Organization of Digital Computers: A Practical Perspective

Setting Up the Tools

Before deep diving into designing an efficient processor, we need to setup the require tools for this purpose. The following tools/skills are required and will be covered in this chapter:

- Overview of **Linux OS**
- Documentations using **Latex**
- Collaboration and revision controlling using **Git**
- HDL Simulator
  - MentorGraphics QuestaSim

Part 1: How to work in Linux environment

Linux, short for Gnu/Linux, is the Unix-like operating environment you will use for this course. This section introduces the required Linux commands and tools that you will need to use during this course. The reason the Linux operating system is chosen, instead of Microsoft Windows or other types of Unix-based OSes, is that most industry-strength CAD tools that we need to design implement, simulate, and verify our logic block are developed for Linux. That's why you need to have the knowledge of how to work with Linux.

Note that you don't have to install Linux on your own computer; instead, your own computer is a client, and you would use a client program to login to a Linux server provided by the course with the CAD tools already installed.

I) Connecting to a Linux Server

This course provides a host or instructional server that has Linux (Unix) operating system installed on it along with all the CAD tools. You need to know the host name or IP address. We also assume you know your account credentials (username and password) on that host. You will run a client program to login to the server.

For UCI Students only:

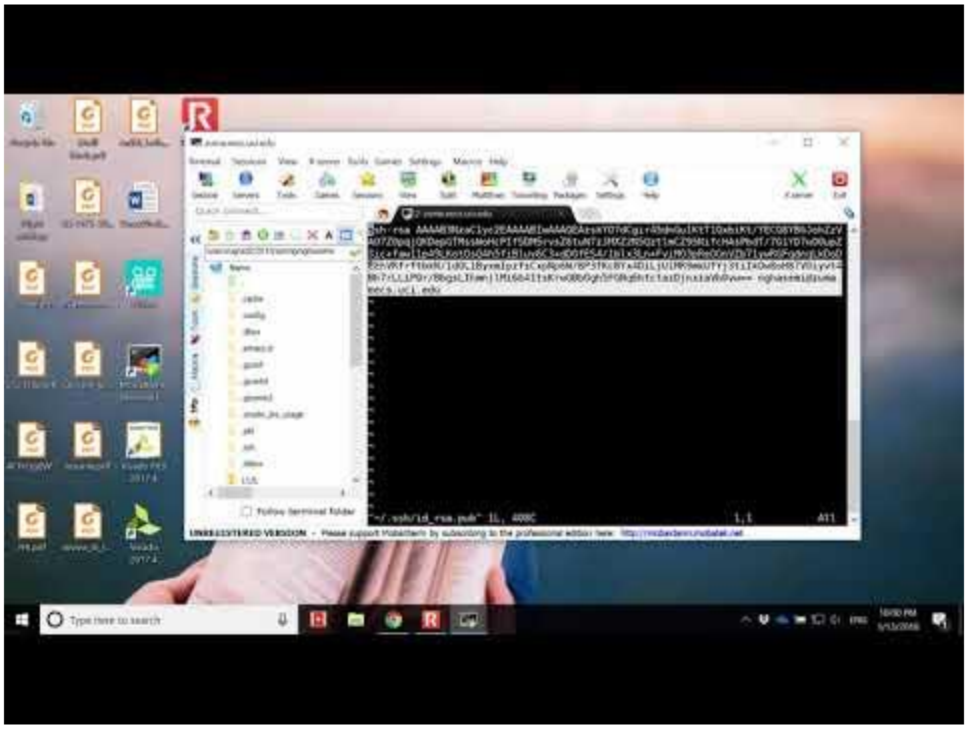
You can actually login to one of following instructional servers to run the CAD tools. The host names are:

zuma.eecs.uci.edu
laguna.eecs.uci.edu
crystalcove.eecs.uci.edu
bondi.eecs.uci.edu

Use your UCINetID and Password to connect to any of these machines. All of these host machines share the same file systems, so when you change a file on one host, you'll see the change on all other hosts.

You will run a client program to connect to an instructional server. You can use any client that supports the SSH protocol. Basic SSH clients are for text mode access. However, to take advantage of the graphical user interface features (e.g., waveform display), you would need an SSH client that is capable of X11 protocol. general-purpose computers can be used, and here we demonstrate one way for Windows and one for Mac users. The client commands for Linux are similar to those on the Mac.

This tutorial explains how to connect to one of the UCI EECS servers using PC or Mac computers.

