Energy Characterization of Embedded Processors for Software Energy Optimization

Tohru Ishihara
Kyoto University

Lovic Gauthier
Kyushu University

Agenda

- Introduction
- Processor Energy Characterization
- Software Energy Optimization

Motivation

- Portable systems execute power hungry applications
- Most functions are implemented by software
- Energy depends on the software running on processor systems

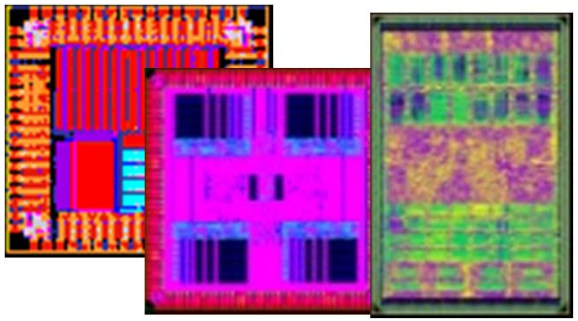
Software energy analysis is necessary for reducing the energy consumption of embedded systems

www.princeton.edu/~wolf/

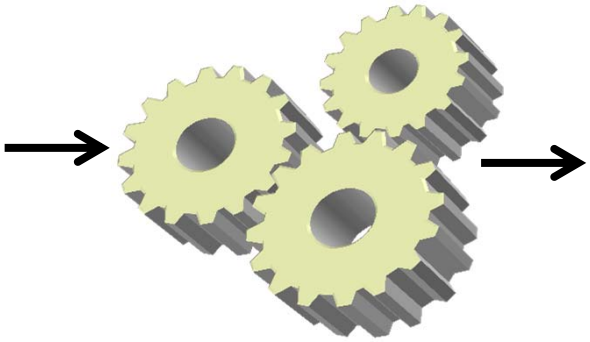
Energy Characterization

- Calculate the energy dissipated for a hardware event using post-layout simulation of a target processor system
- The hardware events include data read, data write, cache miss, program branch execution and so on

Microprocessors & Memory subsystems



Gate-level simulator



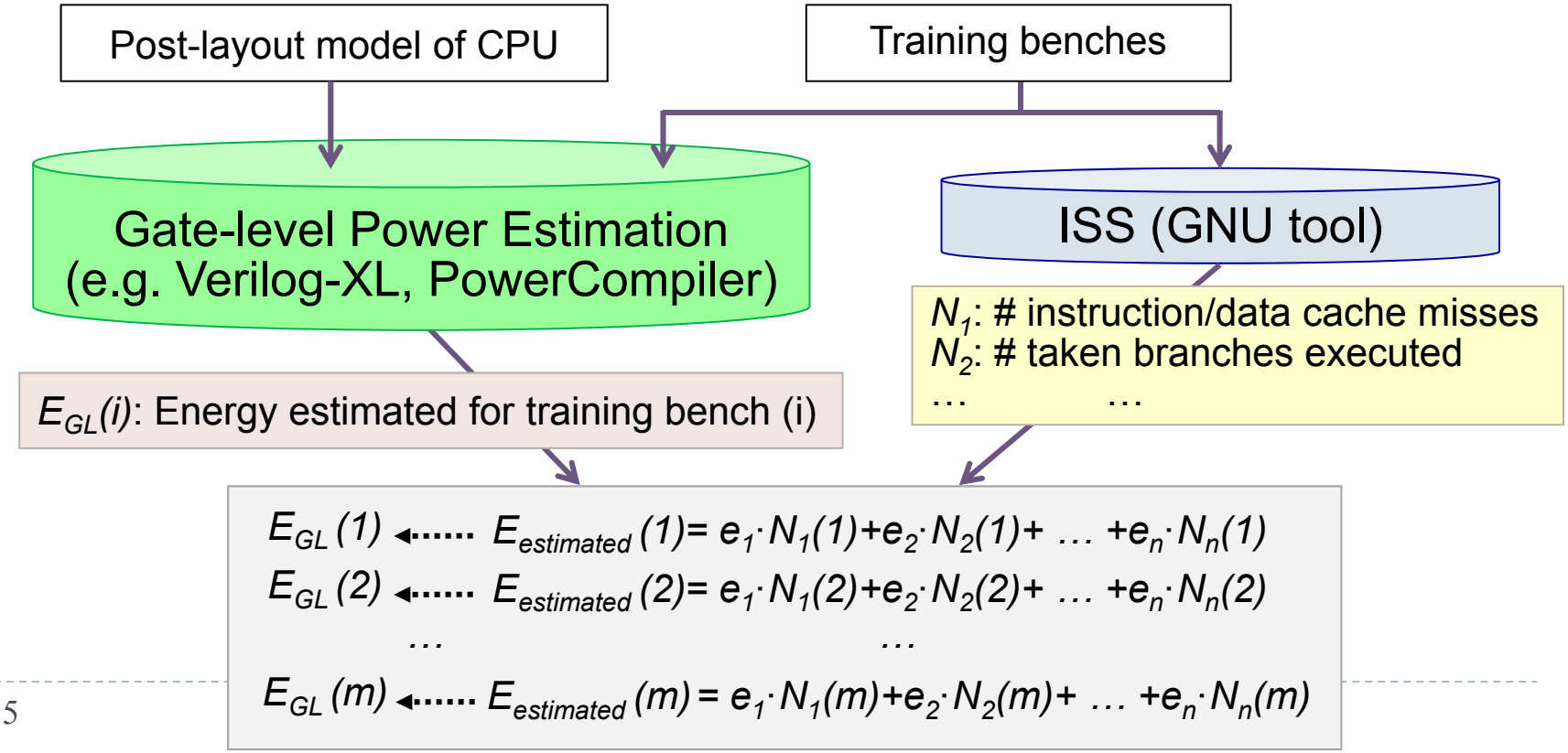
Energy Lookup Table

HW events	Energy
Data read	15 nJ
Cache miss	32 nJ
Branch exec.	58 nJ
…	…

D. Lee, T. Ishihara, M. Muroyama, H. Yasuura, F. Fallha, "An Energy Characterization Frame work for Software-Based Embedded Systems," Proc. of ESTIMedia, pp.59-64, Oct., 2006.

