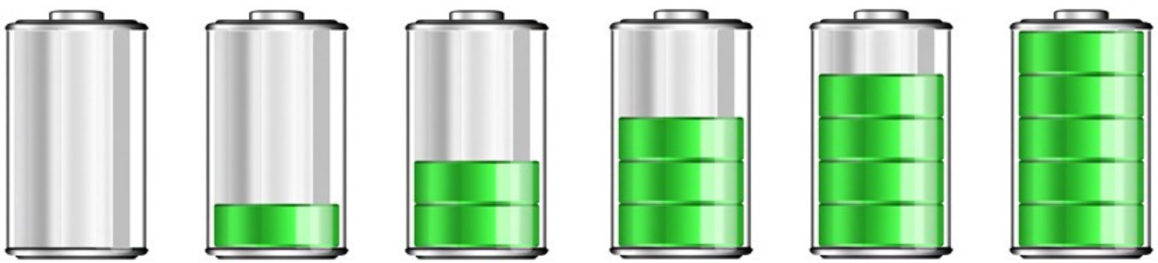
Impact of different compiler options on energy consumption



James Pallister
Reasearch Engineer, Embecosm
PhD Student, University of Bristol
(Supervisor: Kerstin Eder)

Simon Hollis
University of Bristol

Jeremy Bennett
Embecosm

Motivation

- Compiler optimizations are claimed to have a large impact on software:
  - Performance
  - Energy
- No *extensive* study prior to this considering:
  - Different benchmarks
  - Many individual optimizations
  - Different platforms
- This work looks at the effect of many different optimizations across 10 benchmarks and 5 platforms.
- Over 200 optimization passes covered by 150 flags
  - Huge amount of combinations

This Talk

- This talk will cover:
  - Importance of benchmarks
  - How to explore 2^150 combinations of options
  - Correlation between time and energy
  - How to predict the effect of the optimizations
  - The best optimizations

Importance of Benchmarks

- One benchmark can't trigger all optimizations
- Perform differently on different platforms
- Need a range of benchmarks
- Broad categories to be considered for a benchmark:
  - Integer
  - Floating point
  - Branching
  - Memory

Existing Benchmark Suites Considered

- **MiBench**
- **WCET**
- **DSPstone**
- **ParMiBench**
- **OpenBench**
- **LINPACK**
- **Livermore Fortran Kernels**
- **Dhry/Whet-stone**

- Require embedded Linux
- Targeted at higher-end systems
- Multithreaded benchmarks typically for HPC
- Don't necessarily test all corners of the platform

Our Benchmark List

Name	Source	B	M	I	FP	T	License	Category
Blowfish	MiBench	L	M	H	L	Multi	GPL	security
CRC32	MiBench	M	L	H	L	Single	GPL	network, telecomm
Cubic root solver	MiBench	L	M	H	L	Single	GPL	automotive
Dijkstra	MiBench	M	L	H	L	Multi	GPL	network
FDCT	WCET	H	H	L	H	Single	None†	consumer
Float Matmult	WCET	M	H	M	M	Single	GPL	automotive, consumer
Integer Matmult	WCET	M	M	H	L	Single	None†	automotive
Rjindael	MiBench	H	L	M	L	Multi	GPL	security
SHA	MiBench	H	M	M	L	Multi	GPL	network, security
2D FIR	WCET	H	M	L	H	Single	None†	automotive, consumer

Platforms Chosen

ARM Cortex-M0	ARM Cortex-M3	ARM Cortex-A8	XMOS L1	Adapteva Epiphany
Small memory	Small memory	Large memory	Small memory	On-chip and off-chip memory
Simple Pipeline	Simple Pipeline, with forwarding logic, etc.	Complex superscalar pipeline	Simple pipeline	Simple superscalar pipeline
		SIMD/FPU		FPU
			Multiple threads	16 cores

Experimental Methodology

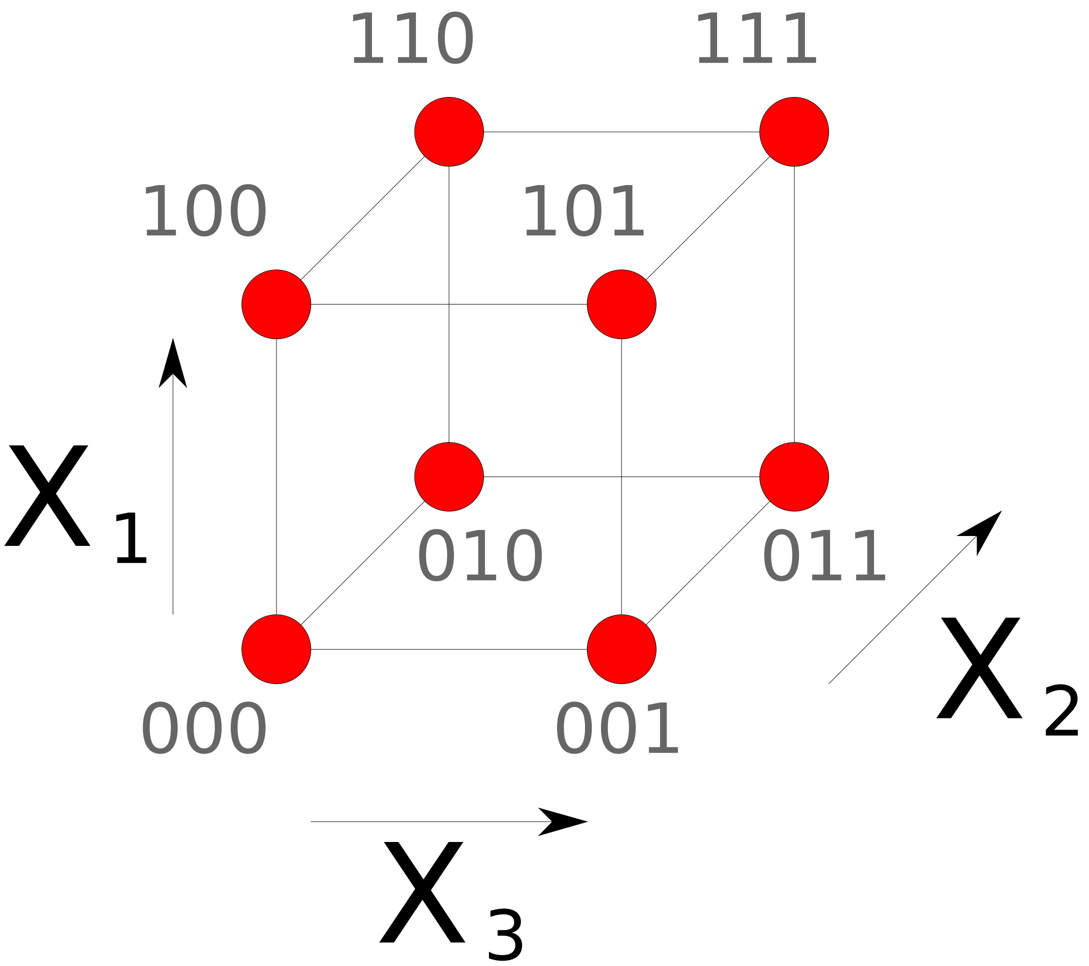
- Compiler optimizations have many non-linear interactions
- 238 optimization passes combined into 150 different options (GCC)
- 82 compiler options enabled by O3
- How to test all of these, while accounting for the interactions between optimizations?

