# PyMTL and Pydgin Tutorial

---

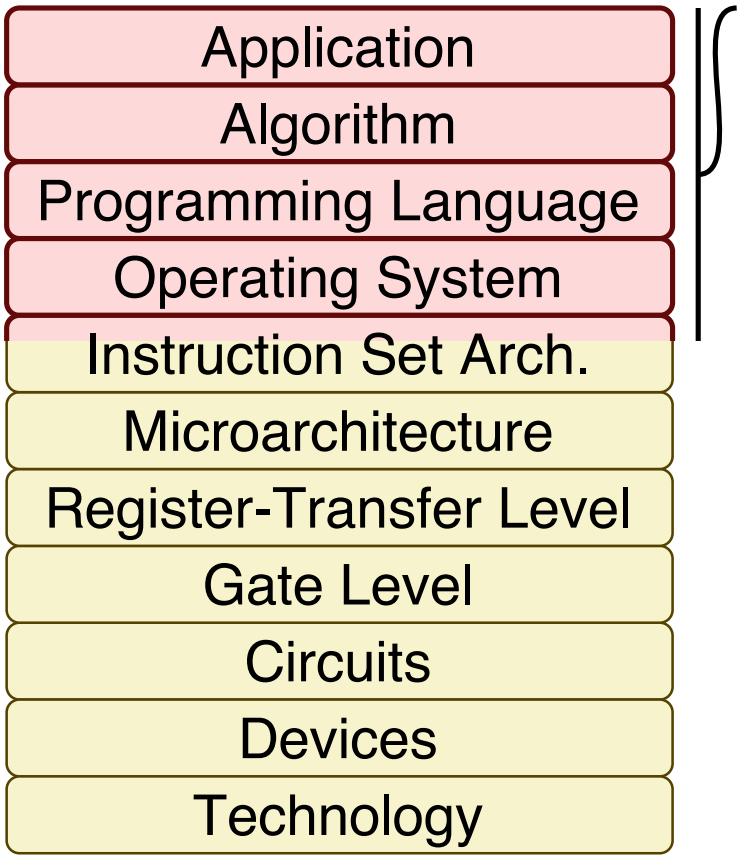
# Python Frameworks for Highly Productive Computer Architecture Research

Derek Lockhart, Berkin Ilbeyi, Christopher Batten

Computer Systems Laboratory
School of Electrical and Computer Engineering
Cornell University

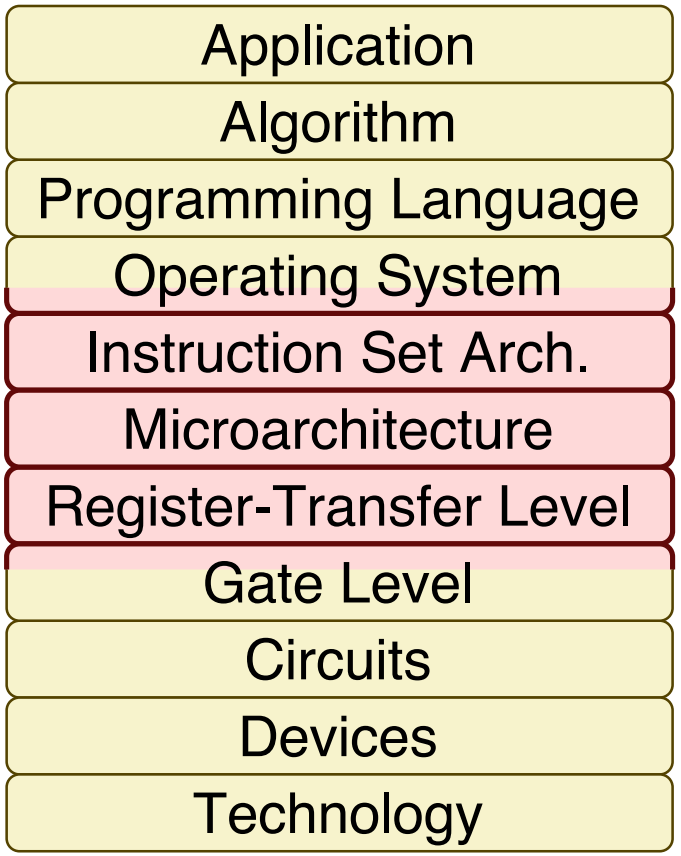
42nd Int'l Symp. on Computer Architecture, June 2015

