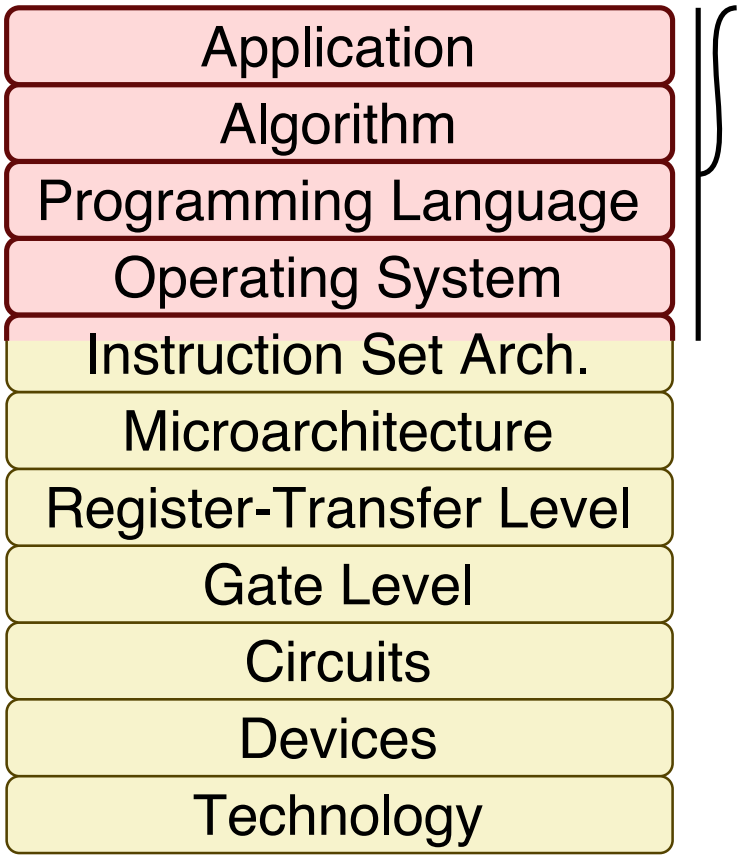
PyPARC: Vertically-Integrated, Highly-Productive Modeling Environment for a Parallel Array of RISC Cores

Christopher Batten

Computer Systems Laboratory
School of Electrical and Computer Engineering
Cornell University

