

RUN TIME PREDICTION OF PERFORMANCE AND ENERGY WHEN FREQUENCY SCALING

David Snowdon, Stefan Petters and Gernot Heiser

David.Snowdon@nicta.com.au



Australian Government

**Department of Communications,
Information Technology and the Arts**

Australian Research Council

NICTA Members



Department of State and
Regional Development



The University of Sydney



Queensland University of Technology



NICTA Partners

RUN TIME PREDICTION OF PERFORMANCE AND ENERGY WHEN FREQUENCY SCALING

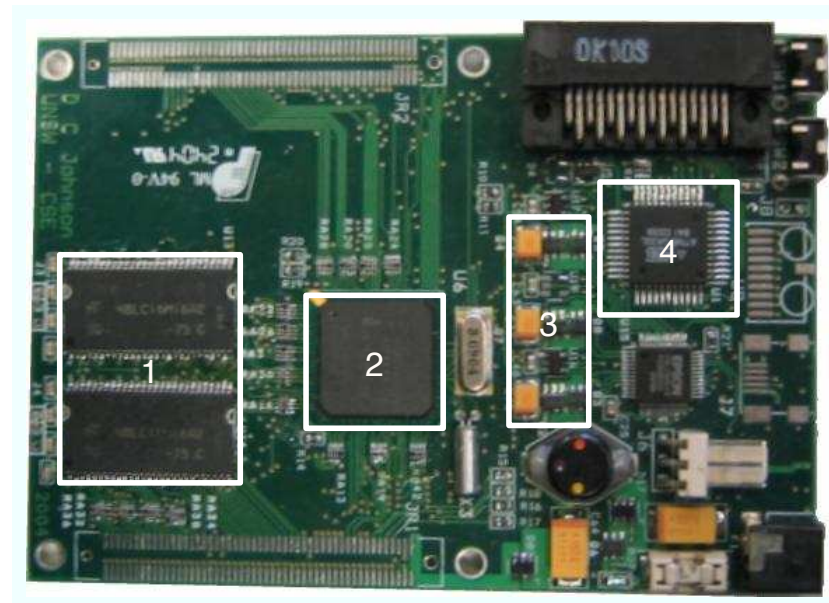
David Snowdon, Stefan Petters and Gernot Heiser