# Typical Research Methodologies: Application-Level



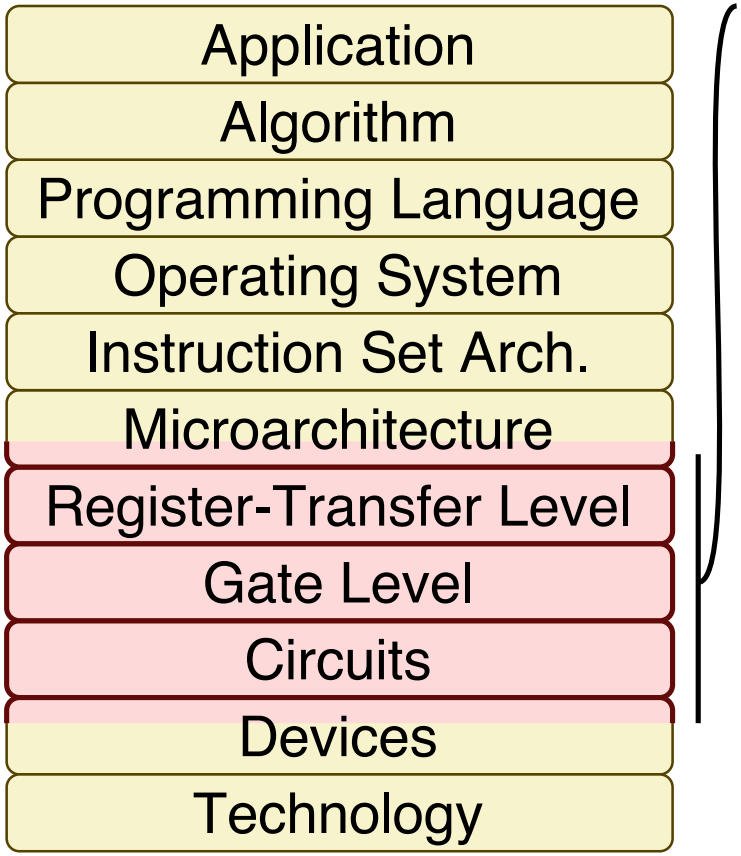

- General Approach
  - Use real machines
  - Use real machines with dynamic instrumentation (e.g., Pin)
  - Use fast instruction set simulators or emulators (e.g., SimIt-ARM, QEMU)
- Benefits
  - Fast execution enables experimenting with large, realistic applications
- Challenges
  - Difficult to explore applications for emerging architectures which do not exist yet