Our Approach

- Run a number of training benches on ISS of a target processor
- Run the same benches on a post-layout model of the processor
- Fit a linear model through regression analysis

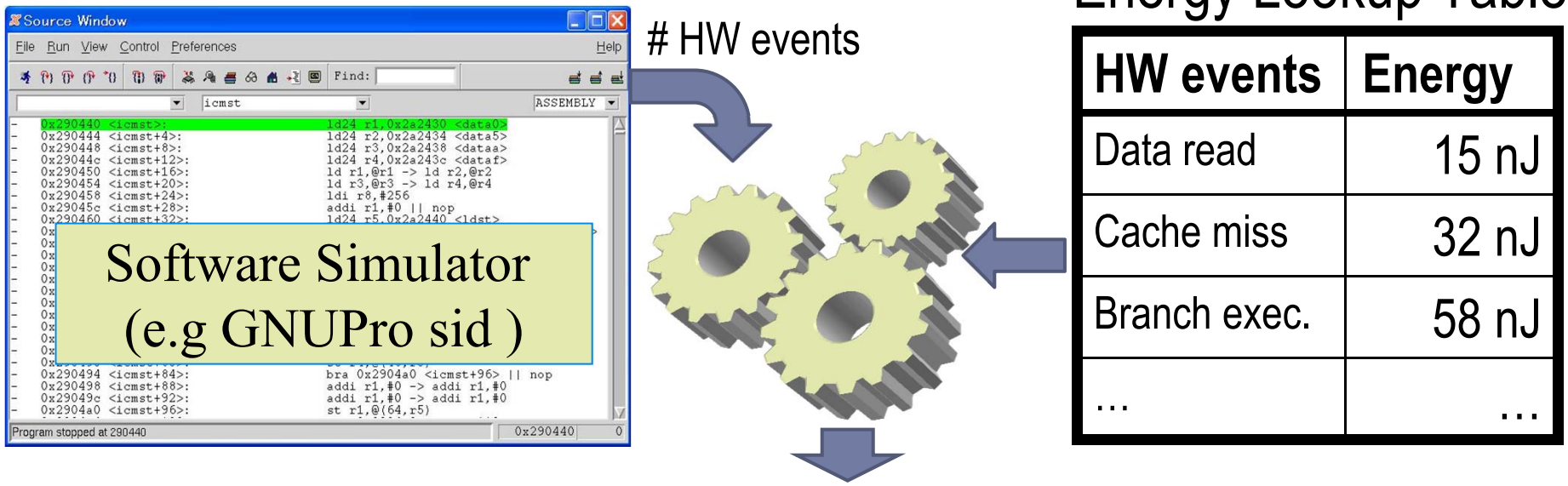


Software Energy Analysis

- A linear expression for the software energy consumption

$$E_{estimated} = \sum e_i \cdot N_i \quad e_i : \text{energy of HW event}, \quad N_i : \#\,\text{HW events}$$

- The number of the hardware events should be counted by instruction set simulator (ISS)