Fractional Factorial Designs

Full Factorial Design

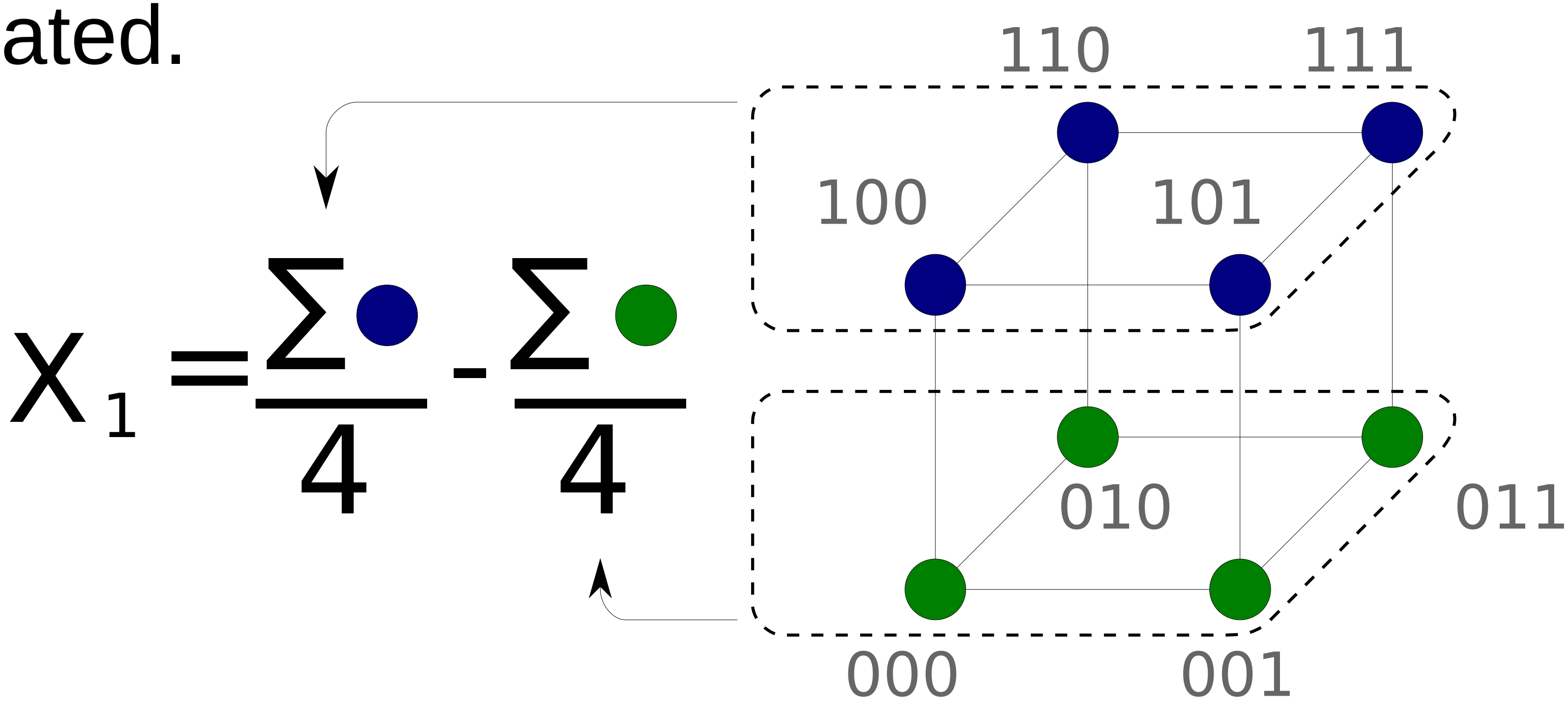
Example:

- 3 options to investigate
- Each option can be on or off (2 level)
- 2^3 tests to be run



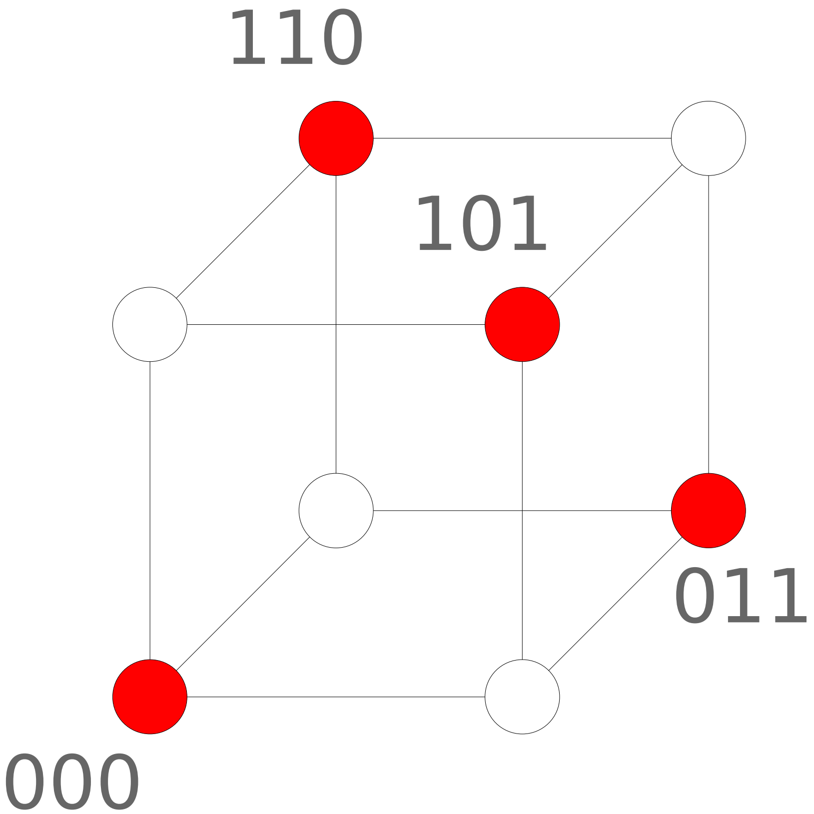
Estimating an Option's Effect

- The effect of a single option can be calculated.