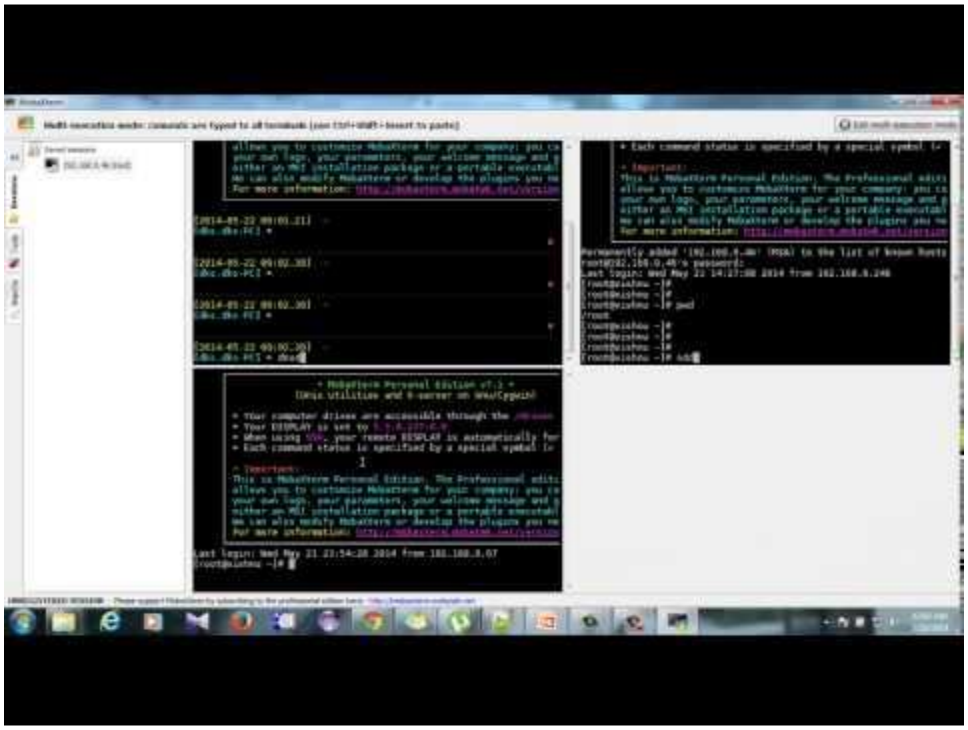


Connecting using PC computer (Microsoft Windows OS)

Microsoft Windows PC need to download a SSH client software to connect to EECS servers. There are multiple SSH clients available, MobaXterm is a good SSH client. The free version is more than enough for our course.
MobaXterm can be downloaded at the following web address:

http://mobaxterm.mobatek.net

Once you get to the web page on your browser, please go to Download tab and download the free version. Here you are given an option to download either a portable edition or an installer edition. Our recommendation is to download an installer edition. Students are more than welcome to download the portable version (Portable version appears to have bugs and might crash once in a while).
Portable version doesn't need installation; you just have to download a zip file, un-compress it and run the application. However if you choose to download the installer edition, please run the setup file and follow all the steps to install.



Connecting using Mac computer

If you just need the command-line interface, just open Terminal.app and type ssh username@hostname to login. To enable the use of graphical user interface, you can download and install **XQuartz** (X11 in older versions). Open this program and type the following command into the local shell:

```
 ssh  -X -Y  UserName@ServerName
```

If you plan to do editing on your local computer, it is also important that you know how to transfer files between your Mac computer and the Server. You can use a GUI version such as FileZilla, or you can use the following command in your local shell to transfer files:

```
 scp -r source-path destination-path
```

Some examples here would help you understand how scp works:

Example-1: We want to copy a file called *cpu.sv* from our computer to my location (112L) in the *zuma* server.

```
  scp -r  cpu.sv  username@zuma.eecs.uci.edu:~/112L/
```

Note-1: That colon (:) symbol after server name indicates the destination directory path in that server.

Note-2: In Linux Bash, tilde (~) by default will be expanded as your home directory path, so you do not need to remember the physical location of your home directory in that machine.

Once you're logged in, type the following command and see if you can see the Machine name you're on:

```
whoami

uname -a

date

lscpu
```

Now before going forward, answer the following question.

Question 1

Have you been able to connect to the Linux machine provided for you?

a

Yes

b

No

II) Which Text Editor to use?

There are various type of text editors for Linux, either command-based or graphical. Here are some of the famous ones:

- vim
- emacs
- nano
- gvim
- nedit
- gedit
- kyle

There are also some editors that let you connect to your remote file location and edit from your computer, like Notepad++. Here we show you how to work with vim in Linux and how to use Notepad++ to connect remotely to the server from a Windows machine.

VIM

If you are a UNIX user, Vim is a highly configurable text editor built to make creating and changing any kind of text very efficient. It is included as "vi" with most UNIX systems and with Apple's macOS [1].

Vim is rock stable and is continuously being developed to become even better. Among its features are:

- persistent, multi-level undo tree
- extensive plugin system
- supports Verilog and VHDL programming languages and file formats
- powerful search and replace

Look for **.vimrc** file on the internet to find useful default settings.

