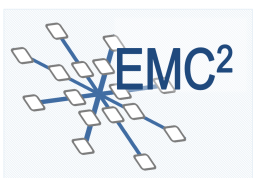
A novel approach for estimating energy consumption at the LLVM IR level

Kyriakos Georgiou, Steve Kerrison, Kerstin Eder

University of Bristol

HIPEAC – EMC^2 workshop, January 24 2017



Things to take away

- We need energy transparency at various levels of software abstraction.
- Energy measurements are not sufficient for energy transparency.
- Bounding energy with Static Resource Analysis (SRA) is a hard challenge.
- We need techniques that are as much architecture- and compiler-agnostic as possible.

(Accepted for publication at ACM TACO)

Presentation Outline

- Motivation and concepts.
- Low level analysis - ISA level energy modeling.
- High level analysis - LLVM IR energy characterization.
- SRA-based energy consumption estimation.
- LLVM IR profiling-based energy consumption estimation.
- Future work.

Motivation

- How much energy does my code consume when executing on a particular architecture?
- What is the effect on the energy consumption for a particular processor when:
  1. choosing an algorithm;
  2. using different coding styles?
- How to detect energy hot spots in my code?
- Is my application energy budget met?
- How to enable energy-aware compilation?

Hard to answer questions!

Current Approaches

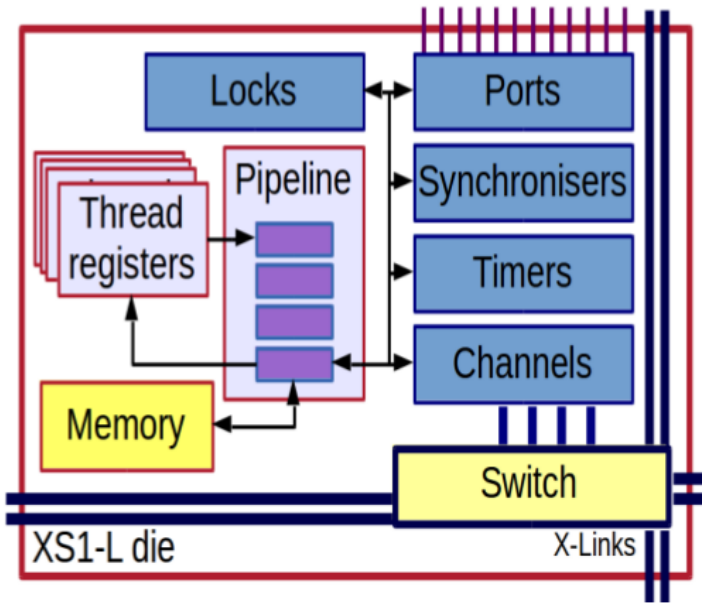
1. Physical Energy Measurements:
   - Not accessible to every software developer (special equipment and advance hardware knowledge needed).
   - Difficult to capture energy consumption bounds with end-to-end measurements.
   - Difficult to achieve fine-grained software energy characterization.
2. Energy Simulation:
   - Usually at low level of abstraction (architecture).
   - Can be significantly time consuming.
   - Can not properly capture energy bounds.
3. Performance Monitoring Counters (PMCs):
   - Their number and availability are still limited in deeply embedded systems.

Low Level energy analysis

XMOS XS1 Architecture

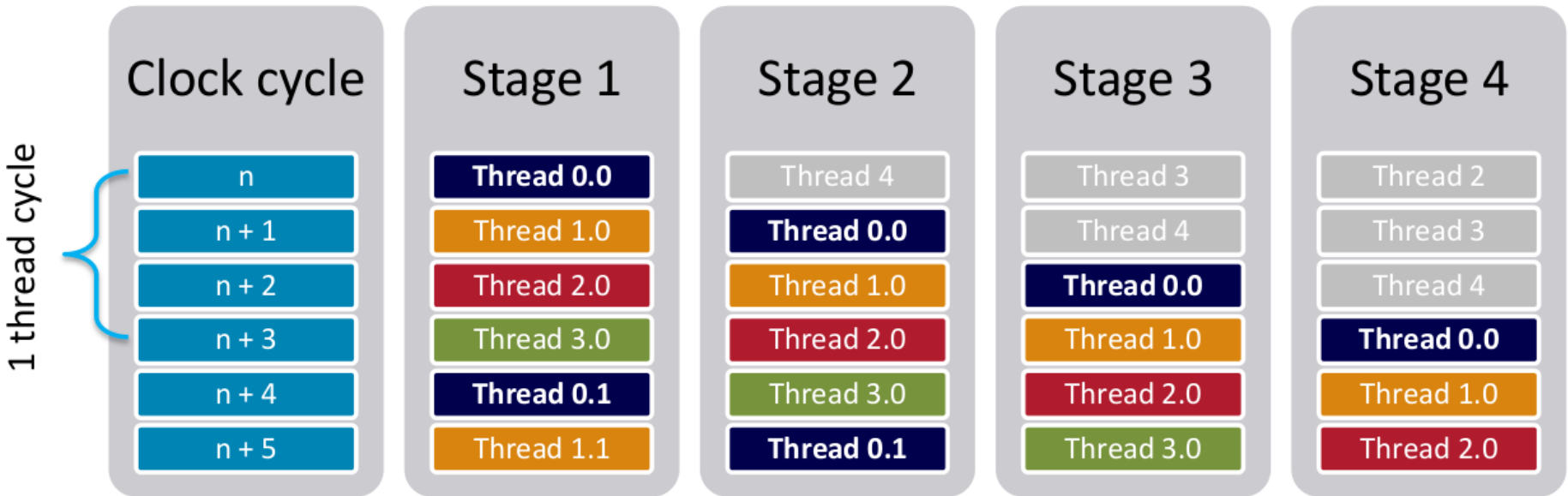
We focus on the XCORE processor, a 32bit multicore microcontroller designed by XMOS.

