ELC 2137 LaTeX Tutorial

John Miller

January 16, 2020

Overview

TeX is a computer language for typesetting or formatting documents created by Donald Knuth in the 1970s. It was expanded by Leslie Lamport in the 1980s, and others since then, to create LaTeX (pronounced "lay-tek" or "lah-tek"). TeX is actually Greek "tau epsilon chi" $\tau\epsilon\chi$, related to our word "technology." For more, see Wikipedia: LaTeX or History of TeX.

It is designed for the author to focus more on the content and less on the formatting, allowing the compiler to take care of it. However, this can be difficult for people who have grown up with WYSIWYG (what you see is what you get) editors like Microsoft Word and PowerPoint.

Recommendation: Work *with* the tools to create great reports, not *against* them.

Getting Started

The ECS lab computers already have MikTeX (compiler) and TeXstudio (editor) installed. MikTeX comes with its own editor (TeXworks), but it's very basic and not as easy to use as TeXstudio. If you want to install them on your personal computer, here are the links:

- `https://miktex.org/`
- `http://texstudio.sourceforge.net/`

If you get stuck, you can find the answer to almost any question by searching the internet for "latex {your question}". For example, try "latex figures". Here are some good references if you need them:

- `https://www.latex-tutorial.com/tutorials/`
- `https://en.wikibooks.org/wiki/LaTeX`

Sections

You'll notice that this document is broken into sections with headings. This is accomplished with the commands `\section*{}, \subsection*{}, \subsubsection*{}`. The heading text is placed inside the braces. Without the '*', the headings have numbers, like "1.4 Title".

Text Formatting & Environments

Text can be formatted with some simple commands, such as:

- `\textbf{bold}` = **bold**
- `\textit{italic}` = *italic*
- `\texttt{typewriter}` = `typewriter` (mono-spaced)
- `\_` and `\&` and `\#` = _ (underscore) and & and #

Lists

The bulleted list above is an `itemize` environment. Similar, the `enumerate` environment creates a numbered list, like:

1. Here is an item.
2. Here is another.
3. The last one.

You can also nest lists, like this:

1. Colors
    (a) Red
    (b) Green
    (c) Blue
2. People
3. Places

or this:

- Colors
- People
- Places
    - Waco
        - ∗ Rogers ECS Building
        - ∗ McLane Stadium
    - DFW
    - Houston

Center Align

Another useful environment is `center`. This aligns whatever content is inside of it in the center of the body.

This sentence is centered.

Tables and Figures - Not floating

Tables and images (figures) are typically “floating” objects, which is explained in the next section. However, they can be included directly with just a few commands. The advantage of this method is you get them exactly where you want them.

We use the `tabular` environment to define a table:

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

If you want it centered, you can put it inside a `center` environment:

Binary	Hex	Decimal
0	0	0
0	1	0
1	0	0
1	1	1

You can include an image using the `\includegraphics` command:

You can even put it in the middle of a sentence if you want. The `\includegraphics` command is explained in detail below.

If you need to include a full-page PDF or something of that sort, this might be a good way to do it:

2N2222

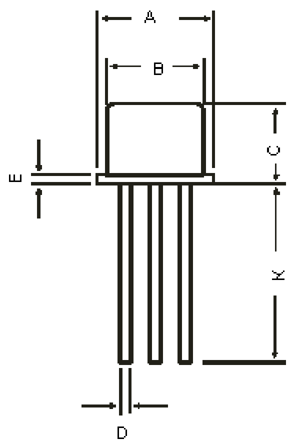
Low Power Bipolar Transistors

multicomp

Features:

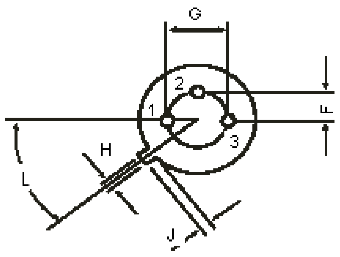
- NPN Silicon Planar Switching Transistors.
- Switching and Linear application DC and VHF Amplifier applications.