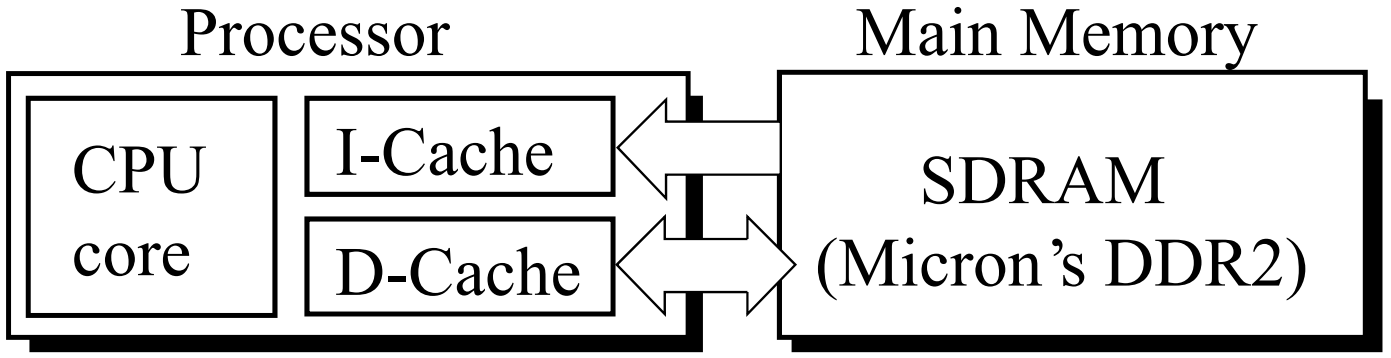


Energy Lookup Table

HW events	Energy
Data read	15 nJ
Cache miss	32 nJ
Branch exec.	58 nJ
…	…

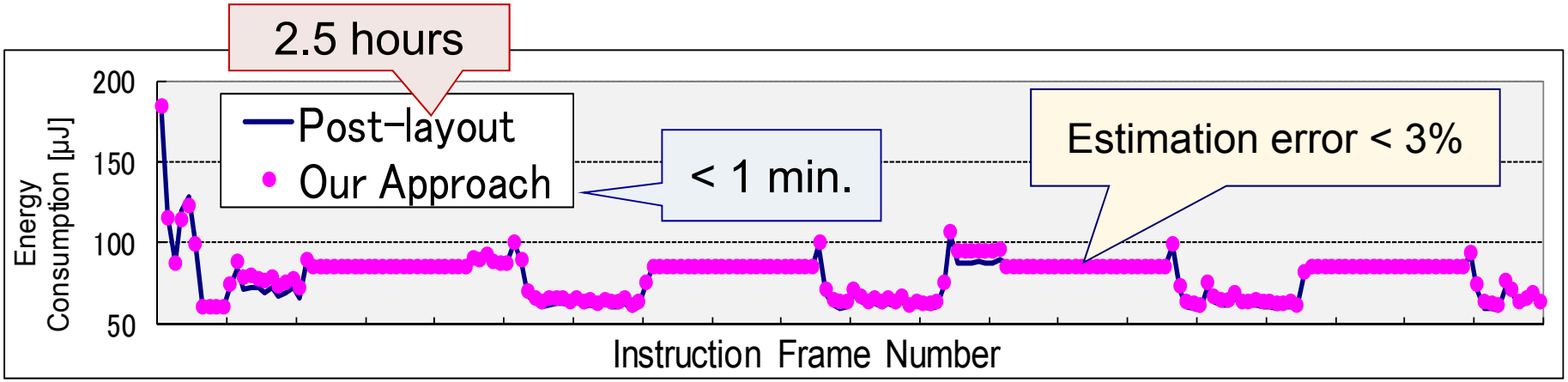
Accuracy Evaluation

- Target system

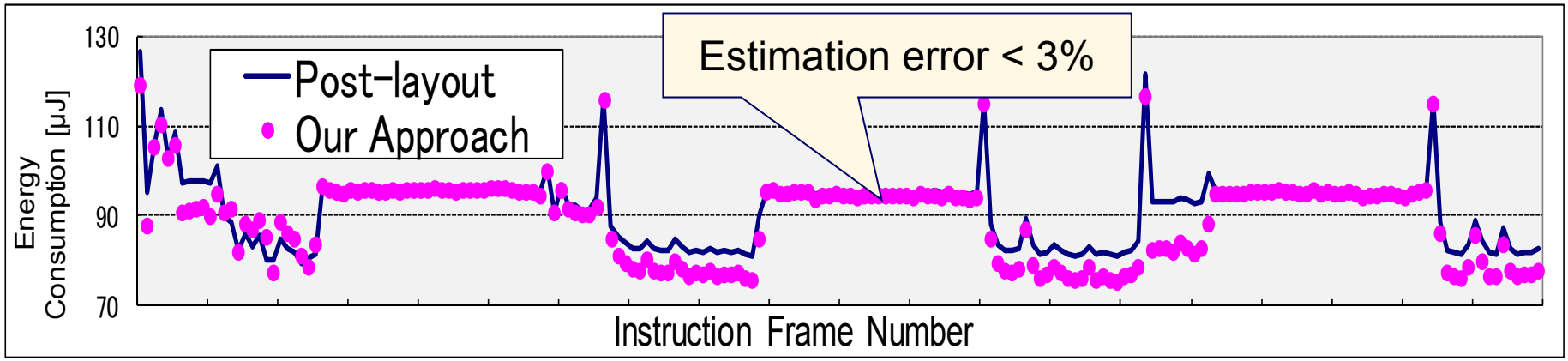


  - Processors
    - M32R-II, SH3-DSP (Renesus)
    - MeP (Toshiba)
  - 0.18μm CMOS library

Results for M32R-II & SH3-DSP



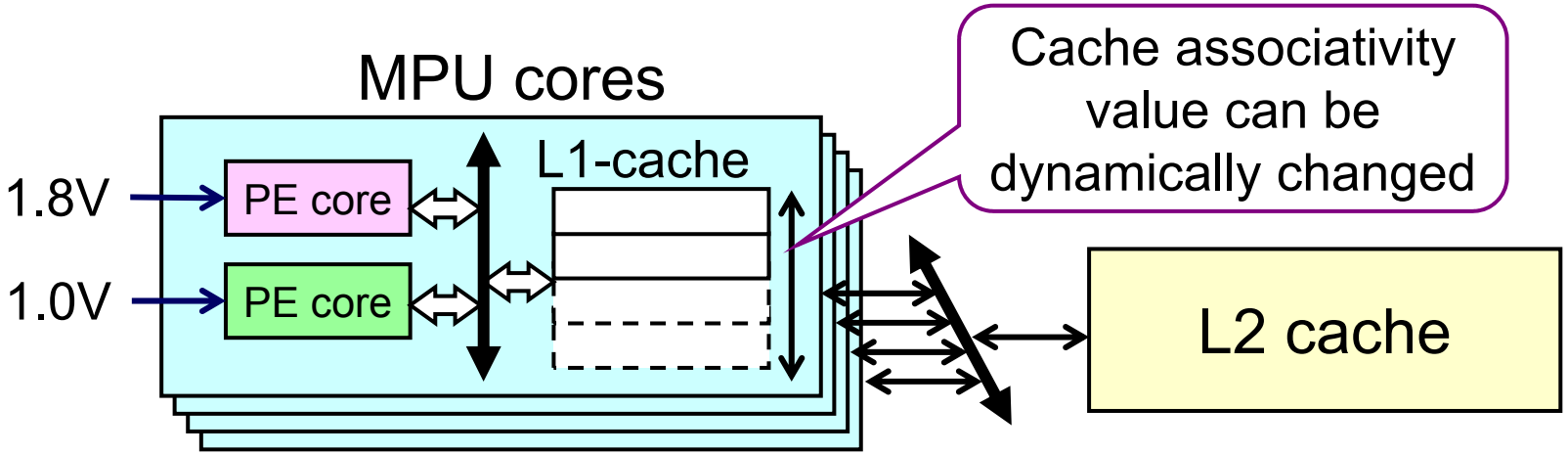
Energy estimation for JPEG encoder executed on a **M32R-II processor**



Energy estimation for JPEG encoder executed on a **SH3-DSP processor**

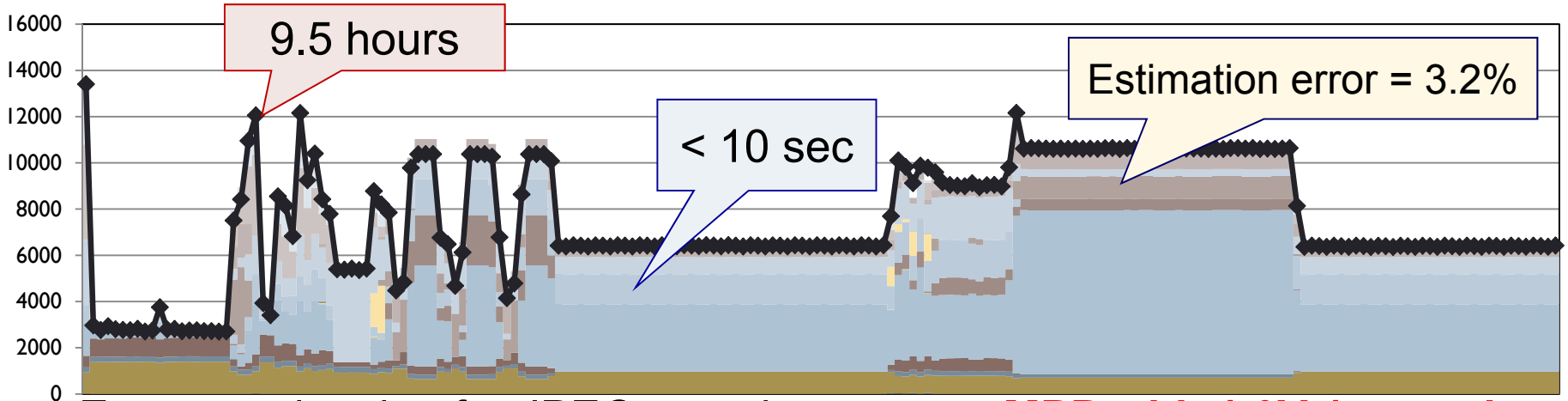
Multi-Performance Processor

- Based on MeP (Toshiba)
  - PEs have the same ISA but differ in their clock speeds and energy consumptions
  - Intra MPU core: a single PE core runs alternatively
  - Inter MPU cores: multiple MPU cores run concurrently