- 64KiB SRAM
- No Cache hierarchies
- Channel based communication between threads and cores
- Instructions dedicated to comms & I/O
  - Not memory mapped
- Peripherals: Software defined interfaces
- Event driven, no idle loops



XMOS XS1 threads/pipeline

- Up to eight threads per core
- Four stage pipeline
- Simple scheduling (no branch prediction)
- At 500MHz, 125MIPS per thread for ¡= 4 threads



ISA Instruction Time Cost

Predictable for the majority of ISA instructions, with some special cases:

- Division
- Communication time is constant on the same core
- Communication time between cores
- Input output on ports time may vary

Low-Level Analysis - Energy Modeling

$$P_{\text{instr}} = I_{leak} * V + (C_{idle} + C_{instr} M_{N_p} O) \cdot V^2 \cdot F$$
$$\text{where } N_p = \min(N_t, 4)$$

- ISA based characterization[1].
- Multi-threaded energy model.
- Complete instruction set.
  - With regression-tree capturing harder to reach instructions.
- Voltage/frequency parameterization.

[1] S. Kerrison and K. Eder. 2015. Energy Modeling of Software for a Hardware Multi-threaded Embedded Microprocessor. ACM Transactions on Embedded Computing Systems 14, 3 (April 2015), 56:1–56:25.

High level energy analysis

Compiler Intermediate Representation Energy Modeling

LLVM is a common optimizer and code emitter.

- LLVM IR is the optimum place for resource analysis and energy optimizations.
- Applicable to many architectures.
- All the information needed for the resource analysis are preserved.
- LLVM IR is closer to the source code than the ISA level.