David.Snowdon@nicta.com.au

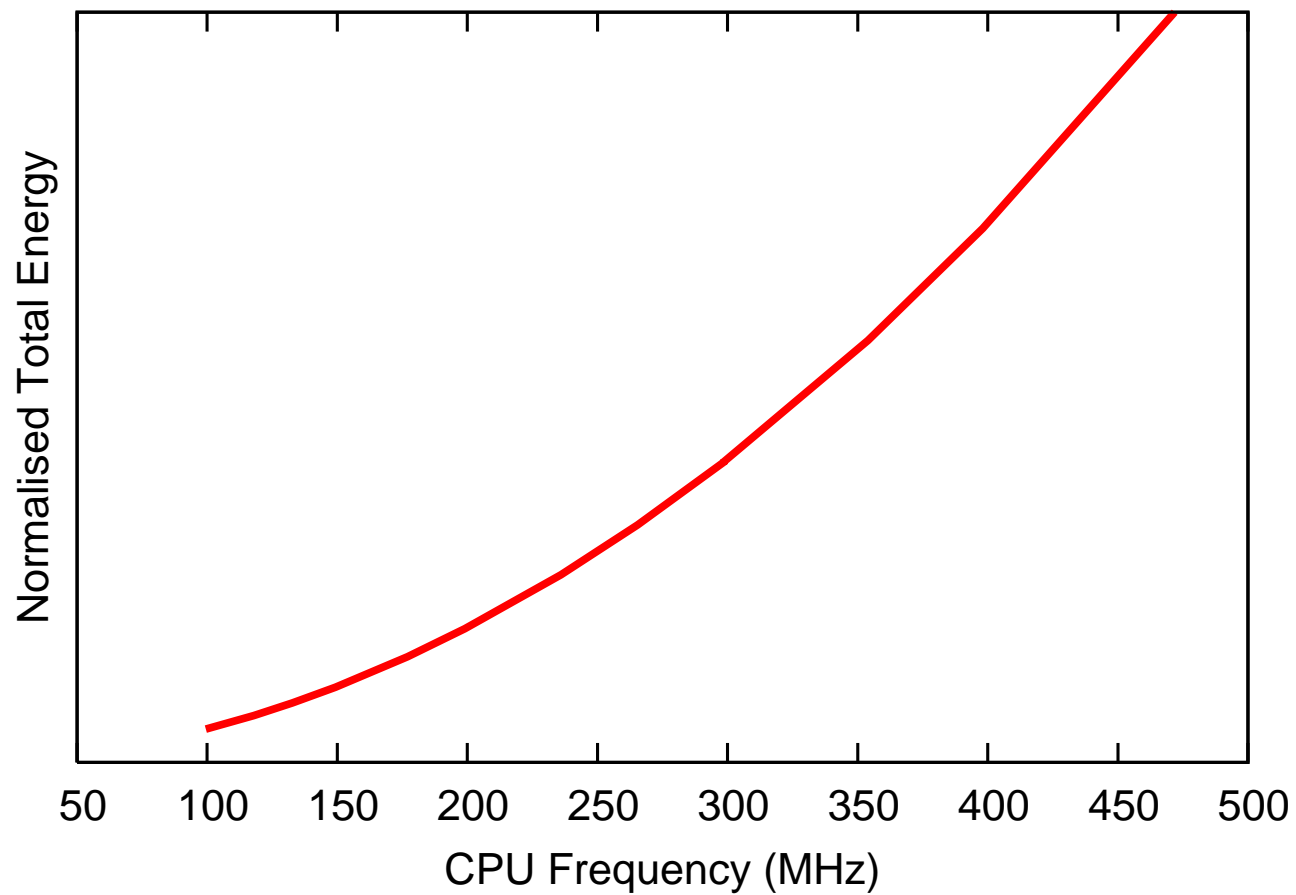
-
- ① Motivation: problems with DVFS
 - ② Modelling performance and energy
 - ③ Evaluation
 - ④ Future work

- Embedded systems are often restricted by battery life.
- Total system energy consumption.

