The challenges of modelling and optimising energy for multithreaded programs

Creating energy efficient software for multi-threaded, multi-core processors

Steve Kerrison
µ Research Group
University of Bristol
steve.kerrison@bris.ac.uk

Contents

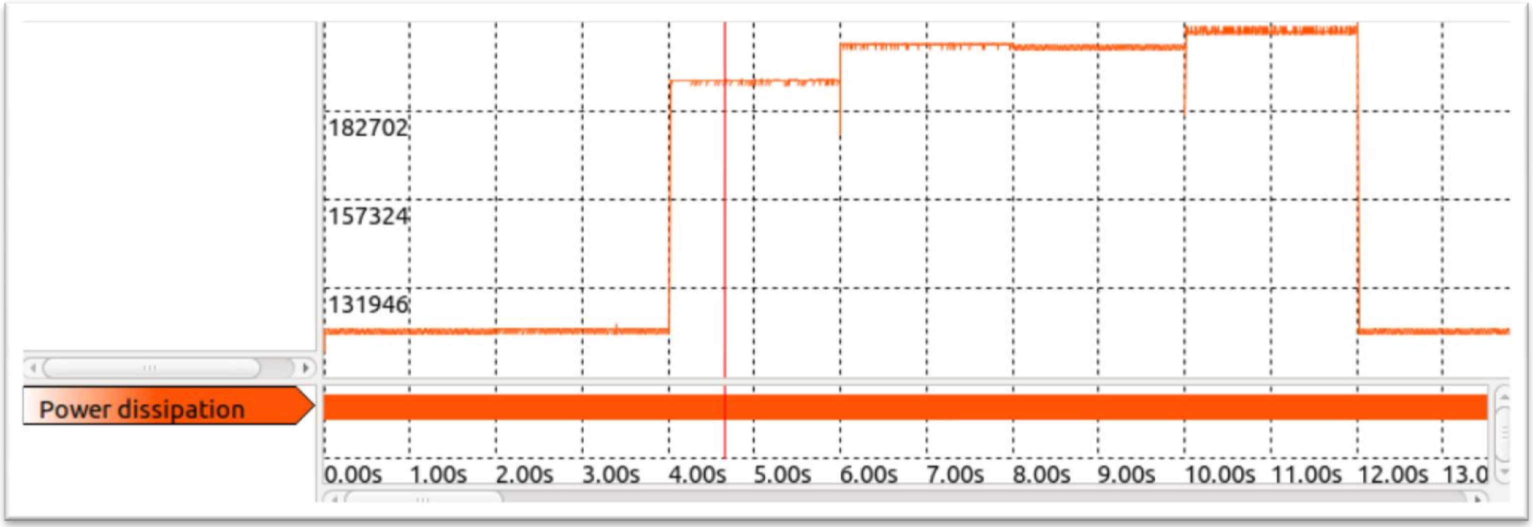
- **The problem**
- XMOS XS1 architecture
- Multi-threading breaks existing models
- Fixing the model
- New tricks
- Unanswered questions

The problem

- Hardware power figures
  - Maximum? Minimum? Typical?
- What will that translate to **for my application?**
- How can **the compiler** help me?

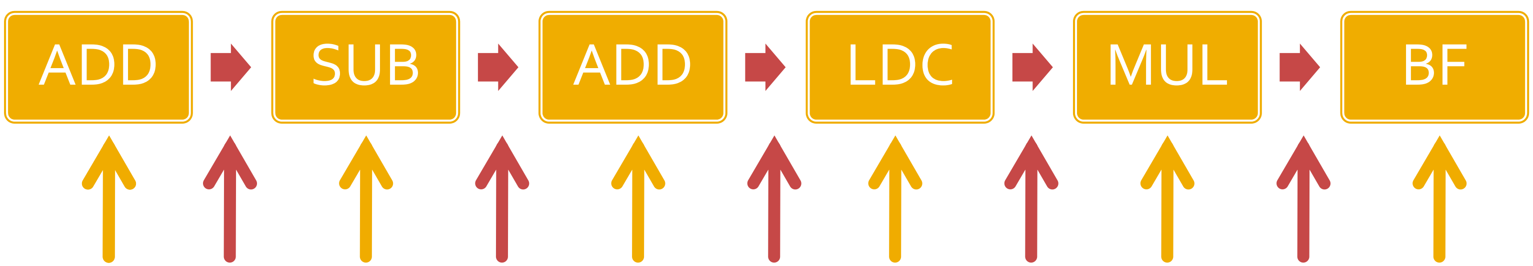
The problem

- Furnish the workflow with this information
  - The developer
  - The compiler
- Packaged in a fast, safe, easy to support way



Traditional methods

- Map processor activity onto some energy metric.
- Requires some cost of executing an instruction.
- **Plus** the cost of switching data & switching between instructions.



$$E = \sum T \left(\boxed{P_i N_i} + \boxed{P_{i,j} N_{i,j}} \right)$$

Traditional methods

- What are we trying to save?