Fractional Factorial Design

- Use a subset of the full factorial design
- Shown here is a 'half fraction'
- 2^(3-1) tests to be run



Loss of Information

- Less runs = less information
- The fewer runs performed, the fewer interactions can be resolved
- The 'resolution' of the fractional factorial design

O1 flags (37 factors)

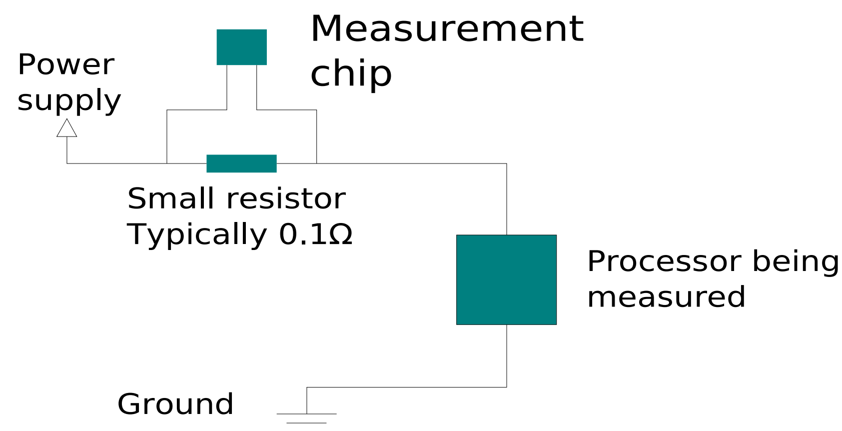
Resolution	Runs Needed
3	256
4	1024
5	2048
6	4096
Full	137438953472

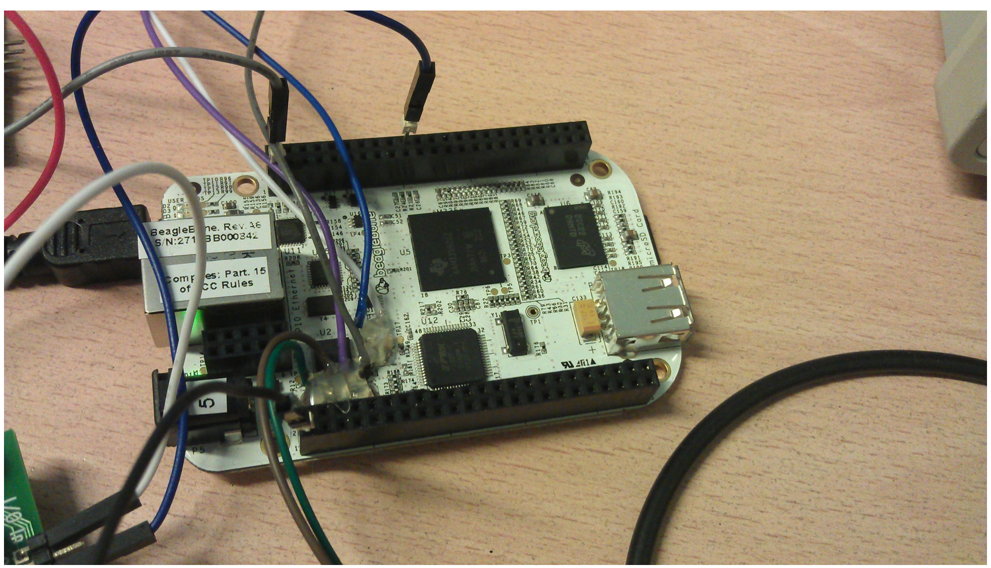
10 hours

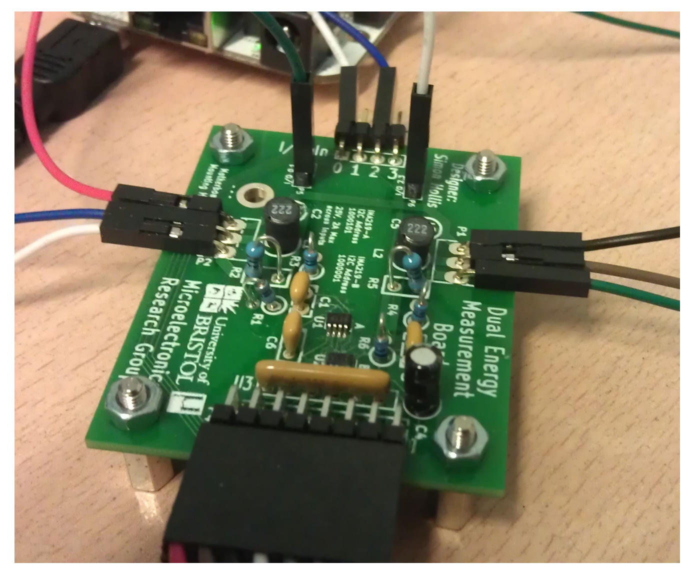
77000 years

Hardware Measurements

- 10 kSamples/s
- XMOS board to control and timestamp measurements
- Integrate to get energy consumption