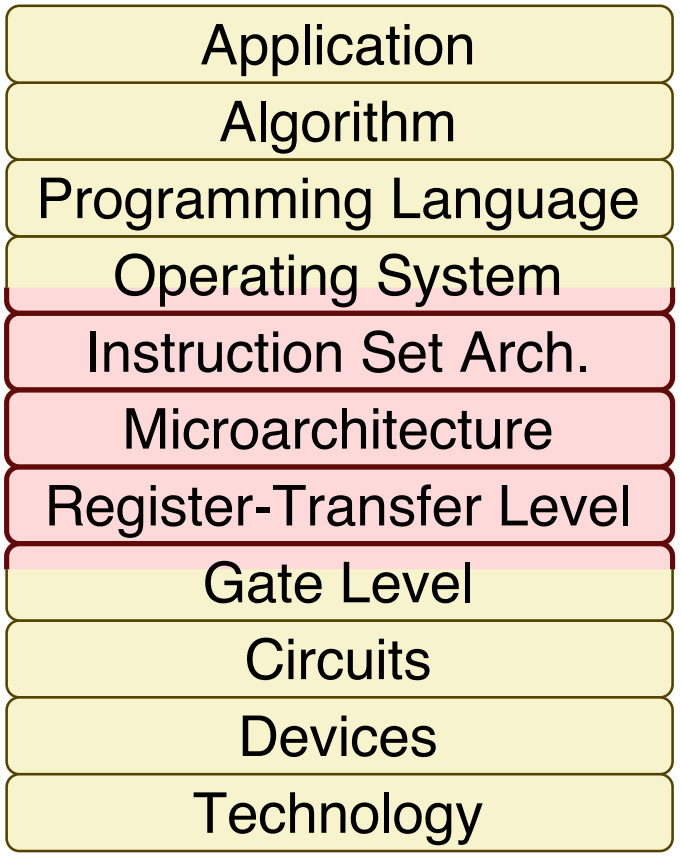
Washington, DC
May 2012

Typical Research Methodologies: Application-Level



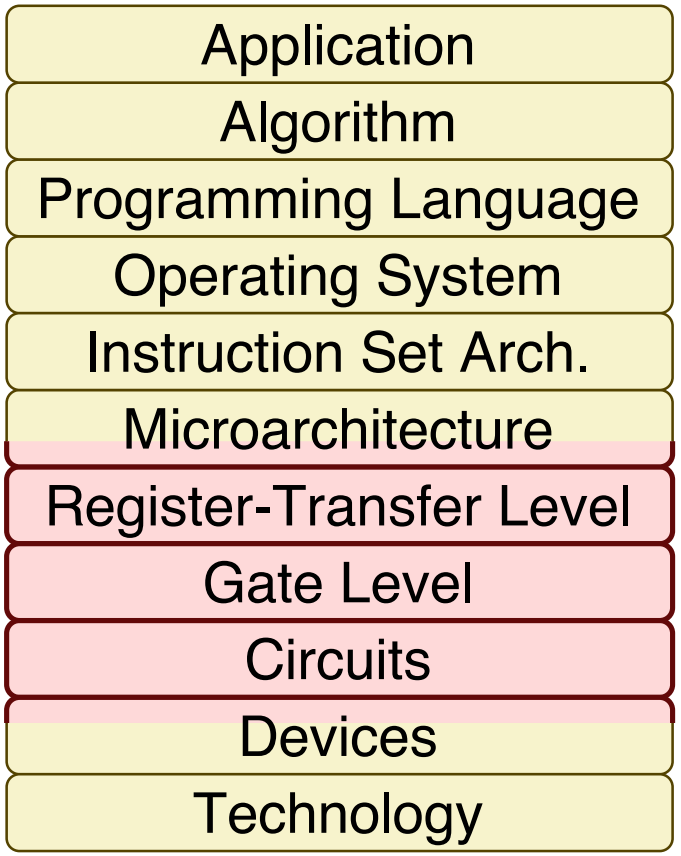
- General Approach
  - Use real machines
  - Use real machines with dynamic instrumentation (e.g., Pin)
  - Use fast instruction set simulators or emulators (e.g., Yocto, QEMU)
- Benefits
  - Fast execution enables experimenting with large, realistic applications
- Challenges
  - Difficult to explore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

