the simulation reaches the `$finish` command in the testbench, QuestaSim will prompt you with a window that says, "Are you sure you want to finish?" (See Figure 3.) This seems like a straightforward question—however, if you select "Yes," QuestaSim will interpret this as you requesting the program to quit altogether. **So, make sure you select "No" if you would like to view your waveforms.** (A simple work-around for this is to replace the `$finish;` on line 34 of `test_tb.sv` with `$stop;`)

   When completing homework 2, take a screenshot of your waveforms to include with your homework submission.

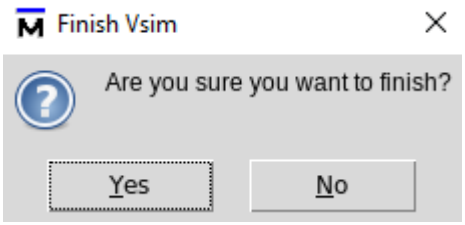


Figure 3. Don't click "Yes" or QuestaSim will exit.

7. Another useful trick for QuestaSim is important in the situation where you have run a simulation, but then decide you want to add new signals to your waveform view. In this case, QuestaSim was not monitoring the values of those signals, and so it will show only blank lines. In this situation, you do not need to *reload* your design, you only want to restart the simulation. In the transcript box, type:

   ```
   restart -f
   ```

   and then re-run the simulation.

   Something similar also occurs when you make a change to an `.sv` source file and want to recompile and simulate. You don't need to re-launch QuestaSim entirely. You can simply recompile the changed file with `vlog`, and then tell your running QuestaSim software to `restart -f`.

   Or you could combine multiple commands with semicolons. One thing I find very helpful is the following set of commands, which I use when I have changed my source file, and I want to re-simulate without quitting the simulator. In the QuestaSim transcript window, type:

Peter Milder
peter.milder@stonybrook.edu

```
vlog +acc *.sv; restart -f; run -all
```

This will recompile all `.sv` files in your directory and re-run the simulation while leaving your waveform window as you have set it up.

8. Another useful trick: imagine you are simulating a very large project, and you have very carefully organized many signals into your waveform view. You don't want to have to set up the wave window every time you start a new simulation. You can save the waveform window format by selecting File → Save in the Wave window. Then you can save a `.do` file that stores the list of signals and their order. Then the next time you open QuestaSim, you can re-load the wave window format with File → Load.

9. QuestaSim's GUI also has lots of other capabilities. You can use it to edit your source files and do push-button compilation (instead of typing things at the command line as described above). As you work, try to play with the tool and find the ways that match your preferred work style.

9. Synthesis using Synopsys Design Compiler

We will be using Synopsys Design Compiler to synthesize our designs targeting a 45nm standard cell library. This will let us accomplish several goals.

- First, we will be able to tell, through warnings and errors in the synthesis tool, if our behavioral description is in fact synthesizable. That is: can the synthesis tool correctly map it to a circuit?

- Second, we will be able to get a realistic estimate of the design's costs in terms of area (typically in μm^2) and power (in μW, mW, etc.).

- Third, we will get an estimate of the system's critical path (and thus its maximum clock frequency) and the location of the critical path in the design. This will allow us to fine tune our design to eliminate long critical paths.

- Fourth, the tool will output the gate-level design in structural Verilog. We can then use this (along with our original testbench and a library with behavioral descriptions of the standard cells) to verify that the synthesized design matches our desired behavior.

The following steps explain how to set up and run Design Compiler.

1. I have provided two script files to help set up your environment. Copy the following two files into your directory:

```
/home/home4/pmilder/ese507/synthesis/scripts/setupdc.tcl
/home/home4/pmilder/ese507/synthesis/scripts/runsynth.tcl
```


Peter Milder
peter.milder@stonybrook.edu

Do this by making sure you are in the correct directory (here, ~/ese507work/hw2), and typing:

```
cp /home/home4/pmilder/ese507/synthesis/scripts/setupdc.tcl .

cp /home/home4/pmilder/ese507/synthesis/scripts/runsynth.tcl .
```

Don't miss the space and the dot at the end. In these lines, `cp` tells the system you are copying a file. The long `/home/home4/...` part tells it the source file. There is a "dot" at the end tells it the destination—the dot means "here."

