Efficient Runtime Power Modeling with On-Chip Power Meters

Zhiyao Xie

Hong Kong University of Science and Technology

ISPD 2023

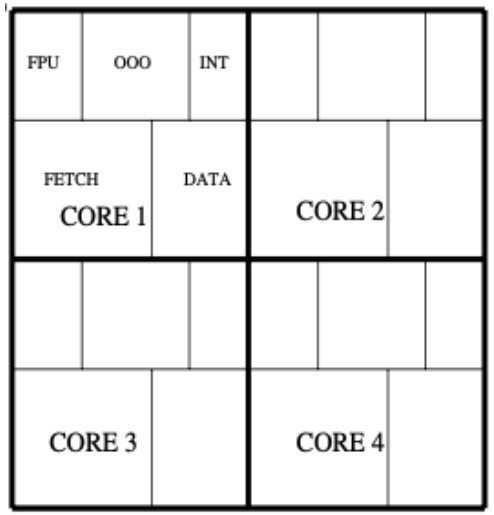
Outline

- Summary of two power modeling **scenarios**
- Summary of multiple power modeling **objectives**
- Introduction to a **general development flow**
- **Recent** results for on-chip power modeling
- Challenges and possible **future** directions

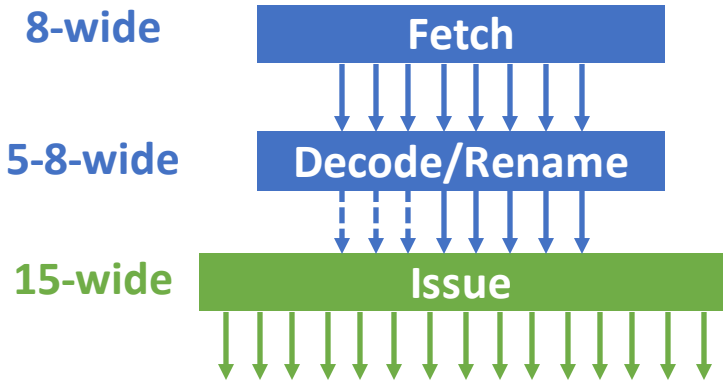
Application Scenarios of Power Models

- Application of **design-time** power models
  - Simulation on realistic workloads
- Application of **runtime** power models
  - Peak power mitigation
  - Voltage droop (Ldi/dt) mitigation