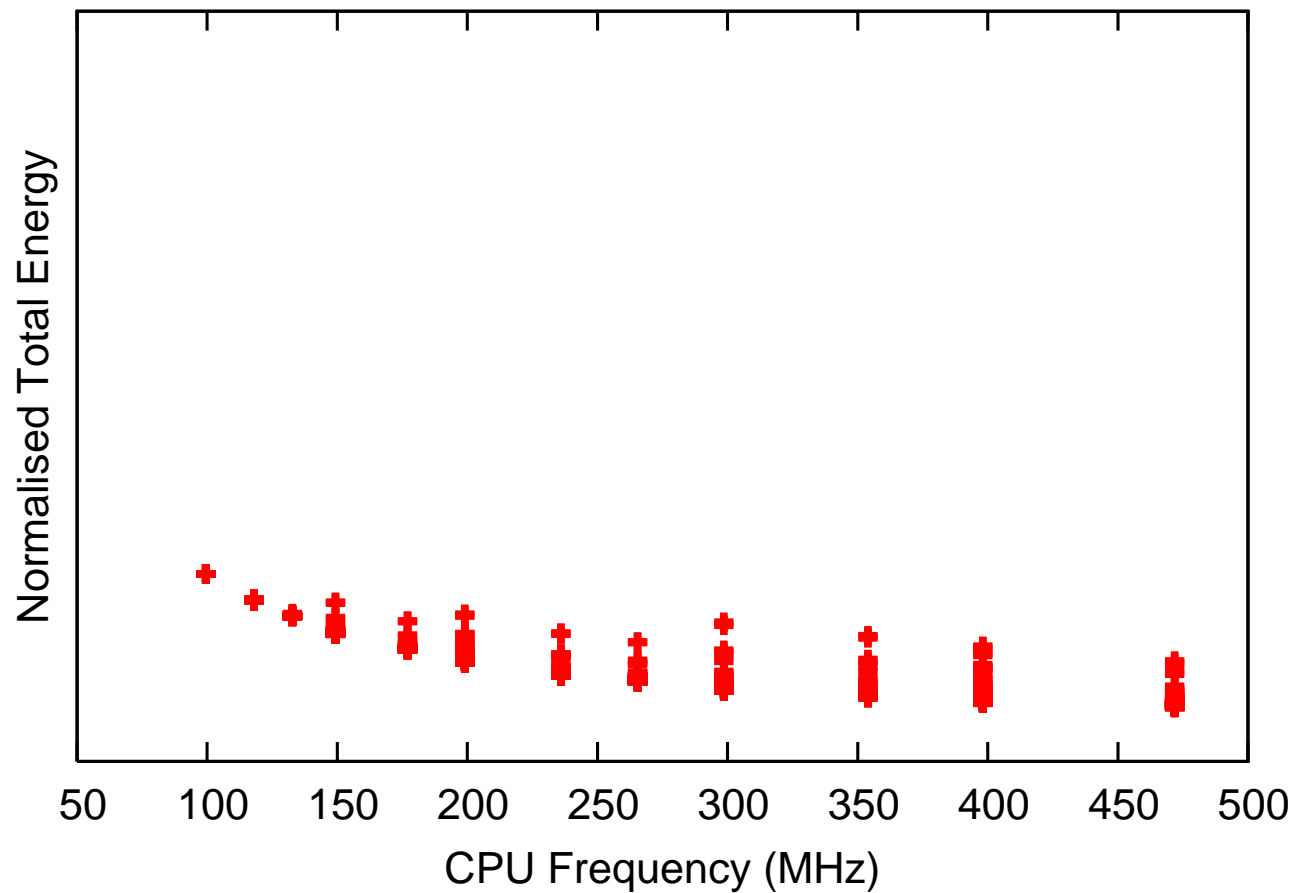
Our work looks at effective DVFS in real systems



Theory: $E \propto V^2$



Practice: (PXA255 based system)



Why?:

Why?: Simple models

→ $P \propto fV^2$

→ $T \propto \frac{1}{f}$

→ $V = F(f)$ and F monotonically increasing

Modern systems aren't simple!

→ Varying number of switches (workload specific!)

→ Multiple frequency domains

→ Frequency independent (static) power

Why?: Simple models

→ $P \propto fV^2$

→ $T \propto \frac{1}{f}$

→ $V = F(f)$ and F monotonically increasing

Modern systems aren't simple!

→ Varying number of switches (workload specific!)