$$P_{static} = V_{core} I_{leak}$$

$$P_{dynamic} = A_{\mu} C_{sw} V_{core}^{2} F$$

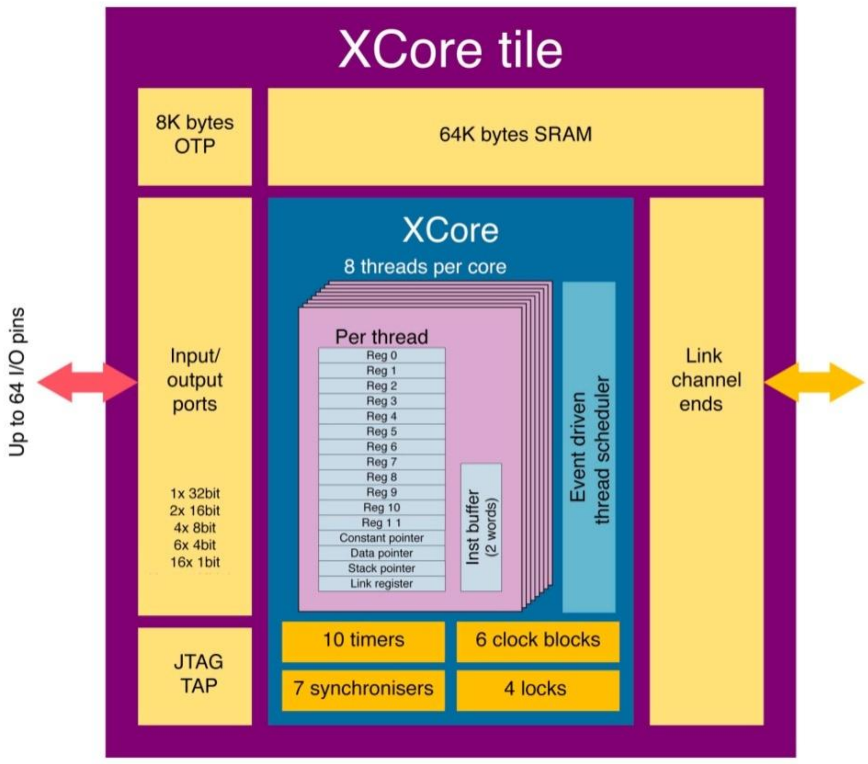
$$E = T(P_{static} + P_{dynamic})$$

- Turn off
- Switch less
- Lower frequency/Voltage
- Take less time

Contents

- The problem
- **XMOS XS1 architecture**
- Multi-threading breaks existing models
- Fixing the model
- New tricks
- Unanswered questions

XMOS XS1



- 32-bit micro-processor for embedded systems.
- ISA incorporates I/O instructions
- Event driven, real-time
- Message passing
- 64KB fast SRAM per core
- Deterministic WCET analysis
- **Hardware multi-threaded pipeline**

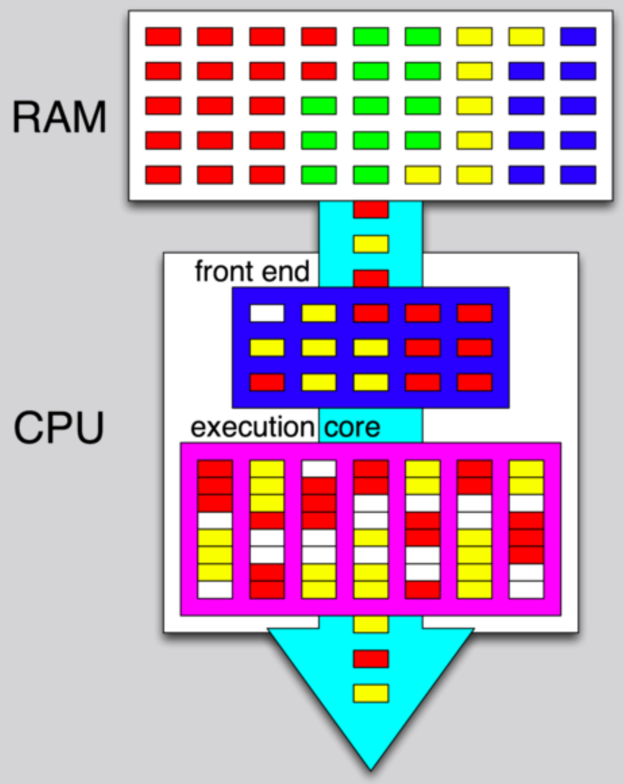
XMOS XS1

- Up to eight threads per core
- Four stage pipeline
- Simple scheduling
- At 500MHz, 125MIPS per thread for <= 4 threads

1 thread cycle

Clock cycle	Stage 1	Stage 2	Stage 3	Stage 4
n	**Thread 0**	Thread 4	Thread 3	Thread 2
n + 1	Thread 1	**Thread 0**	Thread 4	Thread 3
n + 2	Thread 2	Thread 1	**Thread 0**	Thread 4
n + 3	Thread 3	Thread 2	Thread 1	**Thread 0**
n + 4	**Thread 0**	Thread 3	Thread 2	Thread 1
n + 5	Thread 1	**Thread 0**	Thread 3	Thread 2

...and others



A (*very*) abstract representation of SMT

- **Consider also...**

- SMT "hyper-threaded" architectures
- Super-scalar CPUs
- Out-of-order execution

