Designing Generalizable Power Models for Open-Source Architecture Simulators

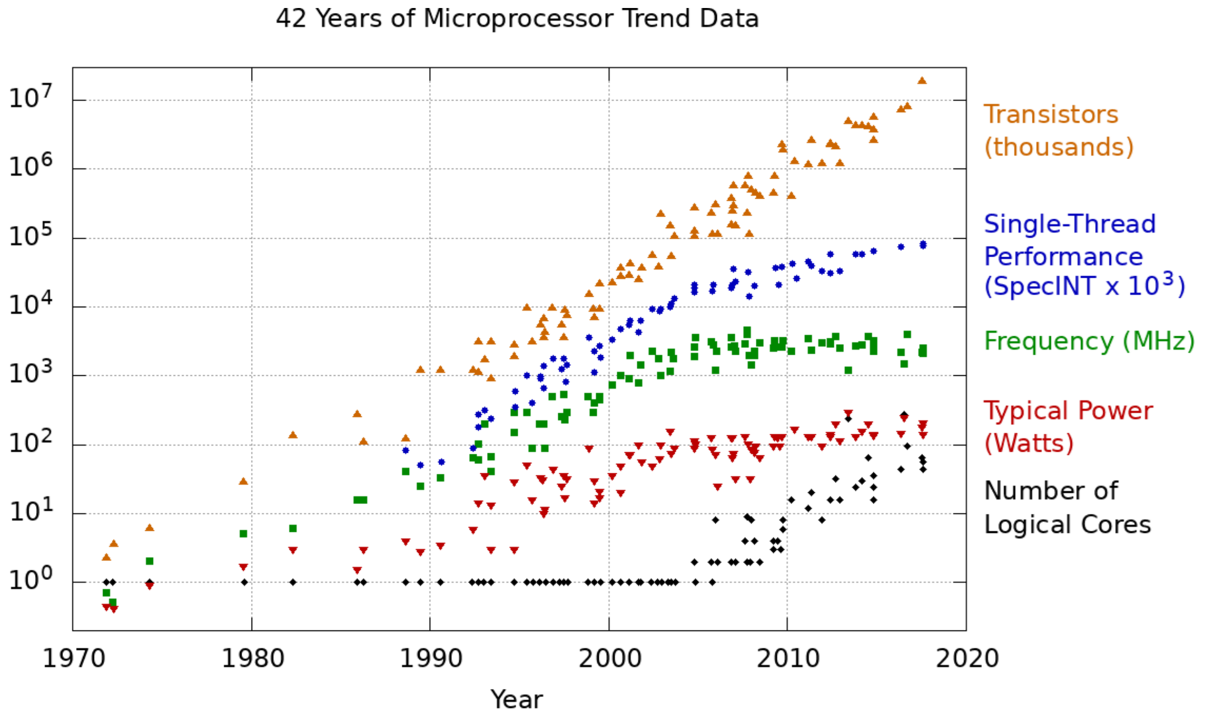
Alex Smith[*], Bobby Bruce[†], Jason Lowe-Power[†], and **Matthew D. Sinclair[*]**

