Accelerating Algorithm Implementation in FPGA/ASIC Using Python
Modeling

- Algorithm Development – MATLAB replacement
  - use Python modules NumPy, SciPy, Matplotlib
- Algorithm Verification
- Floating to Fixed point conversions
  - quickly explore bit width and SNR tradeoffs
- Fixed point precision
  - fixed point data type keeps track of precision
- Linear algebra
- Canned math functions, FFT, filters, ...
- Easy integration of math and control functions
- Simple model re-use via modules
ASIC/FPGA Logic Build

- Custom tools, i.e. DE's ParaCore Architect
- HDL generation
  - simple to make a class to generate HDL
  - Efficient module re-use by wrapping HDL with Python
- MyHDL – code logic in Python, auto-generate Verilog or VHDL
- Powerful scripting language, forget Perl and TCL/TK
  - gen_ise.sh – available on DE website, generates ISE build files
  - project build scripts
  - simulation scripts, regression
Verification

- Use Python `unittest` module to generate test cases
- Generate verification data from Python model
  - simplifies test bench
  - increases test coverage because of ease of generating corner cases and random data from Python
- Allow higher level tests using MyHDL to control simulation
  - Data can be feed from Python to HDL code and back by means of Co-Simulation
- Simple HDL testbench, can be driver by Python generated data for full system verification
- Regression test control
Parks-McClellan Filter Design Example

MATLAB approach

- \( f = [0 \ 0.3 \ 0.4 \ 0.6 \ 0.7 \ 1]; \)
- \( a = [0 \ 0 \ 1 \ 1 \ 0 \ 0]; \)
- \( b = \text{firpm}(17,f,a); \)
- \( [h,w] = \text{freqz}(b,1,512); \)
- \( \text{plot}(f,a,w/pi,\text{abs}(h)) \)
- \( \text{legend}('\text{Ideal}', '\text{firpm Design}') \)

Python approach

- \( f = [0, \ 0.3, \ 0.4, \ 0.6, \ 0.7, \ 1] \)
- \( a = [0, \ 0, \ 1, \ 1, \ 0, \ 0] \)
- \( b = \text{remez}(17, f, a[::2], \text{Hz}=2) \)
- \( [w, h] = \text{freqz}(b,1,512) \)
- \( \text{plot}(f, a, w/pi, \text{abs}(h)) \)
- \( \text{legend}('\text{Ideal}', '\text{remez Design}') \)
- \( \text{show}() \)
Python Matplotlib Plot Bandpass
Fixed Point Modeling

- Use a Python class to create a new data type
- Example class created by DE: DeFixedInt()
- Model signed fixed point data
  - specify integer and fractional width
- Use operator overloading to allow basic arithmetic

- Representing signed fixed point numbers: A(a,b)
  - A – represent signed fixed point number
  - a – integer value representing integer width
  - b – integer value representing fractional width
- Example: A(8,2)
  - 11 bit wide number, 8 integer bits + 2 fractional bits + 1 sign bit
- DeFixedInt available for download at www.dilloneng.com
DeFixedInt() Example

- How addition changes the representation
  - a = DeFixedInt(8,2)
  - b = DeFixedInt(8,0)
  - c = a + b
  - print c.rep
  - A(9,2)

- How addition changes the value
  - a.value = 1.25
  - b.value = 2.0
  - c = a + b
  - print c.fValue
  - 3.25
Why Python?

- Open source object oriented programming
- Intuitive language, even hardware engineers can be productive using
- Single language can handle all programming requirements
  - scripting
  - modeling
  - logic generation
  - verification
- Open source modules available for almost any function
- Concise, explicit format, easy to code, read and maintain
Open Source Resource Summary

- Python - [http://www.python.org]
- numpy - [http://www.numpy.scipy.org]
- scipy - [http://www.scipy.org]
- MyHDL - [http://myhdl.jandecaluwe.com/]
- Octave - [http://www.octave.org/]
- Icarus - [http://icarus.com/eda/verilog]
- matplotlib - [http://matplotlib.sourceforge.net/]
- Dillon Engineering - [http://www.dilloneng.com/]
DE IP Available for ASIC/FPGAs

- World's fastest and most efficient FFT architectures, fixed and floating point
  - Full Parallel
  - Dual Parallel
  - Parallel Butterfly
  - UltraLong FFT
  - 2D FFT
  - Mixed Radix
  - Pipelined
- Floating Point Modules
- AES Encryption/Decryption
DE Services Available

- DSP/HPEC Algorithm Development
- Algorithm-to-FPGA/ASIC
- Image and Video Processing
- Medical IC Development
- RTL Verification
- CPU and SOC Systems
- DSP Fixed/Floating Point Models
- Custom IP Design and Verification