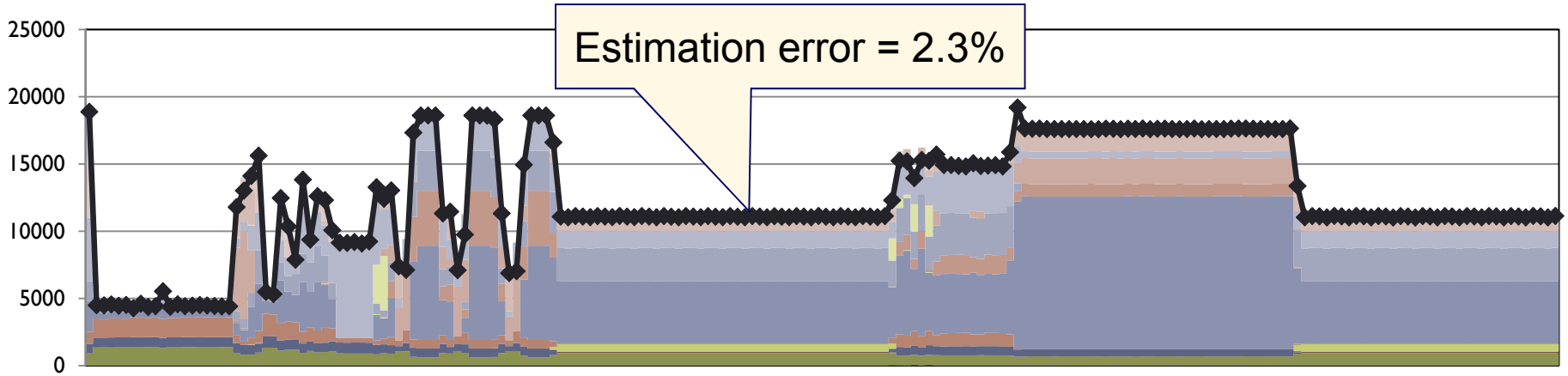


T. Ishihara, “Real-Time Dynamic Voltage Hopping on MPSoCs,” MPSoC 2009, Savannah.

Results for MPP



Energy estimation for JPEG encoder run on a **MPP with 1.0V 1-w cache**

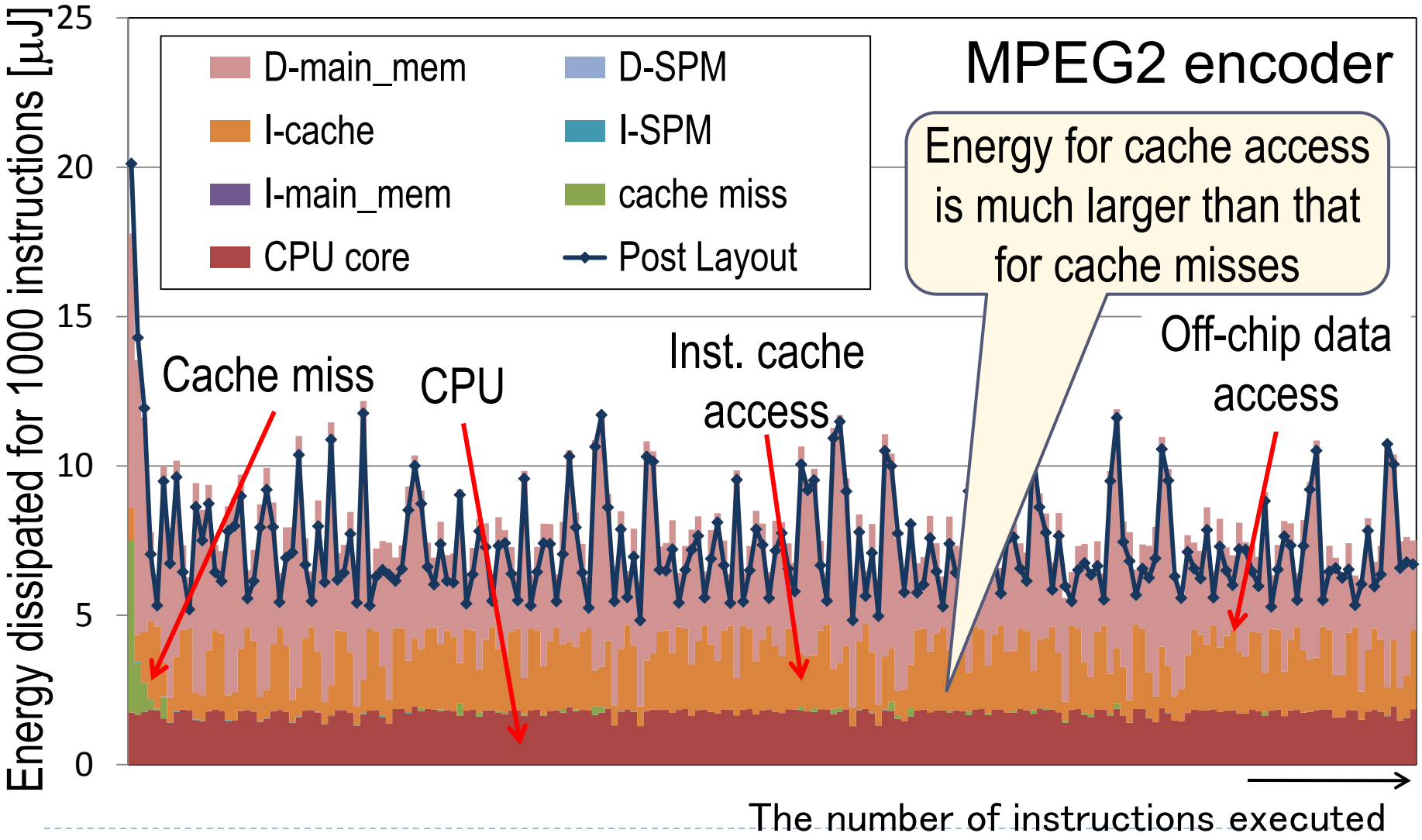


Energy estimation for JPEG encoder run on a **MPP with 1.8V 4-w cache**

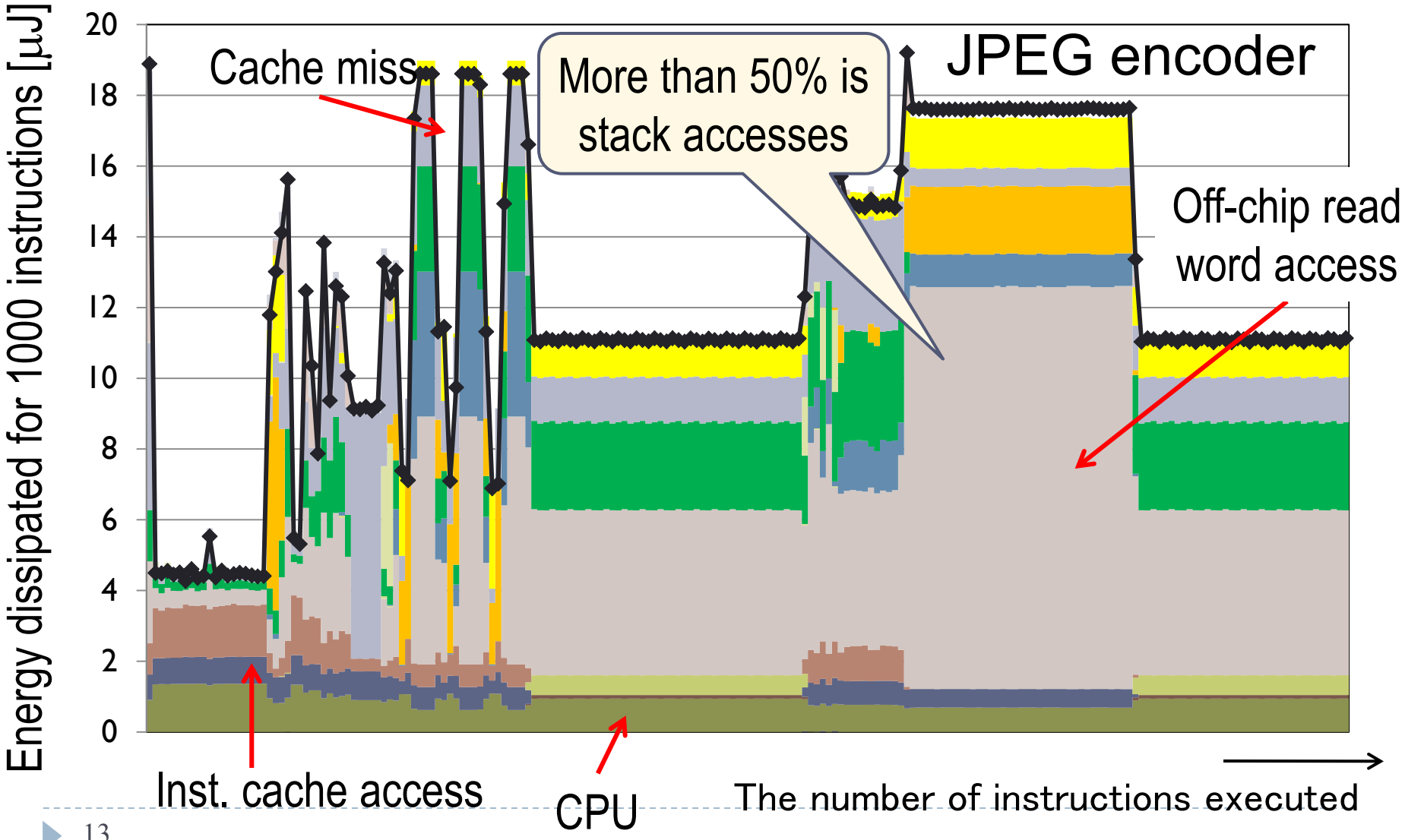
Applications

- Software Energy Analysis
  - ✓ Help finding energy bottleneck