→ Multiple frequency domains

→ Frequency independent (static) power

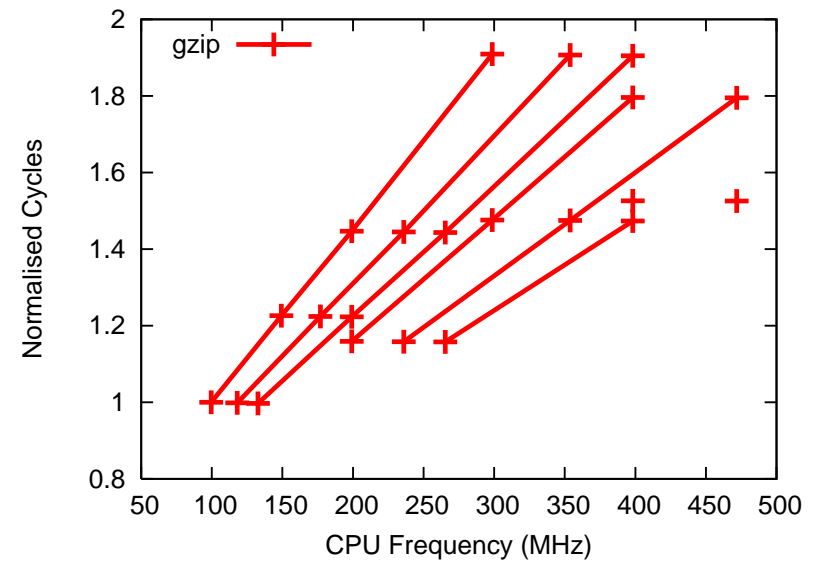
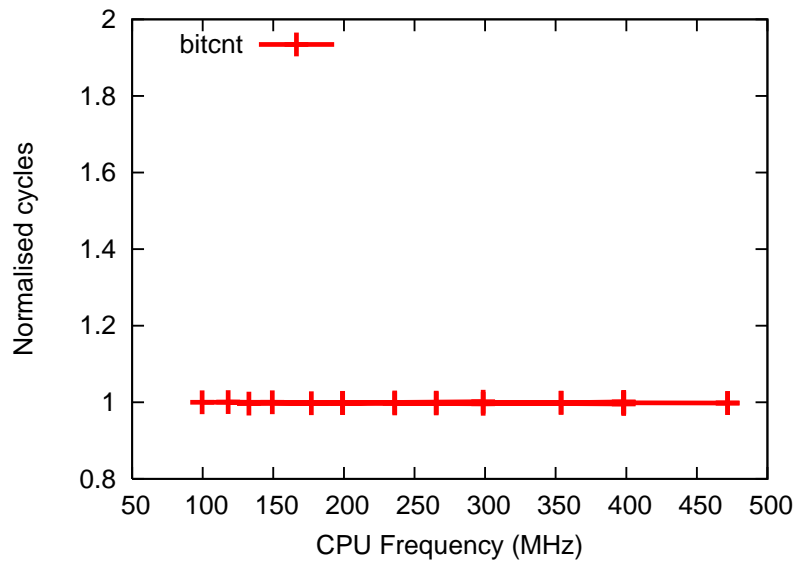
We want to be able to deal with these nuances

EXECUTION TIME MODEL

- Simple execution time model: $T \propto \frac{1}{f_{cpu}}$
- i.e. Constant cycles
- **Problem:** Ignores execution time independent of CPU-clock

EXECUTION TIME MODEL

- Simple execution time model: $T \propto \frac{1}{f_{cpu}}$
- i.e. Constant cycles
- **Problem:** Ignores execution time independent of CPU-clock



Implicaton:

- Memory-bound performance is less dependent on CPU frequency

- Task: predict the execution time of a workload in an arbitrary system configuration
- Low overhead, cross-architectural, dynamic applications

EXECUTION TIME MODEL

- Task: predict the execution time of a workload in an arbitrary system configuration
- Low overhead, cross-architectural, dynamic applications

$$T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \dots$$

EXECUTION TIME MODEL

- Task: predict the execution time of a workload in an arbitrary system configuration
- Low overhead, cross-architectural, dynamic applications

$$T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \dots$$

C_x : characterise a-priori, or online using performance counters

$$C_{bus} = \alpha_1 PMC_1 + \alpha_2 PMC_2 + \dots$$

$$C_{mem} = \beta_1 PMC_1 + \beta_2 PMC_2 + \dots$$

(C_{cpu} inferred from the other results)

EXECUTION TIME MODEL

- Task: predict the execution time of a workload in an arbitrary system configuration
- Low overhead, cross-architectural, dynamic applications

$$T = \frac{C_{cpu}}{f_{cpu}} + \frac{C_{bus}}{f_{bus}} + \frac{C_{mem}}{f_{mem}} + \frac{C_{io}}{f_{io}} + \dots$$

C_x : characterise a-priori, or online using performance counters

$$C_{bus} = \alpha_1 PMC_1 + \alpha_2 PMC_2 + \dots$$

$$C_{mem} = \beta_1 PMC_1 + \beta_2 PMC_2 + \dots$$

(C_{cpu} inferred from the other results)

- 2-parameter: avg 1.7%, max 7%
- CPU frequency only: avg 10%, max 36%

→ Simple CMOS model: $P \propto fV^2$

Problems:

- System power
- Static power and leakage
- Multiple frequency/voltage domains
- Temperature dependence
- Conversion inefficiencies

POWER MODEL

→ Simple CMOS model: $P \propto fV^2$

Problems:

- System power
- Static power and leakage
- Multiple frequency/voltage domains
- Temperature dependence
- Conversion inefficiencies

A (slightly more) realistic model:

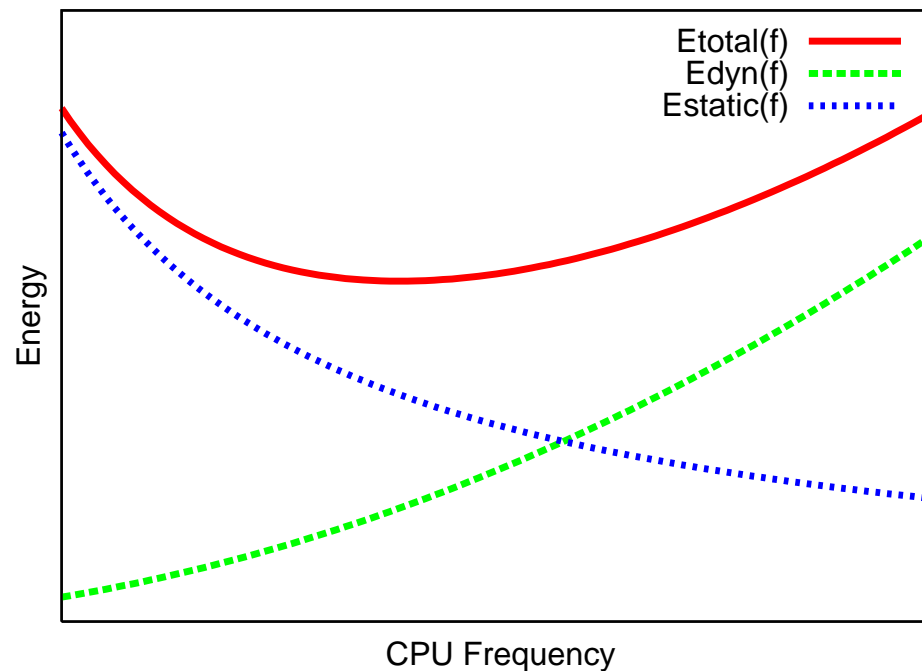
$$P = \sum_{n=0}^N C_n f_n V_n^2 + P_{static}$$

POWER MODEL

The interaction of run-time and static power:

- Dynamic energy increases as frequency increases
- Static energy decreases as frequency increases

$$E_{total} = P_{dyn}\Delta t + P_{static}\Delta t$$



POWER MODEL

Power/Energy model principles:

- *Events* each use an amount of energy
- An event may use energy in more than one voltage domain

For our system:

$$E_{events} = V_{cpu}^2 (\alpha_0 PMC_0 + \dots + \alpha_m PMC_m) + \beta_0 PMC_0 + \dots + \beta_m PMC_m$$

POWER MODEL

Power/Energy model principles:

- *Events* each use an amount of energy
- An event may use energy in more than one voltage domain
- Clocks cycles count as events

For our system:

$$E_{events} = V_{cpu}^2 (\alpha_0 PMC_0 + \dots + \alpha_m PMC_m) + \beta_0 PMC_0 + \dots + \beta_m PMC_m$$

$$E_{freqs} = V_{cpu}^2 (\gamma_1 f_{cpu} + \gamma_2 f_{bus} + \gamma_3 f_{mem}) \Delta t + (\gamma_4 f_{cpu} + \gamma_5 f_{bus} + \gamma_6 f_{mem}) \Delta t$$

POWER MODEL

Power/Energy model principles:

- *Events* each use an amount of energy
- An event may use energy in more than one voltage domain
- Clocks cycles count as events
- Static power models power not related to events or voltages.
- Constant IO power for the benchmarks tested.

