MyHDL vs. PyMTL: Two Python to HDL Packages [Midterm Update]

COMS E6998 - FPGAs Then and Now - Spring 2018

Samuel Beaulieu
srb2208@columbia.edu

ABSTRACT

With the end of Dennard scaling and the slowing of Moores law, engineers must turn to other resources to realize performance gains. These resources come in the form of FPGAs and ASICs. Both utilize specialized hardware, either re-configurable or static, to accelerate certain applications. Designing and implementing these accelerators, however, requires specialized knowledge of VHDL or Verilog. They are also time consuming to design, and challenging to integrate and test. High Level Synthesis offers a way to access the performance gains of FPGAs and ASICs quickly and without having to learn a new language. In this project, we propose to look at the feasibility of designing hardware in Python. Specifically, we compare two Python to Verilog packages, MyHDL and PyMTL, and analyze their feasibility in terms of ease of use, cost, and performance.

1. INTRODUCTION

With the end of Dennard scaling and the slowing of Moores law, engineers must turn to other resources to realize performance gains. These other resources come in the form of FPGAs and ASICs. FPGAs have become common in commercial data centers such as those owned by Microsoft, Amazon, and Intel. Similarly, ASICs have made their impact in Google's data centers and hardware accelerators. Regardless of the hardware choice, both FPGAs and ASICs require hardware design typically done in VHDL or Verilog. Not only does the increased demand in specialized hardware require more, experienced VHDL and Verilog designers, but it also constrains the delivery pipeline. The time it takes to develop, integrate, and verify new hardware is an extremely important metric in the success of a new product.

High Level Synthesis (HLS) offers a way to access the performance gains of FPGAs and ASICs quickly so developers can focus on the algorithms more than the timing and communication details. Instead of using Hardware Design Languages (HDLs), HLS converts an algorithmic description written in a standard programming language like C or Python to hardware. In this paper, we look at the feasibility of two such HLS tools to design hardware in Python. Specifically, we compare two packages, MyHDL and PyMTL, and analyze their feasibility in terms of ease of use, cost, and performance.

2. SPECIFICATION

In this project, we compare and contrast two hardware design packages for Python: MyHDL and PyMTL. We investigate their use in both academic and commercial settings, their support base, their difficulty to set up and use, their restrictions, their benefits, and their performance.

This analysis is based on taking a few Verilog programs of common kernels and re-building them using each of the python packages. We then use the packages to re-develop the programs in Verilog. This provides us a way to compare both python packages to each other as well as to native Verilog.

Once we have the programs in Verilog, we will synthesize them for FPGA using publicly available synthesis tools and and then simulate them. We intend to derive the maximum clock frequency and number of Look Up Tables (LUTs) using synthesis and the number of cycles from simulation. Throughout this process, we will dig into the user base for each Python package and gauge how well their support and documentation is as well as how challenging it was to get the design from Python onto the FPGA. We will also keep in mind any restrictions each package has such as only synthesizing a subset of the language. Finally, we will analyze the performance of each algorithm/package combination to determine which applications each package is better.

2.1 Current Progress

So far we have made progress on the background research and planning for the project. Specifically, we have done the following:

- Background research on development, cost, releases, documentation, and supposed performance for both python packages.
- Looked into support for both packages in terms of location, community size, and tutorials.
- Installed both packages and documented the installation flow and complexity.
- Converted a sample program (provided by the respective python packages) to Verilog for each of the python packages.
- Developed a partial list of Verilog programs to base analysis off of.
- Planned out a path forward for the rest of the semester.

2.2 Milestones

This section outlines the expected milestones for this project. At a general level, we will develop the synthesis and simulation flow and decide on which native Verilog programs to use. Then, we will port the Verilog to python using each of the python packages and compile them back to Verilog. Finally we will synthesize and simulate the native Verilog and the output of each of the python packages and compare results.

1. Install synthesis and simulation software (by Apr. 10)
2. Develop synthesis and simulation flow (by Apr. 13)
3. Convert Verilog programs to each of the python packages and use the packages to convert the programs to Verilog (by Apr. 20)

4. Run synthesis and simulation and compare results (Apr. 27)
5. Write report and finalize results ($\sim$ May 1-10)

2.3 Critical Aspects

A critical aspect of this project is successfully synthesizing and compiling the code for FPGA and then simulating it. Progress in the coming week will be crucial to the success of the project as a whole.

Project Git: https://github.com/saribe0/COMS-E6998-Comparing-MyHDL-and-PyMTL MyHDL: http://www.myhdl.org/start/overview.html PyMTL: https://github.com/cornell-brg/pymtl