Typical Research Methodologies: Architecture-Level



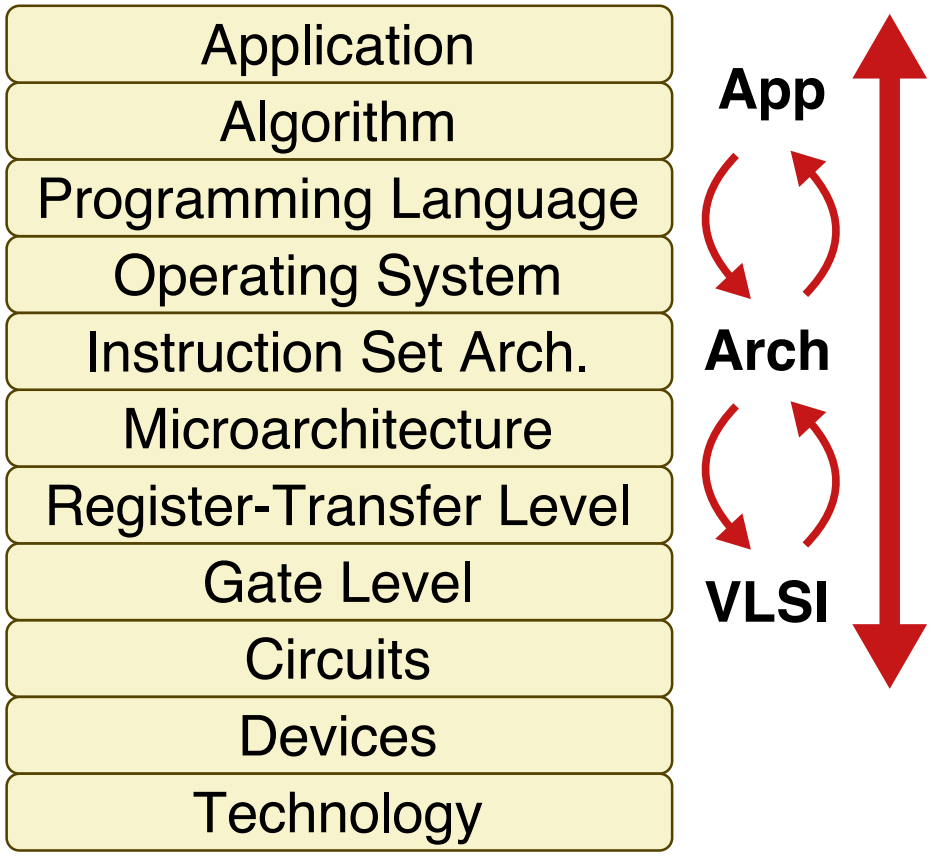
- General Approach
  - Use standard benchmark suite (e.g., Splash2, PARSEC, Rodina)
  - Modify standard cycle-level C/C++ simulator (e.g., SESC, Simics, gem5)
  - Use standard high-level physical modeling tool (e.g., CACTI, Wattch, Orion, McPAT)
- Benefits
  - More accurate than ISA simulation
  - Faster 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