# Typical Research Methodologies: Architecture-Level



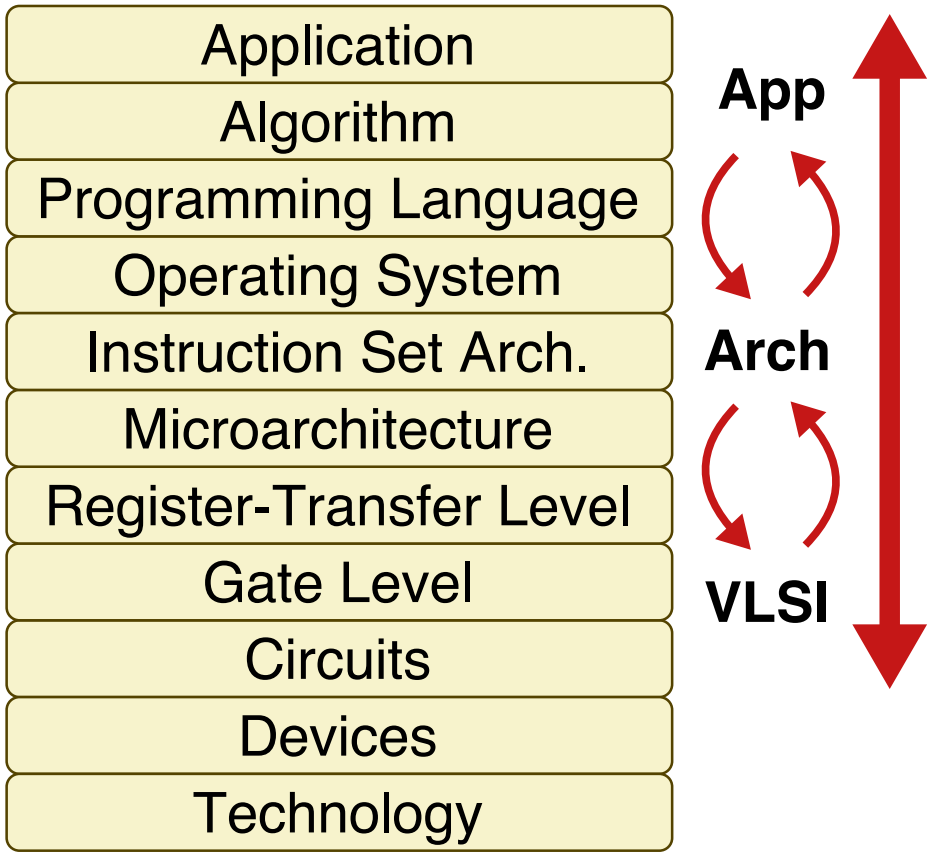

- General Approach
  - Use standard benchmark suite (e.g., Splash2, PARSEC, Rodina)
  - Modify standard cycle-level C/C++ simulator (e.g., SESC, Simics, gem5)
  - Use standard high-level physical modeling tool (e.g., CACTI, Wattch, Orion, McPAT)
- Benefits
  - More accurate than ISA simulation
  - Faster and more flexible design-space exploration than lower-level models
- Challenges
  - Experimenting with large, realistic apps
  - Physical modeling of radically new arch