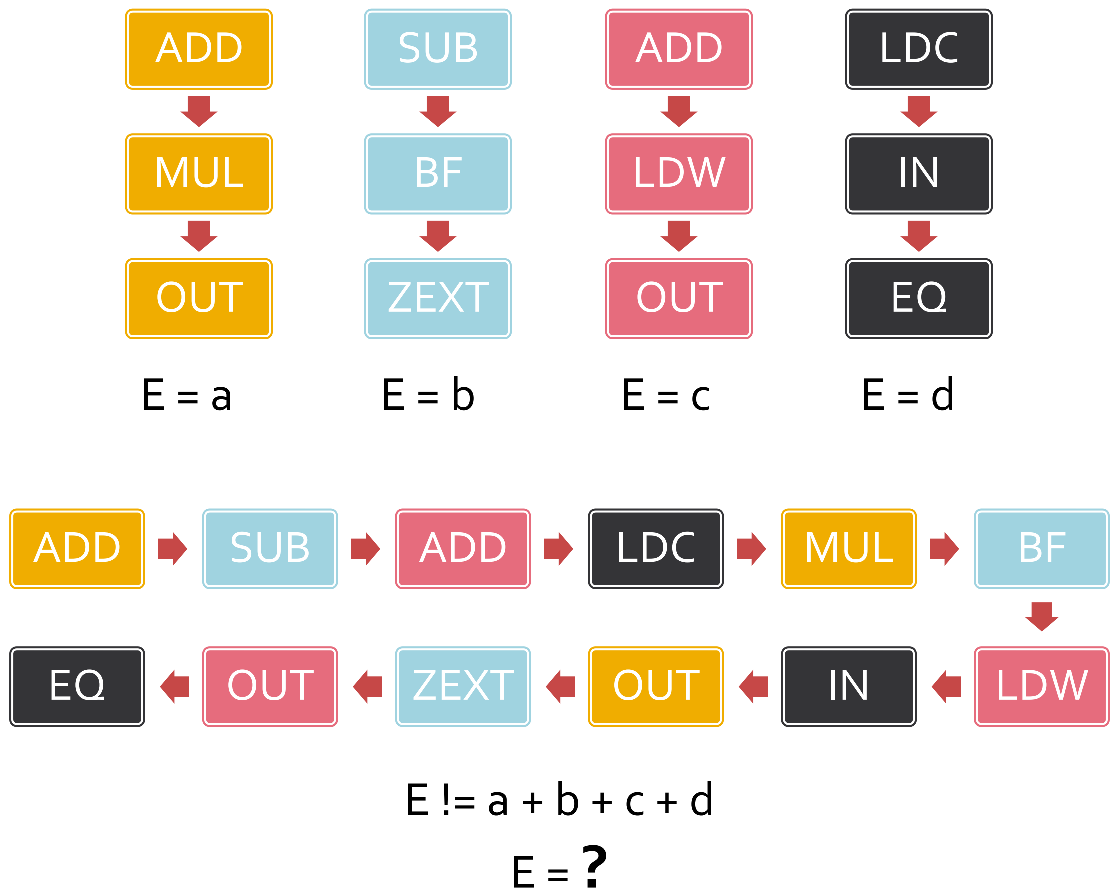
- **All affect switching through the CPU**

Contents

- The problem
- XMOS XS1 architecture
- **Multi-threading breaks existing models**
- Fixing the model
- New tricks
- Unanswered questions