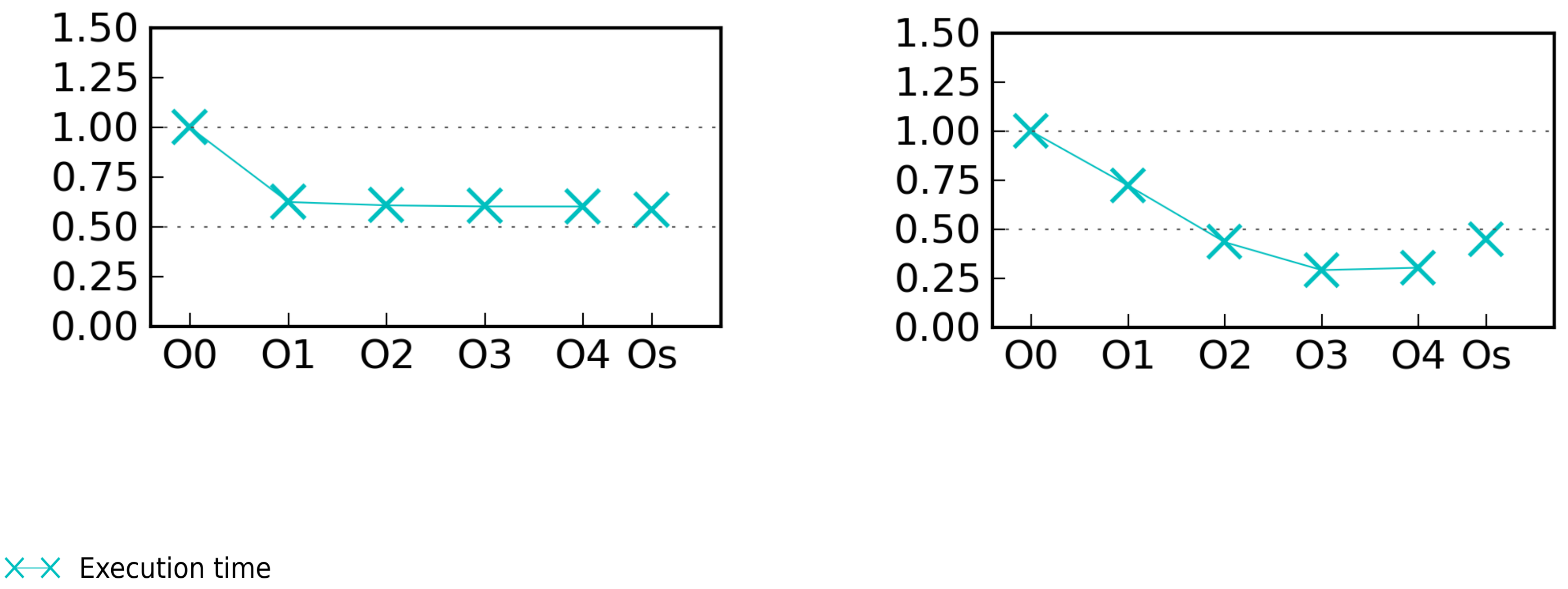
Results

- Energy consumption ≈ Execution time
  - Generalization, not true in every case
- Optimization unpredictability
- No optimization is universally good across benchmarks and platforms