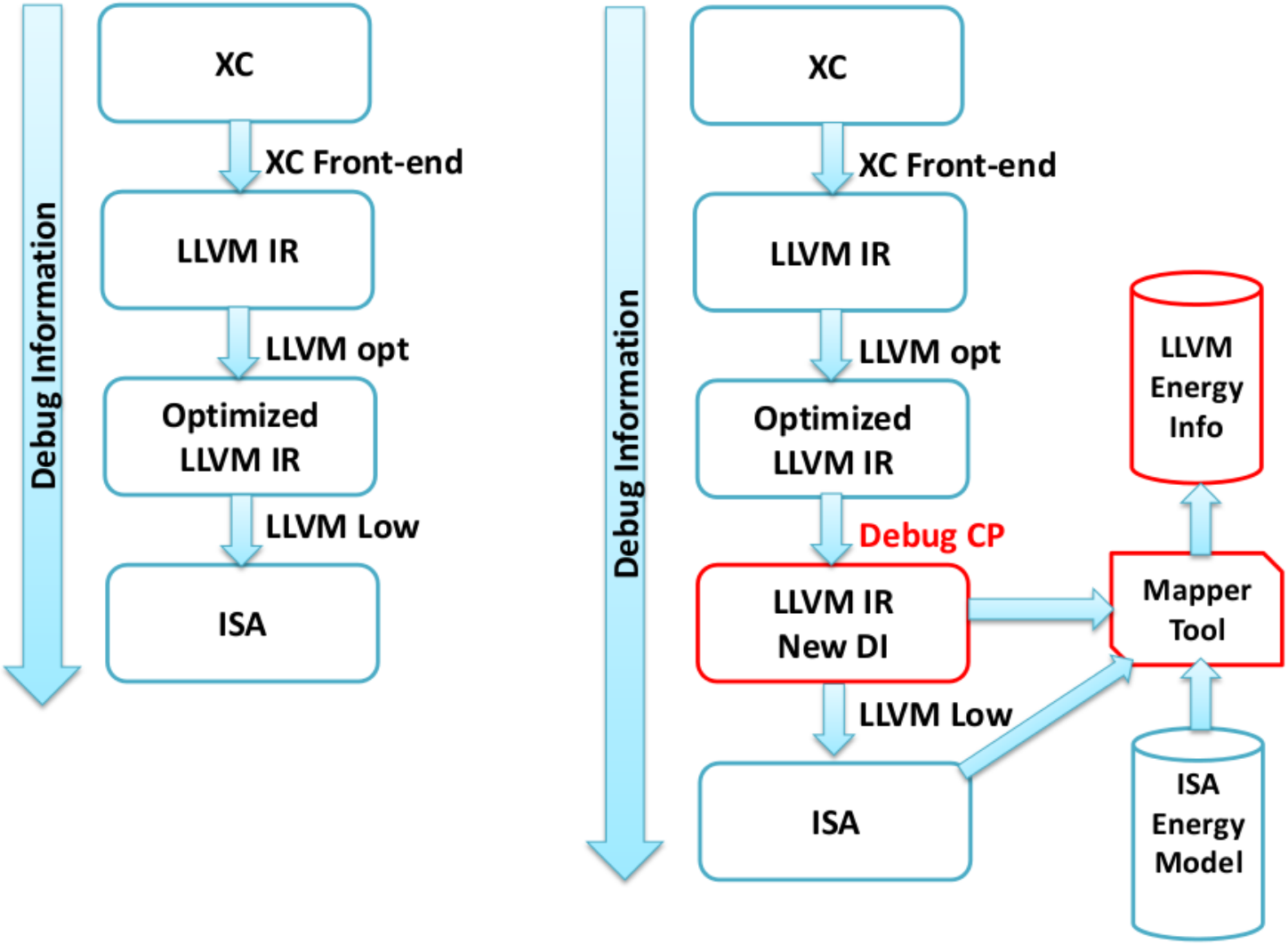
Existing approaches

- Try to directly model energy at IR level.
  - Modeling process needs to be repeated for a new architecture.
  - Can not account for compiler dynamic behavior (code transformations).
  - Can not account for specific architecture behavior (FNOPs).

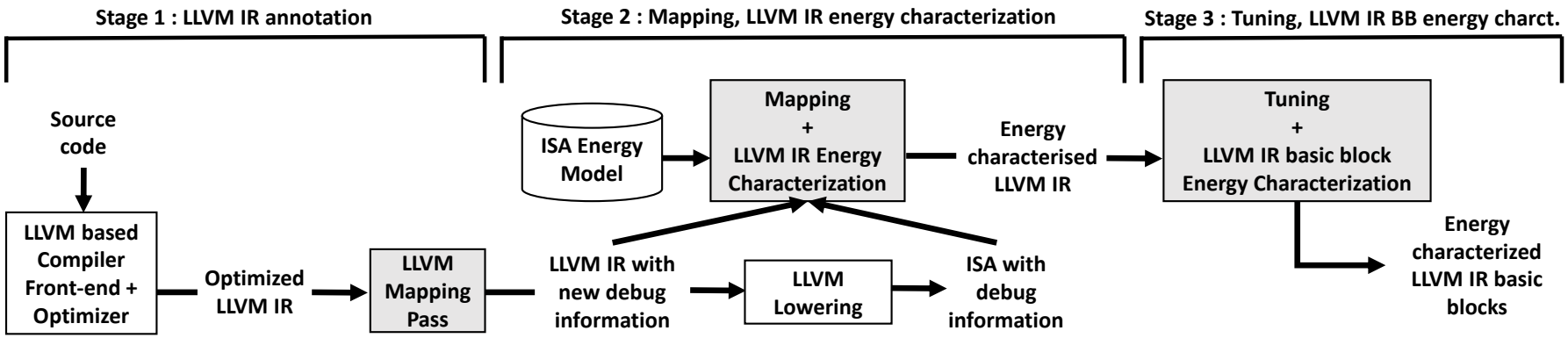
Novel Dynamic Mapping Technique

- Aims to link each LLVM IR instruction of a program with its corresponding machine specific ISA emitted instructions.
- No loss of any energy costings between the two levels.
- It is target agnostic.
- It is dynamic and can account for compiler or architecture specific behavior.
- The technique has been formalized.