Here is a recommended .vimrc file content:

" " Do not use Tabs. Replace Tabs with spaces
set expandtab

" Number of spaces per tab key hit
" 1 tab == 2 spaces
set shiftwidth=2
set tabstop=2

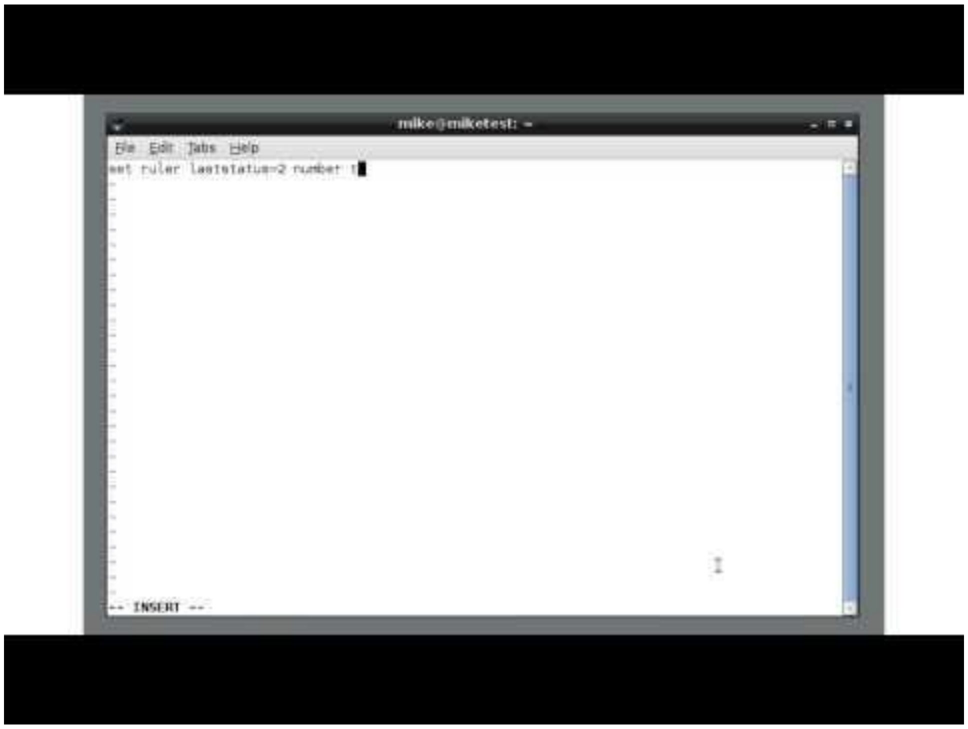
" It's recommended not to write length codes in a line
" Set the Linebreak to 450 characters
set lbr
set tw=450

" Turn on Auto indentation
set ai

" Turn on Wraping of the lines
set wrap

autocmd BufNewFile,BufRead *.sv   set syntax=verilog

autocmd BufNewFile,BufRead *.svi   set syntax=verilog



```
set ruler laststatus=2 number t
-- INSERT --
```

Notepad++

Notepad++ is a free and powerful, open source code editor which runs in the MS Windows environment. You can download the Notepad++ from the following link. Install the 32-bit version.

https://notepad-plus-plus.org/download/v7.4.2.html

Notepad++ installs on Windows OS and then you can install NppFTP plugin to be able to connect to Linux server. The following video tutorial demonstrates how to install NppFTP. plugin

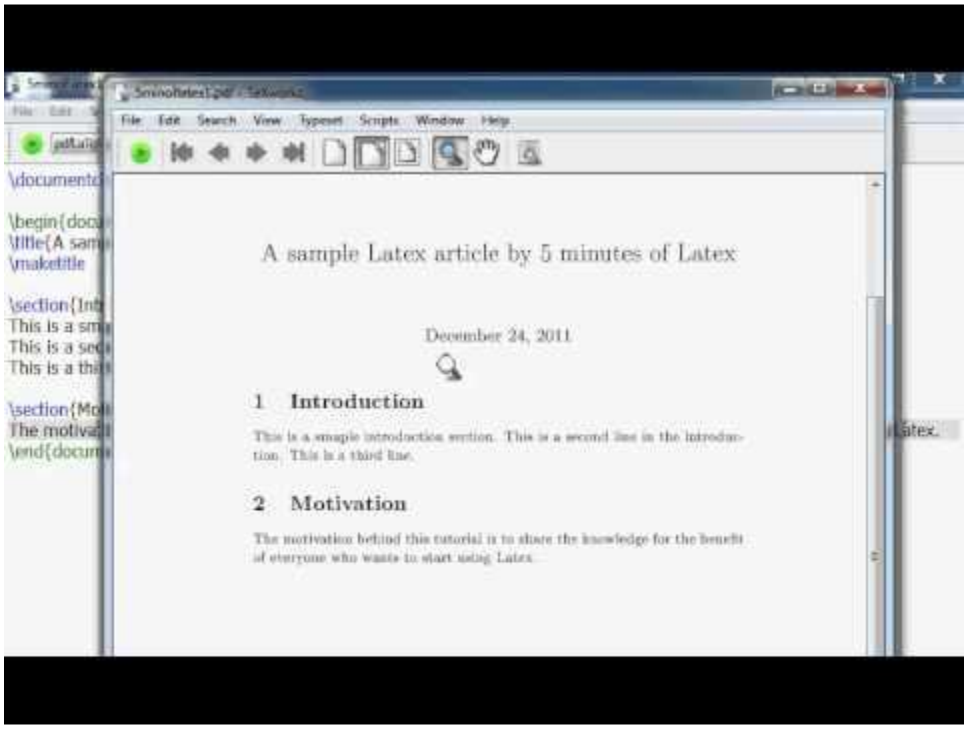
Question 2
No correct answers: No correct answer has been set for this question

Which text editor will you be using during this course?