- Software Energy Optimization
  - ✓ Compiler optimization
  - ✓ OS-based power management

Software Energy Analysis (1/2)



Software Energy Analysis (2/2)

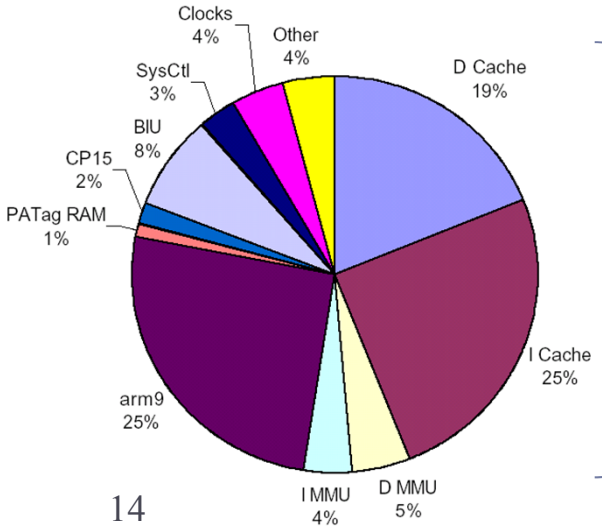


Software Energy Optimization

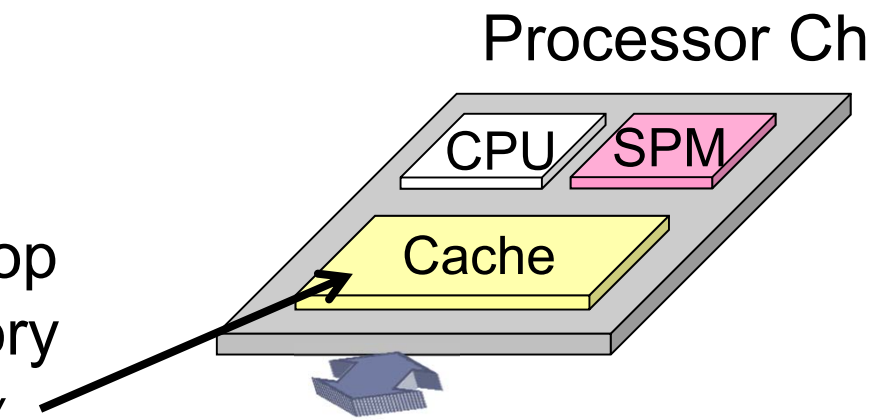
Motivation:

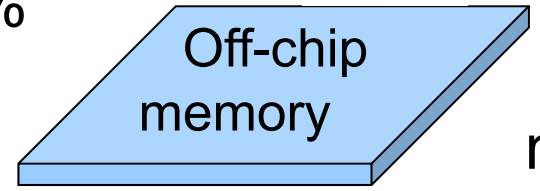
- Memory consumes a large amount of energy
- Memory energy depends on program behavior
- Code optimization contributes to the total energy reduction