# Typical Research Methodologies: VLSI-Level



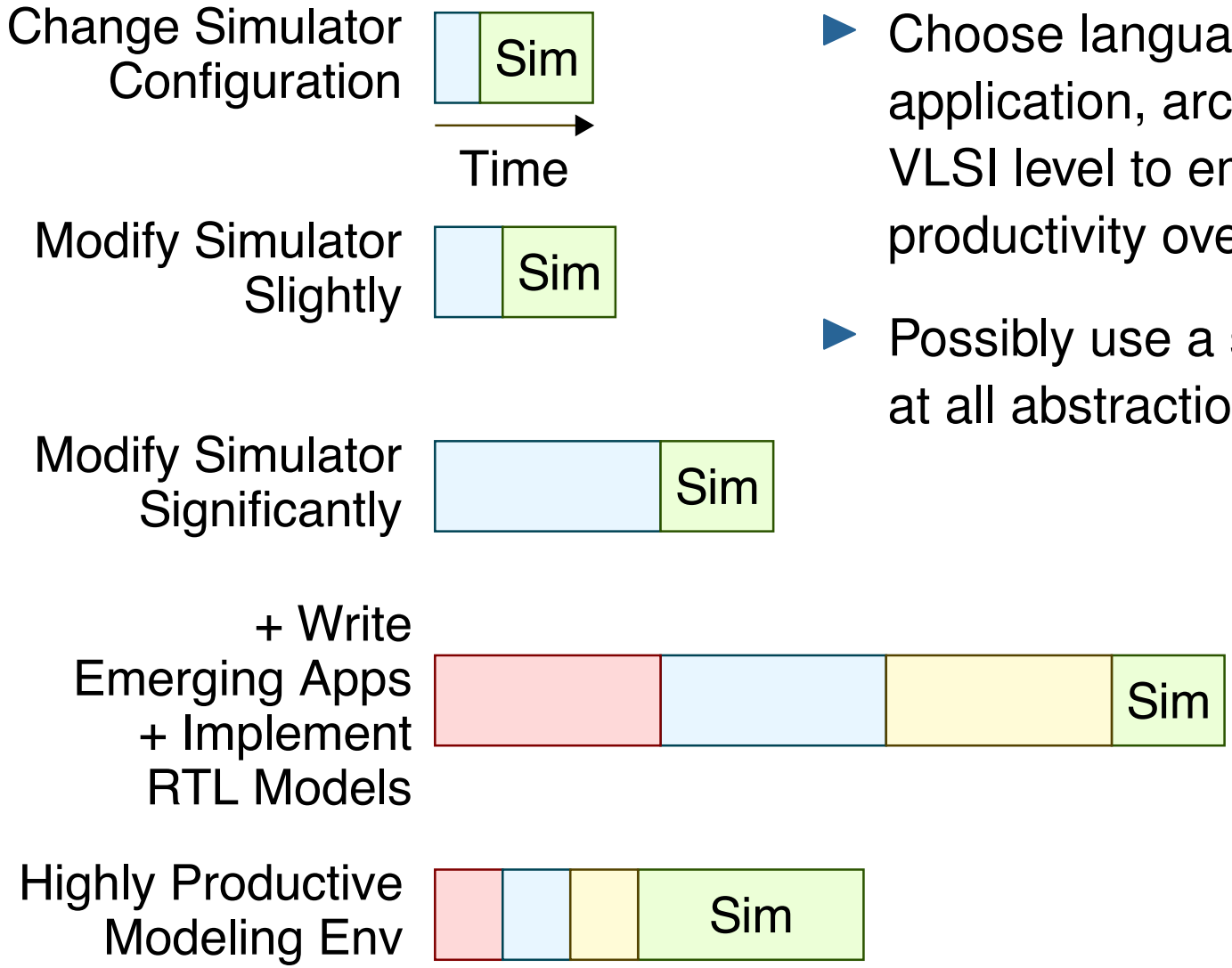

- General Approach
  - Possibly start with open-source IP (e.g., FabScalar, OpenRISC/SPARC, NetMaker)
  - Write small microbenchmarks or embedded applications
  - Implement SystemVerilog/Verilog/VHDL RTL (or Bluespec GAA) model of design
  - Use standard commercial ASIC CAD tools to estimate cycle time, area, energy
- Benefits
  - More accurate physical characterization
  - Increases credibility of design
- Challenges
  - Only small apps possible due to slow sims
  - Cumbersome design-space exploration