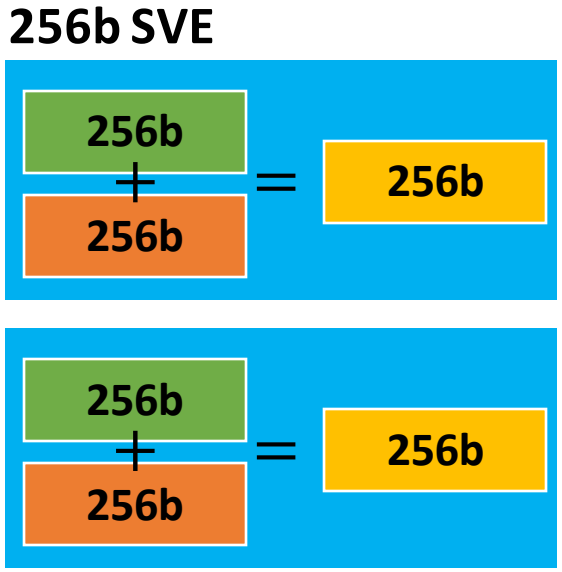
Background: Design-time Power Modeling



Many-core CPU with more transistors



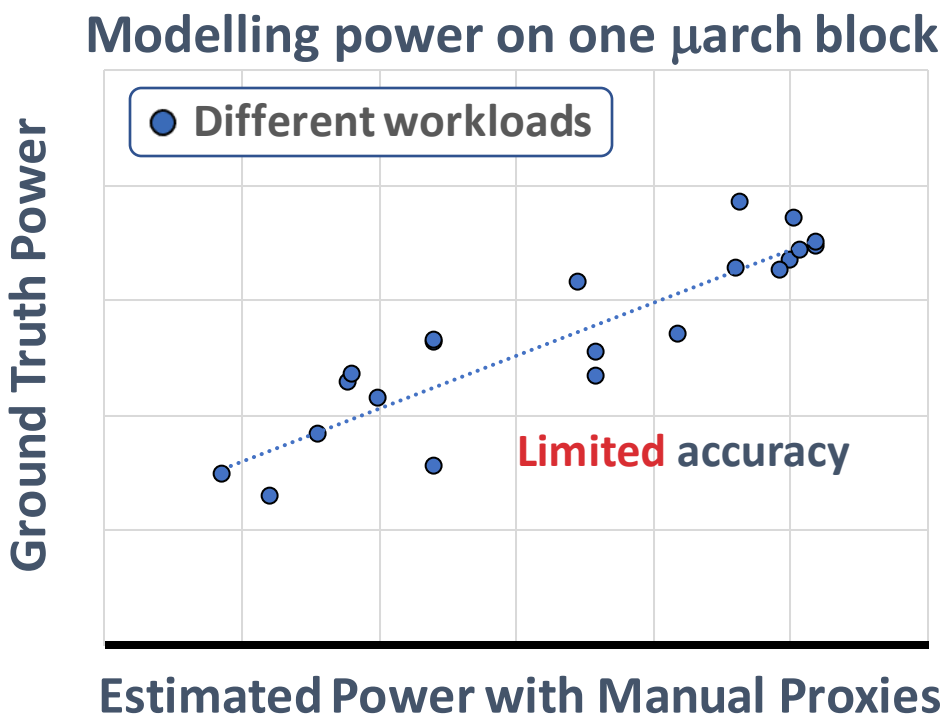
Wider issue

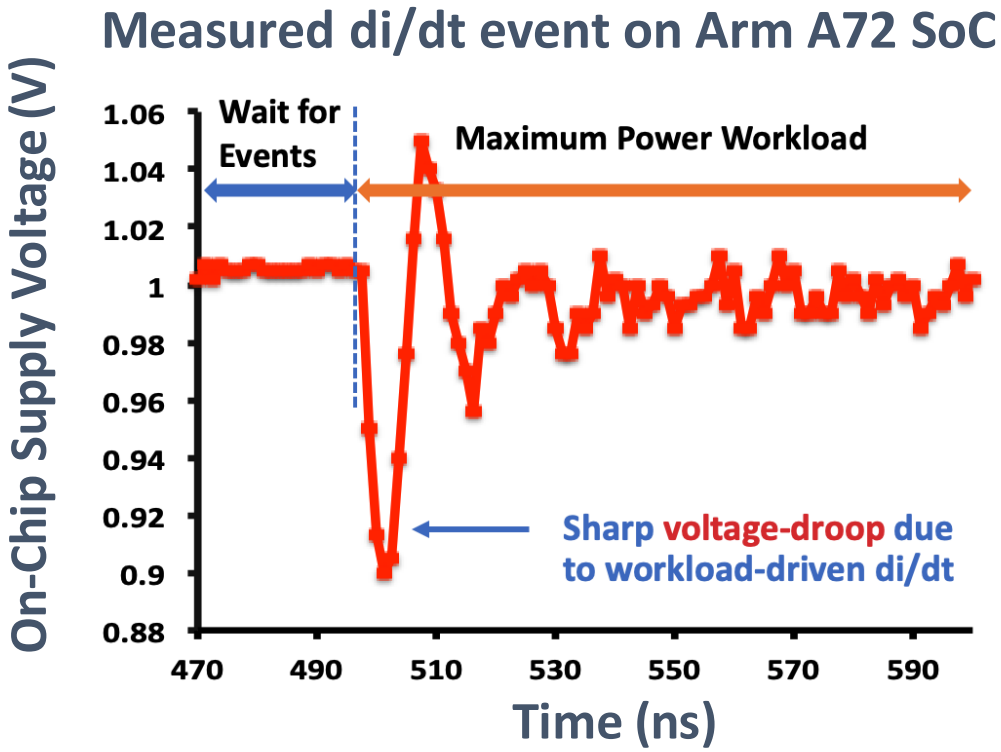


More vectored execution

- Delivering generational performance gains **adversely impacts** IC power
- Power-delivery resources **not keeping pace** with IC power demands
- **Increasing power-sensitivity drives the need for design-time introspection**

Background: Runtime Power Modeling





- **Peak-Power mitigation** requires accurate power estimation to drive throttling
- Abrupt changes in current-demand (**di/dt event**) leading to deep **voltage-droop**

Application Scenarios of Power Models

- Application of **design-time** power models
  - Simulation on realistic workloads
    - **Expensive** and **slow**
    - **Limited** temporal-resolution
- Application of **runtime** power models
  - Peak power mitigation
    - **Difficult to manually** infer proxies
  - Voltage droop (Ldi/dt) mitigation
    - Require very **low** response latency