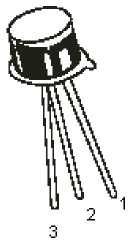
TO-18 Metal Can Package



Dimensions	Minimum	Maximum
A	5.24	5.84
B	4.52	4.97
C	4.31	5.33
D	0.40	0.53
E	-	0.76
F	-	1.27
G	-	2.97
H	0.91	1.17
J	0.71	1.21
K	12.70	-
L	45°	

Dimensions : Millimetres





Pin Configuration:
1. Emitter
2. Base
3. Collector

Tables and Figures - Floating

Tables and figures are intended to be "floating" objects, meaning LaTeX will move them around to make them best fit the flow of the document. You may not think so, but there is a method to what might seem like madness. One of the main reasons to use these environments is that you can provide a caption and reference them in the body, as in:

Table 1 is an example of a truth table. The full logo is shown in Figure 1, while Figure 2 shows how to crop it.

Table 1: Example of a floating table

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

Captions are important for figures and tables. Typically, they should be above a table and below a figure. Labels provide a way to reference a particular figure or table. Note: The label must come **after** the caption.

The `\includegraphics` command has lots of options. Here are a few that you might find useful: (note: `#` = a number you need to fill in)

Adjust size:	`height=#` (length)
	`width=#` (length)
	`scale=#` (0 to 1)
Rotate:	`angle=#` (degrees)
Cropping:	`trim=# # # #` (order is: left bottom right top)
	`clip` (this removes what you just trimmed)
Choose page of a pdf:	`page=#`

(Note: The above list is actually a `tabular` environment, just without any border lines.)

Here are a few examples:

Figure 1 uses the command: `\includegraphics[width=0.5\textwidth]{BUlogo}`. The `width` option scales the whole image to be half of the width of a line of text (using the built-in length command `\textwidth`).

```
\begin{figure}[ht]\centering
    \includegraphics[width=0.5\textwidth]{BUlogo}
    \caption{This is the original logo.}
    \label{fig:original_logo}
\end{figure}
```

This sentence is supposed to be between the figures.

This paragraph comes after both figures. If there is enough text here, LaTeX will put it on the same page as the figures, pushing them to the top. More text. More text. More text. More text.

Figure 1: This is the original logo.

Figure 2: This is the logo cropping the top and bottom.

More text. More text. More text. More text. More text. More text. More text. More text. More text.

You can use the `\clearpage` command to force all floats that have been defined so far to be drawn and then start the next line on a new page. See an example by uncommenting the command in the .tex file of this document.

Code

There is a LaTeX add-on package called "listings" that allows you to easily include code from lots of different languages, including Verilog. At the top of this .tex document, you'll notice some commands for setting up the listings package and "macros" or user-defined commands that make it even easier. There are two ways to include code. You will probably want to use the first one, which reads directly from your code file.

File Inclusion

Listing 1: File-included Verilog code example

```
module example_or(
   input a, b,
   output c
   );

   // c = a OR b
   or(c,a,b);
endmodule
```

You can also include only specific lines.

Listing 2: Selected lines from example file

```
   // c = a OR b
   or(c,a,b);
```

The only downside to this is that you have to tell LaTeX where your file is relative to the location of your LaTeX file. For example, suppose you have a file structure like this:

```
BaseFolder
    Folder1
        report.tex        <-- LaTeX file
        code1.sv
        SubfolderA
            figureA.png
            codeA.sv
        SubfolderB
    Folder2
        figure2.jpg
        code2.sv
```

To include `code1.sv`, which is in the same folder as the LaTeX file, you would use the command:

```
\Verilog{code1.sv}
```

To include `codeA.sv`, you would use the command:

```
\Verilog{SubfolderA/codeA.sv}
```

To include `code2.sv`, you would use the command:

```
\Verilog{../Folder2/code2.sv}
```

This one is a little trickier to understand. The "`..`" tells the compiler to go "up" or "out" one folder level. Thus, it starts at `BaseFolder` and then follows the remaining steps (`BaseFolder -> Folder2 -> code2.sv`).

Note: You can do multiple of these, as in `../../../folder/folder/file`, but it gets confusing pretty fast. This is one reason why we require you to use the folder structure we do.

Direct Inclusion

The other way to include code is to copy and paste it directly into the LaTeX document inside of a `lstlisting` (pronounced "list-listing") environment, like this:

Listing 3: Direct Verilog code example

```
module example_and(
   input a, b,
   output c
   );

   // c = a AND b
   and(c,a,b);
endmodule
```

Notice that the format of the output (in the PDF) is exactly the same.