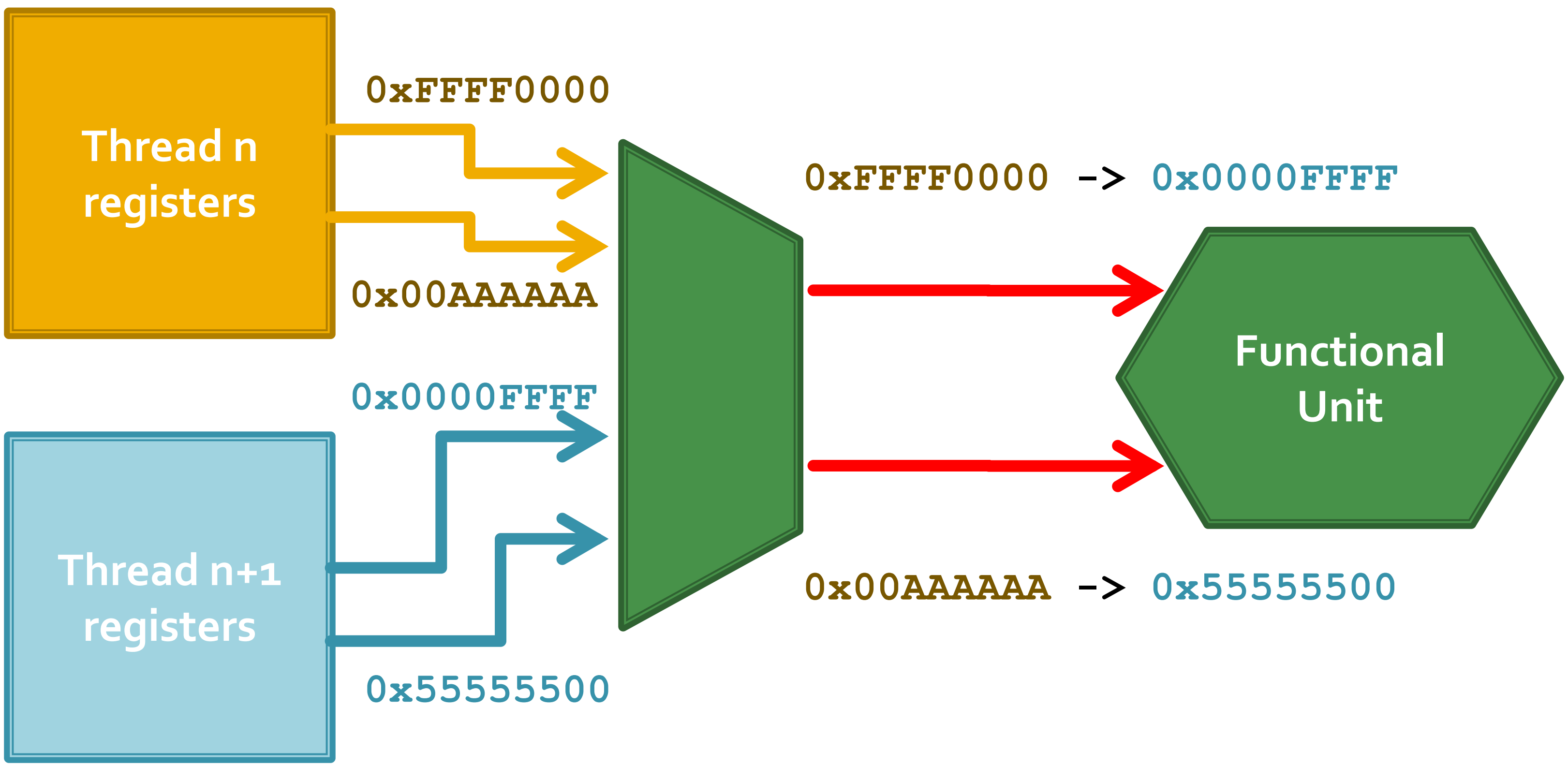
Breaking the model

- ISA-level example
- Model four threads
  - Get power figures
  - Determine energy consumption
- Run at architectural level
  - Interleaving in pipeline changes switching behaviour
  - For better or worse?
  - Regardless, **less accurate!**



Breaking the model

Data-path context switch example



Contents

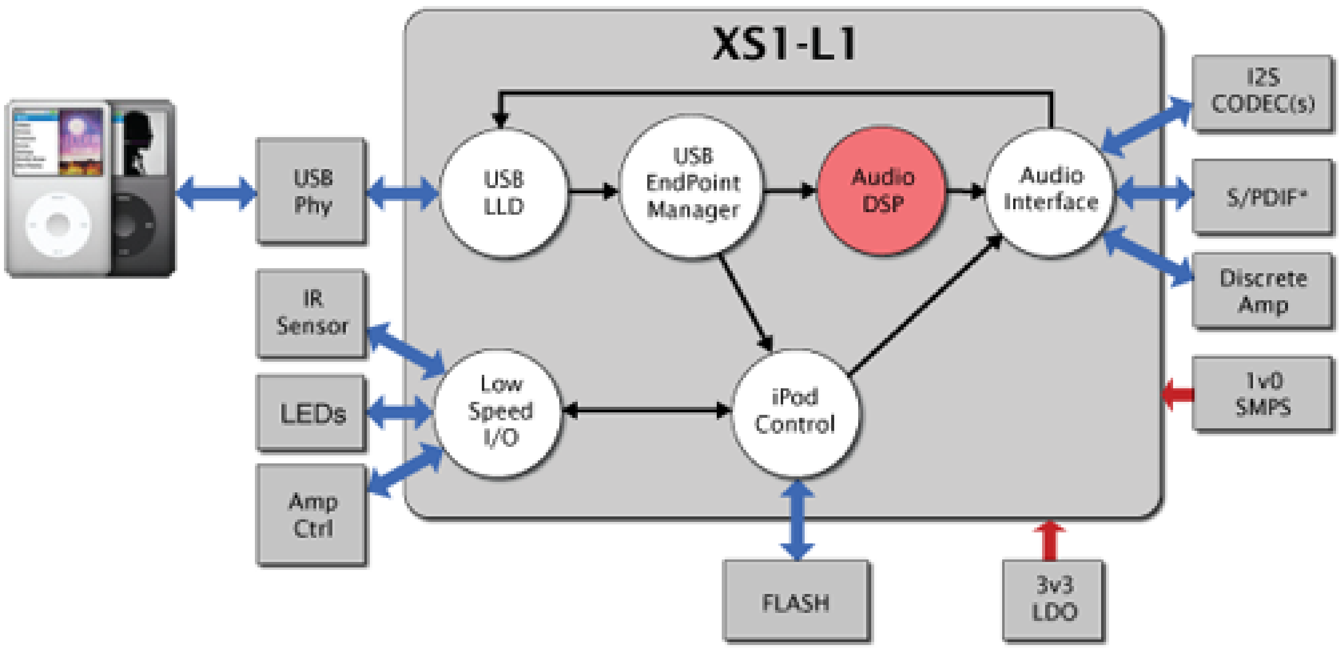
- Introduction
- The problem
- XMOS XS1 architecture
- Multi-threading breaks existing models
- **Fixing the model**
- New tricks
- Unanswered questions

Fixing the model

- Exactly how unpredictable is the pipeline?
- Independent unsynchronised threads – very unpredictable
- Interacting threads – more predictable
  - Known points of synchronisation
    - Explicit sync
    - Communication
    - Events

Fixing the model

- Thread interaction example
  - I/O & **protocols** govern activity
  - More detail gives a **higher accuracy model.**
  - The more we can **predict**, the more **energy we can save**.