Overview of Power Modeling Works

Power Model Types	Accuracy Level ↑	Hardware Overhead if can be Implemented on Hardware ↓	Automation Level ↑	Temporal Resolution ↑	Application Scenario
Standard power simulation [10, 45, 48]	High	N/A (Slow in Simulation)	High	High	Design-time
Micro-architectural-level estimation [8, 21, 30, 33, 41]	Low	N/A (Fast in Simulation)	Medium	N/A or High	
RTL power estimation [7, 28, 29, 31, 44, 50, 54, 57, 59, 60]	Medium	High	High	N/A or High	
Design-time emulation [11, 26, 47, 56]	Medium	Medium to High	High	Medium to High	
Counter-based OPMs [1, 2, 6, 13, 15, 19, 20, 22, 24, 38, 40, 42, 43, 46, 49]	Medium	Low	Low	Low	Runtime
Proxy-based OPMs [12, 35, 39, 52, 53, 61, 62]	Medium	Low to Medium	High	Medium to High	

Design Objectives of Power Models

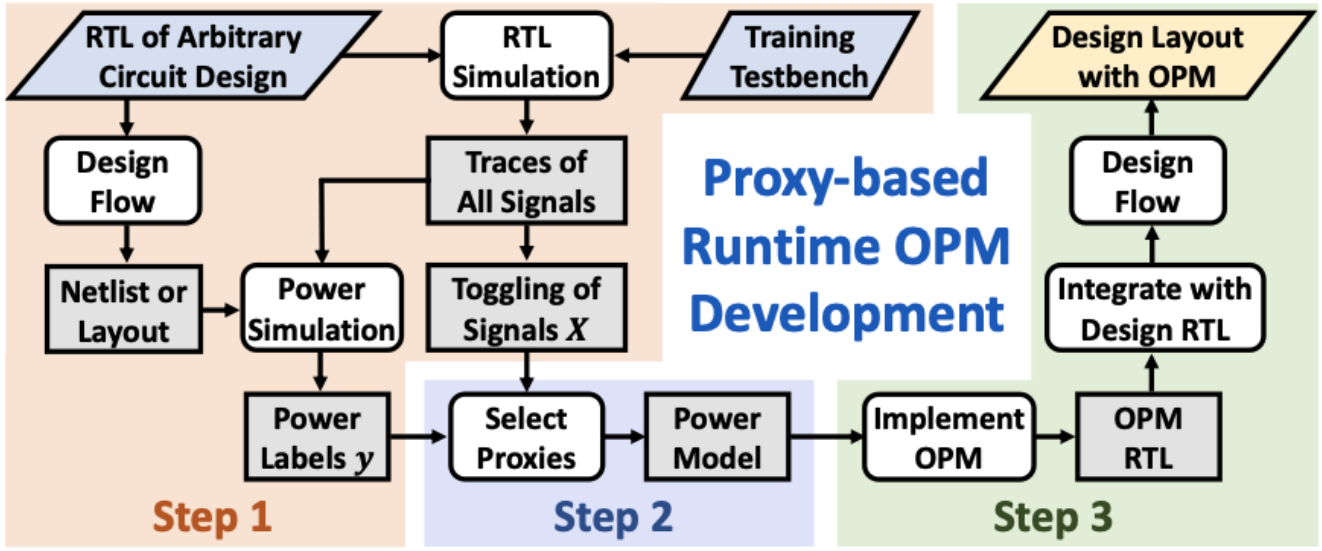
- Objective for both **design-time** and **runtime** models
  - Accuracy ↑
  - Automation-level ↑
  - Temporal resolution ↑
- Objective for **design-time** models
  - Simulation speed ↑
- Objective for **runtime** models
  - Hardware overhead ↓

Overview of Power Modeling Works

Power Model Types	Accuracy Level ↑	Hardware Overhead if can be Implemented on Hardware ↓	Automation Level ↑	Temporal Resolution ↑	Application Scenario
Standard power simulation [10, 45, 48]	High	N/A (Slow in Simulation)	High	High	Design-time
Micro-architectural-level estimation [8, 21, 30, 33, 41]	Low	N/A (Fast in Simulation)	Medium	N/A or High	
RTL power estimation [7, 28, 29, 31, 44, 50, 54, 57, 59, 60]	Medium	High	High	N/A or High	
Design-time emulation [11, 26, 47, 56]	Medium	Medium to High	High	Medium to High	
Counter-based OPMs [1, 2, 6, 13, 15, 19, 20, 22, 24, 38, 40, 42, 43, 46, 49]	Medium	Low	Low	Low	Runtime
Proxy-based OPMs [12, 35, 39, 52, 53, 61, 62]	Medium	Low to Medium	High	Medium to High	