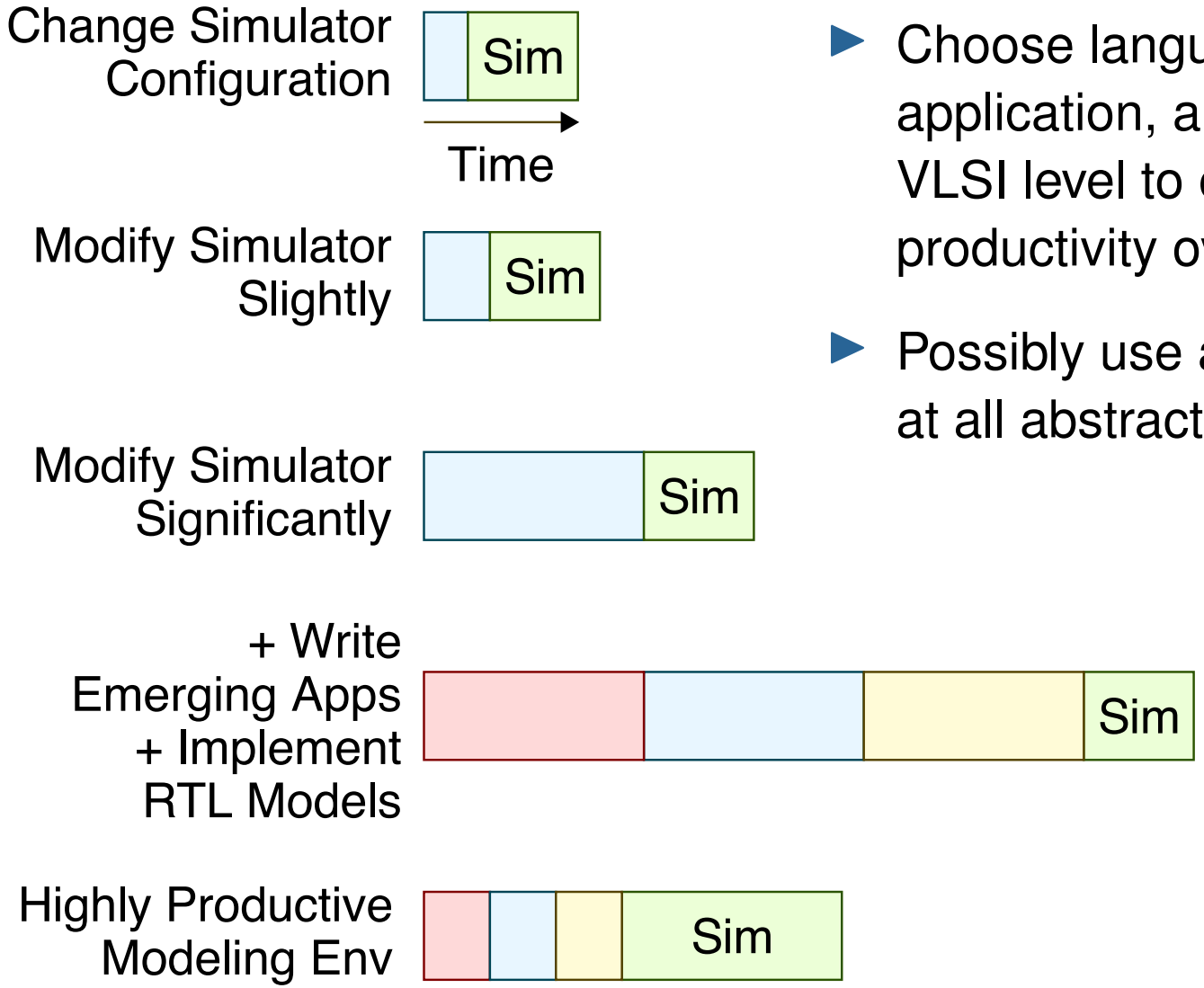
- General Approach
  - Possibly start with open-source IP (e.g., FabScalar, OpenRISC/SPARC, NetMaker)
  - Write small microbenchmarks or embedded applications
  - Implement SystemVerilog/Verilog/VHDL RTL (or Bluespec GAA) model of design
  - Use standard commercial ASIC CAD tools to estimate cycle time, 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