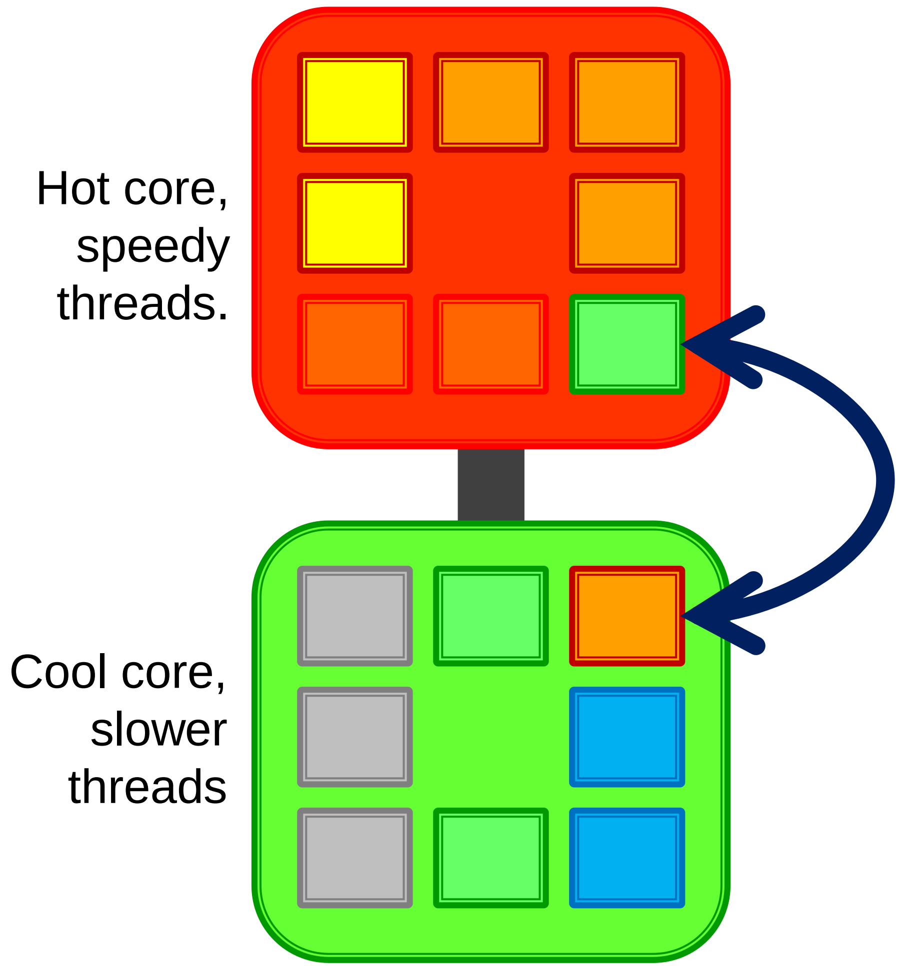
Fixing the model

- Incorporate thread sync/communication.
- Use protocol timing to inform thread schedule.
- Find sections of threads that execute together.
- Identify optimisation candidates.

Contents

- The problem
- XMOS XS1 architecture
- Multi-threading breaks existing models
- Fixing the model
- **New tricks**
- Unanswered questions

Trick 1: Migrate



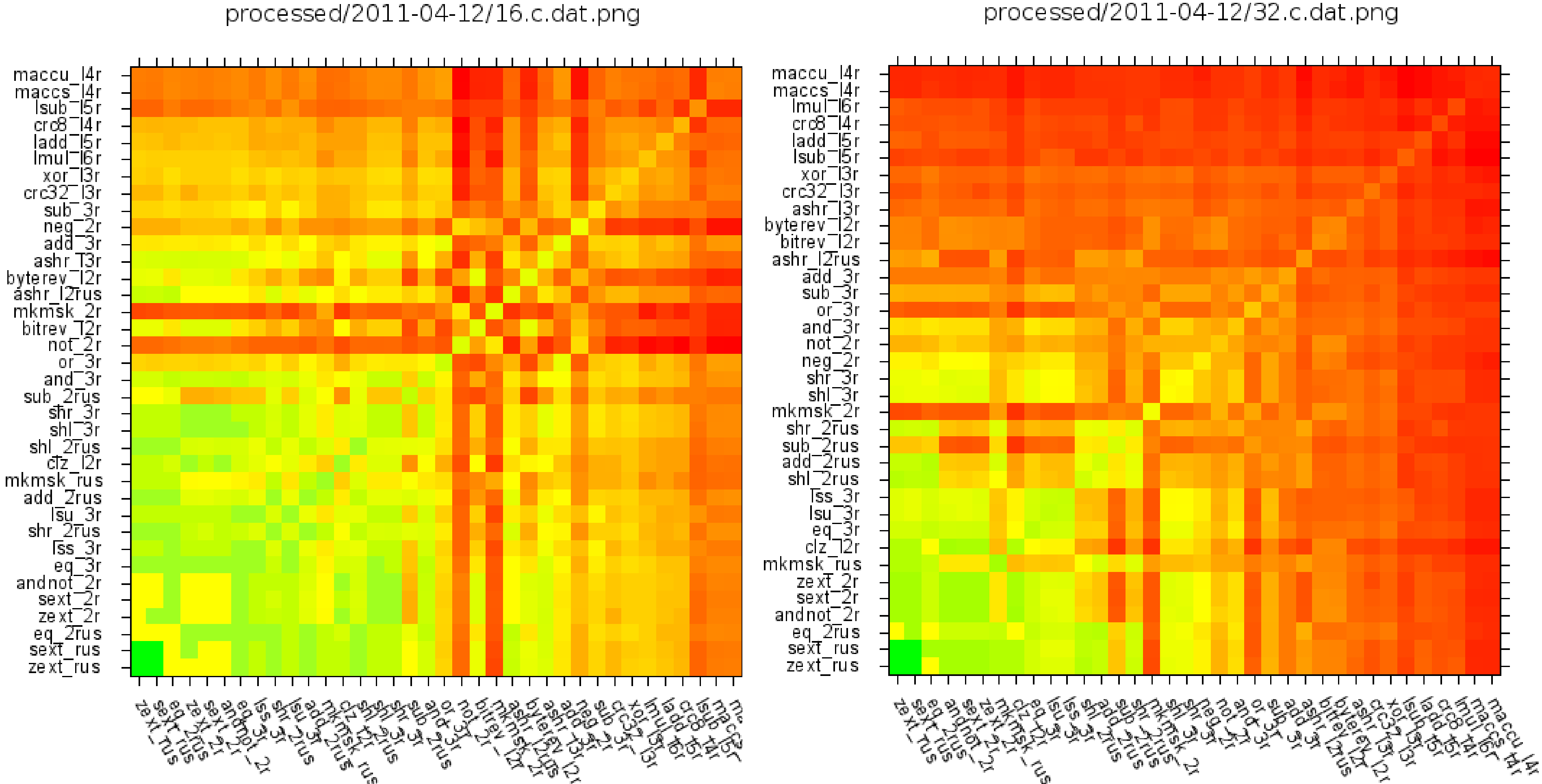
- Assign less demanding tasks to low frequency cores.
- Move tasks that interfere with other optimisation efforts onto a different core
- We have to deal with the **timing implications** of doing this.

