## PyMTL/Pydgin Tutorial Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30am – 8:50am</td>
<td>Virtual Machine Installation and Setup</td>
</tr>
<tr>
<td>8:50am – 9:00am</td>
<td>Presentation: PyMTL/Pydgin Tutorial Overview</td>
</tr>
<tr>
<td>9:00am – 9:10am</td>
<td>Presentation: Introduction to Pydgin</td>
</tr>
<tr>
<td>9:10am – 10:00am</td>
<td>Hands-On: Adding a GCD Instruction using Pydgin</td>
</tr>
<tr>
<td>10:00am – 10:10am</td>
<td>Presentation: Introduction to PyMTL</td>
</tr>
<tr>
<td>10:10am – 11:00am</td>
<td>Hands-On: PyMTL Basics with Max/RegIncr</td>
</tr>
<tr>
<td>11:00am – 11:30am</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>11:30am – 11:40am</td>
<td>Presentation: Multi-Level Modeling with PyMTL</td>
</tr>
<tr>
<td>11:40am – 12:30pm</td>
<td>Hands-On: FL, CL, RTL Modeling of a GCD Unit</td>
</tr>
</tbody>
</table>
Computer Architecture Research Abstractions

Applications

Algorithms

Compilers

Instruction Set Architecture

Microarchitecture

VLSI

Sea of Transistors

Industry Development

Hundreds of Engineers

Academic Research

A Few Researchers

Overview

Pydgin Intro

GCD Instr

PyMTL Intro

Max/RegIncr

ML Modeling

GCD Unit

ISCA 2015

PyMTL/Pydgin Tutorial: Python Frameworks for Highly Productive Computer Architecture Research
Computer Architecture Research Methodologies

Modeling Towards Layout

Greater Simulation Speed

Greater Model Detail

Functional Level
Algorithm and ISA Development
Cycle Level
Design Space Exploration
Register Transfer Level
Area/Energy/Timing Validation and Prototype Development
Computer Architecture Research Toolflows

Different languages, patterns, and tools!

The Computer Architecture Research Methodology Gap

- Functional Level
- Algorithm and ISA Development
- Cycle Level
- Design Space Exploration
- Register Transfer Level
- Area/Energy/Timing Validation and Prototype Development
Great Ideas From Prior Frameworks

- **Concurrent-Structural Modeling**
  (Liberty, Cascade, SystemC)
  Consistent interfaces across abstractions

- **Unified Modeling Languages**
  (SystemC)
  Unified design environment for FL, CL, RTL

- **Hardware Generation Languages**
  (Chisel, Genesis2, BlueSpec, MyHDL)
  Productive RTL design space exploration

- **HDL-Integrated Simulation Frameworks**
  (Cascade)
  Productive RTL validation and cosimulation

- **Latency-Insensitive Interfaces**
  (Liberty, BlueSpec)
  Component and test bench reuse
What is PyMTL?

- A Python EDSL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL EDSL
- A Python tool for simulating PyMTL FL, CL, and RTL models
- A Python tool for translating PyMTL RTL models into Verilog
- A Python testing framework for model validation
The PyMTL Framework

Model / Tool Split

Specification

Test & Sim Harness

Model

Config

Model Instance

Elaborator

Simulation Tool

Translation Tool

User Tool

EDA Toolflow

Visualization

Static Analysis

Dynamic Checking

FPGA Simulation

High Level Synthesis

Tools

Output

Traces & VCD

Verilog

User Tool Output

User Tool

Simulation Tool

 ISCA 2015 PyMTL/Pydgin Tutorial: Python Frameworks for Highly Productive Computer Architecture Research 57 / 125
PyMTL 101: Traditional Model in Python

def max_unit( input_list ):
    return max( input_list )

[3, 1, 2, 0] $\rightarrow$ $f(x)$ $\rightarrow$ 3
PyMTL 101: Model in PyMTL Embedded DSL

```python
def max_unit( input_list ):
    return max( input_list )

class MaxUnitFL( Model ):
    def __init__( s, nbits, nports ):
        s.in_ = InPort[ nports ]( nbits )
        s.out = OutPort( nbits )

@s.tick_fl
def logic():
    s.out.next = max( s.in_ )
```

\[ [3, 1, 2, 0] \rightarrow f(x) \rightarrow 3 \]