The first file is a generic setup file that you will not change. The second file is the script that runs the synthesis job. It will need to be changed based on your design.

2. Edit the file `runsynth.tcl`. Notice at the top that there are five variables you will need to set based on your designs. They provide the clock signal's name, the desired clock period, the reset signal name (if used), the top-level module's name, and the source file with your SystemVerilog (or Verilog or VHDL) design. Note that these variables are already set up for this design (but you will need to change them to use this script with other designs in the future).

   Below that are lines that tell Design Compiler to read and analyze your design, to set up the clock signal at the desired frequency, to run the synthesis job, to report area, power, and timing, and to output the final standard-cell level design as Verilog at `gates.v`.

3. Run this synthesis script by typing:

   ```
   dc_shell -f runsynth.tcl | tee output.txt
   ```

   The first part starts the Design Compiler shell. The `-f` flag points it to your synthesis script. The synthesis tool prints a lot of information to the screen. The final part of the command `| tee output.txt` tells your system to print all of that information to the screen, and also store it in `output.txt`, so you can view it later.

   If you just want to view the output but not save it, just get rid of `| tee output.txt`. If you want to save the output but not view it, then instead run

   ```
   dc_shell -f runsynth.tcl > output.txt
   ```

4. Examine your `output.txt` file to find the area. For area, you will see a line that says, "`Total cell area.`" This will give you the area of your design in μm^2. Note that it also breaks the area down into combinational (gates) and noncombinational (flip-flops).

5. Examine your `output.txt` file to find the power estimate. This will be presented as dynamic power in μW and leakage power (static power) in μW or

Peter Milder
peter.milder@stonybrook.edu

nW. To find the total power, you can add these two values together (keeping in mind 1μW = 1000 nW).

6. Examine the system's output and determine the clock period. At the end of the timing report, if it says `slack (MET)` then it indicates the tool was able to meet your requested cycle time (whatever you asked for in `runsynth.tcl`). If it instead says `slack (VIOLATED)` then it means it was not able to meet timing.

   Initially, the script is set up to target 1ns (1 GHz), and the system can meet it easily. Now, edit `runsynth.tcl` to decrease this target; find the smallest clock period possible[2] where the synthesis tool can meet timing. (When you run again, change the name of the output file so you do not overwrite your previous log.) The system's area and leakage power will likely increase as the requested frequency gets higher. This is because the synthesis tool can often build your logic in different ways; some are faster and larger, while others are smaller and slower. The system's dynamic power will increase as the frequency increases, because dynamic power grows proportionally with clock frequency.

   As you experiment in this step, write down the area, power, and clock frequency of each design you try. Once you have found the highest possible frequency (smallest possible period) for which the system can meet timing requirements, then compare its area and power to the area and power when you synthesized the design at 1ns. ***Include this in your HW2 submission.***

7. Lastly, Design Compiler produced a Verilog file `gates.v` that contains the synthesized standard-cell-level[3] design. First, open up `gates.v` in a text editor and look at it. Note that it contains one module (`myAdder`) and has the same name and input/output ports of our initial design, but the logic consists solely of instantiations of standard cells and flip-flops.

10. Post-synthesis simulation

Lastly, we will perform simulation on `gates.v`, the standard-cell-level Verilog file produced by synthesis. This will verify that the synthesized circuit works correctly.

1. First, let's reset QuestaSim's work directory to ensure you are simulating the synthesized output `gates.v` and not the earlier `test.sv` file. You can do this by making sure you are in your hw2 directory and typing:

   ```
   rm –rf work
   ```

[2] For this assignment, don't bother adjusting the period in increments smaller than 0.1ns. In other words, if your system meets timing at 0.9ns and fails at 0.8ns, you don't need to bother trying 0.85, 0.86, etc. (In later projects you may want to do this but for this warmup it isn't necessary.)

[3] A standard cell is a small group of transistors that provides a simple standard logic function like one or a few gates or a flip-flop. When you look at the `gates.v` file you will see examples of standard cells. We will also discuss this more later in class.