Mapping Technique Logic

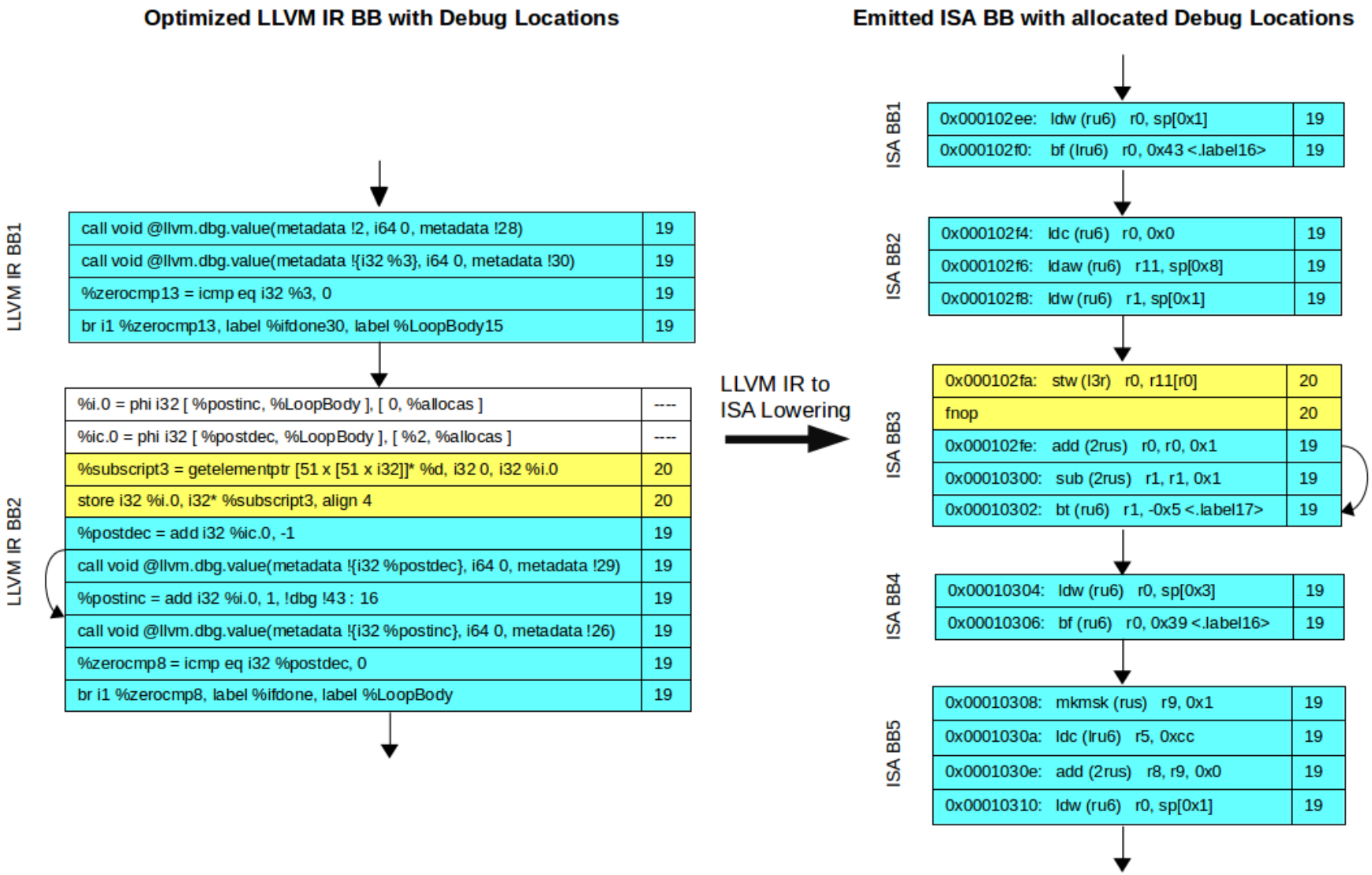


Mapping Technique Overview

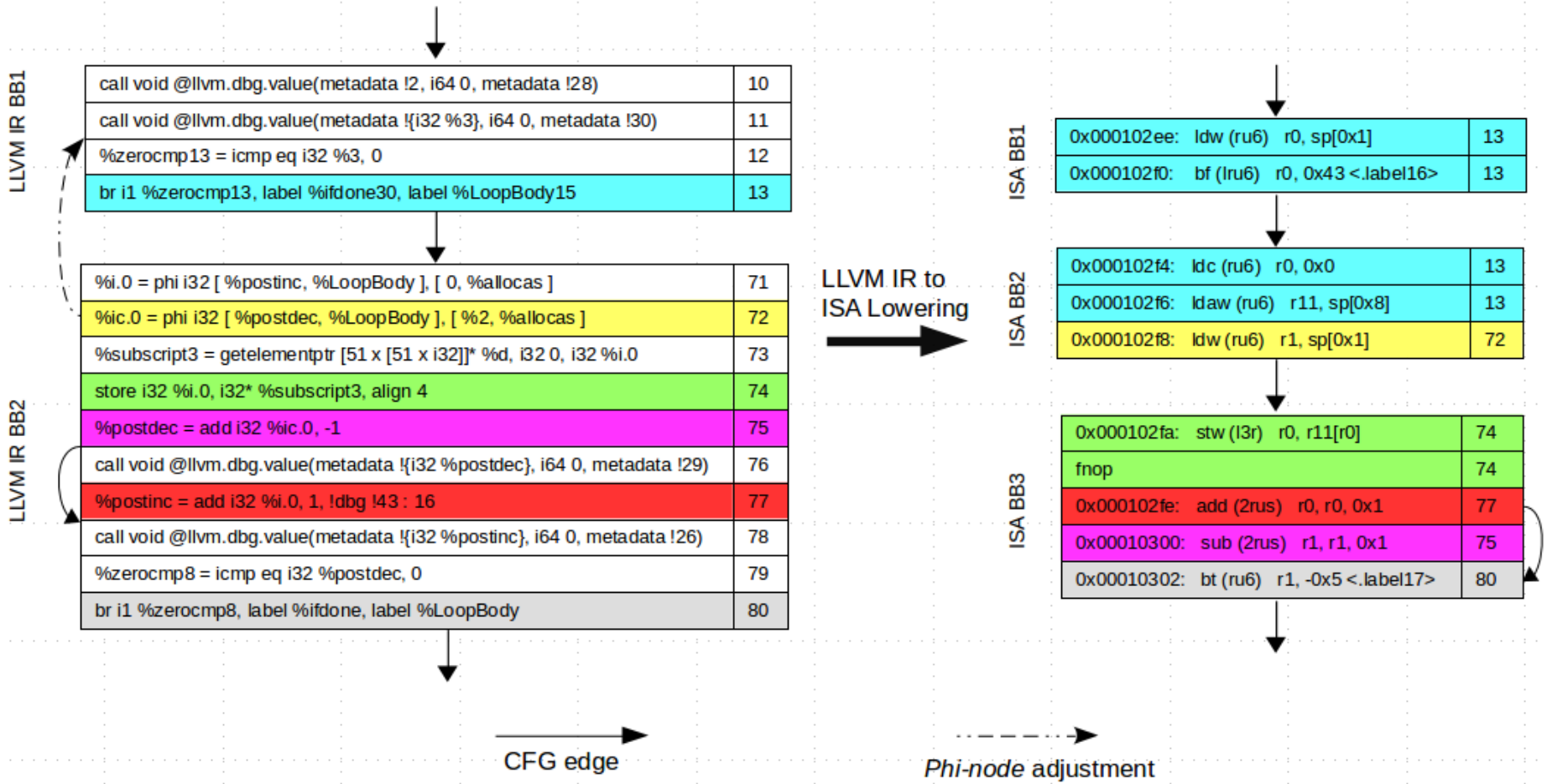


Tunning for FNOPs and Phi-nodes significantly improves BBs energy mapping accuracy.

