Experiences Using a Novel Python-Based Hardware Modeling Framework for Computer Architecture Test Chips

This Poster...

Describes a taped-out 2x2 mm 1.3M-transistor test chip in IBM 130nm designed and implemented using PyMTL, a novel Python-based hardware modeling framework.

Goal of tapeout was to demonstrate the ability of this framework to enable Agile hardware design flows.

PyMTL: A Unified Python-Based Framework for FL, CL, and RTL Modeling

Functional-Level Modeling (FL)
- Behavior

Cycle-Level Modeling (CL)
- Behavior
- Cycle-Approximate
- Analytical Area, Energy, Timing

Register-Transfer-Level Modeling (RTL)
- Behavior
- Cycle-Accurate Timing
- Gate-Level Area, Energy, Timing

What Does PyMTL Enable?

1. Incremental refinement from algorithm to hardware implementation
2. Automated testing and integration of PyMTL-generated Verilog
3. Multi-level co-simulation of FL, CL, and RTL models
4. Construction of highly parameterized RTL chip generators
PyMTL for Computer Architecture Test Chips

Why Build Computer Architecture Test Chips?

Key Aspect of Agile Hardware Design
- Rapid design iteration
- "Building the right thing"
- Reduces cost of validation

Benefits Research
- Builds research credibility
- Highly reliable power and energy estimates for new architecture techniques

Design Methodologies: Large Chips vs. Small Chips

Large-Scale Commercial Chips
- High-volume and high-yield production
- Overcome design challenges with large teams

Computer Architecture Test Chips
- Low-volume and reasonable-yield production
- Overcome design challenges despite small teams and limited resources

→ Provide small teams with highly productive development frameworks to shorten time to tapeout

PyMTL for Agile Hardware Design

PyMTL Framework
- FL Simulation
- CL Simulation
- RTL Simulation
- Post-Synthesis
- Gate-Level Simulation

Highly Automated Standard-Cell Design Flow

Synthesis
- Floorplanning
- Power Routing
- Placement
- Clock Tree Synthesis
- Routing

Power Analysis
- DRC
- LVS
- RCX

Transistor-Level Sim

Post-Place-and-Route
- Gate-Level Simulation

Tape Out

Small teams push RTL to layout with validated gate-level netlist within a day

* Adapted from Yunsup Lee
IEEE Micro 2016

Why Build Computer Architecture Test Chips?

Key Aspect of Agile Hardware Design
- Rapid design iteration
- "Building the right thing"
- Reduces cost of validation

Benefits Research
- Builds research credibility
- Highly reliable power and energy estimates for new architecture techniques

Design Methodologies: Large Chips vs. Small Chips

Large-Scale Commercial Chips
- High-volume and high-yield production
- Overcome design challenges with large teams

Computer Architecture Test Chips
- Low-volume and reasonable-yield production
- Overcome design challenges despite small teams and limited resources

→ Provide small teams with highly productive development frameworks to shorten time to tapeout

PyMTL for Agile Hardware Design

PyMTL Framework
- FL Simulation
- CL Simulation
- RTL Simulation
- Post-Synthesis
- Gate-Level Simulation

Highly Automated Standard-Cell Design Flow

Synthesis
- Floorplanning
- Power Routing
- Placement
- Clock Tree Synthesis
- Routing

Power Analysis
- DRC
- LVS
- RCX

Transistor-Level Sim

Post-Place-and-Route
- Gate-Level Simulation

Tape Out

Small teams push RTL to layout with validated gate-level netlist within a day

* Adapted from Yunsup Lee
IEEE Micro 2016
Detailed Methodology
Using PyMTL for Agile Hardware Design

**HLS**
- C-Based Accelerator Tested in C
- Commercial HLS Tool
- Verilog RTL of Accelerator
- PyMTL Verilog Import

**FPGA**
- Commercial Xilinx-Based FPGA Tools
- FPGA Logic w/ Full Design
- Verilog RTL Modules Specially Annotated for FPGA Synthesis
- PyMTL Verilog Import

**ASIC**
- SRAM Specification
- IBM 130nm SRAM Compiler
- GDS of SRAM Macros
- SRAM Macros Front-End Views
- Standard Cell Front-End Views
- Standard / Pad Cell Back-End Views
- IBM 130nm PDK
- Full-Custom LVDS Receiver GDS & LEF

**PyMTL-Driven Testing Framework**
- PyMTL Simulator w/ Unit Tests and Assembly Test Suite
- VCD Traces
- Synopsys vcat utility

**PyMTL**
- PyMTL FL / CL Models
- Verilog Gate-Level Simulator
- PyMTL-Driven Testing Framework
- Calibre LVS
- Tapeout-Ready GDS of Full Design

**Commercial Xilinx-Based FPGA Tools**
- Post-Synthesis Gate-Level Netlist
- Synopsys Design Compiler
- Post-Synthesis Gate-Level Netlist
- GDS of Full Design
- IC Compiler

**Commercial**
- Tapeout-Ready GDS of Full Design
- PyMTL Verilog Import
- PyMTL to Verilog Translator
- PyMTL FL / CL Models

**Mature full-featured software testing tools**

**IBM 130nm SRAM Compiler**
- PyMTL Verilog Import
- PyMTL RTL of Full Design

**Verilog RTL of Accelerator**
- PyMTL Verilog Import
- Verilog RTL Modules Specially Annotated for FPGA Synthesis

**Standard Cell Front-End Views**
- GDS of Full Design
- DRC-Clean GDS of Full Design
- Tapeout-Ready GDS of Full Design

**IBM 130nm PDK**
- Synopsys Design Compiler
- Post-Synthesis Gate-Level Netlist
- GDS of Full Design
- Calibre DRC

**Calibre DRC**
- Tapeout-Ready GDS of Full Design

**PyMTL Simulator w/ Unit Tests and Assembly Test Suite**
- PyMTL Verilog Import
- PyMTL to Verilog Translator
- PyMTL FL / CL Models

**Verilog Gate-Level Simulator**
- Post-PAR Gate-Level Netlist
- DRC-Clean GDS of Full Design
- Tapeout-Ready GDS of Full Design
PyMTL in Practice: BRG Test Chip 1

The testing platform enables running small test programs on BRGTC1 to compare the performance and energy of pure-software kernels versus the HLS-generated sorting accelerator.

Testing Plans After Fabrication

Taped-out Layout for BRGTC1

2x2mm 1.3M transistors in IBM 130nm
RISC processor, 16KB SRAM
HLS-generated accelerators
Static Timing Analysis Freq. @ 246 MHz

Taped out in March 2016
Expected return in Fall 2016

LVDS Clock Receiver

RISC Processor
Host Interface
HLS Accelerators

SRAM Subbank (4KB)
SRAM Subbank (4KB)
SRAM Subbank (4KB)