Overview

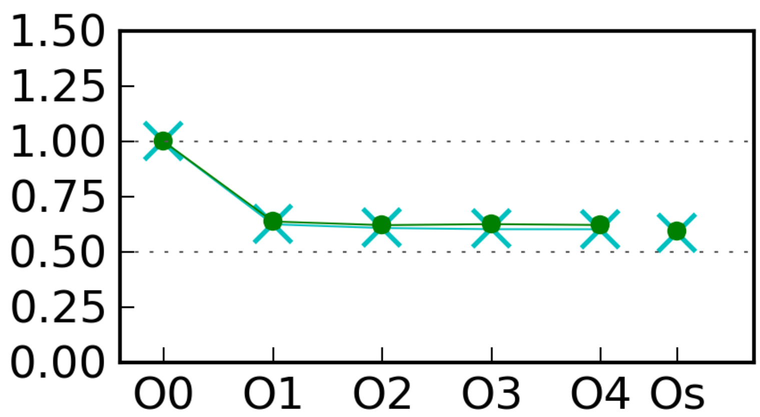
FDCT, Cortex-M0

FDCT, Cortex-A8

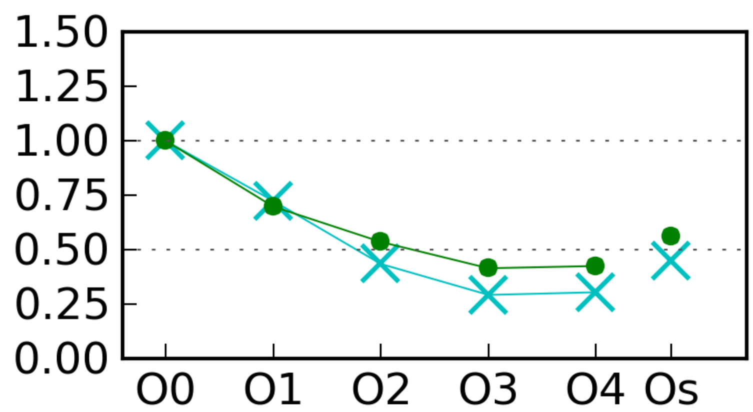


Overview

FDCT, Cortex-M0



FDCT, Cortex-A8

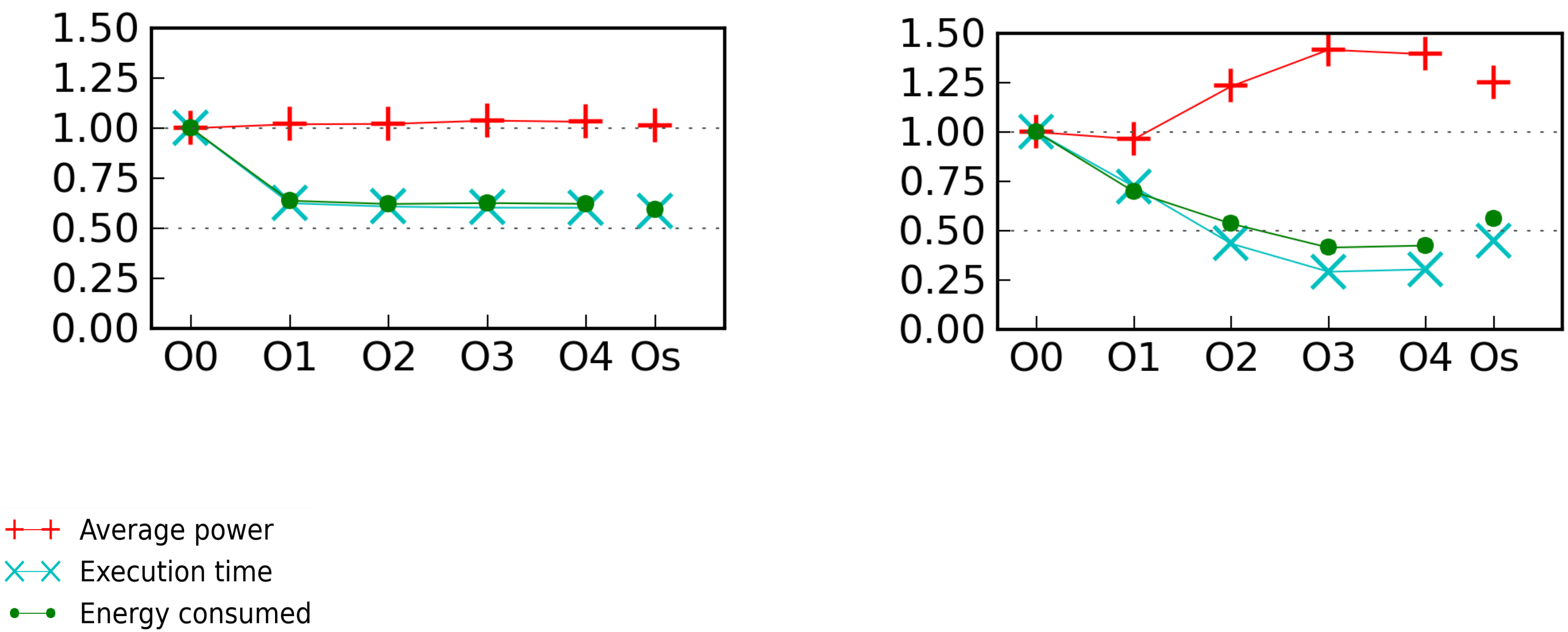


Execution time
Energy consumed

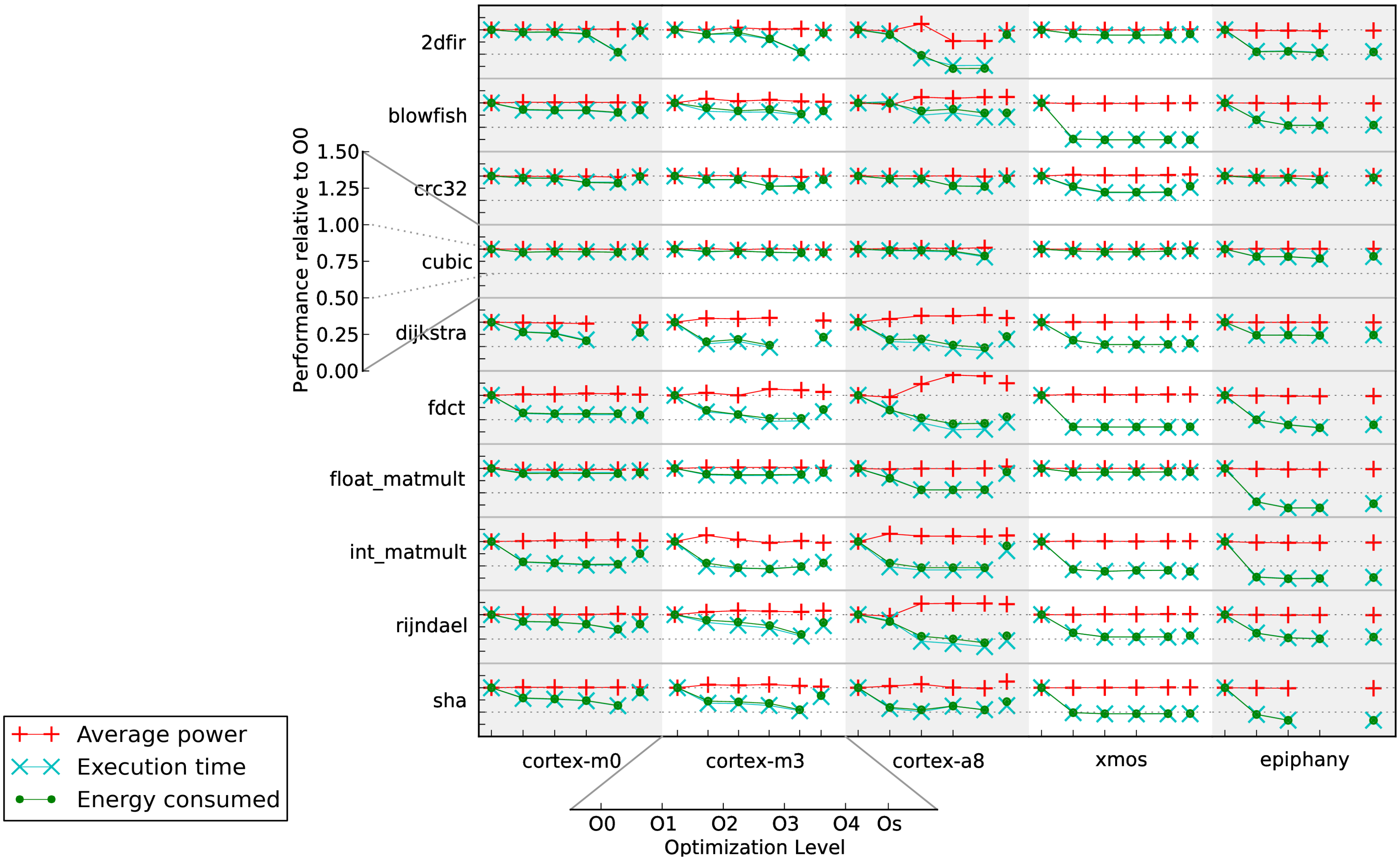
Overview

FDCT, Cortex-M0

FDCT, Cortex-A8