Trick 2: Data width

- Example: 32-bit XCore hardware.
- The full width of the data path is **always active.**
  - No architectural specialisation for narrower data.
- Let's compare the switching with 32-bit and 16-bit values.

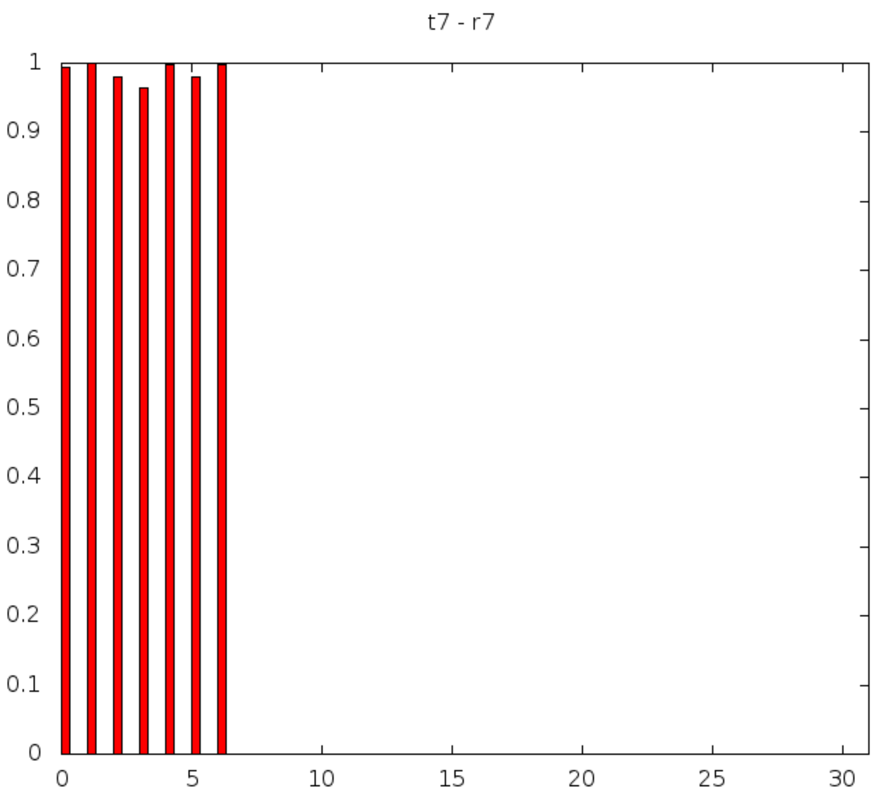
Trick 2: Data width

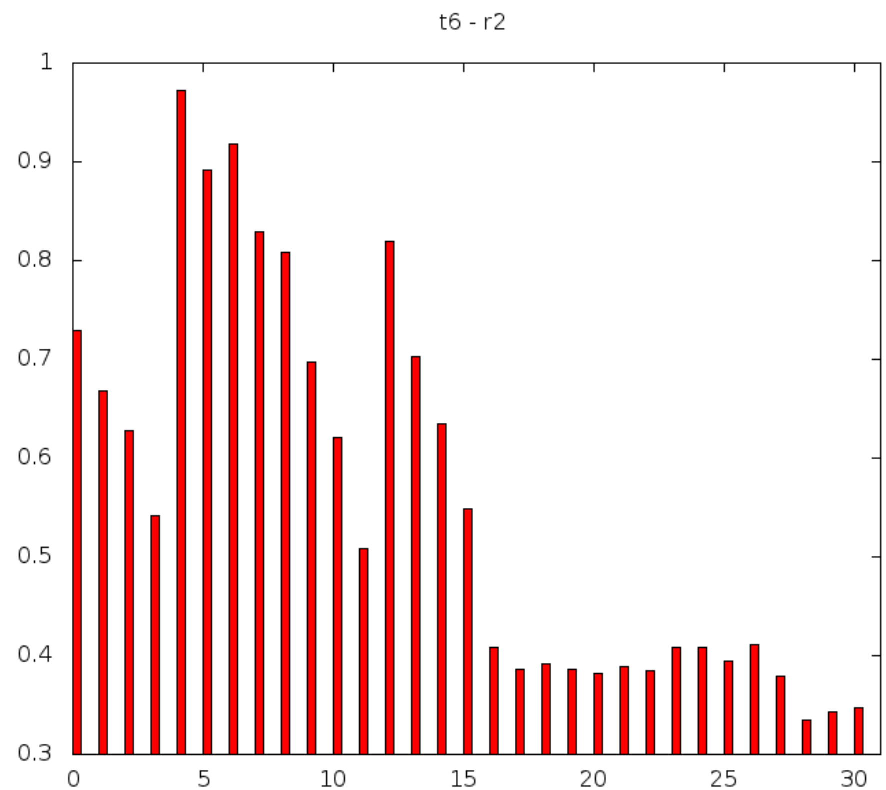
- **10-25%** improvement for some instruction pairings