For our system:

$$E_{events} = V_{cpu}^2 (\alpha_0 PMC_0 + \dots + \alpha_m PMC_m) + \beta_0 PMC_0 + \dots + \beta_m PMC_m$$

$$E_{freqs} = V_{cpu}^2 (\gamma_1 f_{cpu} + \gamma_2 f_{bus} + \gamma_3 f_{mem}) \Delta t + (\gamma_4 f_{cpu} + \gamma_5 f_{bus} + \gamma_6 f_{mem}) \Delta t$$

$$E_{static} = P_{static} \Delta t$$

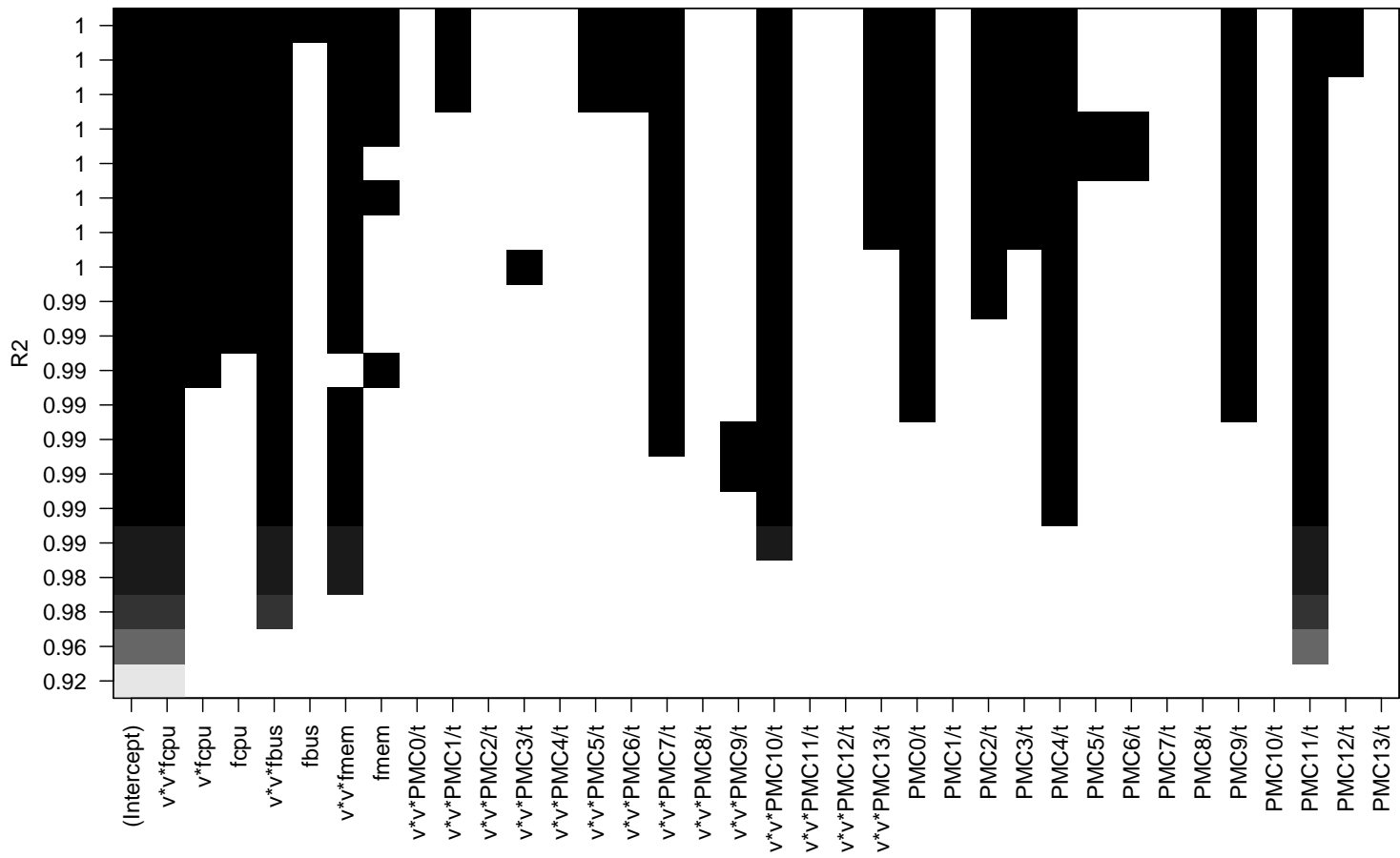
Parameter selection:

- Systematically picking the best model for N counters
- Least-squares regression finds the coefficients

- Typical embedded platform (PLEB 2, XScale based)
- Cycle counter, 2 performance counters, 13 events

- Typical embedded platform (PLEB 2, XScale based)
- Cycle counter, 2 performance counters, 13 events
- 37 benchmarks run to completion at each setpoint for all frequency settings
- 22 frequency setpoints with different f_{cpu} , f_{bus} and f_{mem}
- Voltage varied to three settings for each frequency

- Typical embedded platform (PLEB 2, XScale based)
- Cycle counter, 2 performance counters, 13 events
- 37 benchmarks run to completion at each setpoint for all frequency settings
- 22 frequency setpoints with different f_{cpu} , f_{bus} and f_{mem}
- Voltage varied to three settings for each frequency
- Measurements: Cycles, Frequencies, Performance counters, Energy
- Benchmarks were partitioned for calibration and validation



Power error data (using a measured run-time):

Counters	Param.	R^2	Max Err (%)	Avg Err (%)
1	4	0.983	7.5	2.1
2	5	0.987	6.9	2.3
3	6	0.990	4.8	1.3
4	7	0.992	3.8	1.2
5	8	0.993	3.7	0.9
6	9	0.994	2.9	0.9
6	11	0.995	2.7	0.8

Combining with our performance model:

→ Max: 4.9%, Avg: 1.5% — (Four counters)

Conclusions:

- A time estimate is essential for an energy estimate
- Our method: Low overhead, accurate, generally applicable
- Parameter selection is a useful tool for analysing power
- Energy can be saved by changing the memory/bus frequency

Conclusions:

- A time estimate is essential for an energy estimate
- Our method: Low overhead, accurate, generally applicable
- Parameter selection is a useful tool for analysing power
- Energy can be saved by changing the memory/bus frequency

Future work:

- Temperature, IO, interrupts and DMA are unaccounted for.
- Investigate the effectiveness for other platforms.
- Develop frequency switching policies:
 - Model frequency switching overheads, sleep state energy
 - Operating system and application knowledge

QUESTIONS?



QUESTIONS?