[*]University of Wisconsin-Madison, [†]University of California - Davis

sinclair@cs.wisc.edu

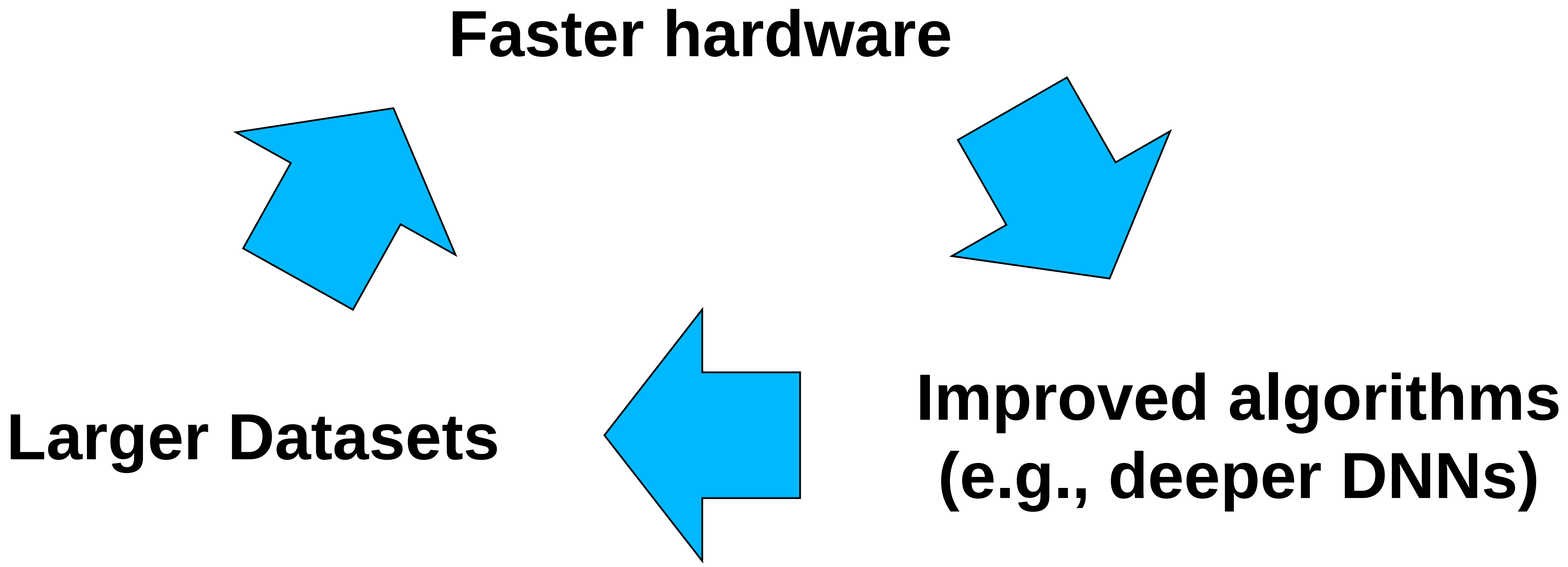
What are Future System Reqs?

- One example – recent DOE CFP Requirements:
  - 100+ exaFLOPs, up to 20 GHz frequency, ≤ 20 MW power
  - But Moore's Law continues to fade …



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Why Is This Important?



Moore's Law has enabled a virtuous cycle of progress in many fields

… slowing of Moore's Law also threatens progress

Modern apps have ravenous (exponential) compute, power needs

Must co-optimize for performance and power

What Is Needed?

⇒ **Need disruptive, cross-layer changes to meet future sys. reqs.**

- Co-design arch, runtime, OS, compiler, network, batch sched.
- For both power and performance (and maybe other factors)
- Increasingly important as transistor sizes shrink

- Typically sim. & modeling tools enable early-stage design exploration
  - Last OSCAR talk: scalably enable accurate co-design for performance
  - But what about power?
- Need credible, open-source modeling infrastructure **for both**

Power Modeling State-of-the-Art

- 5 broad types of power models:
  1. Extrapolate first-principal models (e.g., CACTI, McPAT)
     - Were highly accurate, still widely used … but not updated in 8+ years
  2. Empirical measurement-based models (e.g., AccelWattch)
     - Difficult to generalize beyond specific devices they are measured on
     - Significant accuracy decrease for even minor perturbations
  3. ML-based models
     - Tremendous potential, but accuracy often lacking for previously unseen devices
  4. Tools based on tape-out values
     - Time consuming, expensive, can only happen later in design process
  5. Low-level Spice models
     - Accurate, but often require proprietary information, hard to scale to large systems

Early-stage power model tools divided, arch-specific, out-of-date

What Can We Do?