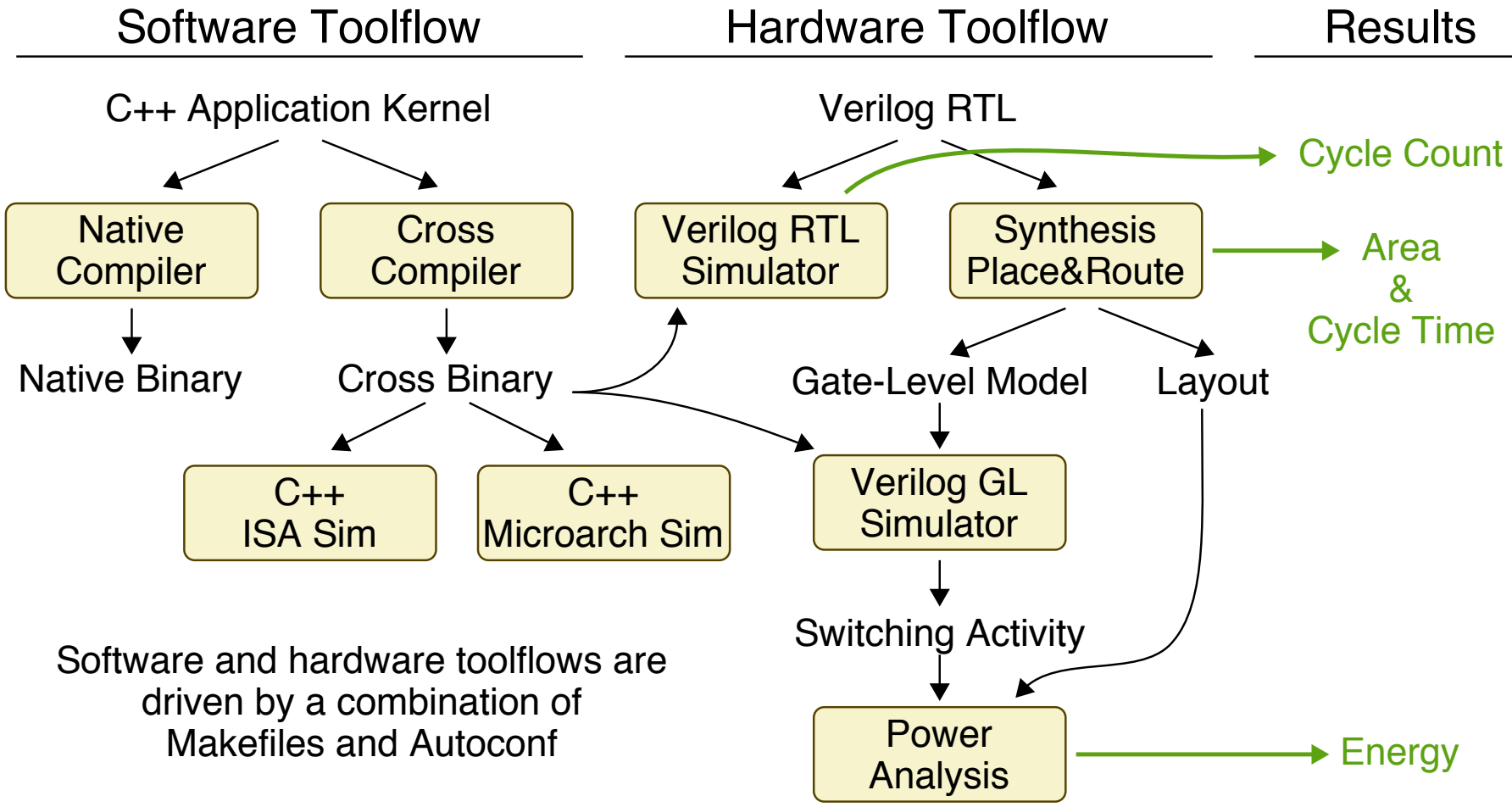
- Unified package with integrated applications, test programs, cross compilers, proxy kernels, full OS kernels, ISA emulators, microarchitectural models, RTL models, ASIC/FPGA CAD scripts, and unit tests
- Support for rapid/iterative design-space exploration across abstraction layers especially for emerging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