# Vertically-Integrated Modeling Environment



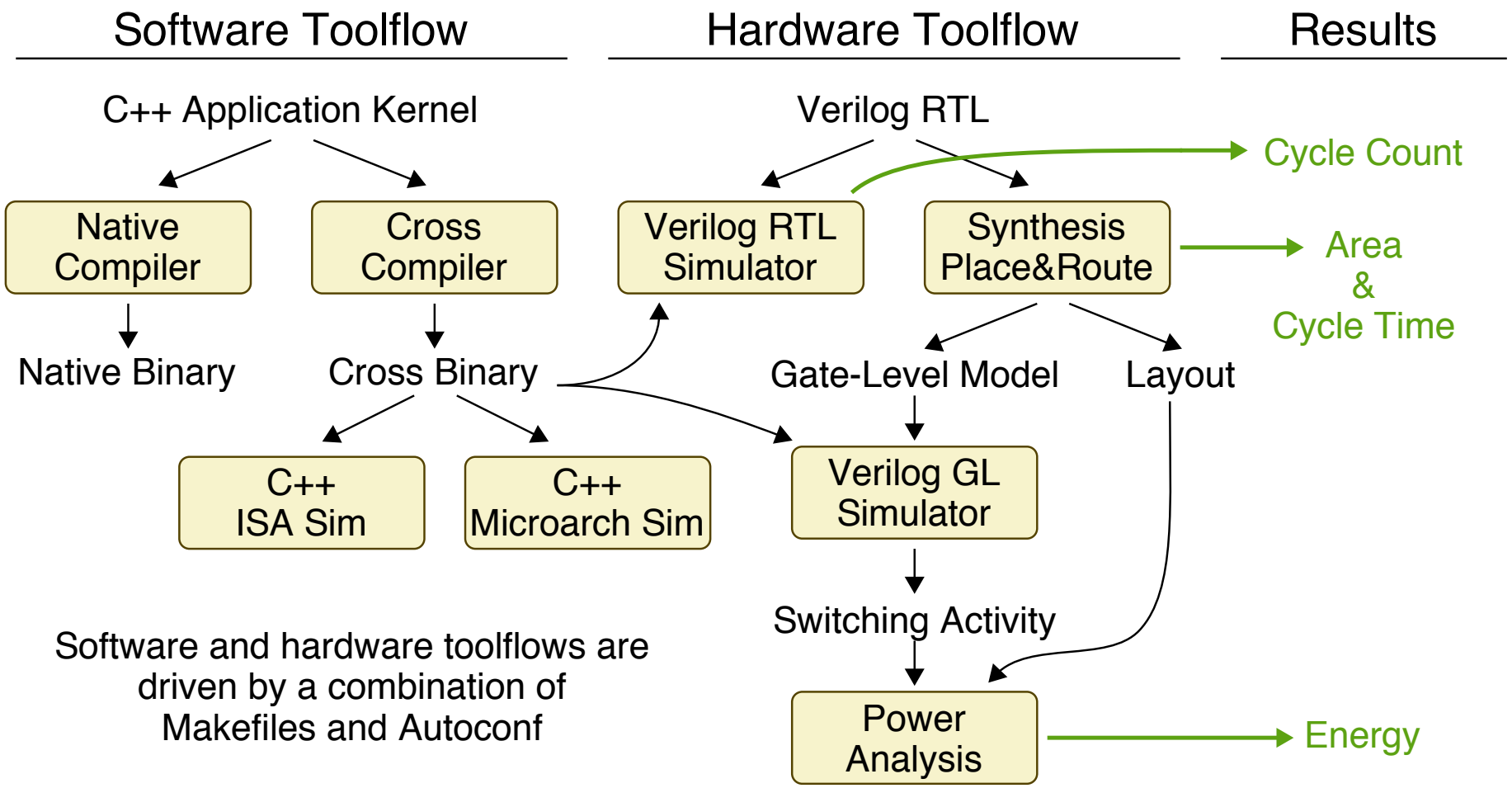

- Unified package with integrated applications, test programs, cross compilers, proxy kernels, full OS kernels, ISA emulators, microarchitectural models, RTL models, ASIC/FPGA CAD scripts, and unit tests
- Support for rapid/iterative design-space exploration across abstraction layers especially for emerging applications and radically new architectures