Power Analysis of ARM920T



On-chop memory 43%

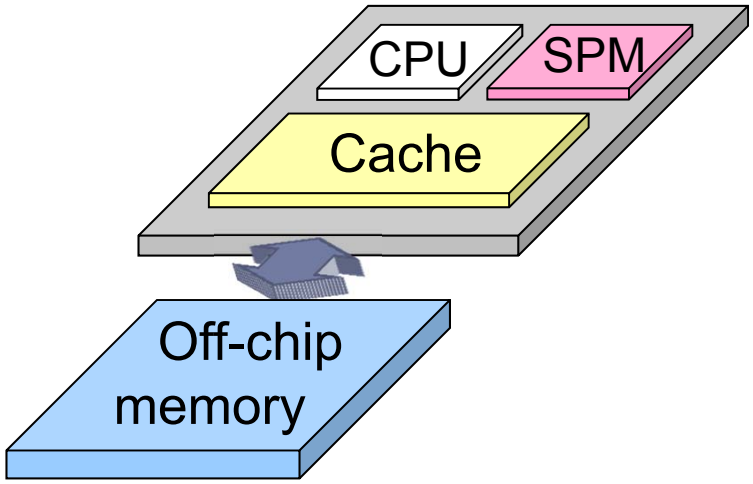




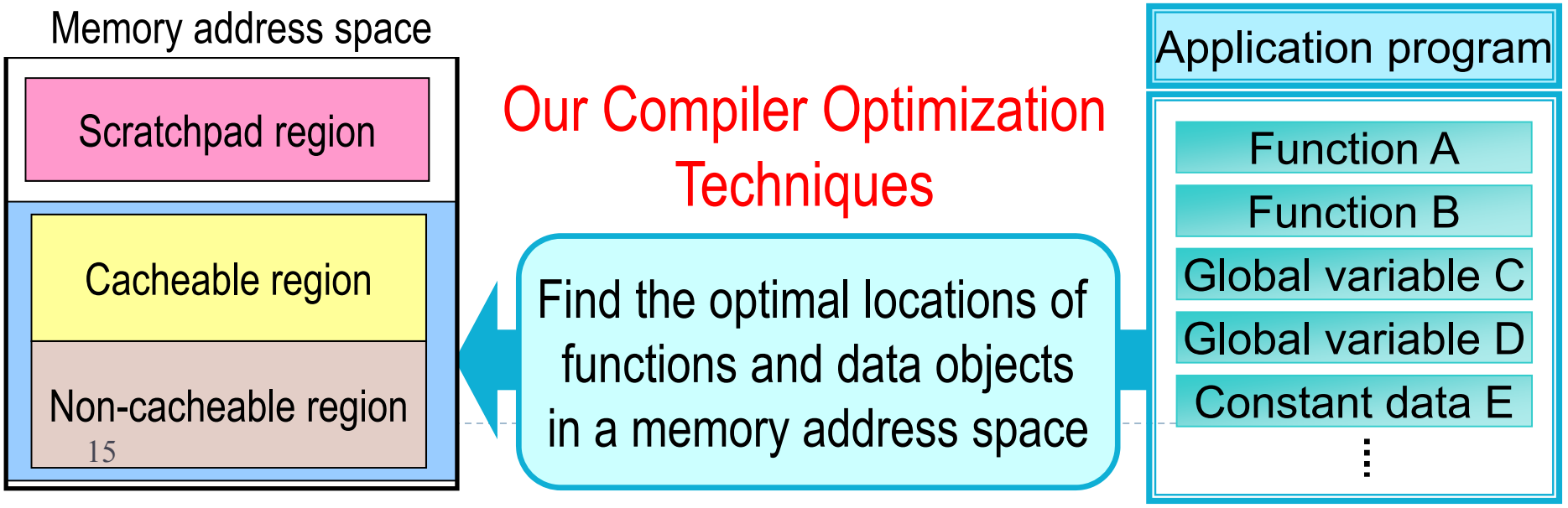
10x of on-chip memory energy

DRAM Chip

Code and Data Allocation



	Size	Latency	Energy
SPM	Small	Small	Small
Cache	Small	Small	Large
Off-chip mem.	Huge	Huge	Huge

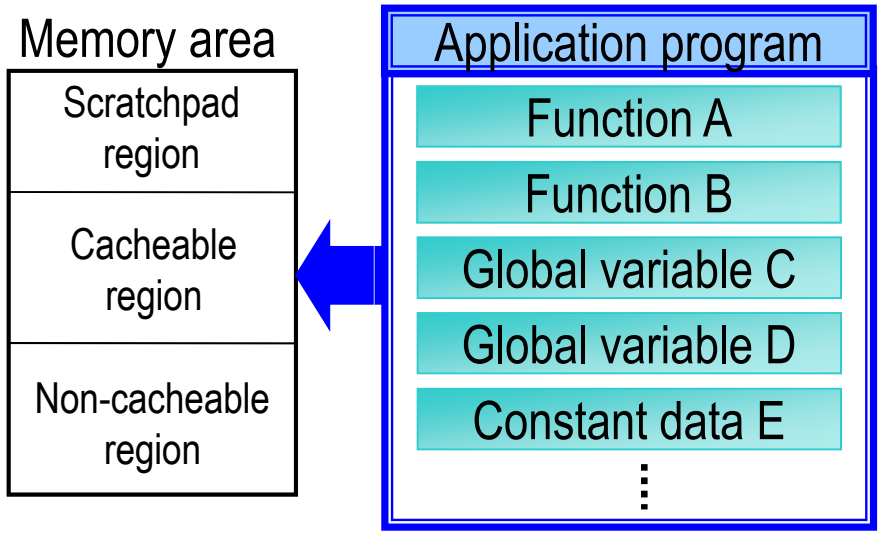


Our Approach

Previous work…

Find code & data placements which
- maximize # SPM accesses
- minimize # cache misses
- minimize # off-chip accesses

Do not always minimize the energy



Our method…

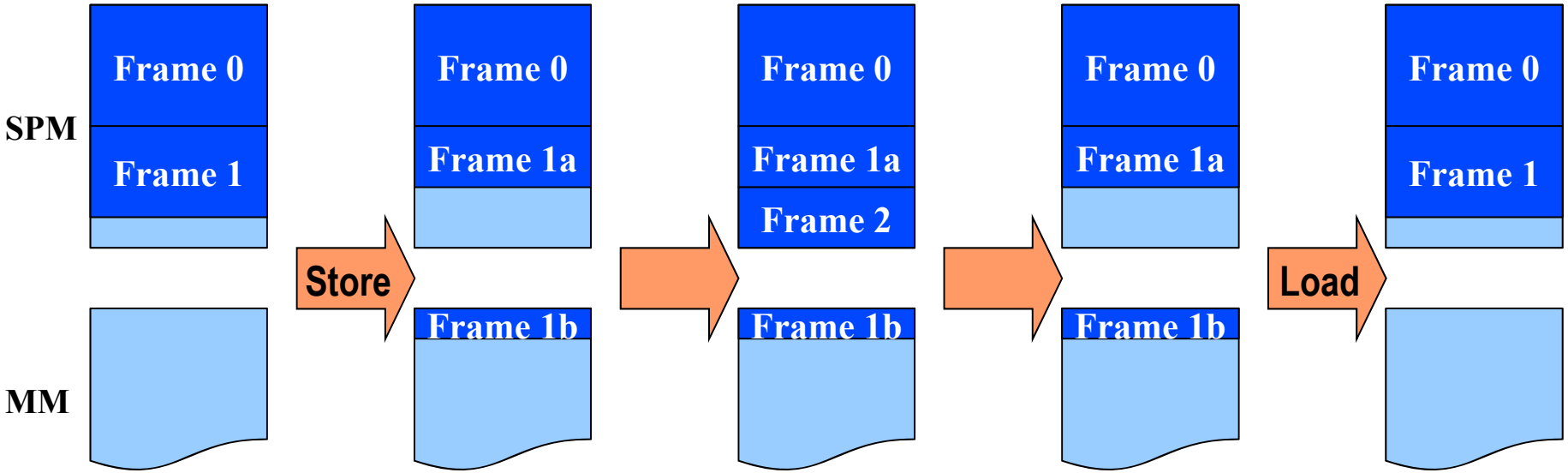
Minimize the total energy consumption estimated by our model through an ILP

Total processor energy reduced by 10%

Y. Ishitobi, T. Ishihara, H. Yasuura, "Code and Data Placement for Embedded Processors with Scratchpad and Cache Memories," Signal Processing Systems 60(2), pp.211-224, August, 2010

