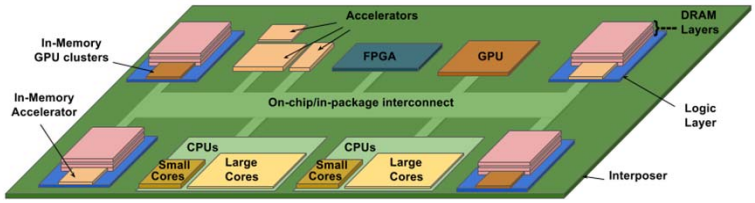
Predictive Modeling for Heterogeneous System Design

Andreas Gerstlauer
System-Level Architecture and Modeling (SLAM) Group
Electrical and Computer Engineering
The University of Texas at Austin
http://slam.ece.utexas.edu

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

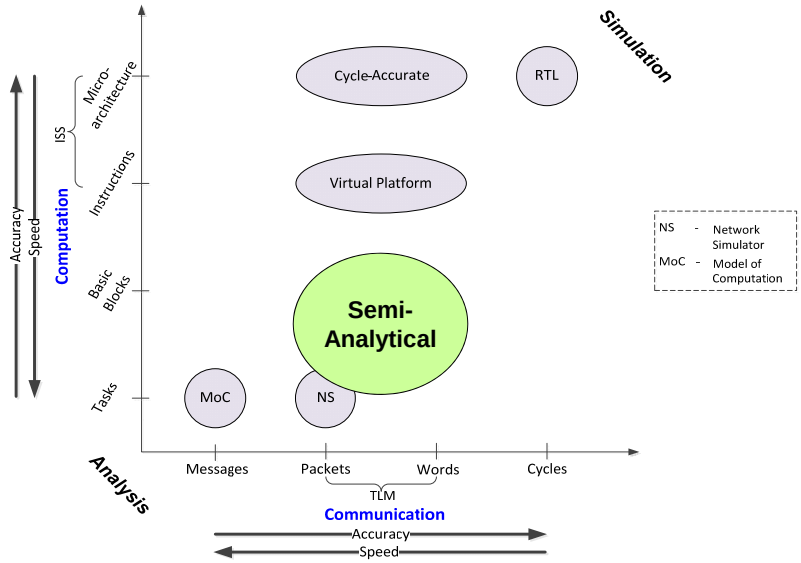
Heterogeneous Systems

- **System complexities and design challenges**
  - Applications and architectures

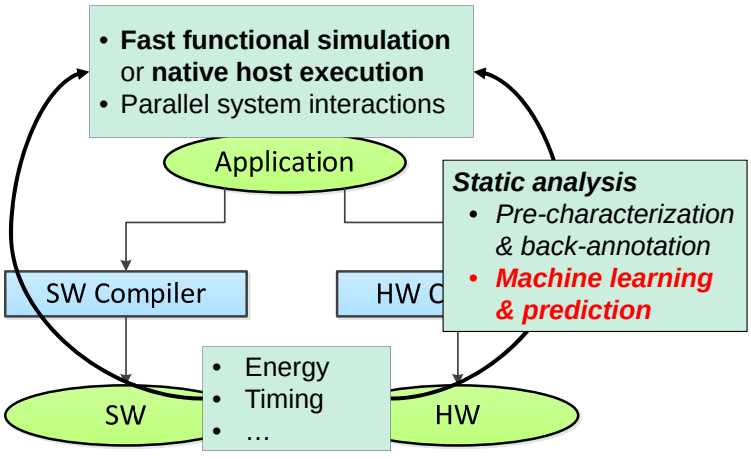


- **Models and tools**
  - For application programmers
  - For operating systems
  - For system architects

Modeling Space



Semi-Analytical Modeling



Learning-Based, Predictive Models

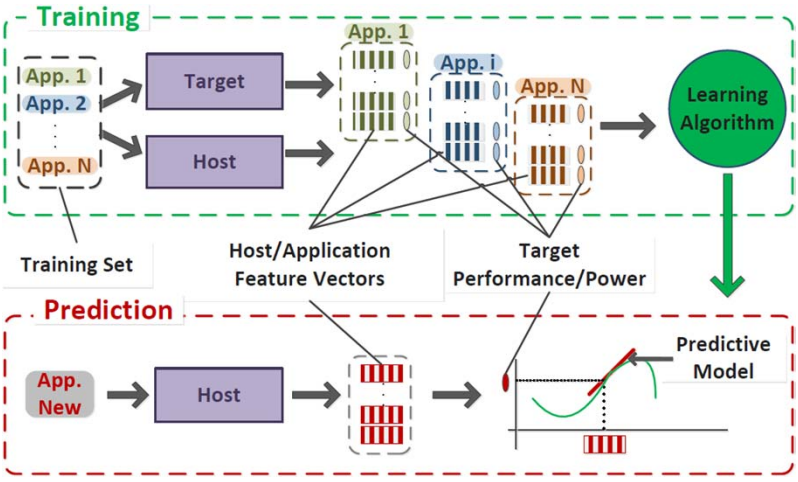
- **Modeling challenges**
  - Dynamic effects in modern systems (uArch, memory, OS)
  - Hard to capture analytically and statically
  - How to provide accuracy w/o detailed, slow simulation?