- Additional Challenges:
  - Each power modeling approach may be “best”
  - Often certain power models easier to integrate with certain simulators
- Insight: decouple “best” power model from simulator integration
  - Don’t pick which power model is the right one
  - Abstract away how simulators integration → plug-and-play power models
- Vision: make power modeling as easy as performance modeling
- Created prototype with gem5’s MinorCPU
- Extending to other components (e.g., GPU, network, main mem)

Today’s Focus: integration with gem5

Outline

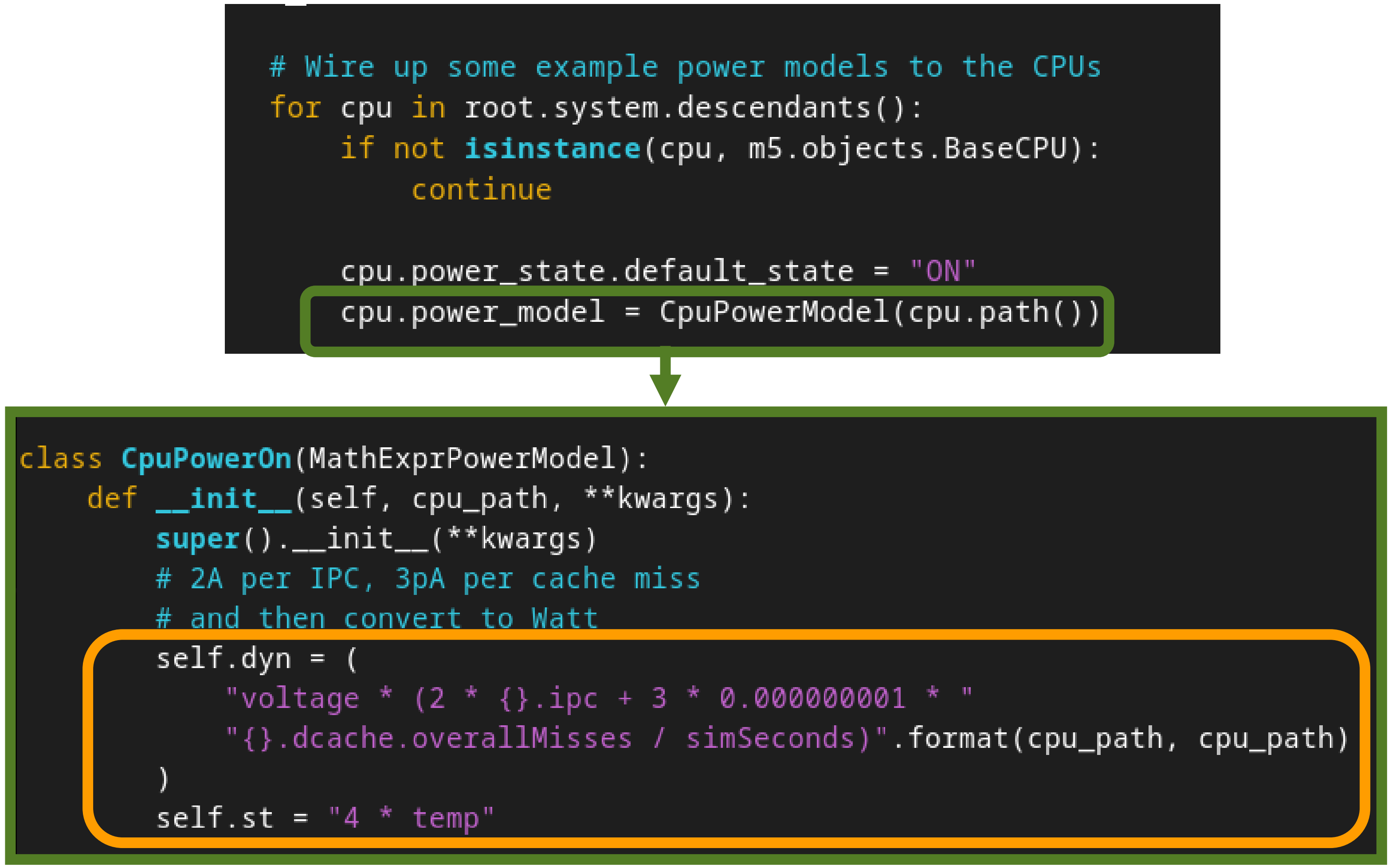
- Motivation
- **Background**
- Design
- Conclusion & Future Work

Why gem5?