Equations

We will not write a lot of equations in 2137. It is included here for completeness and because this is another thing at which LaTeX excels. There are different equation environments. I'll only cover the most basic one here, which is the "in-line" version that uses single dollar signs, like this:

The equation for electrical power is $P = VI$. When delivering power to a resistor, where $V = RI$, this can written as $P = VI = RI^2 = \frac{V^2}{R}$.

For more, see

- `https://www.latex-tutorial.com/tutorials/amsmath/`
- `https://en.wikibooks.org/wiki/LaTeX/Mathematics`

Report Items

Report Format

Unless otherwise specified in the assignment, each report will have the same format: Summary, Results, and Code.

- In the Summary section, you should provide a brief overview of the purpose of the lab and summarize your results.
- In the Q&A section, answer any questions posed in the assignment. Alternatively, your instructor might want you to include these in your summary.
- The Results section will include your simulation waveforms, results tables, pictures of hardware, and any other required items.
- The Code section includes all of the files you wrote or modified as part of the lab. Provided files that you did not change do not need to be included here.

Q&A Formatting

One way to nicely format your responses to questions is the following:

1. How should I format them?

   Use the same numbered list format as in the assignment (like this).

2. Do I need to include the original question?

   Yes, include both the question and the answer to it.

3. Should I put the question on the first line?

   Yes, then skip a line and write your answer. This produces a nicely formatted result.

Simulation Results

You will include screen captures of simulation waveforms from Vivado. The easiest way to do this is to use the Windows Snipping Tool or Snip & Sketch. Use Rectangular mode and select only the relevant part of the simulation window, then save it as a PNG image. You will notice that on Figure 3 you can read the signal names on the left, see the time values across the top, and see all of the signal values up to the stop time of 40 ns.

Note: By default, the background in Vivado is black. If you were going to print your report, we would require you to change it (to save ink). But since we are going to grade everything electronically, it doesn't matter.

Expected Results Table

For each simulation you run, you need to create an "expected results table" (ERT). This table should show what values are expected for each input and output signal at each time step of the simulation. Always position your table above the simulation waveform, as done with Figure 3. Putting the `tabular` environment inside of the `figure` environment ensures that the table and waveform won't be separated by a page break. Note: you can use this example as a template.

Time (ns):	0	10	20	30
a	0	1	0	1
b	0	0	1	1
c	0	0	0	1
s	0	1	1	0

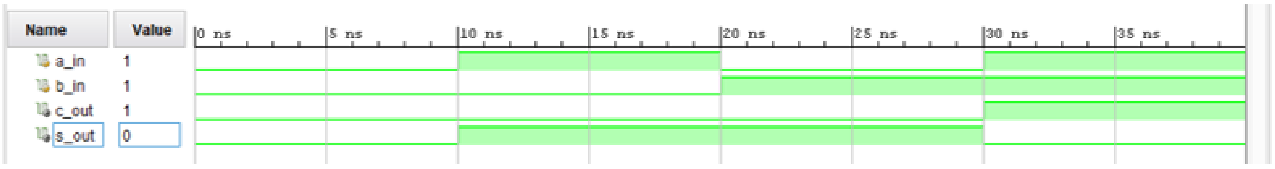


Figure 3: Example of a simulation waveform and ERT

TeXstudio Shortcuts

Some keyboard shortcuts you might find useful:

- F5 – compile to PDF
- Starting typing a command, then use arrows to choose the desired auto-complete command, and press Enter. For example, type `\V` and the first item in the pop-up list should be the `\Verilog` command.
- In tables, TeXstudio will highlight things in red when the number of columns specified doesn't match the number of columns you have in a particular row. Summary: When nothing is red, you have the right number everywhere!
- Text formatting, similar to MS Word:
  - Ctrl+b = **bold**
  - Ctrl+i = *italic*
  - Ctrl+Shift+t = `typewritter (monospaced)`
- Ctrl+Shift+i = insert "`\item `"
- Standard Windows shortcuts:
  - Ctrl+z = undo

- Ctrl+y = redo
- Ctrl+c = copy
- Ctrl+x = cut
- Ctrl+v = paste