Part 2: Documentation using LaTeX

As its official website says, **LaTeX**, which is pronounced «Lah-tech» or «Lay-tech» (to rhyme with «blech» or «Bertolt Brecht»), is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing. Your reports would look much nicer and professional if you prepare them using LaTex. You can install it on Windows, Linux, or MAC. Most of the LaTex IDEs are free [2].

And here is a quick tutorial on how to create your first Latex manuscript:



```
\documentc
\begin{docu
\title{A sam
\maketitle
\section{Int
This is a sm
This is a sec
This is a thi
\section{Mo
The motiva
\end{docum
```

A sample Latex article by 5 minutes of Latex

December 24, 2011

1 Introduction

This is a smaple introduction section. This is a second line in the introduction. This is a third line.

2 Motivation

The motivation behind this tutorial is to share the knowledge for the benefit of everyone who wants to start using Latex.

- Introductory tutorial on timing diagrams: The tikz-timing Package
- Drawing State Diagrams: Drawing graphs with dot

Latex Setup

Have been able to generate a pdf document using LaTex?

a

Yes

b

No

Git

Git is a distributed revision control system which is fast with a rich set of commands.

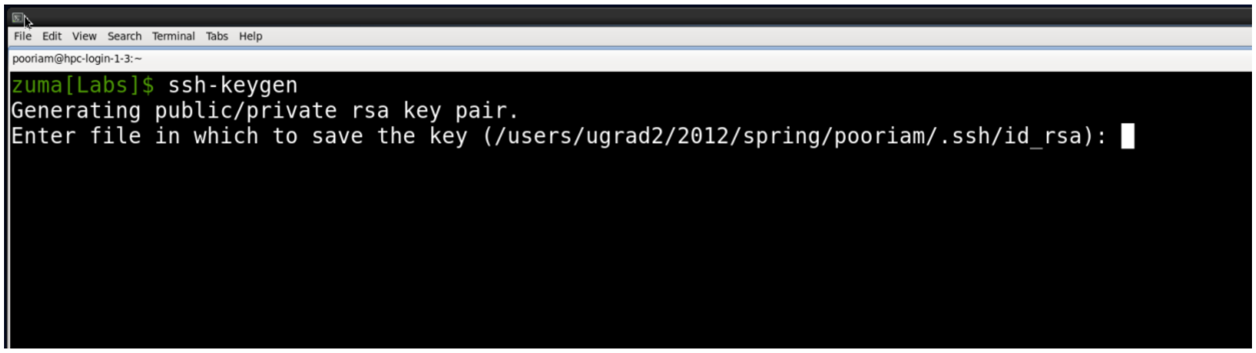
Revision control, also known as **version control** or **source control**, is the process of controlling changes made to files over time. These changes can be to source code project assets or any other files that go into the finished product. It allows multiple collaborators to work on the same portions of a project without worrying that their changes will corrupt or overwrite the changes of other collaborators. The collection of revisions and their side information is called a **repository** or **repo**. The repository represents a step-by-step chronological record of every change made to help project managers revert all or part of the project to a previous state if necessary [3].

You can find a full list of Git commands in the following link:

https://git-scm.com/docs

Step-1: SSH Key

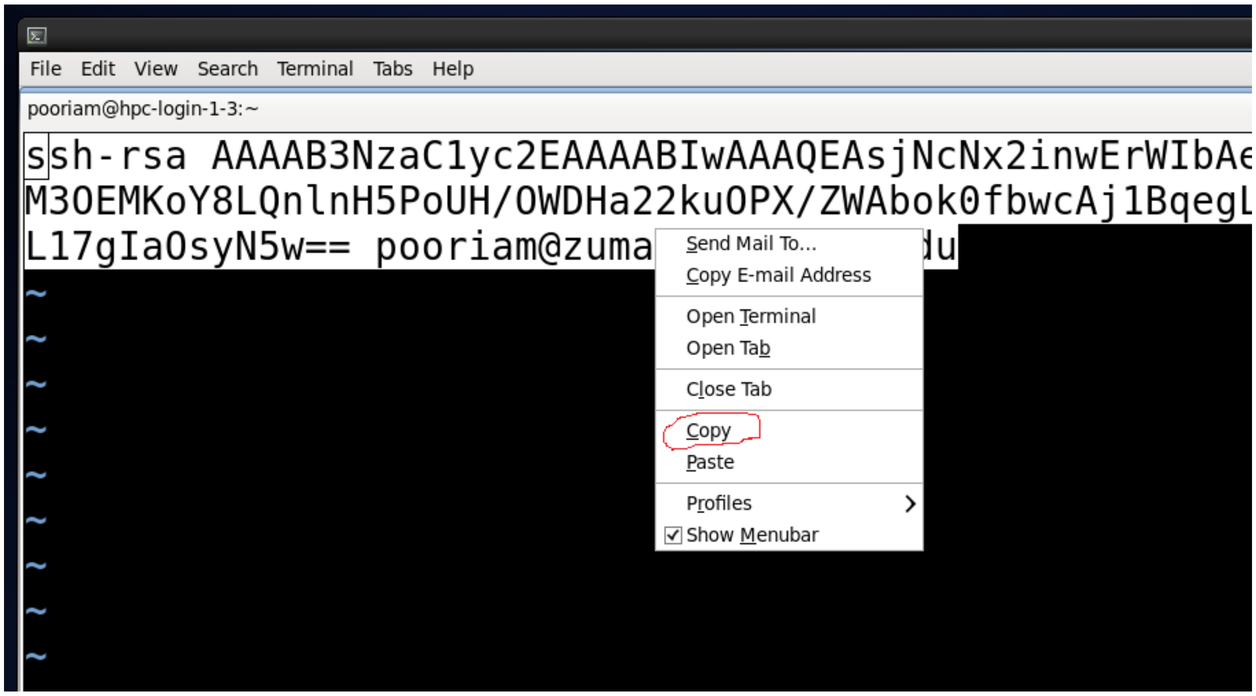
Before starting with Git, there are some steps you need to take to be able to connect securely to GitHub. You need to create a ***rsa key*** and register that on GitHub website. For this purpose, connect to the server (zuma) machine, and type the command "***ssh-keygen***". Then hit Enter until the command finishes.



```
zuma[Labs]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/users/ugrad2/2012/spring/pooriam/.ssh/id_rsa):
```

Then, open the generated public rsa key file and copy the content:

vim ~/.ssh/id_rsa.pub



```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsjNcNx2inwErWIbAe
M3OEMKoY8LQnlnH5PoUH/OWDHa22kuOPX/ZWAbok0fbwcAj1BqegL
L17gIa0syN5w== pooriam@zuma                          du
~
~
~
~
~
~
~
~
~
```

Now, you need to add the copied public rsa key into your GitHub account. Open your GitHub account and then click on your photo (upper right corner), and click on Settings. Now click on SSH and GPG keys. On the SSH keys pane, click on new SSH key, give it a title and paste the copied rsa key and click on green Add button.

Now it's time to start "Collaboration" using Git.

Step-2: Configuration

It is better to set some configuration for easier and more efficient use of Git capabilities. Introduce yourself to Git with your name and public email address before doing any operation. The easiest way to do so is:

```
git config --global user.name "Peter Anteater"

git config --global user.email  panteater@uci.edu
```

Step-3: Clone the repository

Now it's the time to clone the repository and start working. Type the following command in the terminal:

```
git clone git@github.uci.edu:112L/Labs.git
```

Now **Labs** directory is pulled out of the repository and you can use the files inside. You can create your own repository and with your teammates work on the shared code base.

Git Setup

Have you been able to access and clone the git repository?