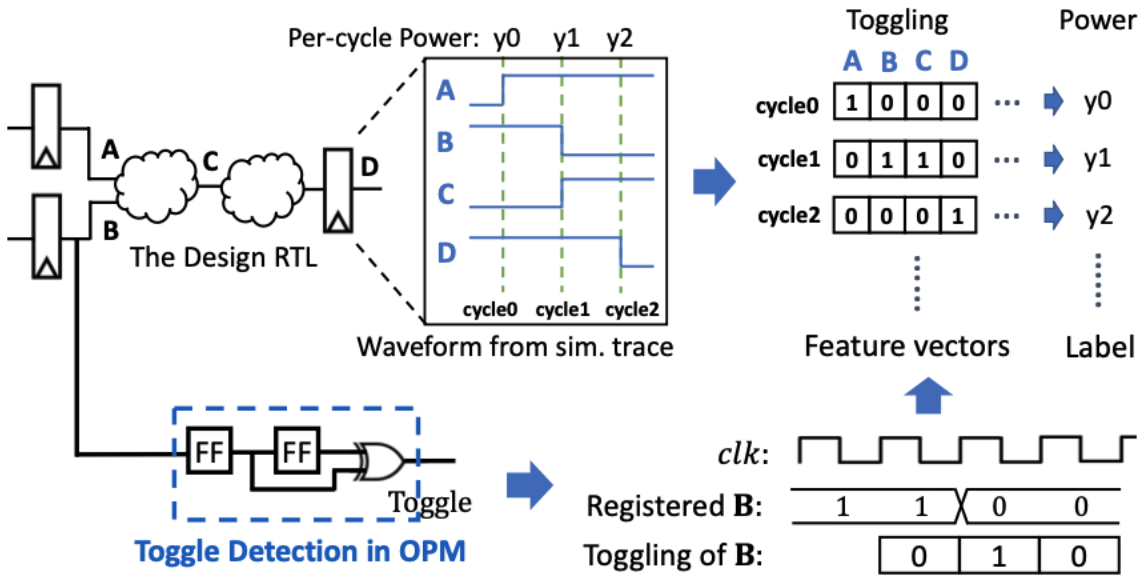
General Development Flow of Proxy-based Power Models

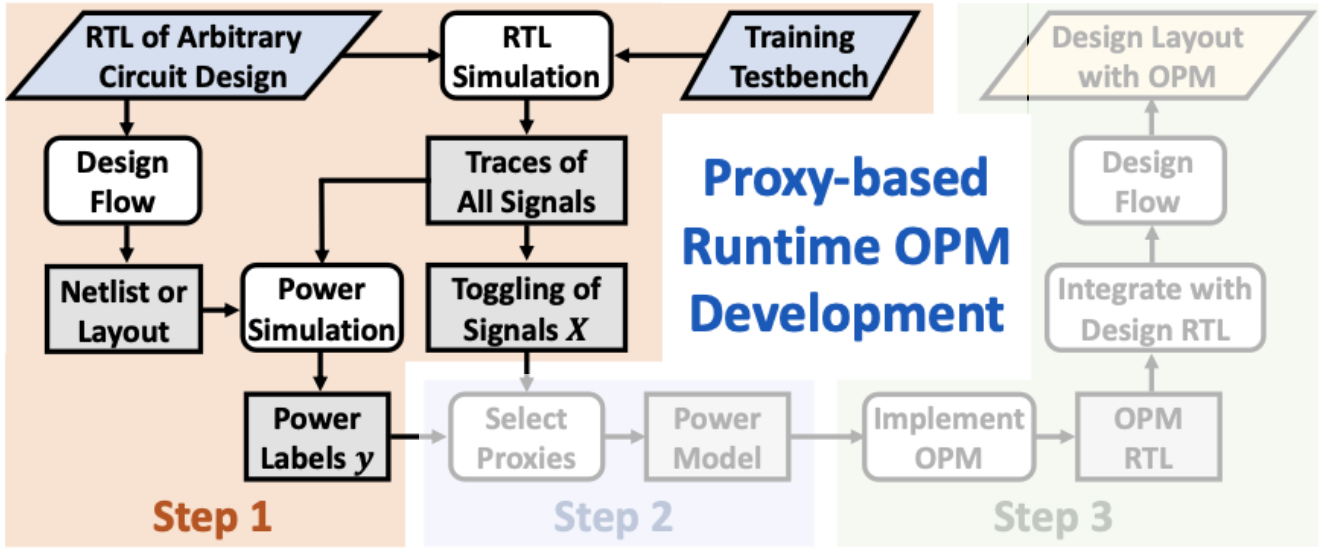
- Step 1. Dataset Generation
  - Given a design RTL, collect corresponding testbenches
  - Generate signal waveforms and ground-truth power values
- Step 2. Power Model Development
  - Develop power model for given design
- Step 3. Hardware Implementation
  - Implement power model on hardware
  - Integrate as part of the target design