`David.Snowdon@nicta.com.au`

`http://ertos.nicta.com.au`

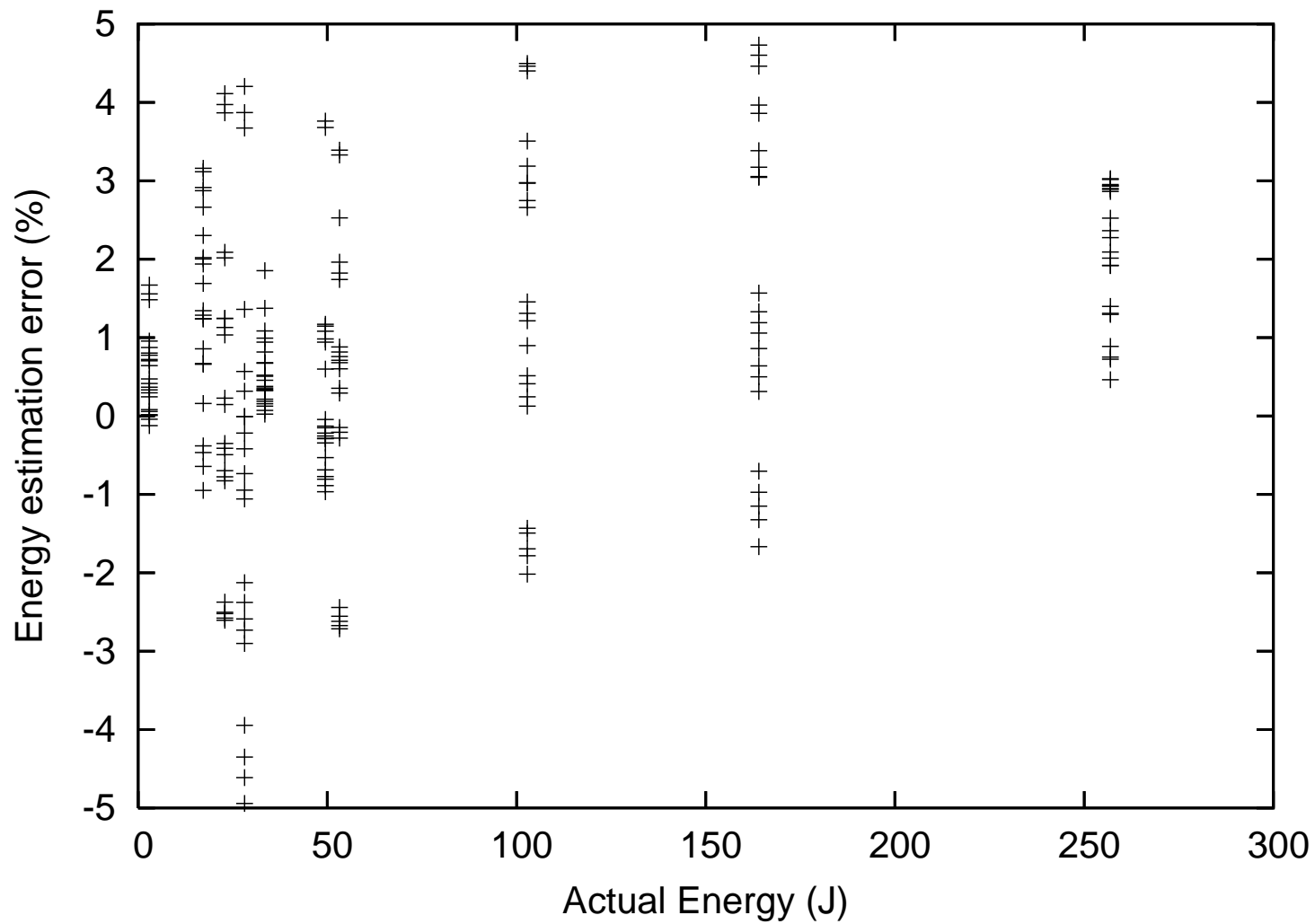
AVAILABLE PMC EVENTS

PMC	Description
0x0	ICache miss
0x1	ICache stall cycles
0x2	Data dependency stalls
0x3	ITLB miss
0x4	DTLB miss
0x5	Branch instruction executed
0x6	Branch mispredicted
0x7	Instruction executed
0x8	DCache buffer stall cycles
0x9	DCache buffer stall
0xa	DCache access
0xb	DCache miss
0xc	DCache write-back
0xd	Software changed the PC

AVAILABLE FREQUENCY SETPOINTS

	f_{cpu} (MHz)	f_{bus} (MHz)	f_{mem} (MHz)
1	99.531	49.766	99.531
2	117.964	58.981	117.964
3	132.71	66.354	132.71
4	149.299	49.766	99.531
5	176.946	58.981	117.964
6	199.064	49.766	99.531
7	199.064	66.354	132.71
8	199.064	99.531	99.531
9	235.929	58.981	117.964
10	235.929	117.964	117.964
11	265.420	66.354	132.710
12	265.420	132.710	132.710
13	298.598	49.766	99.531
14	298.598	99.531	99.531
15	353.894	58.981	117.964
16	353.894	117.964	117.964
17	398.131	66.354	132.71
18	398.131	99.531	99.531
19	398.131	132.71	132.71
20	398.131	199.064	99.531
21	471.858	117.964	117.964
22	471.858	235.929	117.964

ERROR DISTRIBUTION



FIXED MEMORY FREQUENCY