a

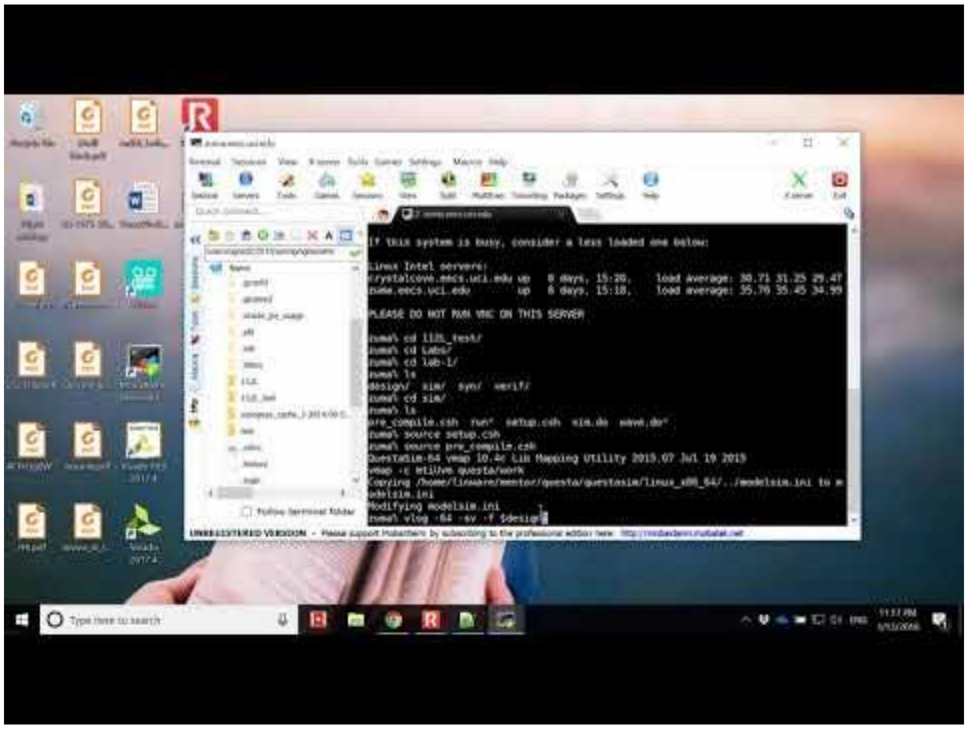
Yes

b

No

Part 3: MentorGraphics QuestaSim

Simulations in Mentor Graphics QuestaSim. In this tutorial, you will learn to compile, optimize and simulate the design. The design and the testbench are written using SystemVerilog. There are steps to simulate these codes.



```
If this system is busy, consider a less loaded one below:

Linux Intel servers:
crystalcove.eecs.uci.edu up   8 days, 15:20,    load average: 30.71 31.25 29.47
zuma.eecs.uci.edu        up   8 days, 15:18,    load average: 35.76 35.45 34.99

PLEASE DO NOT RUN VNC ON THIS SERVER

zuma% cd 115L_test/
zuma% cd Labs/
zuma% cd lab-1/
zuma% ls
design/  sim/  syn/  verif/
zuma% cd sim/
zuma% ls
pre_compile.csh  run*  setup.csh  sim.do  wave.do*
zuma% source setup.csh
zuma% source pre_compile.csh
QuestaSim-64 vmap 10.4c Lib Mapping Utility 2015.07 Jul 19 2015
vmap -c mtiUvm questa/work
Copying /home/linware/mentor/questa/questasim/linux_x86_64/../modelsim.ini to m
odelsim.ini
Modifying modelsim.ini
zuma% vlog -64 -sv -f $design
```

The following is also another simple example which helps you understand better how each command and the corresponding argument works.

How to Simulate a design and its testbench. The steps are explained by taking the example of the design file and testbench. full_adder_4bit.sv is a SystemVerilog code of a 4-bit fulladder and full_adder_4bit_tb.sv is the testbench. The same files are used to explain the steps.

Design the SystemVerilog code for the 32 bit ALU We are considering the SystemVerilog full_adder_4bit.sv file as an example. We could also do different design and play with the language.

```
 module   full_adder_4bit (
   carry_in,
   data_a,
   data_b,
   sum,
   carry_out

  );

  parameter   reg_size = 4;

  input   logic               carry_in;
  input   logic [reg_size-1:0] data_a;
  input   logic [reg_size-1:0] data_b;
  output  logic [reg_size-1:0] sum;
  output  logic                carry_out;

  assign {carry_out,sum} = data_a + data_b + carry_in;

 endmodule
```

a. Open the notepad++ tool.
b. Design using SystemVerilog code.
c. Save the file using **.sv** format. (Example: full_adder_4bit.sv)
d. Upload the file to your account in the server using MobaXterm

Now, design the testbench using SystemVerilog. We are considering the full_adder_4bit_tb.sv as an example.

```
 module full_adder_4bit_tb;

  logic       cin ;
  logic [4-1:0] in_a;
  logic [4-1:0] in_b;
  logic [4-1:0] sum ;
  logic       cout;


  full_adder_4bit dut
  (
     .carry_in  (cin ),
     .data_a    (in_a),
     .data_b    (in_b),
     .sum       (sum ),
     .carry_out (cout)
  );

  initial begin
     #10;
     cin  <= 1'b1;
     in_a <= 4'b0001;
     in_b <= 4'b0011;
     #10;
  end

endmodule
```

a. Now, design the testbench using the SystemVerilog code using notepad++.
b. Save the file using .sv format. (Example: full_adder_4bit_tb.sv)
c. Upload the file to your account in the server using MobaXterm

Copy the below mentioned files to your account on the server. Make few modifications to these files before uploading them to your online account on the server.

Now upload all the below mentioned files to the same location on the server where the design and testbench were uploaded before.

The following files exist in the Labs repository under questa directory. Open the README file in that directory first, read it thoroughly and then you understand the following steps better.

1. pre_compile.csh

2. setup.csh

3. rtl.cfg

4. tb.cfg

5. sim.do

Go to the folder location in your online account in the server, where the files are uploaded. Use the following Linux commands on the server to do the compilation, optimization, simulation and to view the waveform.

Note: All these Linux commands should be executed in the folder location where these files are uploaded to the server.

1. source setup.csh

This source command will load the file into the command prompt. It reads and executes the commands from the file setup.csh

2. source pre_compile.csh

This source command will load the file into the command prompt. It reads and executes the commands from the file pre_compile.csh

3. Compile the System Verilog design file: full_adder_4bit.sv

Compilation is the process of reading in SystemVerilog source code, decrypting encrypted code, and analyzing the source code for syntax and semantic errors. Implementations may execute compilation in one or more passes.

Run **vlog -64 -sv -f rtl.cfg** on the command line.