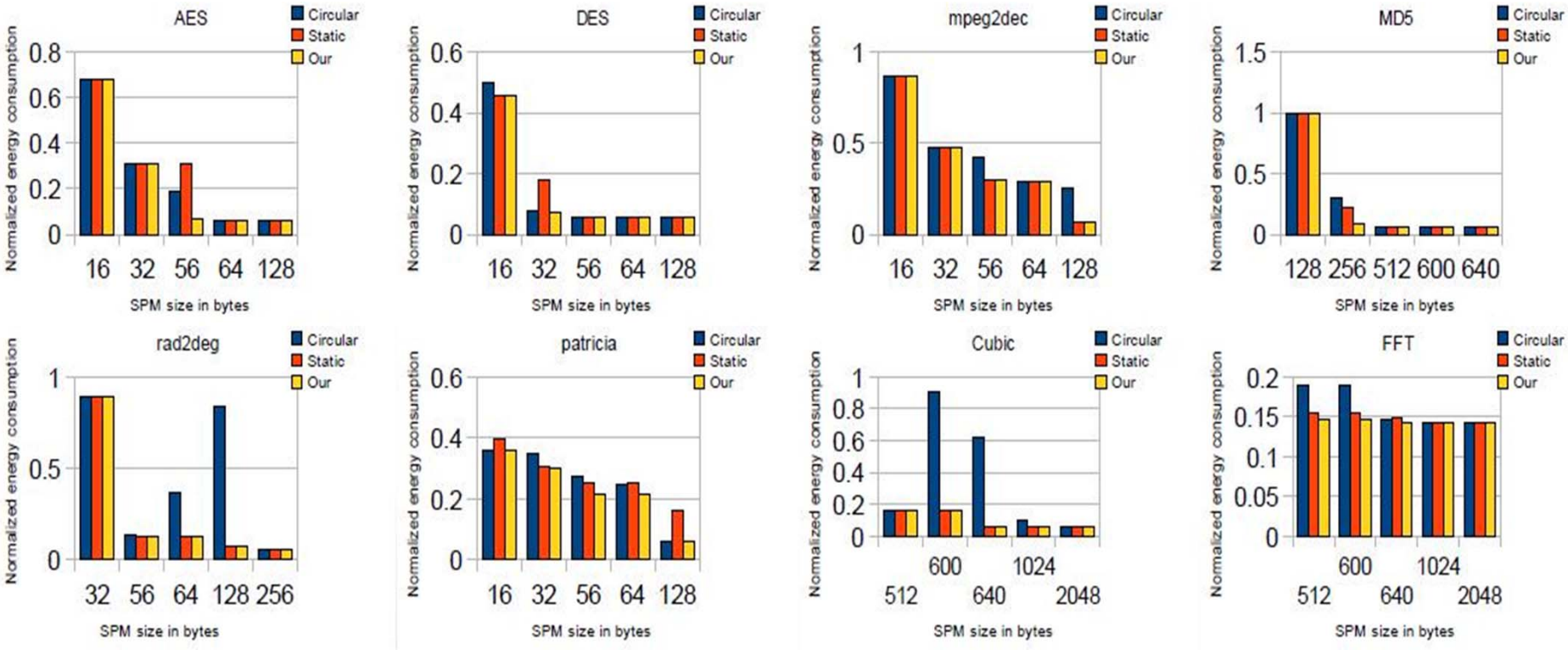
Stack Allocation

- Optimization is done at compile time through an ILP
  - Find the best locations of stack frames based on profiling
  - Store/Load inserted before and after call instructions
- Frames are dynamically moved between SPM and MM
  - Stack frame is generated in SPM when the function is called
  - Store frames into MM if there is no space left in the SPM



Stack Allocation Results

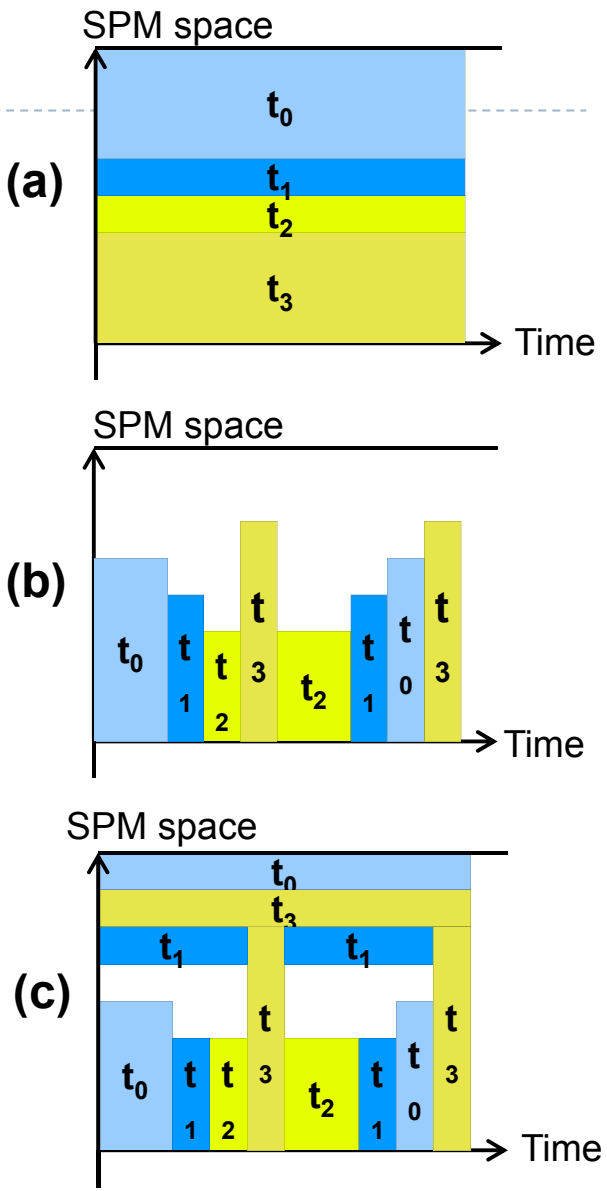
- Circular: Evict oldest frame first if there is no space left in the SPM
- Static: Place frames in the SPM only if SPM does not overflow
- Ours: Placement is optimized by the Integer Linear Programming



L. Gauthier, T. Ishihara, “Optimal Stack Frame Placement and Transfer for Energy Reduction Targeting Embedded Processors with Scratch-Pad Memories,” Proc. of IEEE Workshop on Embedded Systems for Real-Time Multimedia, pp.116-125, Oct., 2009.