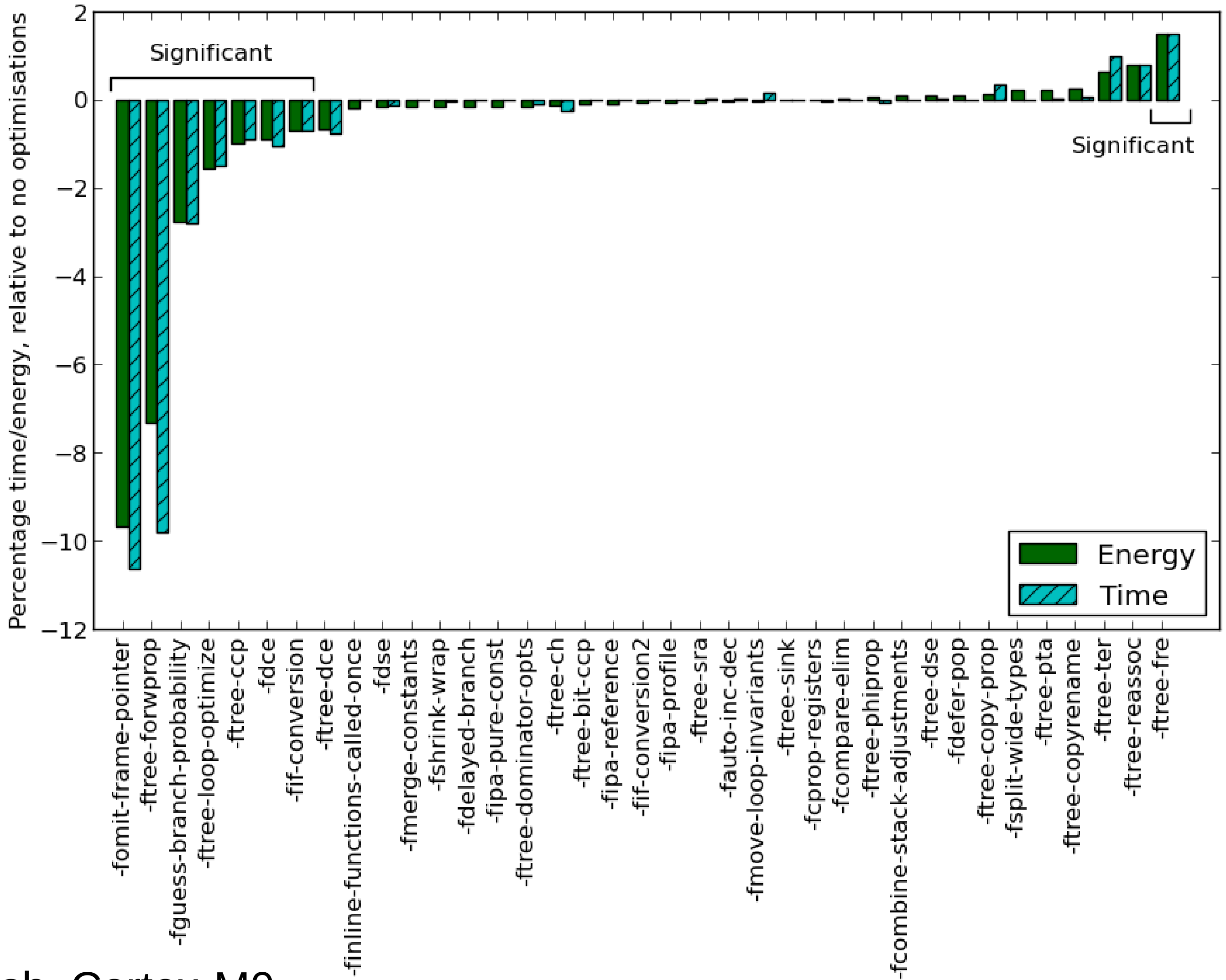


Overview



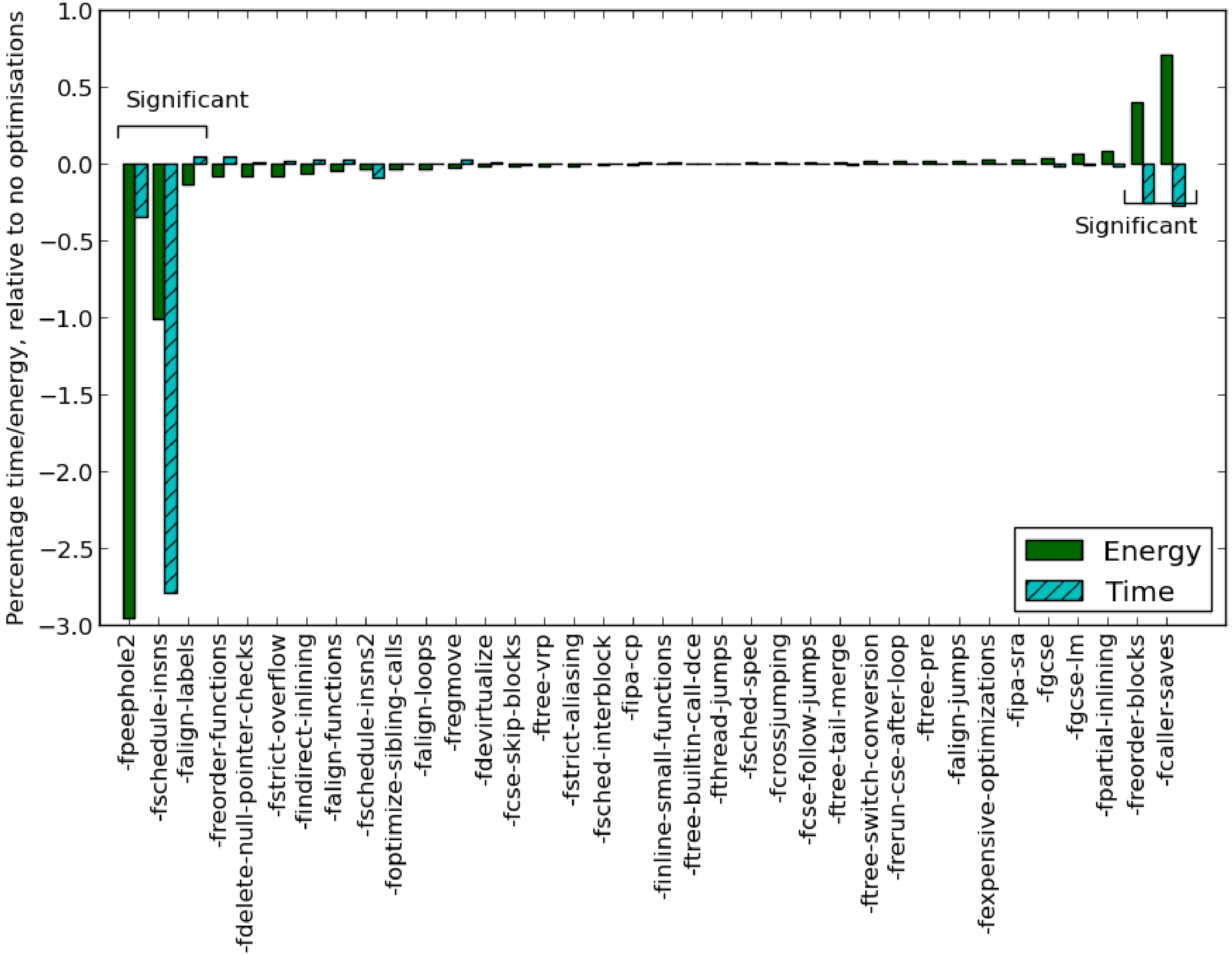
2dfir
blowfish
crc32
cubic
dijkstra
fdct
float_matmult
int_matmult
rijndael
sha
Performance relative to O0
1.50
1.25
1.00
0.75
0.50
0.25
0.00
cortex-m0
cortex-m3
cortex-a8
xmos
epiphany
O0
O1
O2
O3
O4
Os
Optimization Level
Average power
Execution time
Energy consumed