Highly-Productive Modeling Environment



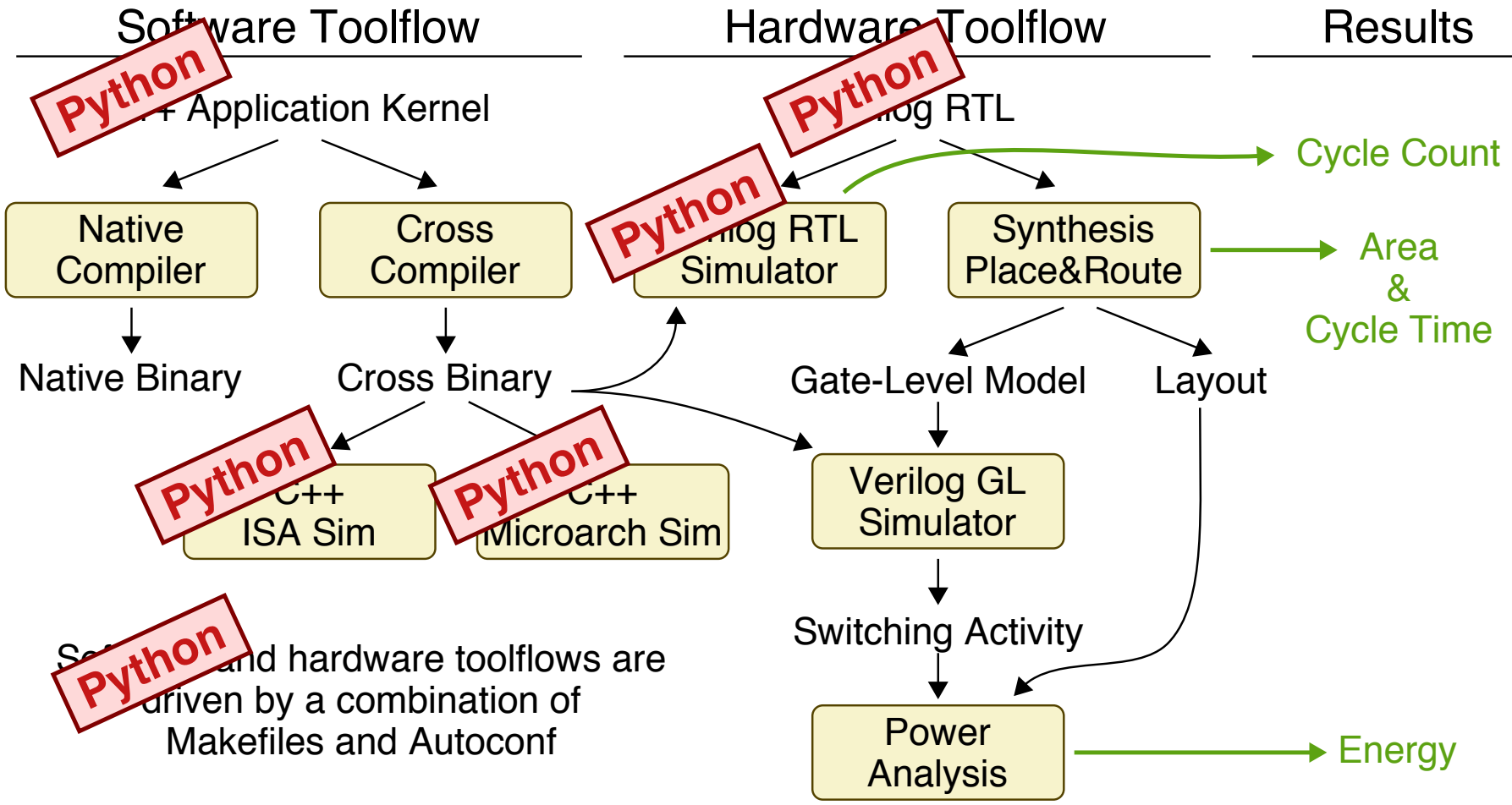
- Choose language at the application, architecture, and VLSI level to emphasize productivity over performance
- Possibly use a single language at all abstraction levels

Our Current Research Methodology



First step towards improved vertical integration (single environment) and productivity (C++ modeling framework, testing framework, standard modules), but still not good enough