Trick 2: Data width

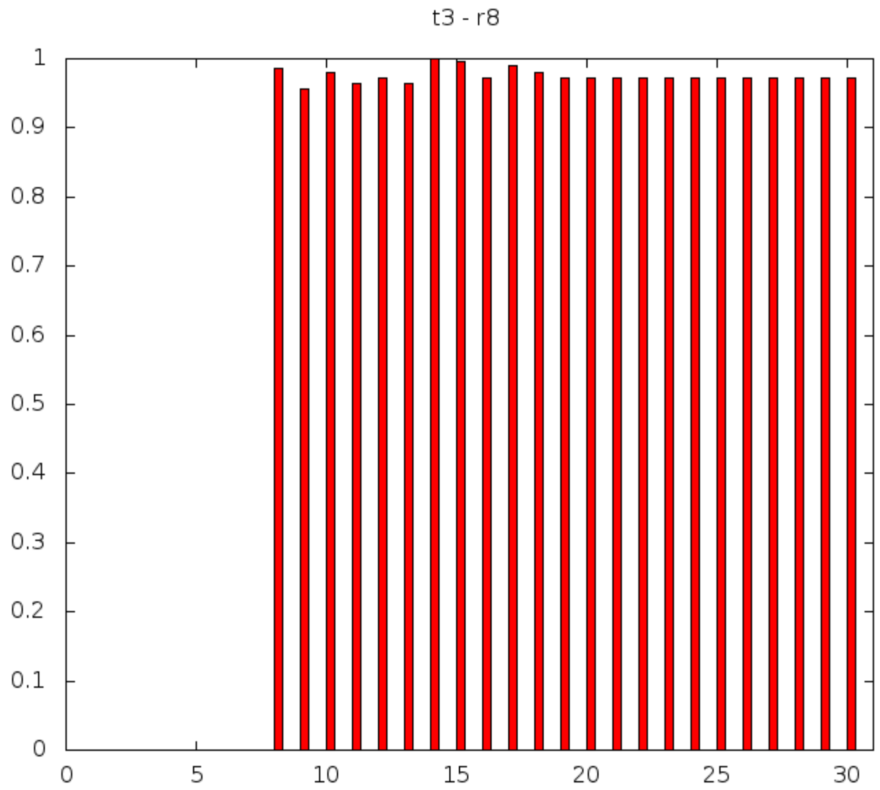
- Per-bit entropy sampled over the **lifetime** of a register.

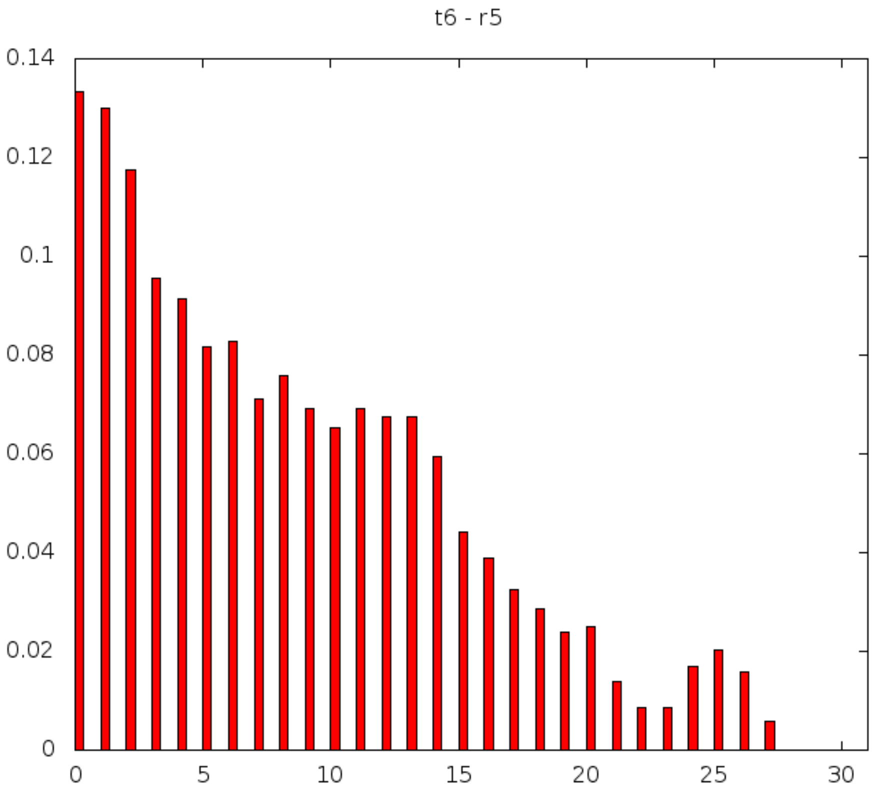




Trick 2: Data width

- Per-bit entropy sampled over the **lifetime** of a register.