Time = Energy



O1 Flags, Blowfish, Cortex-M0

Time ≈ Energy



O2 Flags, Blowfish, Cortex-M3

-fpeephole2

Constant folding, strength reduction, algebraic simplification

-fschedule-insns

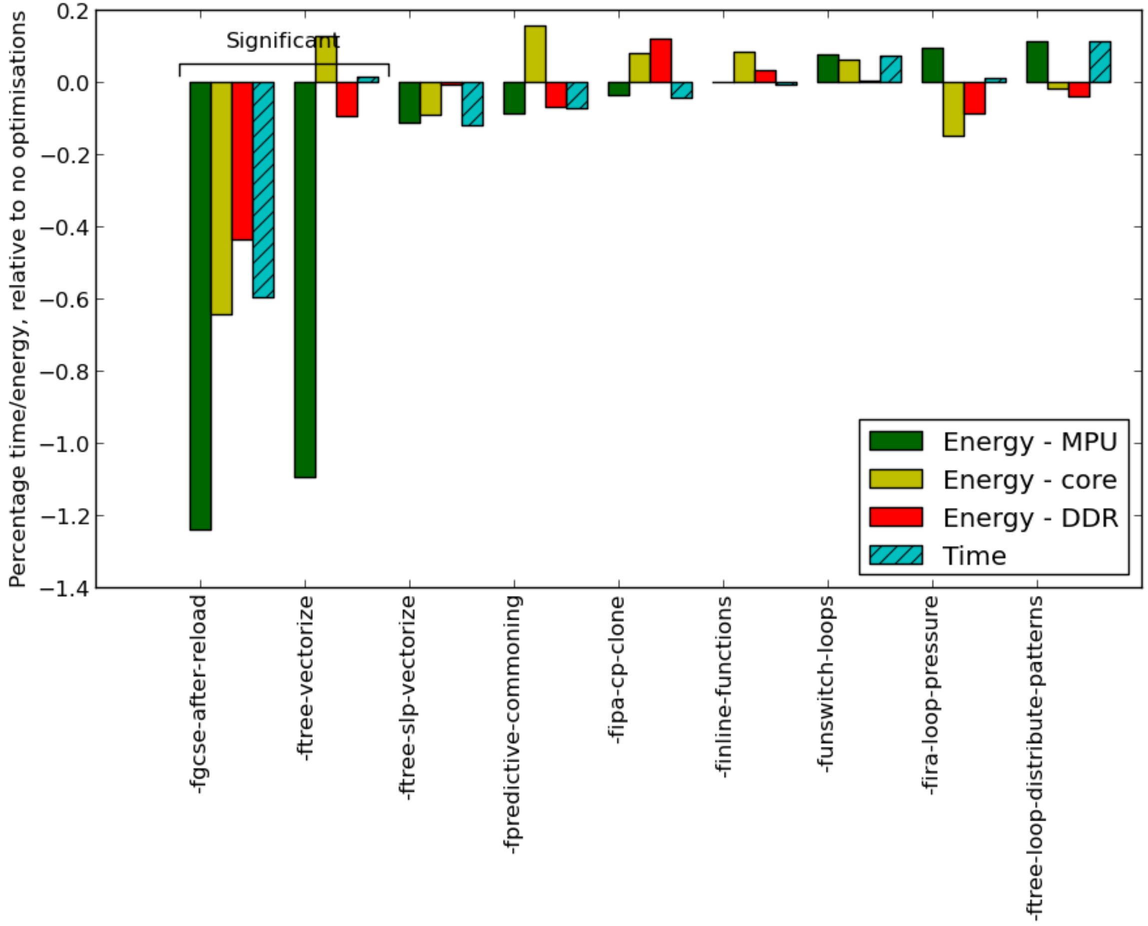
Reorder instructions to reduce execution stalls

-fcaller-saves

"Enable values to be allocated in registers that will be clobbered by function calls, by emitting extra instructions to save and restore the registers around such calls."