Multi-Task System

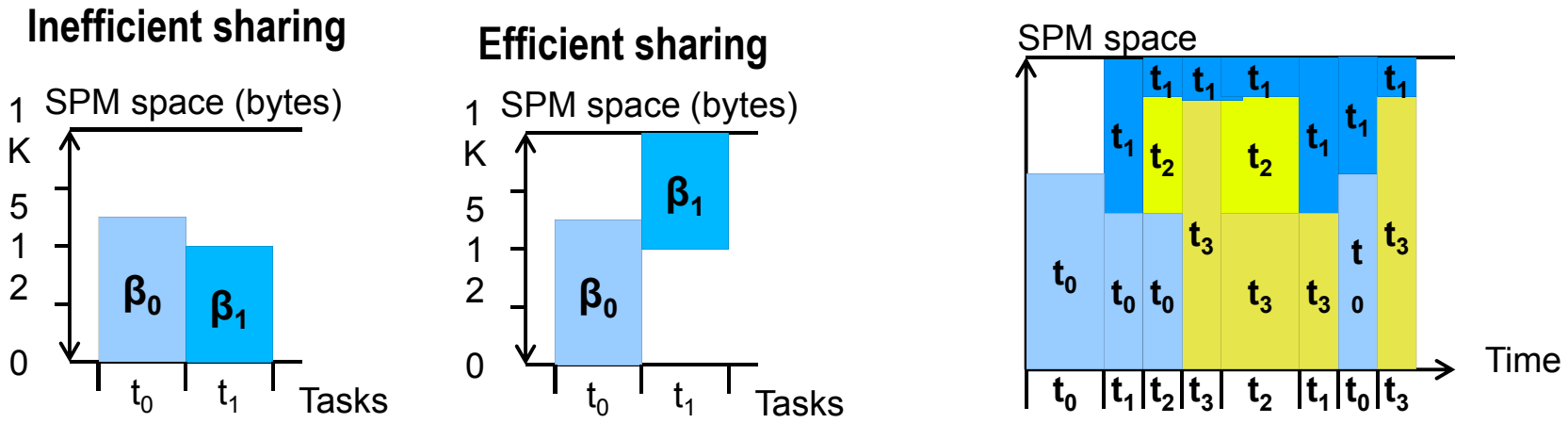
- The SPM is shared among tasks
- Several previous techniques
  - (a) Spatial sharing
    - ✔ No management required
    - ✔ Very small part of the SPM for each task
  - (b) Temporal sharing
    - ✔ Totally of SPM space to each task
    - ✔ SPM update at context switches
  - (c) Hybrid
    - ✔ Best of both approaches
    - ✔ Compile time profile-based assignment to SPM



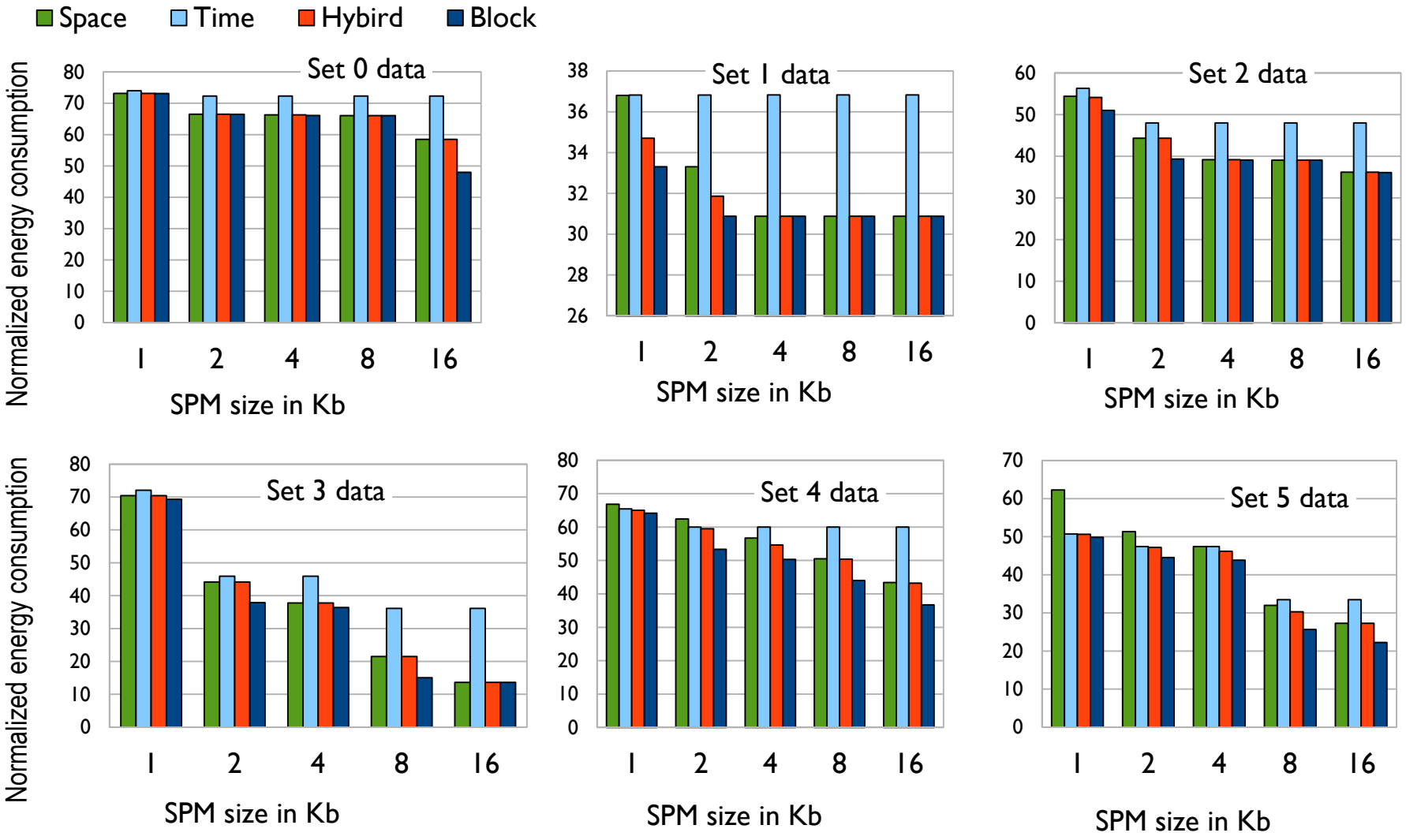
H. Takase, H. Tomiyama, and H. Takada. *Partitioning and allocation of scratch-pad memory for priority-based preemptive multi-task systems*. in DATE '10

SPM Sharing for Multi-Task

- At compile time:
  - Assign an SPM space (i. e., block) to each task
  - Find memory objects to place in each block
  - Find an address for each block in SPM for minimizing overlaps
- At run time:
  - Copy only a part of overlapping with coming task



Multi-Task Results



L. Gauthier, et al. "Minimizing Inter-Task Interferences in Scratch-Pad Memory Usage for Reducing the Energy Consumption of Multi-Task Systems," Proc. of CASES, pp.157-164, Oct., 2010.

Summary

- A fast accurate model for SW energy consumption
- The error of our approach is 5% on an average and 20% at the maximum case
- Code and data placement techniques drastically reduce the SW energy

Future work

- Refine stack placement technique and target heap object (WIP)