Peter Milder
peter.milder@stonybrook.edu

Here `rm` is the command to remove a file or directory. The `-r` tells it to operate recursively, allowing it to remove the directory. Then `work` tells it the name of the file or directory to remove.

2. Next, you can compile `gates.v` by typing `vlog gates.v` in your terminal. (Important: do not compile `test.sv` here.)

3. Next, we also need to compile the simulation model of the standard cells. This file is provided for you in the course directory. Compile this library on a lab computer by typing:

```
vlog /home/home4/pmilder/ese507/synthesis/verilog/
NangateOpenCellLibrary.v
```

   (Note: this command should be entered entirely on a single line.)

   If you are unfamiliar with using the terminal, this is a good time for you to take note of something called "Tab completion." This is a method where you can type a portion of a directory or filename and then press the TAB key on your keyboard. This instructs the terminal to try to fill in the rest of the name for you. For example, try typing `vlog /home/home4/pm` and then press TAB. You will see the terminal fills in the following `ilder/` . Then you can type `ese507/s` and press TAB again, and it will fill in the `ynthesis/` . For more information, search for "tab completion" online or read this:
   https://en.wikipedia.org/wiki/Command-line_completion#Example

4. Now, we just need to compile the testbench. You can re-use your original testbench. Compile it with `vlog test_tb.sv` as before.

5. Now, you can simulate your standard-cell-level design! Just like before, type

```
vsim -c tbench -do "run -all"
```

   which will simulate as before. (Except now it is simulating each of the individual standard cells.) You will get a few warnings about "Too few port connections" but you can ignore these.

   Verify that the design still simulates correctly. If you like, you can also run the simulation in GUI mode and observe the behavior of the individual cells.

   If you ever find a design that does not simulate correctly after synthesis, it means that something went wrong in the synthesis process. If that happens, go back and look at the `output.txt` log file produced by Design Compiler and look for warnings. (Later in class, we will discuss in more detail what kinds of things can go wrong and how to identify and fix them.)

Appendix A. Frequently Asked Questions

If you have other questions, please post them on the class Piazza discussion board.

Peter Milder
peter.milder@stonybrook.edu

1. ***I don't know anything about Linux. Help!?***

   Below are answers for very common questions. You can also find lots of information online. A good starting point is the "Learning the Shell" section of linuxcommand.org.

2. ***How can I edit files?***

   The absolute simplest things you can use are `nano` or `gedit`, which are very basic.
   To use `nano`, just type:

   ```
   nano nameoffiletoedit
   ```

   To use `gedit`, type:

   ```
   gedit nameoffiletoedit &
   ```

   However, it ***really*** is worthwhile to learn `emacs` and/or `vim`. Plenty of information about these editors can be found online.

   If using `nano`, you will interact entirely using your keyboard. The bottom of the screen will show a menu of commands that you access with the control key. For example, where it says `^X Exit`, this means hold control and press X to exit.

   If you are using MobaXterm on Windows, you can also open files on your local Windows computer using whatever editor you want by using the file browser on the left side of the terminal. Then, any edits you make to the file will be automatically saved on the server.

   You can also use SSH capabilities built into modern text editors like VSCode, Sublime Text, and others to directly edit files. (If you use VSCode, I suggest trying the sshfs plugin.)

3. ***Every time I login to a lab computer, I get a weird message that says "id: cannot find name for group ID XXXX." Is this a problem?***

   No, don't worry about it.

4. ***When logging to the Linux computer, I received the following message. Is it a problem?***

   ```
   /usr/bin/xauth: file /home/home5/<username>/.Xauthority
   does not exist
   ```

   No, you can safely ignore this.

5. ***How do I figure out which directory I am in, and how do I navigate directories?***

   You can type `pwd` to print your current directory. When you first login, you should be in a directory like `/home/home5/yourusername/`

   You navigate directories with the `cd` command. If you type

Peter Milder
peter.milder@stonybrook.edu

```
cd subdirname
```

you go “down” into the subdirectory you typed. If you want to go "up" a level, you type

```
cd ..
```

(That's a space followed by two dots.)

You can also combine the two. For example, if you are in a directory with subdirectories `a` and `b`, you can do:

```
cd a
```

Now you are in `a`. Then you can type

```
cd ../b
```

Now you told it to go up one level and then down into `b`.