When Time ≠ Energy

- Complex pipeline
- -ftree-vectorize
  - NEON SIMD unit
  - Much lower power



O3 Flags, 2DFIR, Cortex-A8

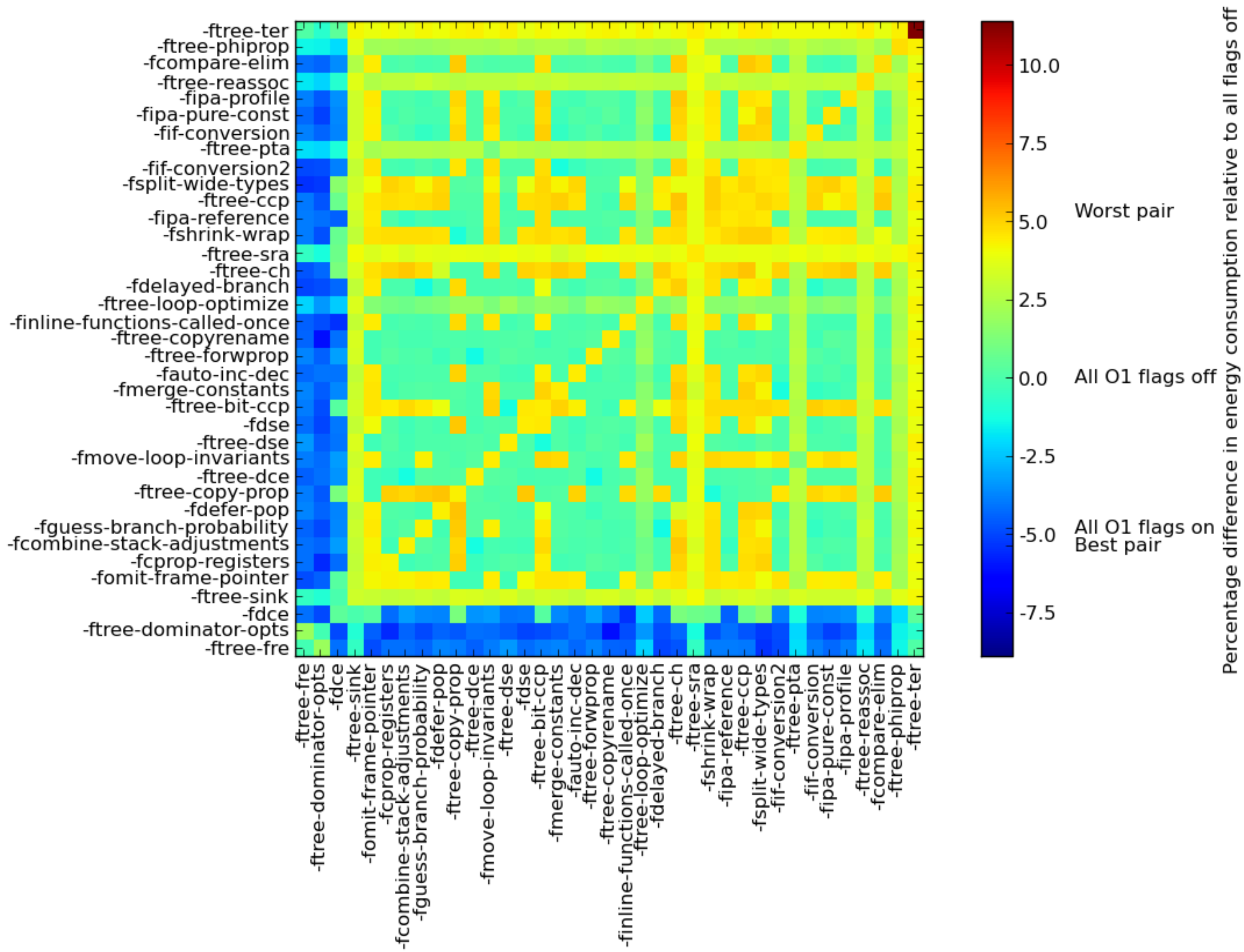
Conclusion: Mostly, Time ≈ Energy

- Highly correlated
- Especially so for 'simple' pipelines
- Little scope for stalling or superscalar execution

- Complex pipelines:
  - Still a correlation
  - But more variability
  - SIMD, superscalar execution
- To get the most optimal energy consumption we need better than "go fast"

Optimization Unpredictability

- Pairs of optimizations on top of O0
- Possibly higher order interactions occurring?



O1 Flags, Cubic, Cortex-M0

The Best Three Optimizations for Energy

Benchmark	Cortex-M0	Cortex-M3	Cortex-A8	Epiphany
2dfir	E	T, G, H	N, G, C	H, A, D
blowfish	B, J, E	J, B, G	K, B, E	D, P, H
crc32	F	F	F, G	
cubic	A, I	A, I	A	A, I, O
dijkstra	I, A, B	F, I, A	F, I, A	
fdct	J, G, D	J, G, K	M, K, J	A, H, D
float_matmult	C, E	C, E, G	N, L	D, H, A
int_matmult	C, E, B	C, L, F	L, N, M	A, H, D
rijndael		B, C, R	K, B, S	
sha	B, C, E	C, B, F	C, B, M	D, C, Q

ID	Count	Flag	ID	Count	Flag	ID	Count	Flag
A	11	-ftree-dominator-opts	B	10	-fomit-frame-pointer	C	10	-ftree-loop-optimize
D	7	-fdce	E	7	-fguess-branch-probability	F	7	-fmove-loop-invariants
G	7	-ftree-ter	H	6	-ftree-ch	I	6	-ftree-fre
J	5	-ftree-forwprop	K	4	-fschedule-insns	L	3	-finline-small-functions
M	3	-fschedule-insns2	N	3	-ftree-pre	O	1	-fcombine-stack-adjustments
P	1	-fipa-profile	Q	1	-ftree-pta	R	1	-ftree-sra
S	1	-fgcse	T	1	-fpeephole2			

Conclusion

- Time ≈ Energy
  - True for simple pipelines
  - Mostly true for complex pipelines
  - Good approximation
- Optimization unpredictability
  - Difficult to predict the interactions between optimizations
- Commonality across platforms
  - Instruction set plays a role
  - Common options for the ARM platforms, but not Epiphany

What does this mean?

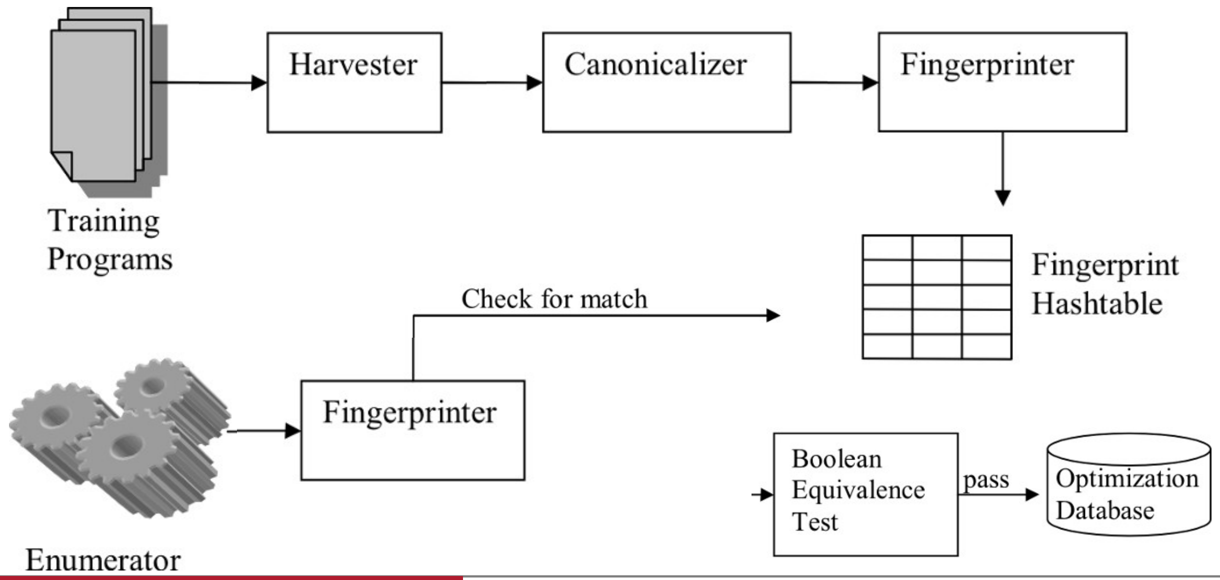
For the Compiler Writer

- Current optimization levels (O1, O2, etc.) are a good balance between compile time and performance/energy.
- Never completely optimal
- Machine learning
  - MILEPOST
  - Genetic algorithms
  - **MAGEEC**

- Current optimizations targeted for performance
- **Few (if any) optimizations in current compilers are designed to reduce energy consumption**

What am I doing now?

- Superoptimization!
  - For energy
- Partially based on *Peephole Superoptimizers,* S. Bansal, 2006
- Find new optimizations for energy efficiency.



More Info

- Academic paper
  - *http://arxiv.org/abs/1303.6485*
- Embecosm Blog – *http://www.embecosm.com/blog/*
  - Superoptimization
  - Benchmarking
  - Compiler optimizations write-up
  - Coming up: Hardware energy measurements
- Dataset Download
  - *http://www.cs.bris.ac.uk/Research/Micro/compileroptions.jsp*
- Code
  - *https://github.com/jpallister/lowpower-benchmarks*
  - *https://github.com/jpallister/stm32f4-energy-monitor*
- UoB Research Projects
  - *http://www.cs.bris.ac.uk/Research/Micro/*