- Widely used, popular, open-source simulator
  - Models CPUs, GPUs, and accelerators in a single ecosystem
  - Supports many layers of the computing stack – enables deep co-design
  - **Extensible** through C++/Python modules
- Significant software engineering effort
  - Validated, cycle-level fidelity across a range of components
  - Rigorous functional regression testing ensures stability
    - On-going work: performance regression testing
  - Stable interfaces reduces churn

Current Power Modeling in gem5

- Power modeling API takes user-defined equations as strings
  - **Simulation statistics** passed in as variables (e.g., cache hits)



```
# Wire up some example power models to the CPUs
for cpu in root.system.descendants():
    if not isinstance(cpu, m5.objects.BaseCPU):
        continue

    cpu.power_state.default_state = "ON"
    cpu.power_model = CpuPowerModel(cpu.path())
```

```
class CpuPowerOn(MathExprPowerModel):
    def __init__(self, cpu_path, **kwargs):
        super().__init__(**kwargs)
        # 2A per IPC, 3pA per cache miss
        # and then convert to Watt
        self.dyn = (
            "voltage * (2 * {}.ipc + 3 * 0.000000001 * "
            "{}.dcache.overallMisses / simSeconds)".format(cpu_path, cpu_path)
        )
        self.st = "4 * temp"
```

Current Power Modeling in gem5 (Cont.)

- Power modeling API takes user-defined equations as strings
  - **Simulation statistics** passed in as variables (e.g., cache hits)
  - **Limitation**: Difficult for users to express complex functions, novel models
- Partial McPAT integration
  - But not updated in many years …
- Partial DVFS & thermal support
- Some ISAs have better power support (ARM [Reddy PATMOS'17])
  - But also not updated in many years …