➢ **Intuition**
  - Performance and power on two platforms is correlated
  - Such correlations are non-trivial
  - Can we learn them?

➢ **Predict for target while running natively on host**
  ➢ Bridge gap between analysis and simulation

Learning-Based, Predictive Models

- **Learning-based analytical cross-platform prediction (*LACross*, w/ L. K. John) [IJPP'17]**



Software Models

- **Predict on target CPU while running on host CPU**
  - Using hardware counters on host as features
  - Predict target performance and power
  - At program phase level

➢ **Instrumentation-based [DAC'16, IJPP'17]**
  - Compiler-based instrumentation at basic block granularity
  - Collect features and train/call model every *N* basic blocks

➢ **Sampling-based [DATE'17]**
  - Source-oblivious at binary level using timer interrupts
  - Sample alignment during training

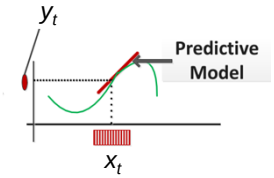
Learning Formulation

- **Given training set (x_i, y_i)**
  - $x_i \in \mathbb{R}^d$: d-dimensional counter feature vector from host
  - $y_i \in \mathbb{R}$: reference performance/power on target
- **Want to find function $F(x_i) \approx y_i$**
  - Fundamentally non-linear

➢ **Locally linear approximation $F_t(x_t)$ at input x_t**

$$F_t(x_t) = \theta_t^T x_t$$

  - Around neighborhood of x_t
  - LASSO regression to solve for θ_t



Experimental Setup

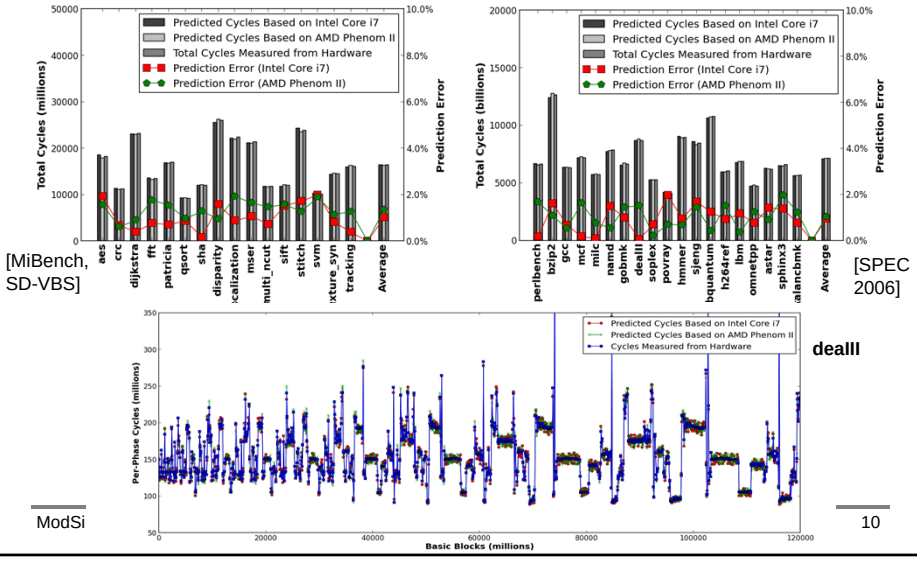
- **Platforms**
  - Target: Samsung ARM A9/A15 Exynos
  - Host: Intel Core i7 / AMD Phenom II
- **Host counters**
  - Instrumentation-based: 14 / 8 counters
  - Sampling-based: 6 counters

Host Counters
Instructions
Cycles
Total Cache Misses
Total Cache References
Total Branches
Total Branch Misses

- **Training set**
  - 157-284 programs of ACM-ICPC competition
- **Test set**
  - 7 programs from MiBench and 8 programs from SD-VBS
  - 19 programs from SPEC CPU 2006
  - 13 Java & Python benchmarks from DaCapo/PyBench