If you want to go back to your home directory, you can just type `cd` with nothing after it; this is a quick shortcut so you don't have to type the full home directory path.

6. ***What does it mean when there is a tilde character ~ in a path?***

   This is a shortcut for your home directory. So, if you have `~/ese507work/` then this is equivalent to `/home/home5/yourusername/ese507work/`

7. ***If I start working on one lab computer, can I move to another?***

   Yes. Your home directory is shared over the network and the machines are all set up the same way. You can switch from one lab machine to another with no problem.

8. ***How do I see what files and subdirectories are in a directory?***

   You can use the `ls` command.

9. ***What does it mean when some filenames start with a dot? Like `.cshrc.cad`?***

   Filenames that start with a dot are hidden by default. When you run `ls`, those files won't show up. If you want to view them, instead type `ls –a`

10. ***How can I easily view the contents of a text file?***

    The `less` command is a quick way to scroll through a text file. Just type `less filename` (where filename is the name of the file you want to view). You can scroll up and down, and when you are done, press `q` to quit.

Peter Milder
peter.milder@stonybrook.edu

11. ***When I try to open something with a GUI (gedit or QuestaSim in GUI mode for example), I get an error like "Cannot open display: Localhost:13.0". What does this mean?***

    It means your X11 forwarding is not working correctly. Try the X11 steps above again, and make sure you set X11 forwarding correctly as explained above.

12. ***How do I get files on and off the lab computers?***

    If you are using MobaXterm on Windows, you can use the file browser on the left side of the terminal to upload/download files.

    Otherwise, the best way is to use SCP. From Windows, the program WinSCP (www.winscp.net) works very well and is easy to use. On Linux or Mac, it's fairly easy to use SCP from the command line (e.g. http://www.hypexr.org/linux_scp_help.php).

    If you are off campus, you can point your SCP program to `eceap1.ece.stonybrok.edu`. Your files are synchronized there as well.

13. ***Everything worked yesterday, but now today I logged in again and none of the tools will run. What's wrong?***

    You may have forgotten to run `source ese507setup-csh` or `source ese507setup-bash` after you logged in. (See Section 5.) This step needs to be done every time you log in to the computer. (Or you can set up your system to automate this.)

14. ***When I try to SSH into a computer, I get warning about a host key differing like the following. What do I do?***

```
$ ssh eceap2.ece.stonybrook.edu
Warning: the ECDSA host key for 'eceap2.ece.stonybrook.edu'
differs from the key for the IP address '129.49.69.207'
Offending key for IP in /Users/pmilder/.ssh/known_hosts:42
Matching host key in /Users/pmilder/.ssh/known_hosts:81
Are you sure you want to continue connecting (yes/no)?
```

    Just type "yes" and press enter to connect.

15. ***When I tried to SSH into a computer, I get a "Remote host identification has changed" warning, like the following. What do I do?***

```
$ ssh eceap1.ece.stonybrook.edu
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-
middle attack)!
It is also possible that a host key has just been changed.
```


Peter Milder
peter.milder@stonybrook.edu

...

This warning means that something has changed on the computer you are trying to connect to. Typically, this means that our departmental IT staff have updated the computer recently. You can clear this warning by running:

```
ssh-keygen -R eceap1.ece.stonybrook.edu
```

