Patrick Austin

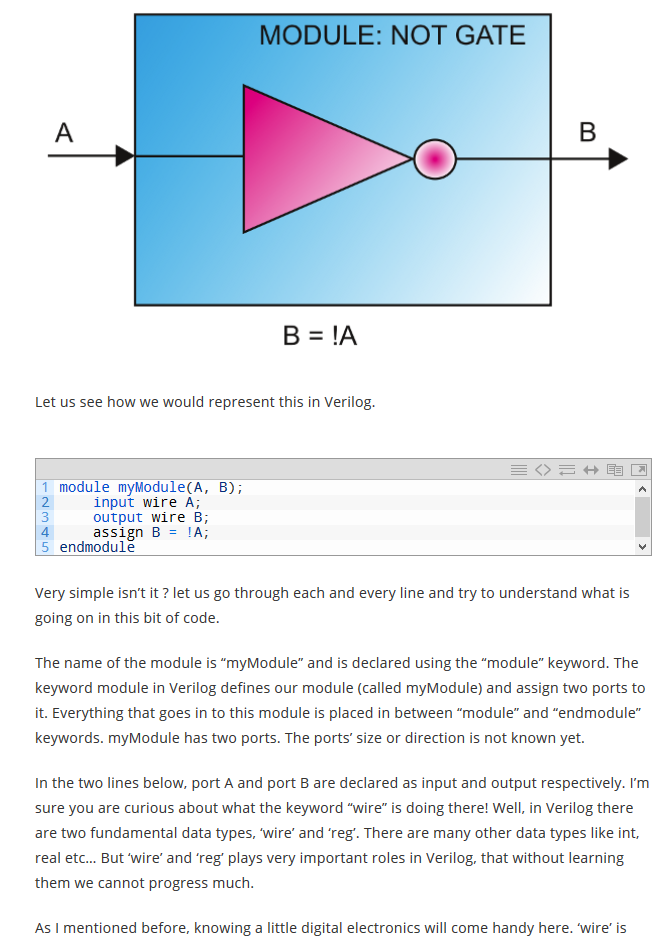
CPE 301 - 1104, Fall 2016

Homework 9

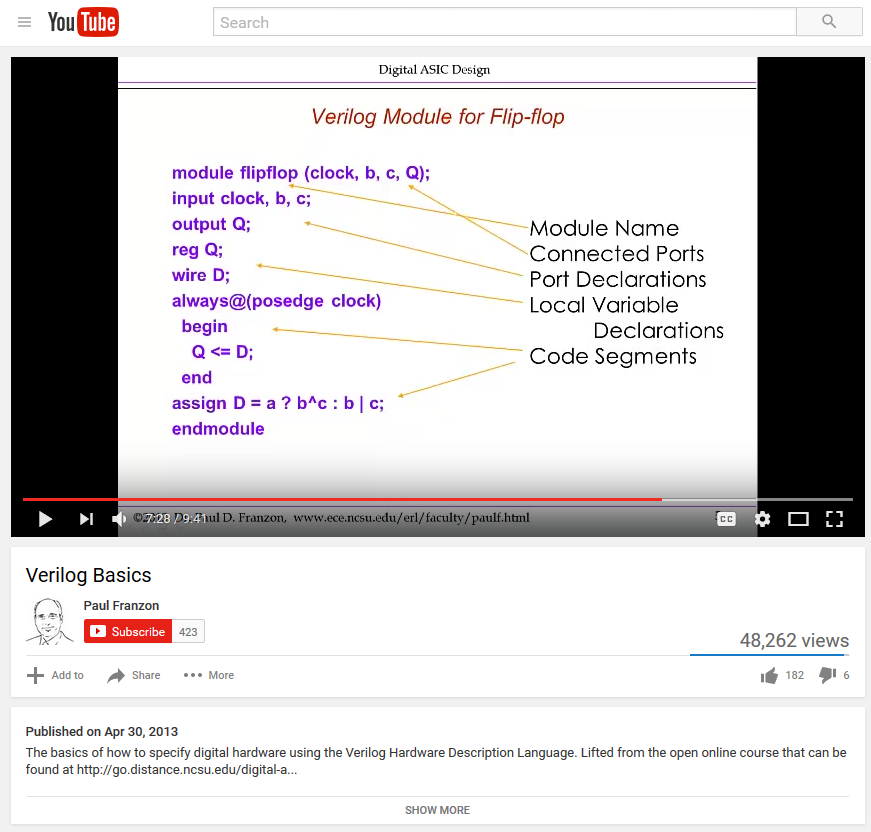
**Part 1.** 12/4/2016

Verilog tutorial 1: <https://docs.numato.com/kb/learning-fpga-verilog-beginners-guide-part-1-introduction/> and subsequent pages