LLVM-IR/ ISA Mapping Example

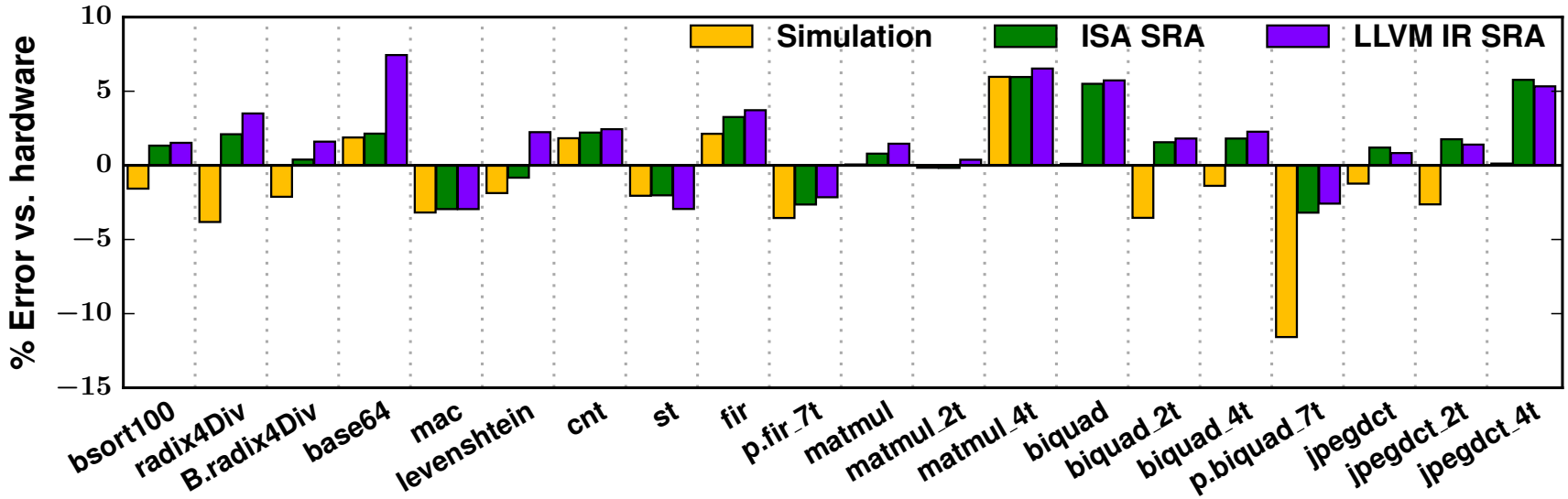


LLVM-IR/ ISA Mapping Example



Energy Consumption Estimation Using Static Resource Analysis

SRA Results



Highlights:

- SRA based on the Implicit Path Enumeration Technique (IPET).
- Simulation-based estimation is the baseline for best achievable accuracy.
- Overestimation up to 5.7% for ISA SRA, up to 7.4% for LLVM IR SRA.
- Max observed underestimation of 4%.
- LLVM IR SRA results are within one percentage point error of ISA SRA results.

Why soft energy bounds?

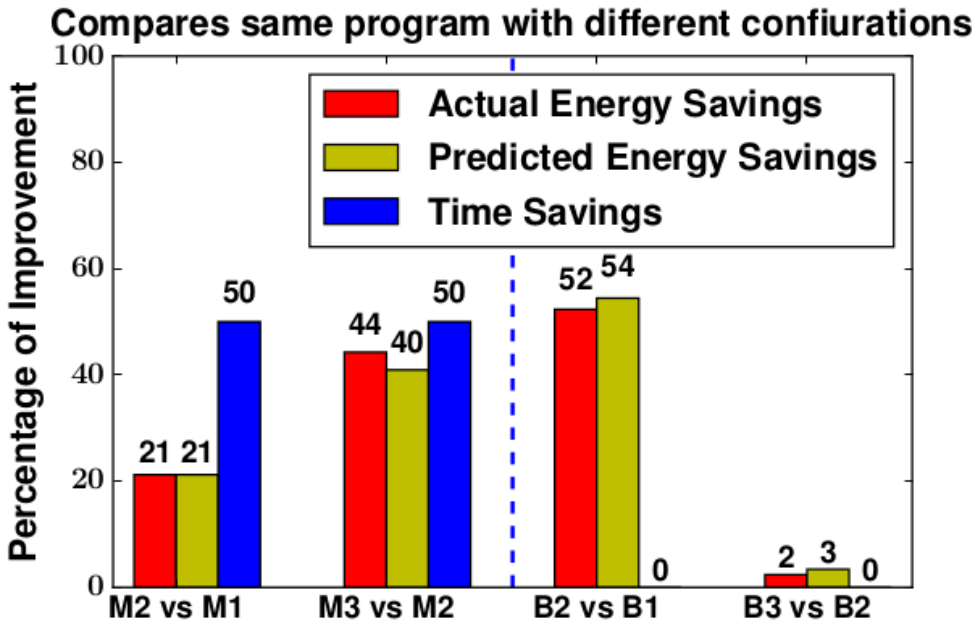
- Energy is data depended.
- Use of a data-insensitive energy model and SRA method.
- Finding the data that will trigger the worst case energy consumption is an NP-hard problem[2].
- No method can approximate tight energy consumption upper bounds within any level of confidence[2].
- Worst case energy consumption observed: pseudo-random-generated data.
- Still the best option among the available techniques.

[2] J. Morse, S. Kerrison, and K. Eder, On the infeasibility of analysing worst-case dynamic energy, http://arxiv.org/abs/1603.02580.

Design Space Exploration using SRA

Benchmark	ID	Configuration			
		C	T	V	F
MatMult (4 pairs of 30x30 matrices)	M1	1	1	1	450
	M2	1	2	1	450
	M3	1	4	1	450
Biquad Filter	B1	1	1	1	450
	B2	1	7	0.75	150
	B3	2	7	0.7	75

C: Number of cores used
T: Number of threads used
V: Core(s) Voltage in Volts
F: Core(s) Frequency in MHz



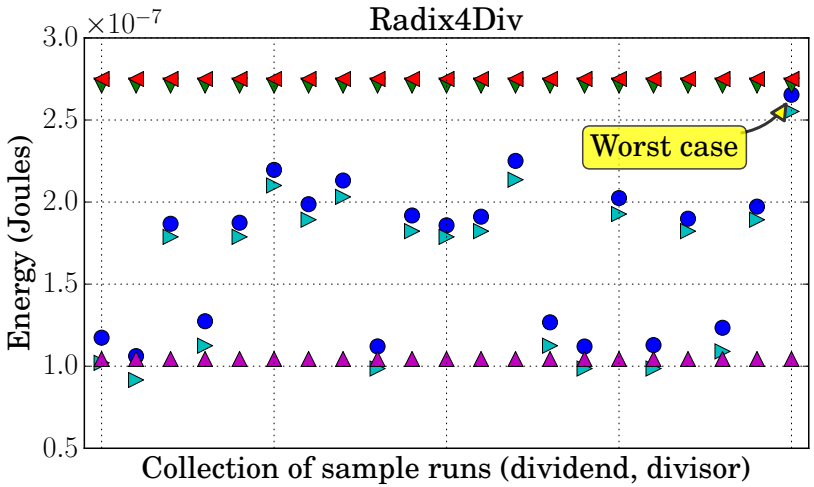
- The worst-case execution path is actually the dominant execution path.
- Time predictions are omitted due to the architectures deterministic nature.
- The small error of estimation allows for comparison between different versions.