Step 1. Dataset Generation

- **Testbench generation**
  - Select/generate testbenches for data collection
  - **How to know which testbench to select?**
- Toggling activity collection
  - Perform RTL simulation, generate .fsdb/.vcd
- Ground-truth power generation
  - Accurate power simulation
- Training and testing data split
  - Choose testbenches for train & test
  - **How to approximate real scenarios?**

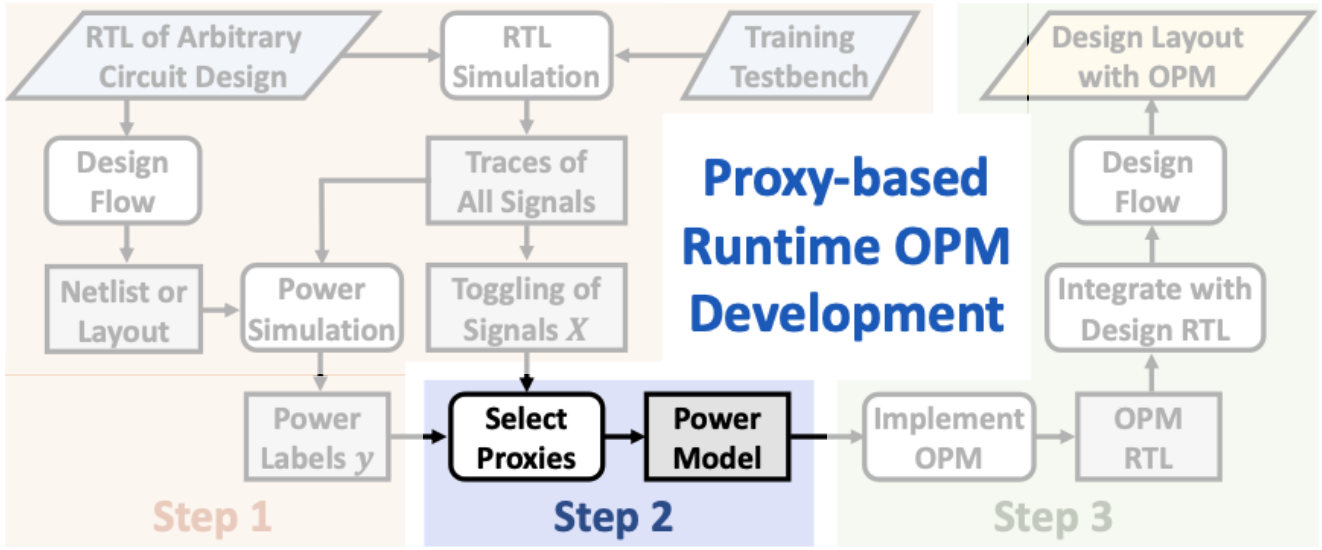




Step 2. Power Model Development

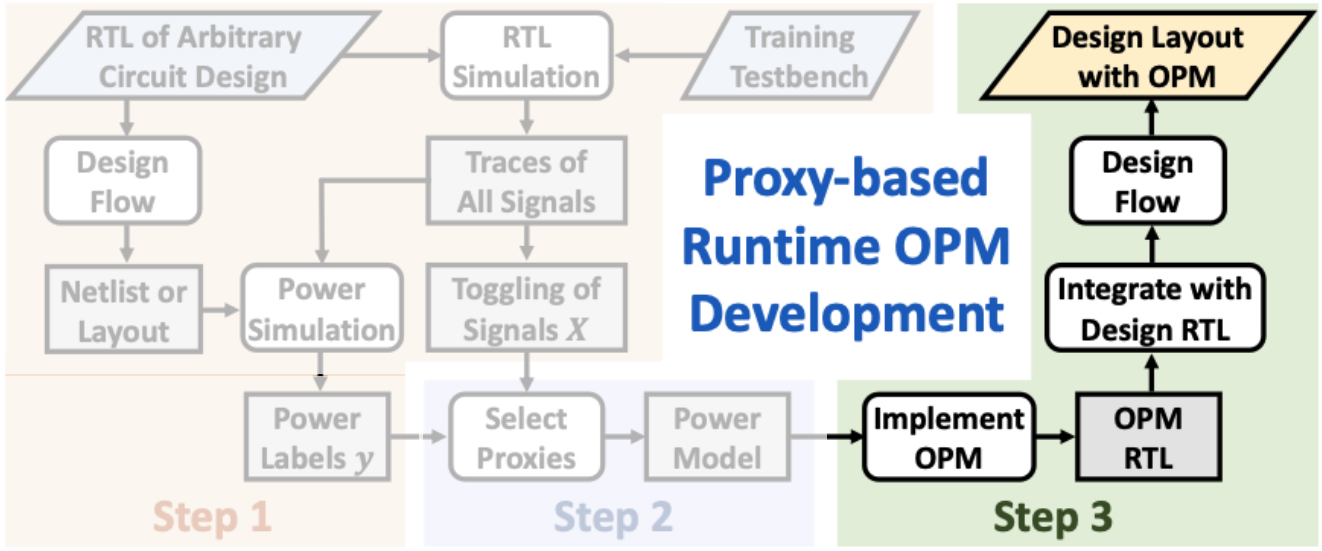
- Model selection
  - Consider accuracy/hardware cost trade-off
  - Linear or other very fast hardware-friendly models
- Temporal resolution selection
  - Trade-off with accuracy/ hardware cost
- **Model input (proxies) selection**
  - Determines accuracy/hardware cost