# Highly Productive Modeling Environment

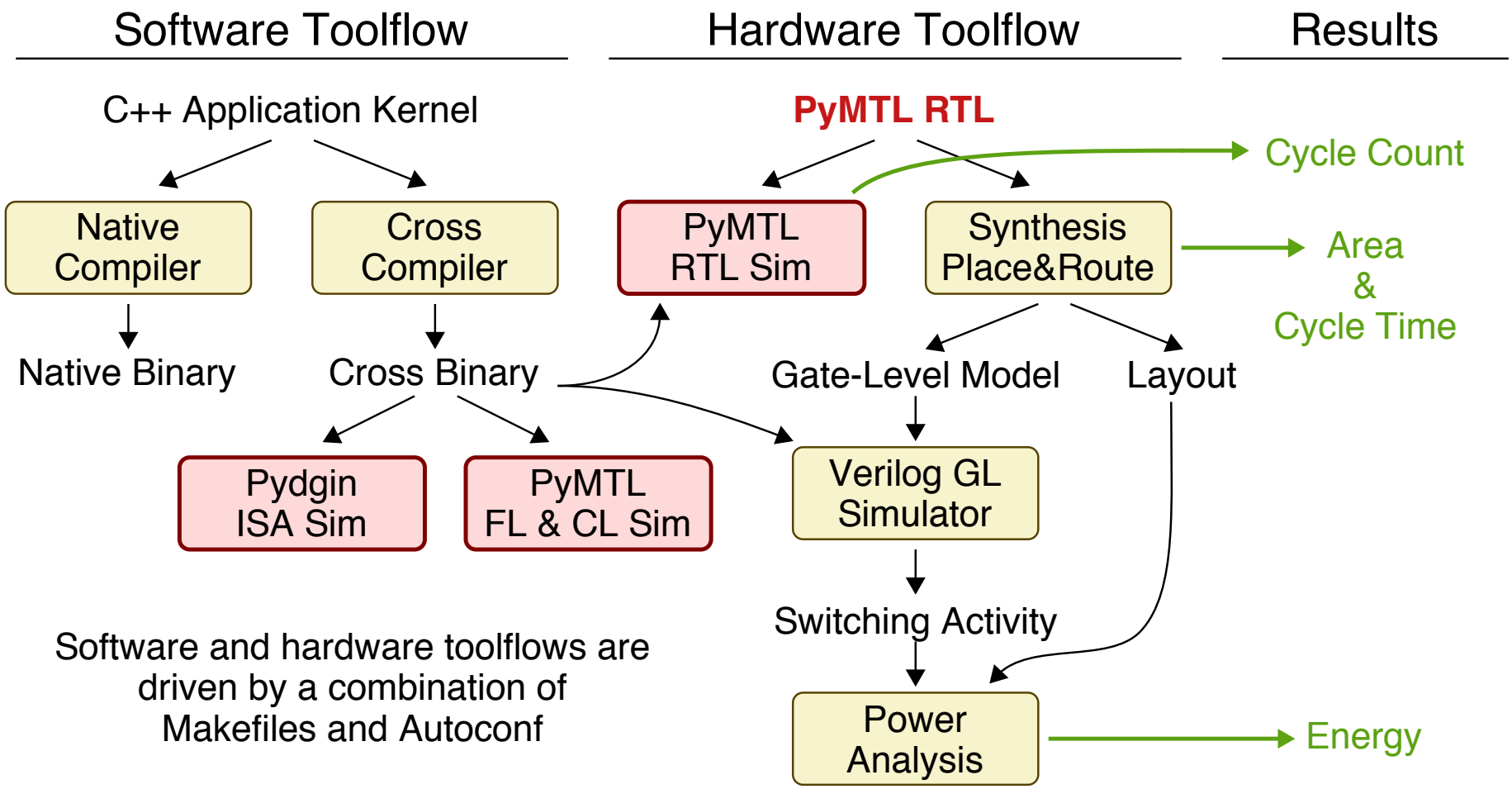



- Choose language at the application, architecture, and VLSI level to emphasize productivity over performance
- Possibly use a single language at all abstraction levels

