Python FPGA Development
MYHDL.ORG (Where have you been!)

Python EcoSystem!!

Abstract

Introduction
MyHDL is an open source Python package that lets you go from Python to silicon. With MyHDL, you create a Python model of the hardware system, which you can then simulate, and validate, export to Verilog or VHDL, and take it to a FPGA.

Why
The key advantage of doing hardware design in Python with MyHDL is that it manages your complexity far better than the alternatives. You now have a single language for designing, simulating, testing, and scripting. This is a modern high level language that allows you to abstract the details, and focus on the design. This is particularly useful for beginners, but is even more important for experienced designers as their designs get more and more complex. It is especially useful if you want to move your design between Verilog and VHDL.

How
Please visit our website MyHDL.org. Download the free open source python and MyHDL packages. Join the mailing list. Sign up to edit the wiki. Or better yet, sign up for a workshop.

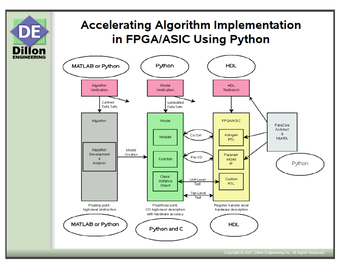
The MyHDL Class
Please sign up for our class if you are interested in an upcoming workshop. We are targeting python developers who want to program FPGAs.

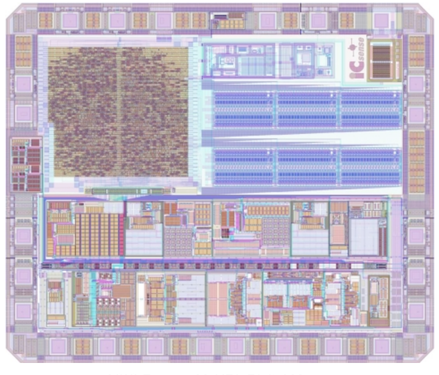
The goal of the MyHDL project is to empower hardware designers with the elegance and simplicity of the Python language.

MyHDL is a free, open-source package for using Python as a hardware description and verification language.

Modeling

Python's power and clarity make MyHDL an ideal solution for high level modeling. Python is famous for enabling elegant solutions to complex modeling problems.





ASIC Proven, MyHDL Digital Macro

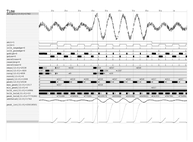
Simulation and Verification

The built-in simulator runs on top of the Python interpreter. It supports waveform viewing by tracing signal changes in a VCD file.

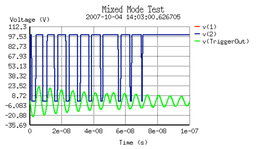
With MyHDL, the Python unit test framework can be used on hardware designs.

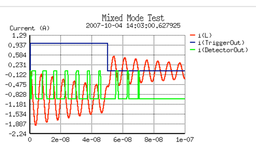
Conversion to Verilog and VHDL

Subject to some limitations, MyHDL designs can be converted to Verilog or VHDL. This provides a path into a traditional design flow

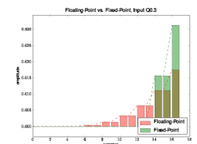


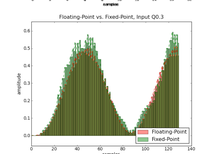
Complex Simulations





Cosimulation with Spice





Fixed-Point Analysis

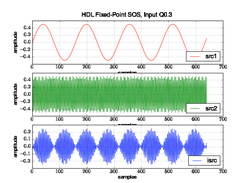
Why?

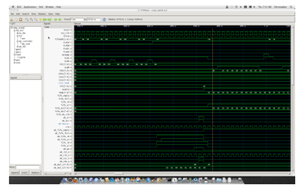
- Manage Complex Designs
- New to Digital Hardware Design
- Scripting Languages Intensively Used
- Modern Software Development Techniques for Hardware Design
- Algorithm Development and HDL Design in One Environment
- Require Both Verilog and VHDL
- VHDL Too Verbose
- SystemVerilog Too Complicated
- You have been TCL'd too much
- Google uses Python

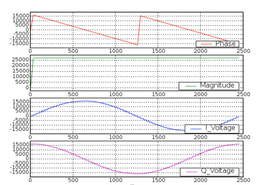
```
def bitonicSort(a, z, dir):

    n = len(a)
    k = n//2
    w = len(a[0])

    if n > 1:
        t = [Signal(intbv(0)[w:]) for i in range(k)]
        loSort = bitonicSort(a[:k], t[:k], ASCENDING)
        hiSort = bitonicSort(a[k:], t[k:], DESCENDING)
        merge = bitonicMerge(t, z, dir)
        return loSort, hiSort, merge
    else:
        feed = feedthru(a[0], z[0])
        return feed
```







Acknowledgements
Jan Decaluwe is the creator of MyHDL and www.myhdl.org. Content from www.myhdl.org contributed by Jan Decaluwe and Thomas Traber was used for this poster.

Jan Decaluwe www.jandecaluwe.com
MyHDL www.myhdl.org
DSPtronics www.dsptronics.com
Dillon Engineering www.dilloneng.com