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

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 require more, experienced VHDL and Verilog designers, but it also constraints 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 intend to compare and contrast two hardware design packages for Python: MyHDL and PyMTL. We aim to 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 project does not intend to compare the results of using these packages in comparison to native Verilog or VHDL. Instead, we will take the viewpoint of someone who only knows Python and would still like to develop a design and synthesize it on an FPGA.

To reach these goals, we will first compile a few algorithms in Python that range from being sequential to potentially parallel and compile them to Verilog using these packages. Next we will compile and install the programs on an FPGA and look at the performance benefit seen by each algorithm. 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 Milestones

This section outlines the expected milestones for this project. At a general level, we will, for each package, install the software and determine the flow for a small Python program. Next we will develop three more complex designs. One that is inherently sequential by nature, one that is parallel, and one that is somewhere in between. Finally, each program will be run on the FPGA and their performance results analyzed.

1. Install the software and convert a basic program to Verilog/VHDL (by Mar. 16)
2. Compile and run the program on an FPGA (by Mar. 23)
3. Develop the benchmark algorithms (by Apr. 6)
4. Use the packages to convert the programs to Verilog/VHDL and install on an FPGA (by Apr. 20)
5. Analyze performance and compile findings (Apr. 27)
6. Write report and finalize results ($\sim$ May 1-10)

2.2 Critical Aspects

A critical aspect of this project will be running the code on the FPGA to determine how well the packages actually did at generating the hardware descriptions. This will require a small FPGA and the software required to compile and run Verilog or VHDL on it. With no prior FPGA experience (at least in the last 4 years), this might be a challenge.

Another critical aspect is actually compiling and running the packages. From an initial search, these are the only two tools publicly released for Python to VHDL/Verilog. Both seem promising but if something goes significantly wrong, we may switch to C to VHDL/Verilog packages.

MyHDL: http://www.myhdl.org/start/overview.html
PyMTL: https://github.com/cornell-brg/pymtl