# Previous Vertically Integrated Methodology



# Python-Based Vertically Integrated Methodology

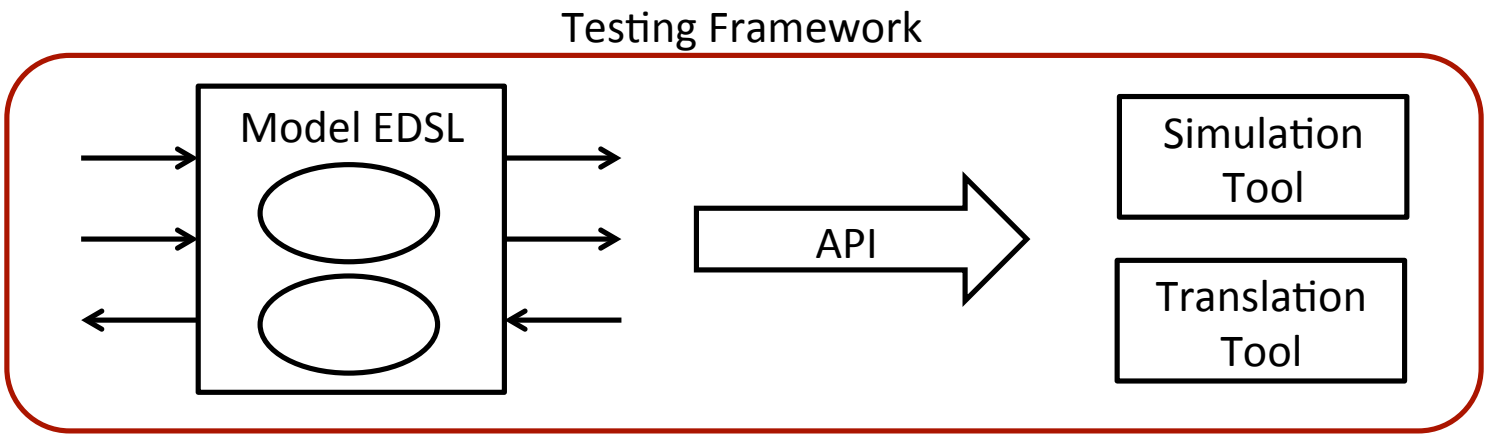


# Why Python?

- Python is well regarded as a highly productive language with lightweight, pseudocode-like syntax
- Python supports modern language features to enable rapid, agile development (dynamic typing, reflection, metaprogramming)
- Python has a large and active developer and support community
- Python includes extensive standard and third-party libraries
- Python enables embedded domain-specific languages
- Python facilitates engaging application-level researchers
- Python includes built-in support for integrating with C/C++
- Python performance is improving with advanced JIT compilation

# 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



# What is PyMTL for and not (currently) for?

- **PyMTL is for ...**
  - Taking an accelerator design from concept to implementation
  - Construction of highly-parameterizable RTL chip generators
  - Rapid design, testing, and exploration of hardware mechanisms
  - Quickly prototyping models and interfacing them with GEM5
  - Interfacing models with imported Verilog
- **PyMTL is not (currently) for ...**
  - Python high-level synthesis
  - Many-core simulations with hundreds of cores
  - Full-system simulation with real OS support
  - Users needing a complex OOO processor model “out of the box”
  - Users needing an ARM/x86 processor model “out of the box”
  - Users needing a mature modeling framework that will not change

# What is Pydgin?

Pydgin is a Python-based framework for productively generating very fast instruction-set simulators

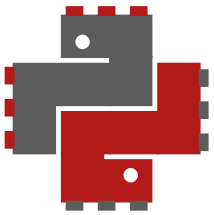

- Flexible, productive, pseudocode-like ADL syntax
- ADL embedded in a popular, general-purpose language
- Tracing-JIT generator applies across many different ISAs
- Leverages advancements from dynamic-language JIT research
- Capable of simulating RISC instruction sets at 100's of MIPS