  - Data mining for more automation
  - **Trend is to include more candidates**
  - **What is the best selection method?**

Works	Proxy Candidate	Temporal Resolution	Claimed Overhead
[56] 2015	Registers	Per-cycle	16%
[39] 2018	Registers	> 1K cycles	7%
[62] 2018	Module I/O signals	100s cycles	4 – 10%
[26] 2019	All RTL signals	100s cycles	N/A
[12] 2020	Module I/O signals	>1K cycles	2-20%
[53] 2021	All RTL signals	Per-cycle	< 1%
[52] 2022	All bits of RTL signals	Per-cycle	< 0.1%



Step 3. Hardware Implementation

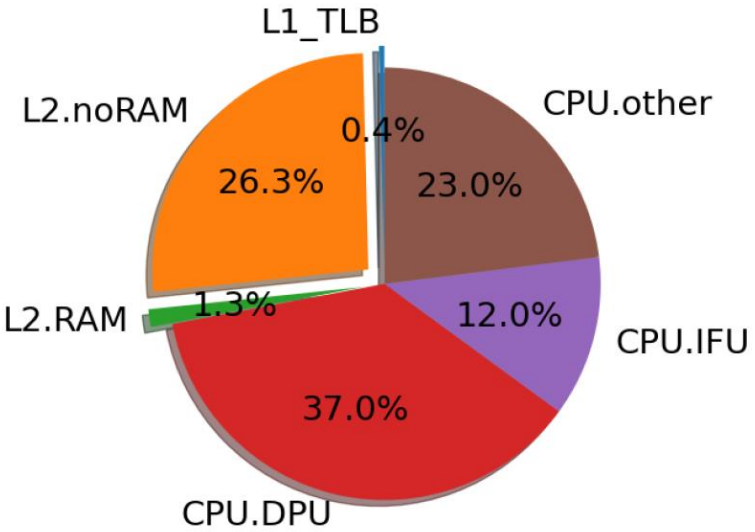
- **Reduction of hardware cost**
  - Weight quantization
  - Reduce costly multipliers & counters
  - Design to support different temporal resolution
- Integration with target design
  - Initialize on-chip power meter (OPM)
  - Connect design proxies to the OPM