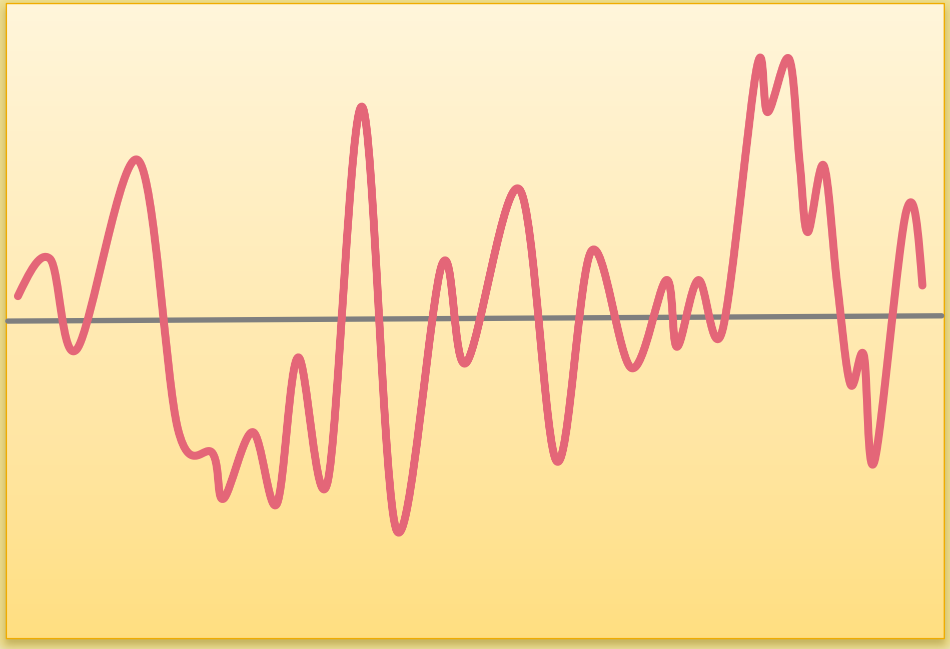


Trick 2.5: Precision

- Do we really need all 12-bits of that ADC sample?
- If we reduce precision of some data, what will the cumulative error be?
- How do we choose an appropriate trade-off between error, accuracy and energy?

Trick 3: Format

- Think of a sequence of audio samples.
- How often do we cross *zero*?
- What happens in twos complement when we do this?
  - 10 -> -10
  - 00001010 -> 11110110
  - 7 of 8 bits switch!



Trick 3: Format

- Think of a sequence of audio samples.
- How often do we cross *zero*?
- What happens in twos complement when we do this?
  - 10 -> -10
  - 00001010 -> 11110110
  - 7 of 8 bits switch!
- So what about sign-magnitude?
  - 10 -> -10
  - 00001010 -> 10001010
  - 1 bit switch!
  - 5-15% reduction seen by changing format.
- Unfortunately:
  - Redundant zero representation
  - No hardware support.

Contents

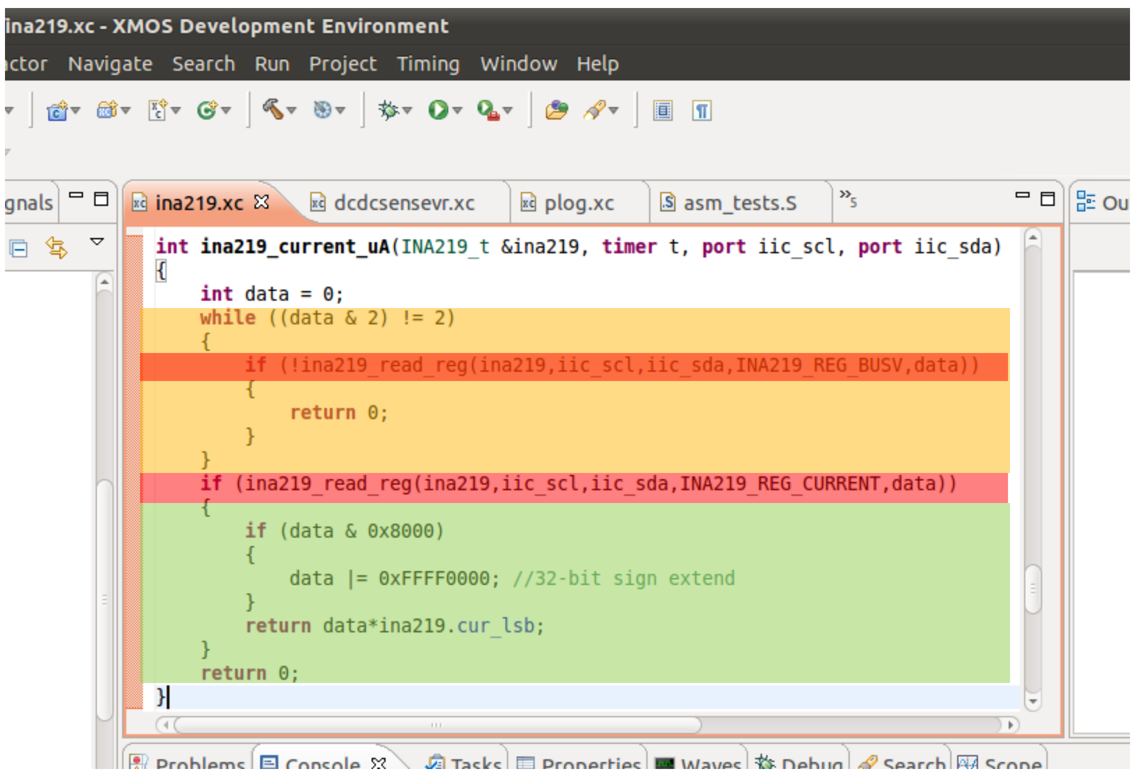
- The problem
- XMOS XS1 architecture
- Multi-threading breaks existing models
- Fixing the model
- New tricks
- **Unanswered questions**

Unanswered questions

- How do we knit together timing data, thread schedules and the energy cost of code segments?
- To what extent can inter-thread optimisation be performed?
  - How great are the benefits and when do they become worth the effort?

Unanswered questions

- What's the best way to present energy data to the developer?
- How do we minimise the amount of additional annotation required?
  - We want to improve workflow, not complicate it



Thank you

Questions welcomed…

More information

XMOS: http://www.xmos.com

UoB Micro research group: http://www.cs.bris.ac.uk/Research/Micro

Me: steve.kerrison@bris.ac.uk