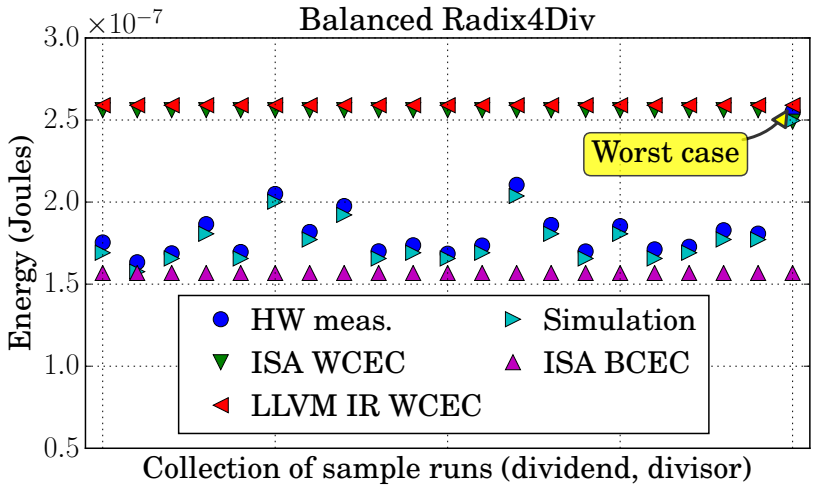
Energy consumption trends for parametric benchmarks, using regression analysis

Benchmark	Regression Analysis (nJ)	x
Base64	$f(x) = 54.9x + 62.3$	string length
Mac	$f(x) = 15x + 21.1$	length of two vectors
Cnt	$f(x) = 2.4x^3 + 17.6x^2 + 5.7x + 34.5$	matrix size
MatMul	$f(x) = 14x^3 + 17.1x^2 + 4.3x + 34$	size of square matrices
MatMul_2T	$f(x) = 18.1x^3 + 20.3x^2 + 5.7x + 112$	size of square matrices
MatMul_4T	$f(x) = 21x^3 + 23.3x^2 + 7.1x + 213.1$	size of square matrices

- Programmers/ users can predict a program's energy consumption under specific parameter values
- Embedding such equations into an operating system (e.g. library function calls), can enable energy aware decisions:
  - for scheduling tasks
  - checking if the remaining energy budget is adequate to complete a task
  - downgrade the quality of service and complete the task with less energy