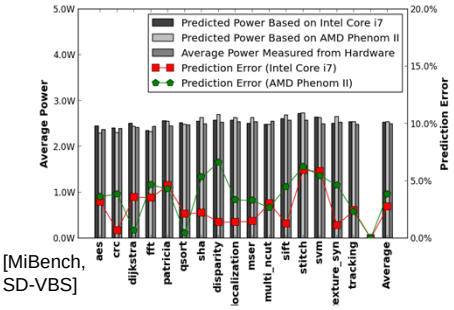
LACross Performance Results

- **95% per-phase accuracy @ 500 MIPS speed**
  - Phase granularity of 5,000 basic blocks

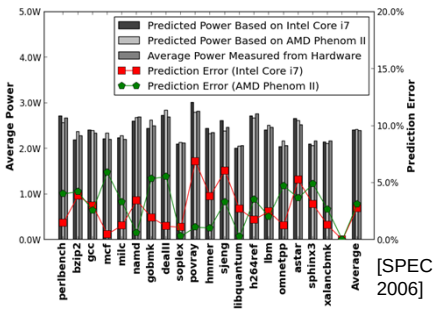


LACross Power Results

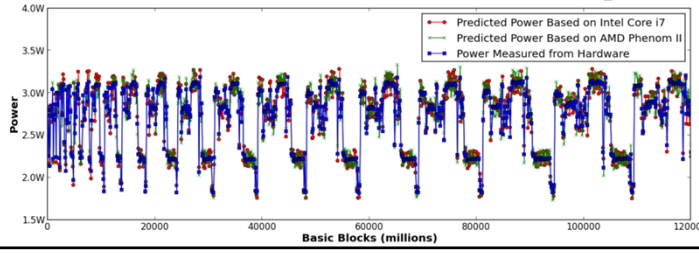
- **90% per-phase accuracy @ 600 MIPS speed**
  - Phase granularity of 20,000 basic blocks



[MiBench, SD-VBS]



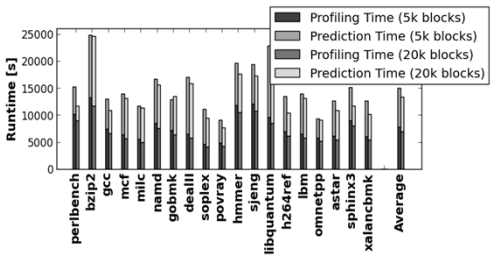
[SPEC 2006]



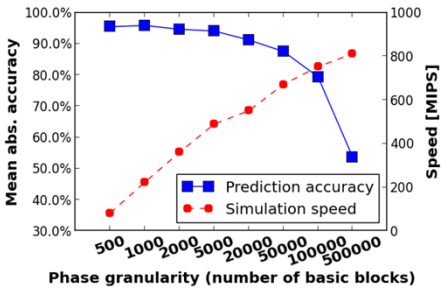
dealII

Instrumentation-Based Speed & Accuracy

- **Accuracy & speed vs. phase granularity**
  - Finer granularity requires more prediction overhead
  - But: more & better training data w/ finer granularity
    - Phase similarity: number of unique phases *decreases* linearly
  - Runtime also limited by hardware counter support on host
    - Multiple runs needed to collect all counters

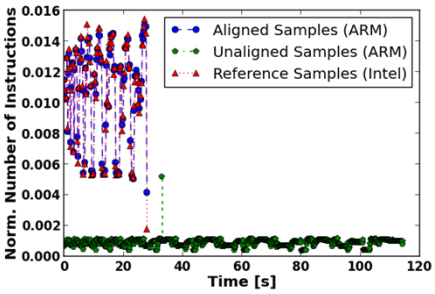


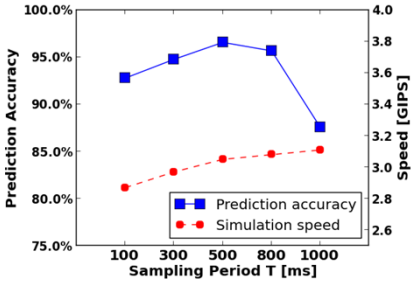
[SPEC 2006]



Sampling-Based Results

- **Speed & accuracy increase with coarser host sampling *T***
  - Better alignment, until lack of training data (*T* > 500ms)