Foundation to build off/learn from

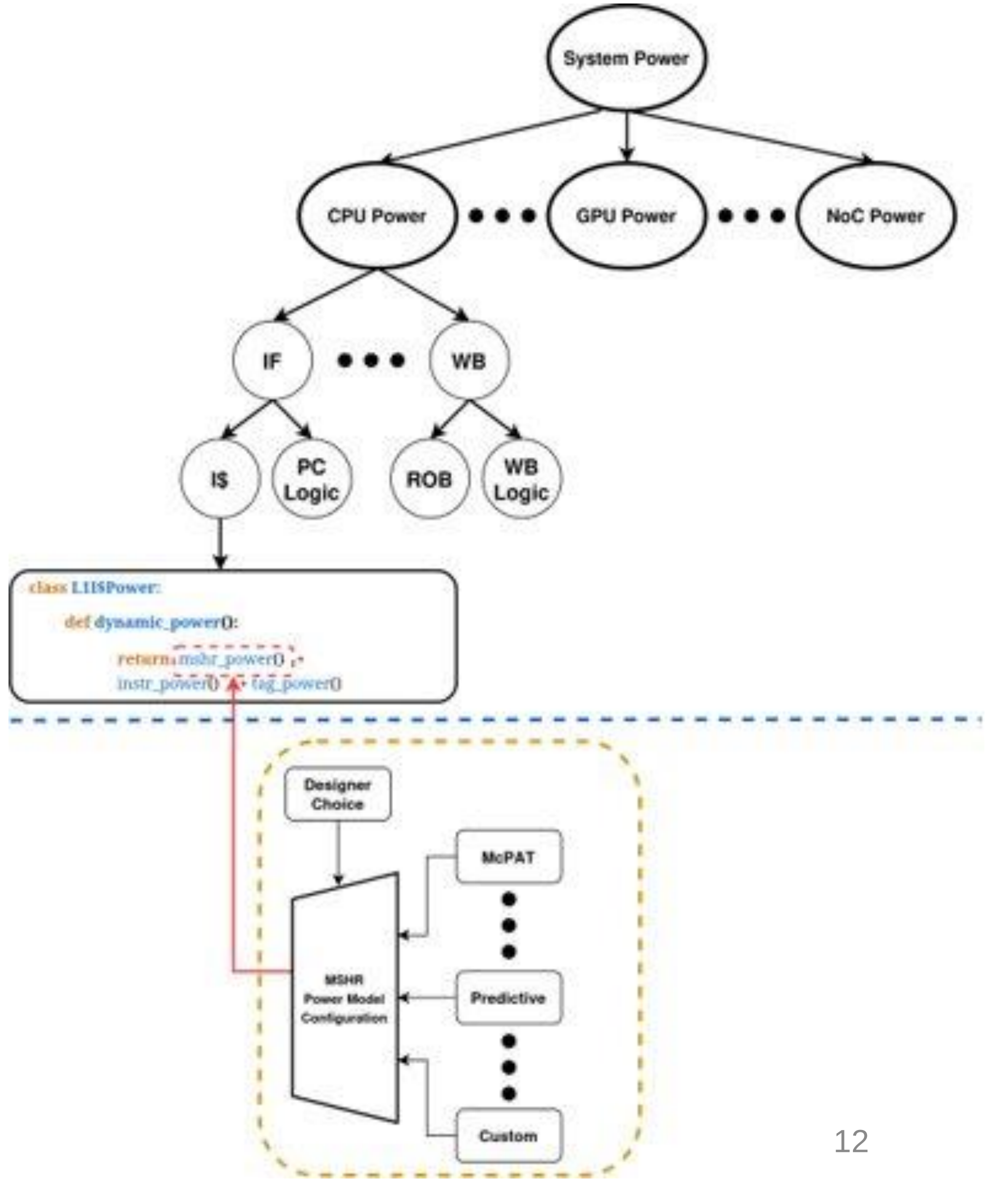
Outline

- Motivation
- Background
- **Design**
- Conclusion & Future Work

Extending the Power Modeling API

- ***How should we extend the power modeling API?***

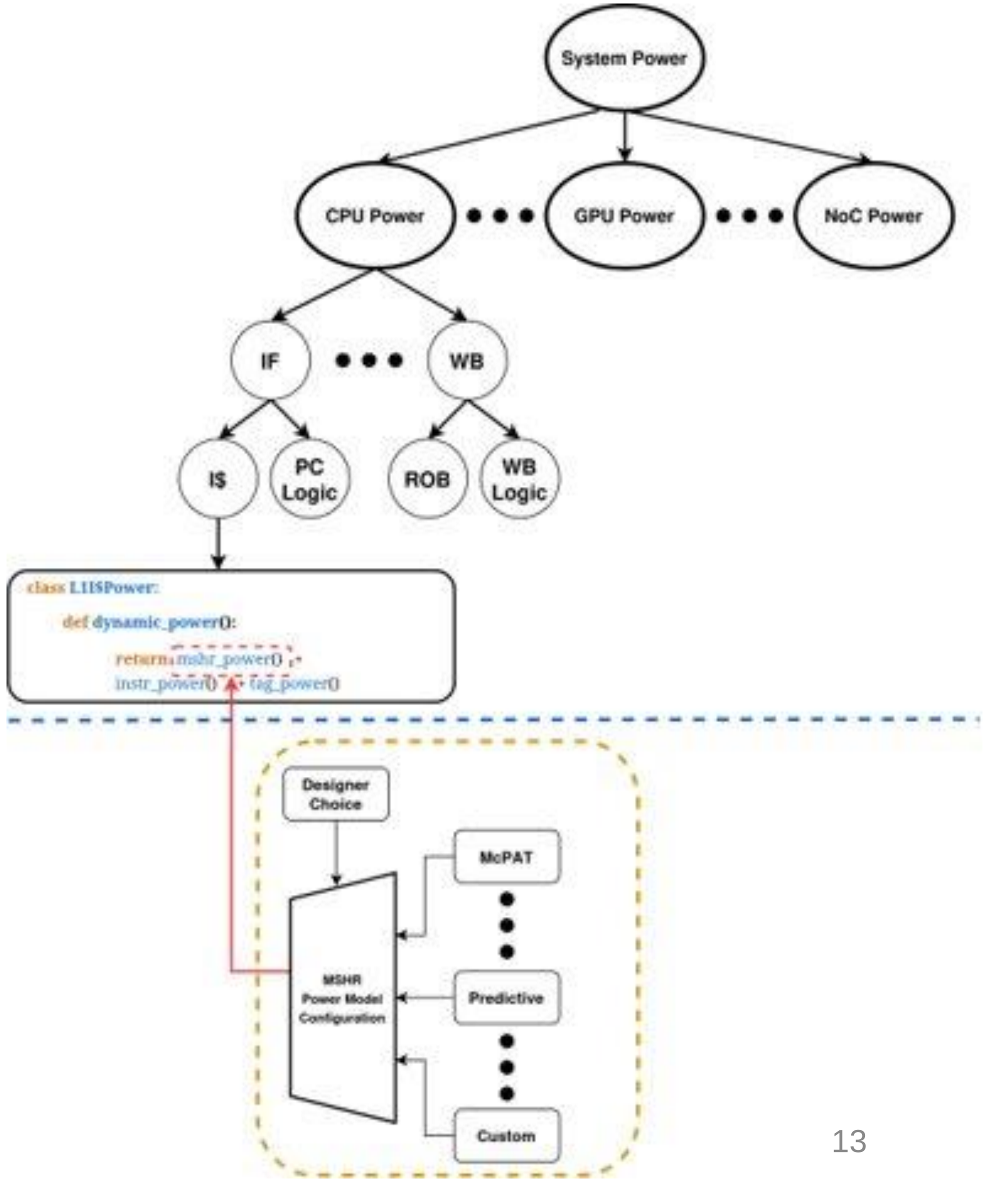
- **Give the user an interface which provides fine-grained customization**



```
class L1ISPower:
    def dynamic_power():
        return mshr_power() +
        instr_power() + tag_power()
```

Extending the Power Modeling API (Cont.)

- **Hierarchical system of components**
  - Overall power model: sum of components
  - Separate simulator, power model
- Break model into 3 Key Pieces:
  - Simulator organizes hardware into sub-groups (e.g., known good models)
  - Power model(s): express static, dynamic power per component
  - Interface: pick between power models for a given component



```
class L1ISPower:
    def dynamic_power():
        return mshr_power() +
        instr_power() + tag_power()
```

Putting It All Together

- **Uses gem5’s Python front-end**
  - Can change power model choice as easily as cache size (no recompile)
  - Richer way to add support, easy to modify
  - Could use different power models for different components
- **Architecture-agnostic**
  - New power model “just” provides static, dynamic power values per component
  - Simulator handles rest of integration (plug-and-play)
- Can integrate and compare/validate different power models
  - **Existing** power models (e.g., McPAT)
  - **Pre-built** power models
  - **Custom** power models (e.g., in-house)

Validation

- Created power model for gem5’s MinorCPU
- Integrated this model with gem5’s emitted stats
- Validation: different static, dynamic power values properly impact predicted power at different levels of hierarchy
- Ongoing: extend to additional components

Outline

- Motivation
- Background
- Design
- **Conclusion & Future Work**

Conclusion

- Future systems need to balance power, performance even more
- But power models are out-of-date, brittle, or proprietary
- Insight: decouple simulator power model integration, power model
  - Simulator devs: focus on how power should be integrated …
  - … without worrying about specifics of underlying power model
- Potential Benefits:
  - Easily support & simple to change between many different power models
  - Better maintainability – separate power model and simulator design
  - Easier to integrate new power models (e.g., for novel accelerators)
- **Make power modeling as easy as performance modeling**