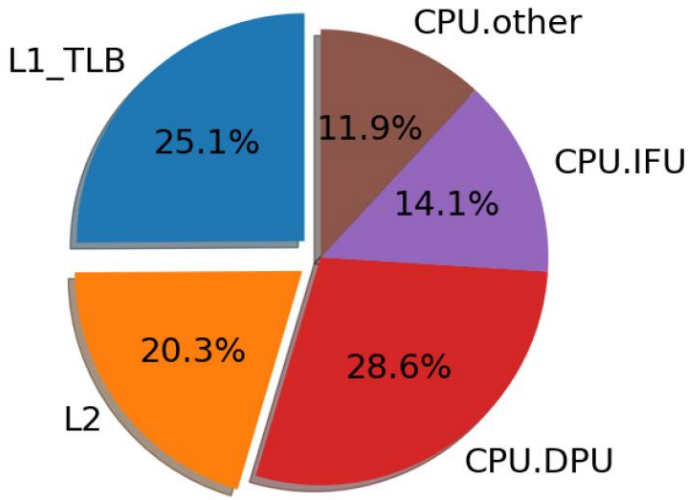


Result: Experiment Setup and Basic Statistics

#RTL Signal	#Register	#RTL Bit	#Standard Cell	#Macro
155 K	67 K	578 K	603 K	66

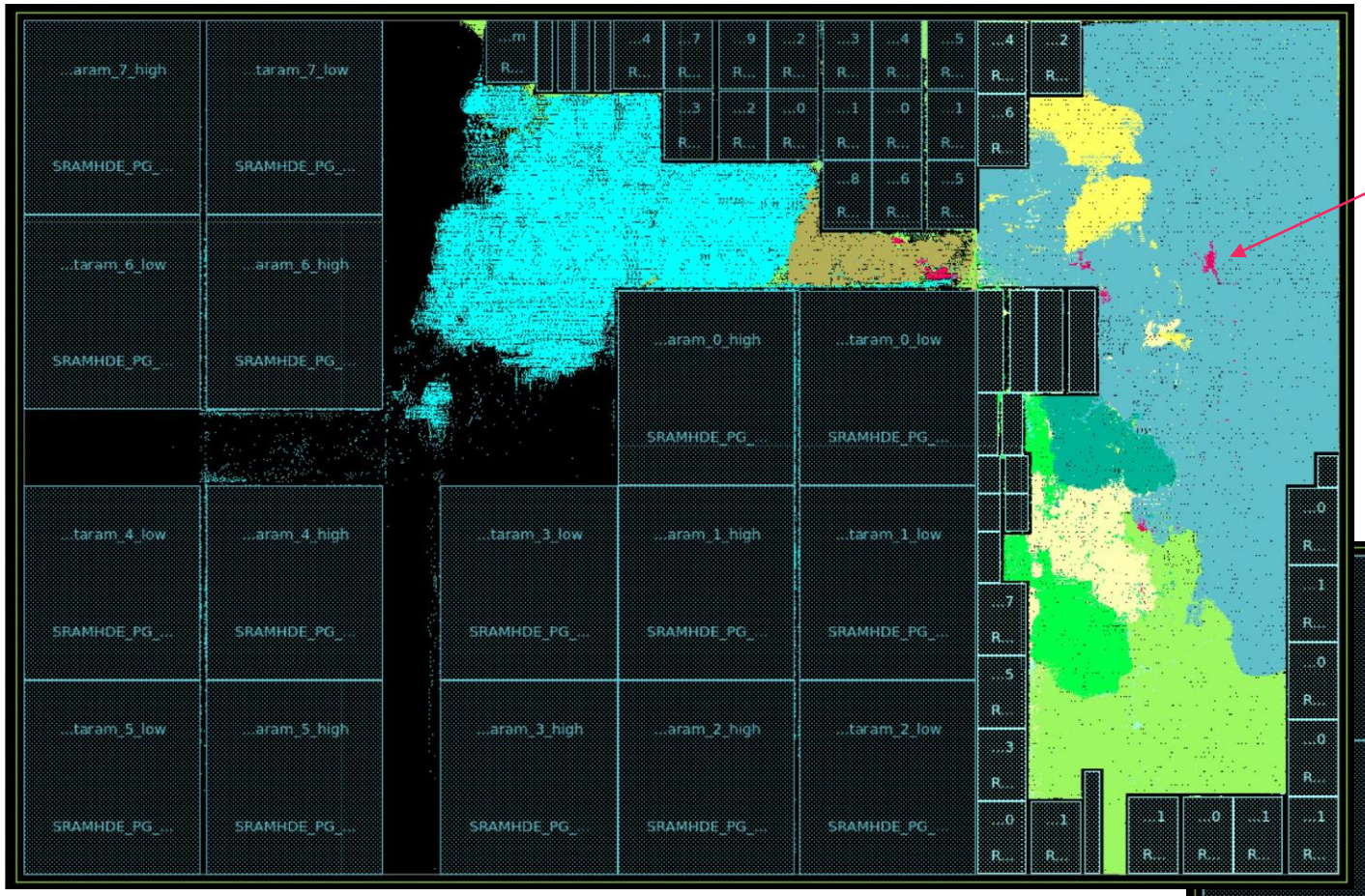
Statistics of the micro-processor used in experiment.





Left: Distribution of all 578 K RTL bits as candidates.

Right: Distribution of power.