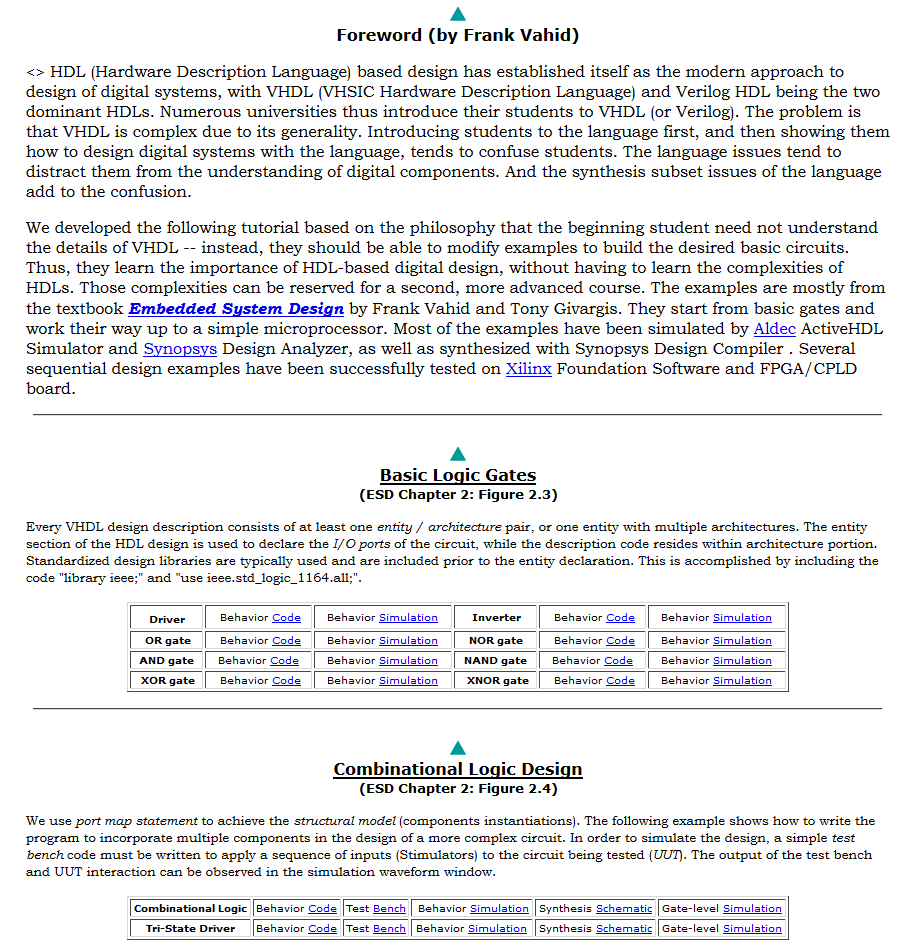
Representative page:



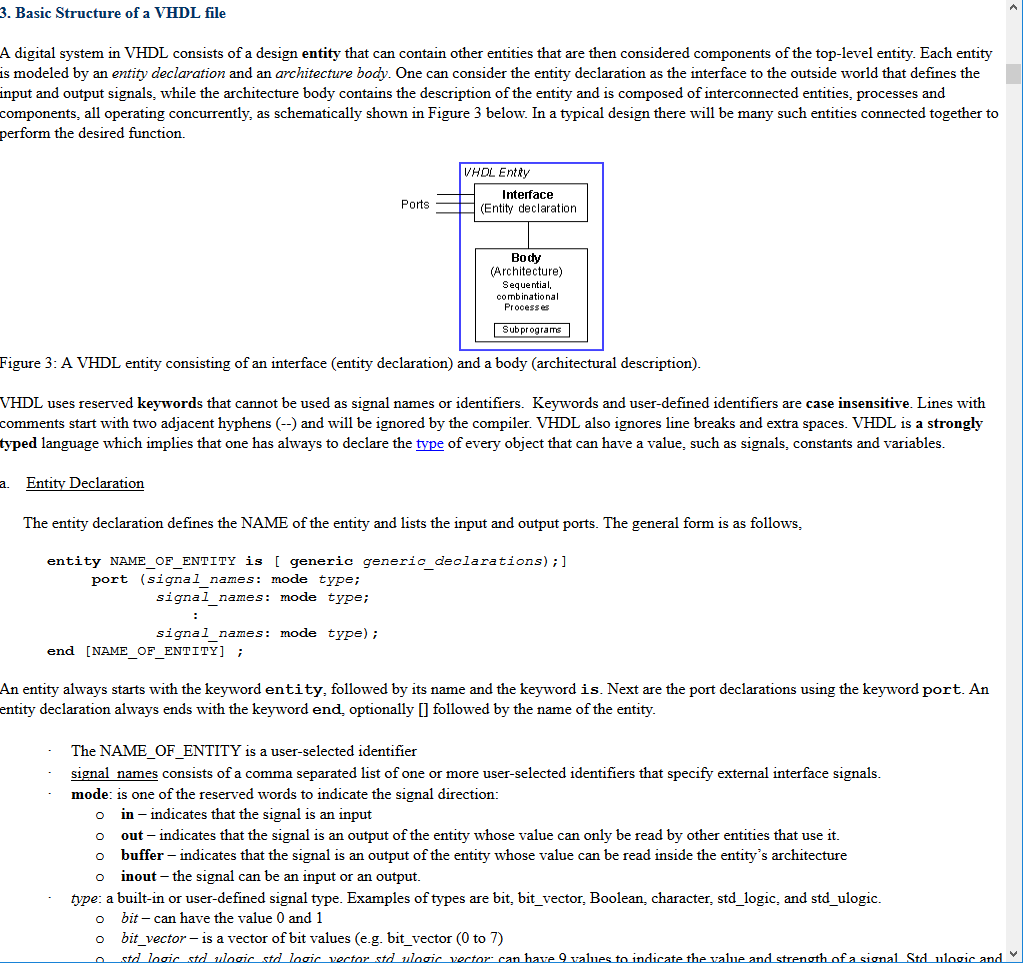
Verilog tutorial 2: <https://www.youtube.com/watch?v=vVYOV9MP5BA> (and subsequent videos on same channel)



VHDL tutorial 1: <http://esd.cs.ucr.edu/labs/tutorial/> , and linked code samples



VHDL tutorial 2: <https://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html>



**Part 2.**

In this essay I will briefly discuss what I learned from my web search on Verilog and VHDL. I will give a brief overview of history, similarities, and differences of the two languages, and then discuss some pros and cons of the tutorials linked in the previous section.

The website I am most indebted to with regards to comparing the two languages is located here: <http://www.angelfire.com/in/rajesh52/verilogvhdl.html>. Despite being an old website, this article features a useful point by point comparison of the languages, which I will draw on heavily in this discussion. At the highest level, VHDL and Verilog are both languages used to model the workings of a real-world circuit. One might use these languages to create a model of a circuit to test its function before building it, or because actually building such a complex circuit by hand would be too arduous. One may be using the VHDL or Verilog model as code to program an FPGA, which will map the function of the modeled circuit onto actual hardware by appropriately configuring the FPGA’s array of gates.

VHDL originated from a US Department of Defense project seeking to standardize the simulation and description of integrated circuits being used in military equipment. It was first standardized in 1987. Verilog was developed by a company called Gateway Design Systems in the mid-1980s to simulate the behavior of hardware, before being acquired by Cadence Design Systems in 1990, who put the language in the public domain. It was first standardized in 1995. Both have been subject to numerous standardized revisions and extensions in the years following.

As the languages are both designed to simulate real-world circuits, similarities should be expected. Both languages are similarly technically capable of modeling circuits, with baseline VHDL having slightly more power than baseline Verilog (Verilog comes in multiple flavors/forks/branches, which I won’t go into here). The choice of which to use comes down to the differences, which I will now discuss.