Here, vlog is a command which compiles the Verilog source code/SystemVerilog extensions (Here it is full_adder_4bit.sv)

-64 represents that vlog uses 64-bit executable

-sv is used to enable the SystemVerilog features and keywords

-f specifies the argument file with command lines arguments, which allows to use the complex arguments once again without retyping.

rtl.cfg is the file which has the name of the SystemVerilog file: full_adder_4bit.sv

4. Compile the System Verilog file: full_adder_4bit_tb.sv

Run **vlog -64 -sv -f tb.cfg -work work** on the command line.
Here, vlog is a command which compiles the Verilog source code/SystemVerilog extensions (Here it is full_adder_4bit_tb.sv)
-64 represents that vlog uses 64-bit executable
-sv is used to enable the SystemVerilog features and keywords
-f specifies the argument file tb.cfg , which allows to use the complex arguments to be used once again without retyping.
tb.cfg is the file which has the name of the SystemVerilog testbench: full_adder_4bit_tb.sv
-work work. The first work is a command option which specifies a logical name or path name of a library that is to be mapped to the logical library work. Here the name of this specified library is work.

5. Optimize (Elaboarte) the SystemVerilog design: full_adder_4bit.sv

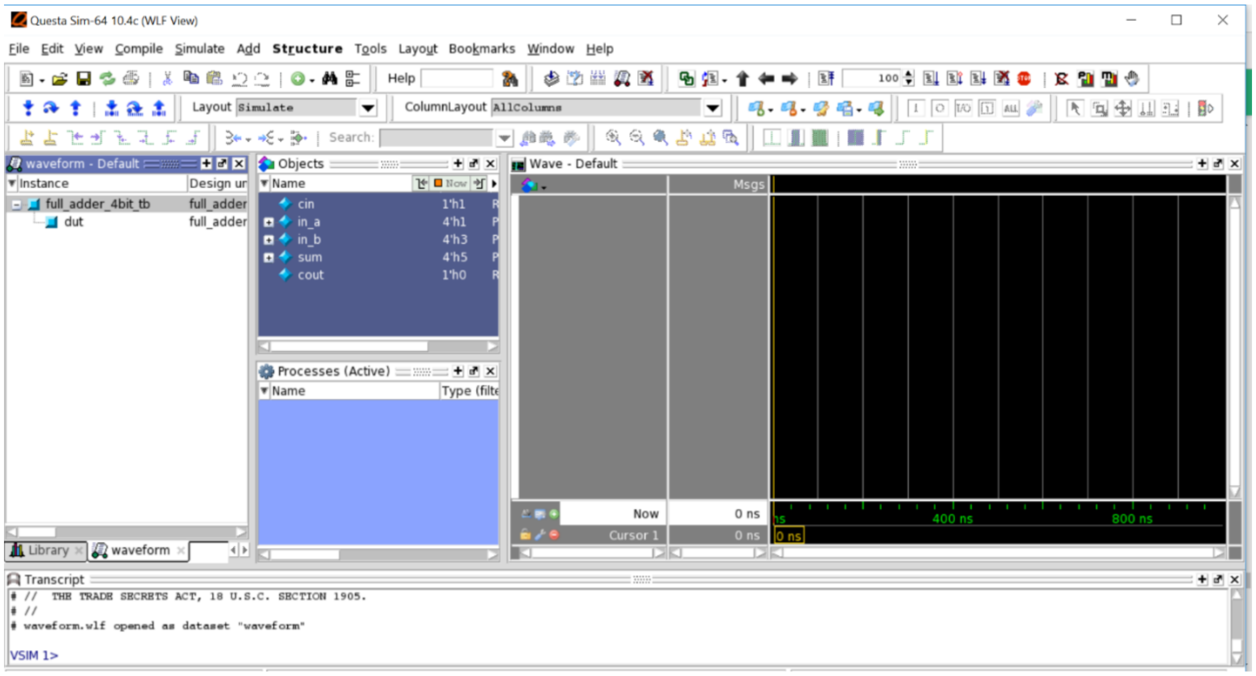
Elaboration is the process of binding together the components that make up a design. (in software terms, elaboration is like "linking" object files that have been compiled separately) These components can include module instances, program instances, interface instances, checker instances, primitive instances, and the top level of the design hierarchy. Elaboration occurs after parsing the source code and before simulation; and it involves expanding instantiations, computing parameter values, resolving hierarchical names, establishing net connectivity and in general preparing the design for simulation [4].

Run **vopt -64 full_adder_4bit_tb -o tb_top_opt +acc -work work** on the command line
Here, vopt is used to do the global optimization on the design after the compilation has been done using vcom (for VHDL design) or vlog -64 represents that vopt uses 64-bit executable
-o allows us to designate the name of the optimized design file
+acc provides visibility into the design for debugging purposes
-work work, the first work is a command option which specifies a logical name or path name of a library that is to be mapped to the logical library work. Here the name of this specified library is work.

6. Simulate the Design: full_adder_4bit.sv using the testbench: full_adder_4bit_tb.sv
Run **vsim -64 -c tb_top_opt -do sim.do** on the command line.
Here, vsim command is used to simulate the VHDL design or an entity/architecture pair or a verilog module/systemVerilog extension or an optimized design.
-64 represents that vsim uses 64-bit executable
-c specifies that the simulator will run in command-line mode

tb_top_opt this is the name of optimized design which we want to simulate.
-do sim.do tells the vsim to use the commands specified in the do file.