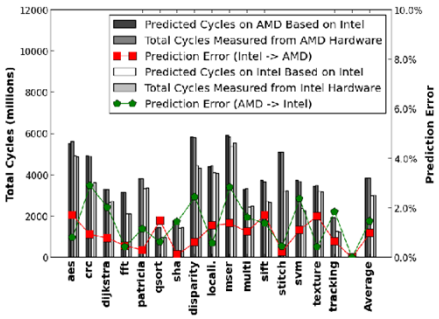


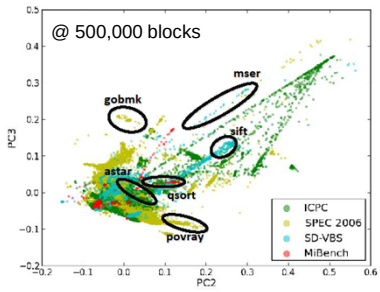


➢ **96% accuracy @ 3 GIPS (*T* = 500 ms)**
  - No instrumentation overhead (6x faster)
    - Fewer counters, coarser granularity, but requires more training
  - 2x faster than running native on ARM target

Software Prediction Questions

- **Host/target pairs**
  - ARM from x86, x86-to-x86
  - From simple to complex?
- **Prediction features**
  - Which counters?
  - Other information?
- **Training set**
  - Larger granularity requires larger training set
  - Optimal training set?
  - ➢ Generate synthetic training set (Genesys) [SAMOS'16]



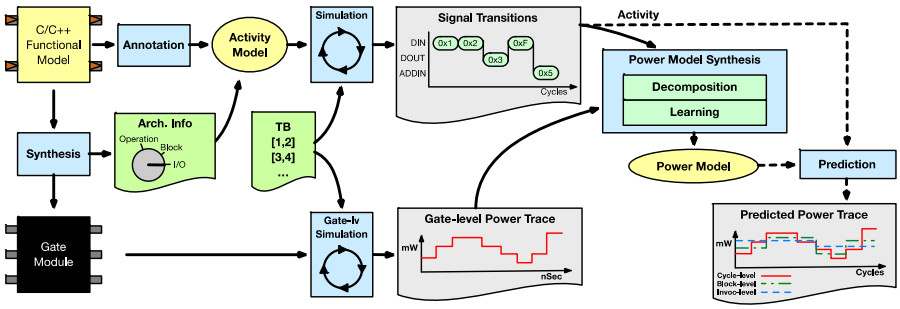


Other Predictive Cross-Platform Models

- **GPU performance models (Intel/UC Riverside, P. Brisk)**
  - GPU-to-GPU prediction using performance counters
  - Commercial GPUs to predict pre-silicon hardware
- **FPGA high-level synthesis models (UC Riverside, P. Brisk)**
  - Predict FPGA performance of code regions of interest
  - Running on host CPU, using hardware counters
- **Heterogeneous ISA models for OSs (UCSD, D. Tullsen)**
  - Predict performance on different CPU cores
  - Use prediction to make OS scheduling decisions
- **CPU benchmark performance models (Harvard, D. Brooks)**
  - Predict benchmark performance from CPU specifications

Hardware Accelerator Models

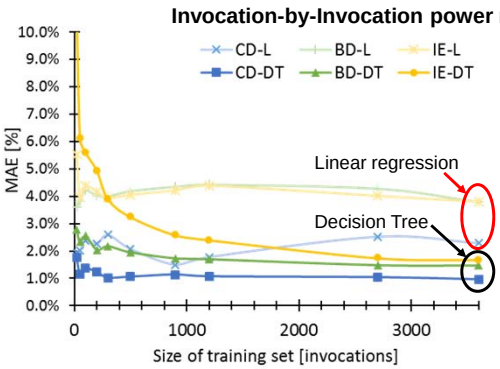
- **Hardware power models**
  - White / grey / black box [DATE'15 / TODAES'18 / ICCAD'15]
  - Operation / block / I/O activity from functional simulation
  - Predict gate-level power at cycle / block / invocation level



➢ **Data-dependent, Fast**

Learning Formulation

- **Dedicated, domain-specific learning formulations**
  - Structural model decomposition & feature selection
- **Advanced, non-linear regression models**
  - Traditional, not deep learning w/ small training size

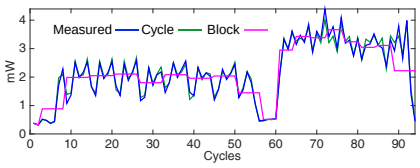


- L : Linear regression
- DT: Decision Tree regression

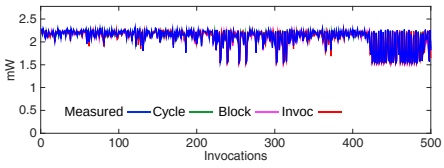
- CD : Cycle decomposed model
- BD : Block decomposed model
- IE : Invocation ensemble model