To generalize, the article above describes VHDL as a more complex yet more powerful language, with Verilog being less capable but easier to learn. One reason for this is data typing between the two languages. VHDL allows user-defined data types, and even the standardized default ones can be complicated- in the std\_logic type there are nine possible values, indicating different strength and values of a signal. Verilog, by contrast, uses predefined data types and users may not create their own. In the practice of the tutorials above, only the types wire and reg are used, and their behavior and interaction is mostly self-explanatory. Again, VHDL gives the user more power at the cost of more complexity.

Syntactically, Verilog is more like C and Verilog is more like Ada, though neither comparison is perfect. For C programmers who may not know Ada, like students in this class, Verilog might be a natural choice of personal preference due to readability for that reason alone. The strongly typed nature of VHDL, discussed above, presents another source of difficulty. VHDL code will be longer and more verbose than a Verilog counterpart, arguably being more precise and/or less readable.

VHDL has a more powerful suite of tools for high-level modeling, ie modeling a circuit at an algorithmic level rather than a gate-to-gate one. Verilog only gains such functionality when extended by what the article above calls the PLI, the programming language interface. In general, VHDL has more and better tools for managing a large project, like the ability to package procedures and functions for re-use, and the ability to create libraries.

As a kind of general conclusion, Verilog is less powerful, models at a level closer to the actual logic gates, and is easier to understand. VHDL is more powerful, is capable of modeling circuits on a more abstract level, and is more complex.

I will now discuss the tutorials for each language shown in Part 1.

The first tutorial goes quite slowly and starts at the very basics of Verilog. It has a significant apparent benefit of being up to date (dated at April 2016 as of my access), goes quite slowly and starts with very simple examples, and focuses on moving Verilog code to the actual FPGA board. Many other tutorials are quite old, or focus solely on the properties of simulating in the language itself without getting into the details of the FPGA interface process, which is obviously of significant interest for this class. If this tutorial has a disadvantage, it may be the rather informal and loose style of the writing, which does contain some typos.

The second Verilog tutorial comes in the form of a video series from an online class on ASIC design. The video format has upsides and downsides, but as a “modern” student I find it to be a valuable resource, albeit only one tool in one’s repertoire. The video is only a few years old, so it does represent a reasonably up to date resource. One downside of the video format is the lack of tangible example code that can be tested directly and used as a resource. Also, being a series of short videos, this series is mostly a high-level, conceptual fly-by tutorial, which is useful in its own way but is definitely not into the nitty-gritty.

The first VHDL tutorial has a primary downside of being quite old, published circa 2001. However, it has the benefit of focusing heavily on reusable example code snippets which are used to build more and more complicated circuits, which can be referred to and utilized to tangible effect without having to go into the weeds of VHDL, which as described above are significant. The information is nicely presented, in the form of comparable code and schematics for both high-level algorithmic simulations and low-level logic gate simulations. On the flipside, the ‘basics’ of the language aren’t presented in much depth. It should also be noted that this tutorial does not go into depth about the process of moving the code onto an actual FPGA, which is a bit of a downside as well.

The second VHDL tutorial contrasts nicely with the first. This quite involved tutorial goes into the nitty-gritty where the first one presented examples that floated above it. That makes this tutorial useful as a reference to the previous one, if nothing else. It treats the more complex VHDL system of data types in a lot of depth, talks about how to build libraries in VHDL, treats behavioral vs structural vs dataflow approaches to modeling a circuit in VHDL, etc. This is a level of complexity we probably don’t need for this class, but again it makes a useful resource. This article is, unfortunately, on the older side as well.

In the course of assembling this report I feel like I have picked up a good basic understanding of the two languages, their similarities and differences, and a suite of resources and references to explore these topics in more detail, or just to refresh my memory if and when FPGA programming becomes a useful tool.

List of References

Learning FPGA And Verilog: A Beginner’s Guide, <https://docs.numato.com/kb/learning-fpga-verilog-beginners-guide-part-1-introduction/>, (copied 12/4/2016)

Verilog Basics, Paul Franzon, Digital ASIC Design, <https://www.youtube.com/watch?v=vVYOV9MP5BA>, (copied 12/4/2016)

VHDL Tutorial: Learn by Example, UC Riverside, <http://esd.cs.ucr.edu/labs/tutorial/>, (copied 12/4/2016)

VHDL Primer, University of Pennsylvania, <https://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html>, (copied 12/4/2016)

VHDL and Verilog Modeled and Contrasted, <http://www.angelfire.com/in/rajesh52/verilogvhdl.html>, (copied 12/4/2016)