Energy Consumption Variation



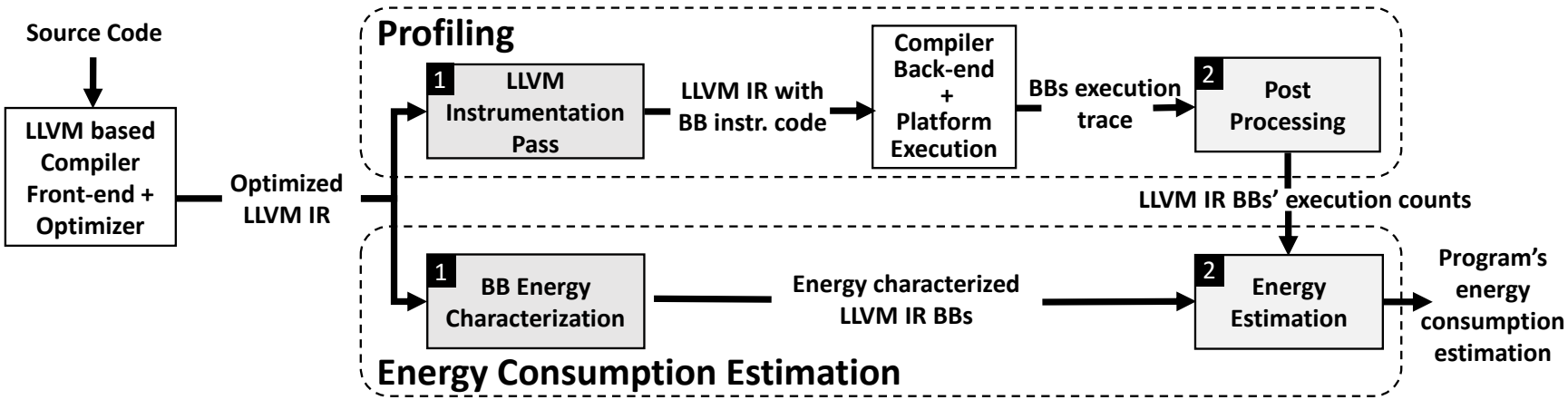


Profiling-Based Energy Consumption Estimation

Advantages

- Captures the actual case energy consumption.
- Energy estimations directly into the LLVM IR.
- As much target-agnostic as possible.
- No energy overheads due to instrumentation instructions.
- Can significantly outperform simulation based estimations.
- Allows for fine-grained energy characterization of software components at LLVM-IR level.

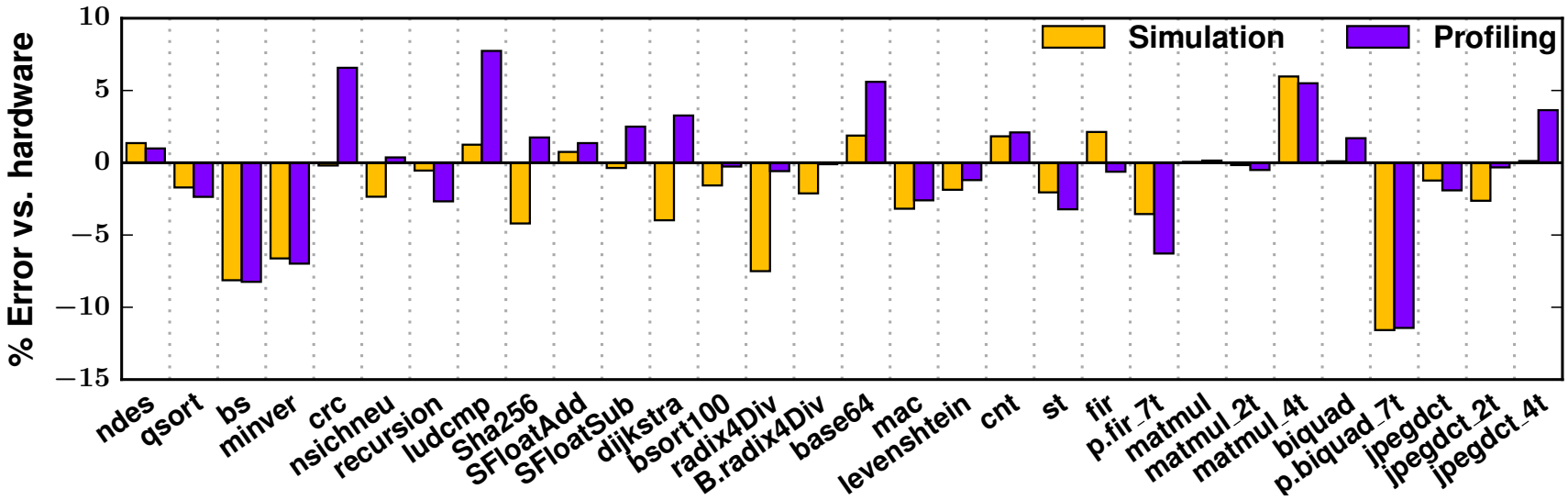
Profiling-Based Energy Cons. Estimation Overview



Highlights:

- Emits Basic Blocks traces.
- Block tracing rather than counters.
- Instrumentation code is inserted at the LLVM IR.
- Depends on the mapping technique.
- Clean copy of LLVM IR for the energy estimation.

Profiling-Based Energy Cons. Estimation Results



Highlights:

- The average error obtained for the profiling-based estimations is 3.1%, and 2.7% for the simulation-based estimations.
- The profiling results demonstrate a high accuracy with an average error deviation of 1.8% from the ISS.

Profiling VS Simulation Estimation Performance

- Simulation performance is governed by the complexity of the program's underlying algorithms.
- Profiling performance is mainly governed by the program size, since retrieving the BB execution counts incurs a negligible execution time overhead.
- Examples:
  - SFloatAdd benchmark, with a $O(1)$ complexity but big code size: negligible performance gain over the ISS estimation.
  - Matrix multiplication benchmark ($30x30$ size matrices), $O(n^3)$ but small code size: a 381 times speedup over the simulation estimation.
- Mapping most time consuming part of profiling.
- There is plenty of space for optimization.

Future work

- Work in progress: extending the analysis to ARM Cortex M series processors.
- Use mapping technique and profiler to perform energy-specific optimizations.
- Account for external to the core activity.

Thank you!

Questions?

Kyriakos.Georgiou@bristol.ac.uk
Steve.Kerrison@bristol.ac.uk
Kerstin.Eder@bristol.ac.uk