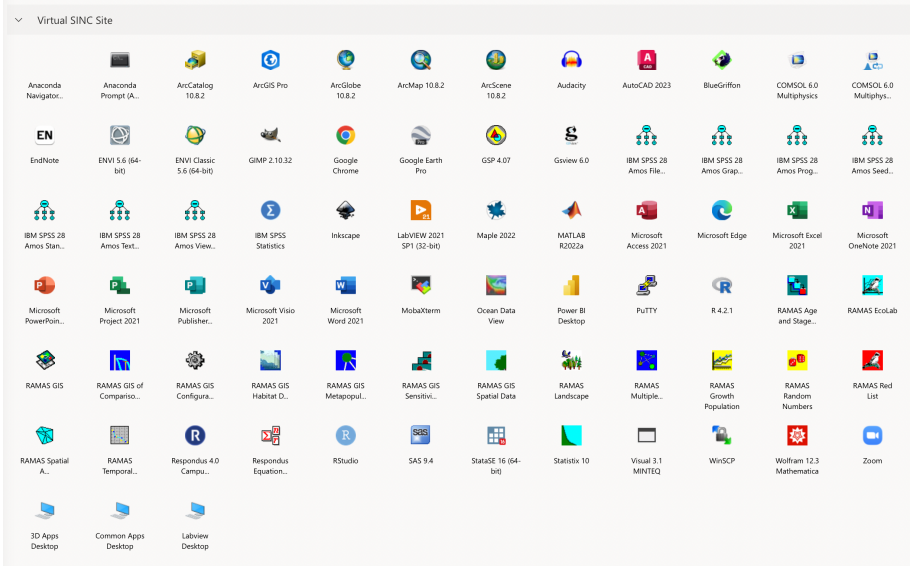
You should replace `eceap1.ece.stonybrook.edu` with the name of the computer you were trying to SSH into that caused the warning.

Appendix B. Connecting Remotely via the Virtual SINC Site

If you have trouble with running our software tools on your own computer, or if you find that your Internet connection is too slow to use QuestaSim's GUI mode via X11, another option to try is Stony Brook's Virtual SINC Site. The basic idea is that you can connect to a Remote Desktop session in your, and then run MobaXTerm from the remote computer. This can be done without installing any software on your computer, and it may help with connection speed issues because the connection speed between the Remote Desktop and our CAD lab likely has lower latency than the connection between your home and our CAD lab.

- First, go to https://it.stonybrook.edu/services/virtual-sinc-site and click on the "Launch Virtual SINC Site" button. You will need to log on with your Stony Brook Net ID (and authenticate with Duo).

- After logging in, you will see a page that lists available software to run:



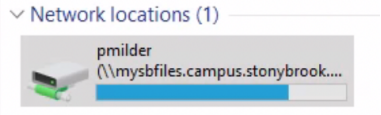
Select "Common Apps Desktop" to load a full Remote Desktop. (You could instead opt for loading MobaXTerm directly from this screen but I believe it's beneficial to have the full desktop, which can make it easier to do things like upload/download files.

Peter Milder
peter.milder@stonybrook.edu

- You will need to log in again. The username should be your full stonybrook.edu email address (which should be auto-filled in for you). For the password use your NetID password. It may also prompt you for what permissions you want to allow. I suggest enabling File Transfer, which would

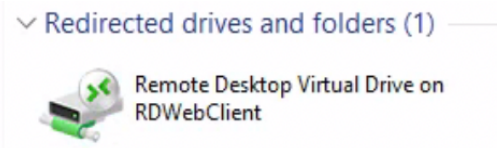
- After logging in, you will see a standard Windows desktop. You can then follow the directions in Section 4.1 above (skipping the steps where you would install MobaXTerm since it is already installed on the Remote Desktop).

- Files stored on the Remote Desktop are removed after you log out. However, each user has a shared network drive that you can use for storage. To see it, double click on "This PC" and find the "Network locations" category. There you will see a drive with your username:



  Anything stored here will be saved and accessible from Virtual SINC Sites (or in-person Stony Brook SINC Sites).

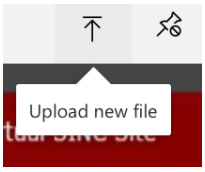
- If you need to upload files to the remote desktop or download files from the remote desktop to your computer, there is built-in functionality to do this. On the remote desktop, double click on "This PC." There you should see a category called "Redirected drives and folders" with an entry "Remote Desktop Virtual Drive on RDWebClient."



- Double click on Remote Desktop Virtual Drive and you will see it includes a folder named "Downloads" and "Uploads."

    - To download a file from the Remote Desktop to your computer, copy it to the "Downloads" folder in the virtual drive. The software will then download it using your browser.

    - To upload a file from your computer to the Remote Desktop, find the Upload icon near the top-right corner. It looks like an upward arrow:



Peter Milder
peter.milder@stonybrook.edu

Click the arrow, select your file, and this file will be uploaded an appear in the “Uploads” folder in the virtual drive.

Peter Milder
peter.milder@stonybrook.edu