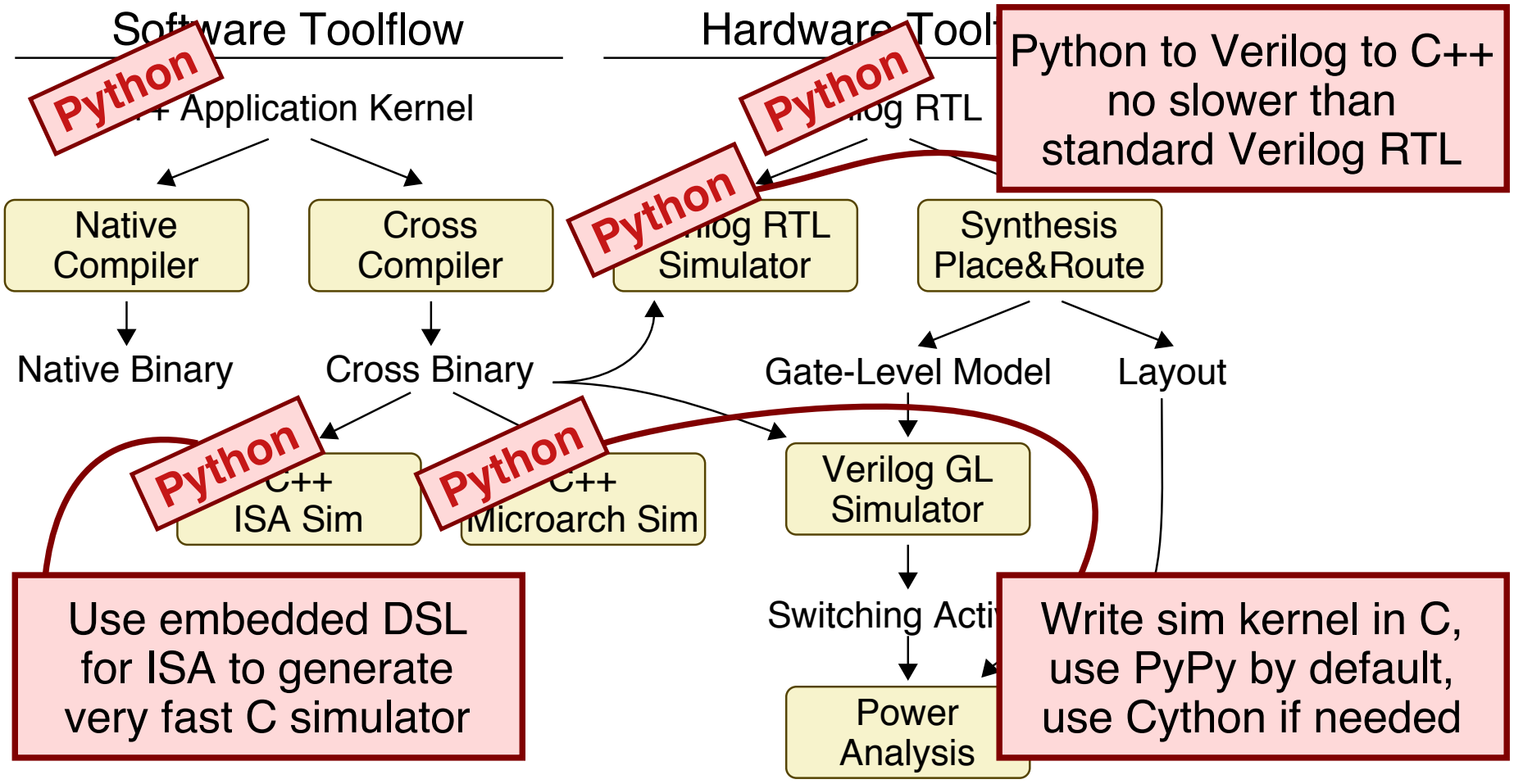
PyPARC: Python-Based Modeling Environment



PyPARC: Python-Based Modeling Environment

- Key Benefits
  - Python is well regarded as a highly productive language
  - Leverage extensive libraries to simplify development
  - Leverage robust unit testing infrastructure across all abstraction layers
  - Enables powerful static elaboration for both microarch and RTL models
  - Enables embedded DSLs (e.g., for instruction sets or assertions)
  - Easier to engage application-level researchers
  - Easier to compose different levels of abstraction into single model
- Many key technologies already exist or serve as proof of concepts
  - Python RTL to Verilog RTL translator (inspired by MyHDL)
  - Verilog RTL to C++ translator (via Verilator)
  - Python just-in-time compilation (via PyPy)
  - Python to C/C++ translation for apps and critical modules (via Cython)
  - Flexible Python-based build system (via waf)

But Isn't Python Too Slow?



Take-Away Questions

Should we be focusing on a vertically integrated and highly productive modeling environment to enable radically new application, architecture, and VLSI research?

Can Python offer a way to unify modeling across abstraction layers and thus facilitate integration and productivity?