# What is Pydgin for and not (currently) for?

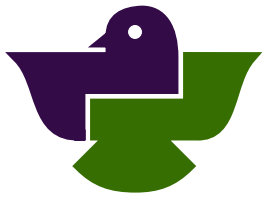
- **Pydgin is for ...**
  - Building your own very fast instruction set simulators
  - Experimenting with emerging research instruction sets
  - Easily instrumenting an instruction set simulator for early analysis
- **Pydgin is not (currently) for ...**
  - Multi-core simulations (planned for the near future)
  - Full-system simulation
  - Users needing an ARMv8/x86 simulator “out of the box”
  - Users needing a mature modeling framework that will not change

# PyMTL/Pydgin in Practice

- **PyMTL/Pydgin in Research**
  - PyMTL CL modeling used in recent XLOOPS accelerator project
  - PyMTL/Pydgin modeling used in current HLS accelerator work
  - PyMTL/Pydgin modeling used in current data-parallel accelerator work
  - Pydgin used in porting PBBS to our PARC instruction set
- **PyMTL/Pydgin in Teaching**
  - Graduate-Level Complex Digital ASIC Design Course
  - Undergraduate-Level Computer Architecture Course



PyMTL



Pydgin

PyMTL: A Unified Framework for Vertically Integrated Computer Architecture Research

[ MICRO 2014 ]

https://github.com/cornell-brg/pymtl

Pydgin: Generating Fast Instruction Set Simulators from Simple Architecture Descriptions with Meta-Tracing JIT Compilers

[ ISPASS 2015 ]

https://github.com/cornell-brg/pydgin

# PyMTL/Pydgin Project Sponsors

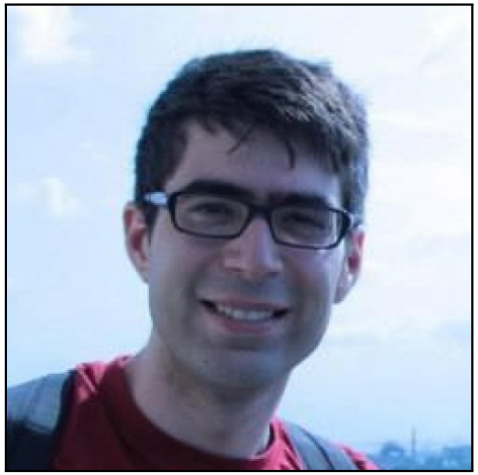

Funding partially provided by the National Science Foundation through NSF CAREER Award #1149464


Funding partially provided by the Defense Advanced Research Projects Agency through a DARPA Young Faculty Award

# PyMTL/Pydgin Tutorial Organizers


## Derek Lockhart

- N$^{th}$-Year Ph.D. Candidate, ECE, Cornell University
- Graduating this summer and heading to Google Platforms
- Research Interests: Hardware design methodologies, computer architecture, VLSI design
- Lead researcher/developer for PyMTL framework
- Co-Lead researcher/developer for Pydgin framework



## Berkin Ilbeyi

- 3$^{nd}$-Year Ph.D. Candidate, ECE, Cornell University
- Research Interests: Computer architecture, just-in-time compilation, novel hardware/software interfaces
- First “real” user of PyMTL framework for XLOOPS project
- Co-Lead researcher/developer for Pydgin framework

# PyMTL/Pydgin Tutorial Schedule

| | |
|---|---|
| 8:30am – 8:50am | Virtual Machine Installation and Setup |
| 8:50am – 9:00am | *Presentation:* PyMTL/Pydgin Tutorial Overview |
| 9:00am – 9:10am | *Presentation:* Introduction to Pydgin |
| 9:10am – 10:00am | *Hands-On:* Adding a GCD Instruction using Pydgin |
| 10:00am – 10:10am | *Presentation:* Introduction to PyMTL |
| 10:10am – 11:00am | *Hands-On:* PyMTL Basics with Max/RegIncr |
| 11:00am – 11:30am | Coffee Break |
| 11:30am – 11:40am | *Presentation:* Multi-Level Modeling with PyMTL |
| 11:40am – 12:30pm | *Hands-On:* FL, CL, RTL Modeling of a GCD Unit |