Hardware Modeling Results

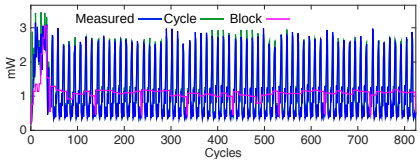
- **Pipelined 2D-DCT**
  - Cycle-by-cycle trace



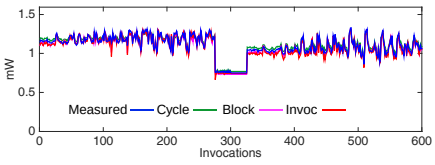
  - Invocation-by-invocation trace



- **Pipelined HDR weight comp.**
  - Cycle-by-cycle trace



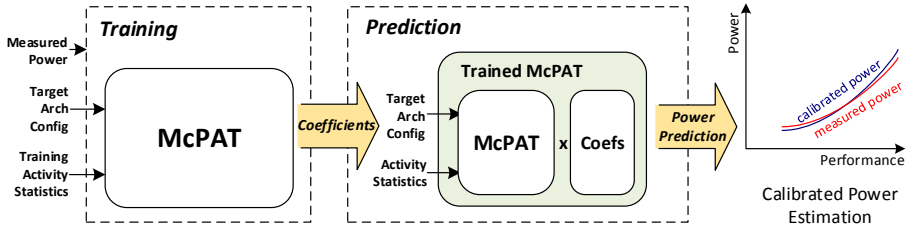
  - Invocation-by-invocation trace



➢ **> 97% accuracy @ 1Mcycles/s speed**
  - 2,000-10,000x faster than gate-level, 100x-500x faster than RTL

CPU Power Models

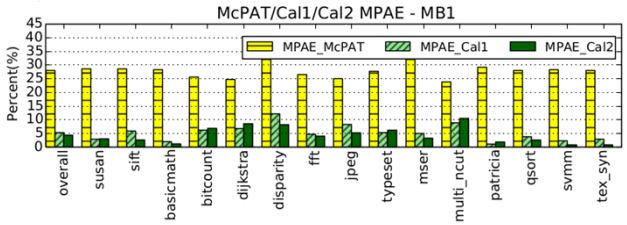
- **PowerTrain [ISLPED'15]**
  - Learning-based calibration of library-based models
  - Against post-silicon hardware measurements



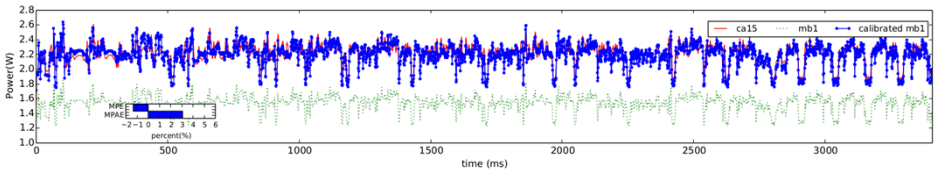
- **Learn CPU micro-architecture models (on-going)**
  - At cycle-accurate and component granularity
  - From gate-level training

PowerTrain Results

- **Comprehensive & accurate power prediction**
  - 15-fold cross-validation w/ 4% avg. MPAE



  - Spec CPU 2006 **gcc** trace w/ 3% MPAE



➢ **General, automatic post-silicon power model calibration**

On-Going Work (w/ L. John, P. Brisk)

- **Cross-platform models for heterogeneous system design**
  - Model accuracy vs. speed, learning formulations
  - Prediction targets, host/target combinations
  - Prediction metrics (reliability, thermal, …)
  - Model interpretability, feature ranking
  - ➢ Architecture design, programming, runtime/OS mgmt.
- **Prediction-enhanced simulation**
  - Combine statistical sampling with prediction
- **Prediction for time series data**
  - Program phase behavior, runtime management
- **Architecture-independent prediction**
  - Predict from source code or IR features

Summary & Conclusions

- **Predictive cross-platform modeling**
  - Run on a host, predict for a target
  - Advanced machine learning to capture correlations
  - Combination of simulation (host) & analysis (learning)
- **Learning-based performance and power prediction**
  - CPU-CPU, GPU-GPU, accelerators/FPGAs, …
  - More than 95% accuracy at native host speeds
  - Programming, OSs, architecture definition
- **Extensions to other domains**
  - Hybrid simulation and prediction
  - Time series data, other metrics and targets
  - Architecture-independent prediction
  - …