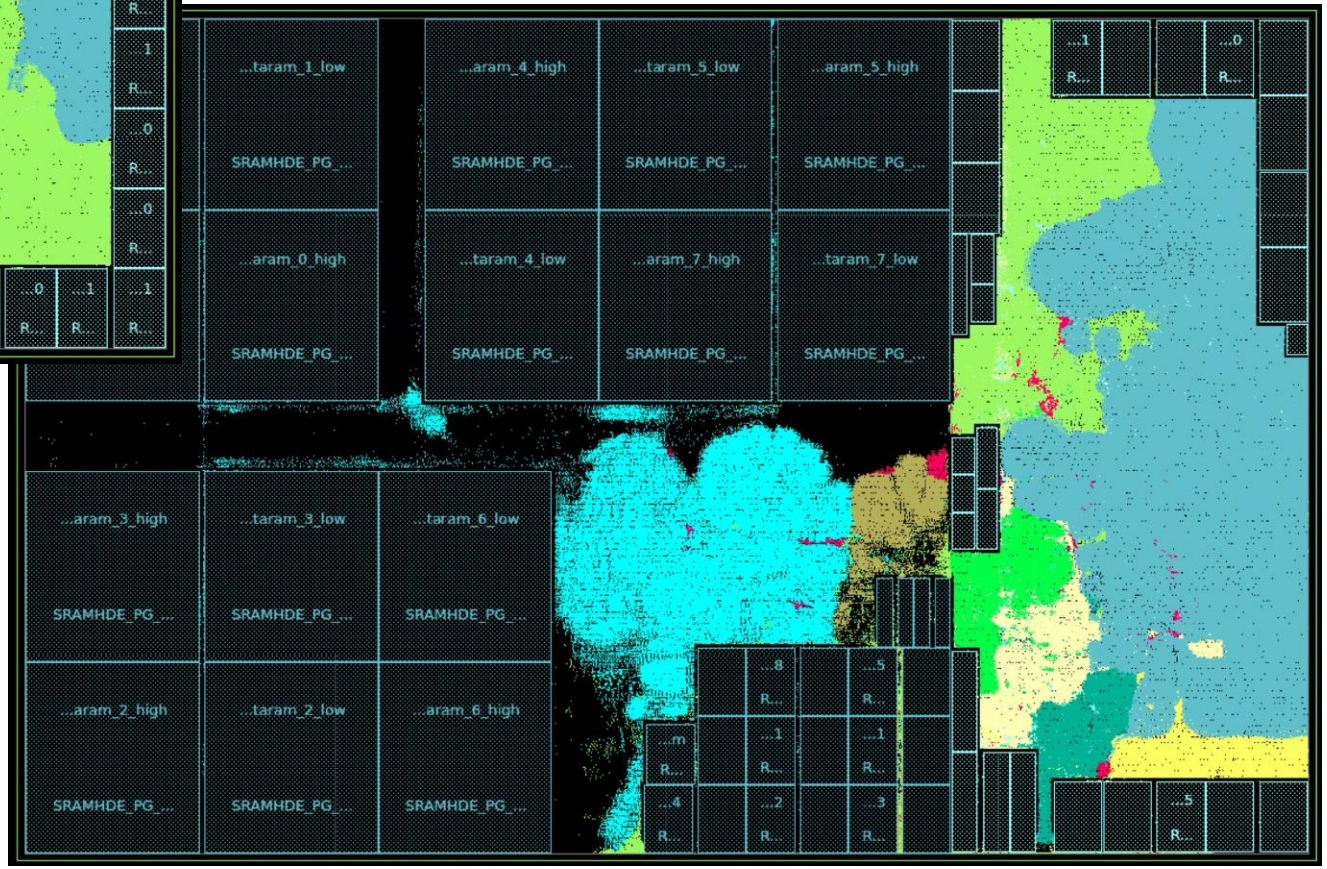


Pink regions are OPM on the layout

Area overhead **0.16%** on this layout.

The MAE = 9.5% and R = 0.951

APOLLO method [MICRO'21]



Area overhead **0.04%** on this layout.

The MAE = 9.5% and R = 0.954.

DEEP method [ICCCAD'22]

What's Next? Possible Future Improvements

- For design-time models, more generalization
  - Generalize across different designs
  - Support fine-grained vector-based power estimation
- For runtime on-chip models, better co-design of hardware cost and algorithm
  - Early evaluation of 'real' hardware cost, use it to guide co-optimization
- Prediction of power consumption in advance?
  - Is it possible to capture 'future' chip activities on-chip?

Conclusion

- We categorized and summarized various **power modeling solutions**
- We introduced a general power model **development flow**
- We demonstrate some **recent results** in on-chip power modeling
- We analyzed existing challenges and potential **future directions**