7. View the Waveform
Run **vsim -64 -gui -view waveform.wlf**
Here, vsim is also used to view the results of the previous simulation run when
-view switch is invoked
waveform.wlf is the simulation file. It is opened to view the waveform.
After running the command, a window will pop up. That window is shown below



Instance	Design unit
full_adder_4bit_tb	full_adder
dut	full_adder

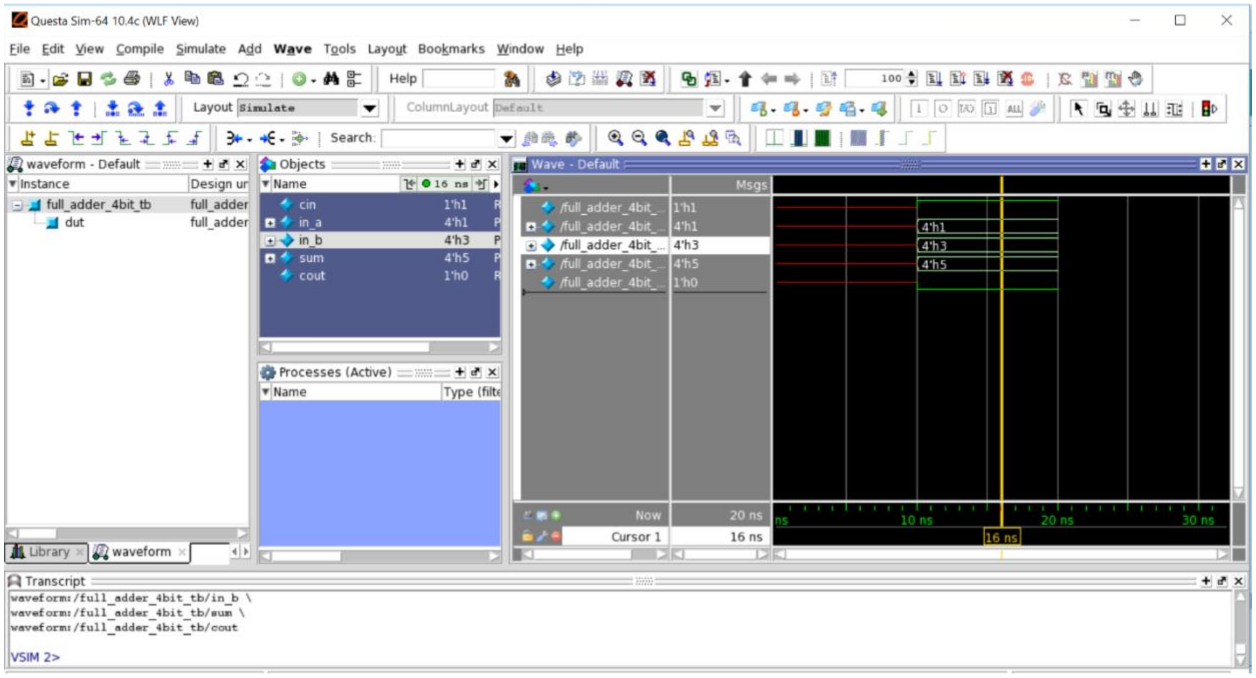
Name	Now
cin	1'h1
in_a	4'h1
in_b	4'h3
sum	4'h5
cout	1'h0

```
Now        0 ns
Cursor 1   0 ns
```

```
# //  THE TRADE SECRETS ACT, 18 U.S.C. SECTION 1905.
# //
# waveform.wlf opened as dataset "waveform"

VSIM 1>
```

8. Step to add waveform in the simulation window Use the mouse and right click on the full_adder_4bit_tb in the waveform default tab. Select Add to/Wave /all items in design. Simulated waveform should appear on the right side of the screen as shown below. You may right click on the waveform window and choose "Zoom Full" to fit the waveform.



```
waveform:/full_adder_4bit_tb/in_b \
waveform:/full_adder_4bit_tb/sum \
waveform:/full_adder_4bit_tb/cout

VSIM 2>
```

Check the waveform based on the desired functionality. If the waveform does not follow the desired functionality, then make the changes in the design or testbench.

We have made an intentional mistake in the testbench. Find it and fix it :)

If you get the desired results and functionality, then your design is good enough to be proud of your efforts!

Exercise

In order to review the digital design basics and syntax, it is highly recommended that you try to implement the following Digital Blocks in a parametrized way.

- Multiplexer
- Decoder

- Encoder
- Comparator
- Half Adder
- Full Adder
- Adder/Subtractor
- Sign extender
- Shift Register
- Complementer

Logic Design

Which of the following listed digital design block can be a sequential circuit?

a

Sign Extender

b

Shifter

c

Adder/Subtractor

d

Multiplexer

Abbreviations

References

[1] vim.sourceforge.io

[2] www.latex-project.org

[3] https://git-scm.com/

[4] IEEE Approved Draft Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in *IEEE P1800/D4a, July 2017* , vol., no., pp.1-1317, Jan. 1 2017