



An Application-Specific and Adaptive Power Management Technique

Bernhard Egger Jaejin Lee Heonshik Shin
School of Computer Science and Engineering
Seoul National University
Korea





Overview

- Motivation & Goals
- Advanced Power Manager (APM)
 - characterization of the target device
 - APM regions
 - detecting periodic behavior
- Implementation & Experiments
- Conclusion

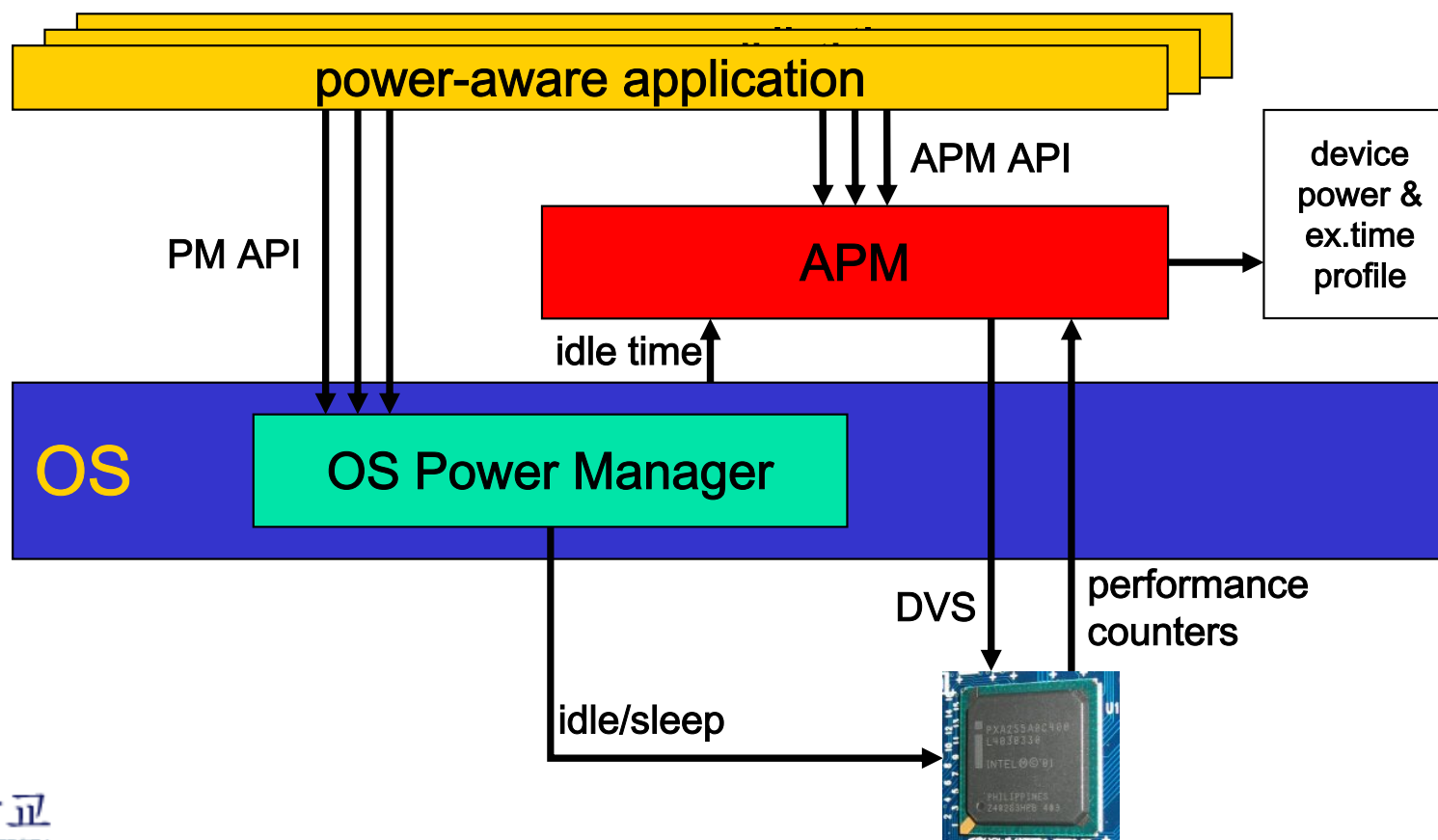


Motivation & Goals

- reduce total system energy
 - on a multi-purpose OS
 - consider peripherals
 - without modifying the scheduler
- exploit
 - idle time
 - CPU stalls
- rely on
 - idle time reported by the OS
 - the CPU's performance counters

System Overview

- Advanced Power Manager (APM)
 - enhances the conventional OS power manager
 - invoked by power-aware applications



Dynamic Voltage Scaling

- DVS:

- $P_{chip} = ACfV^2 + I_{leak}V + P_{short}$

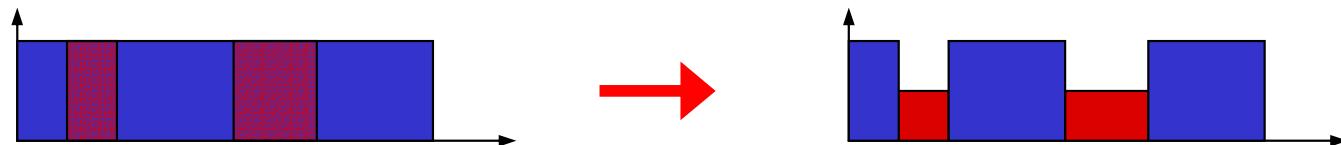
- exploit slack and/or idle time

- idle time: no task ready to run
 - slack time: task finishes before its deadline (RTOS)



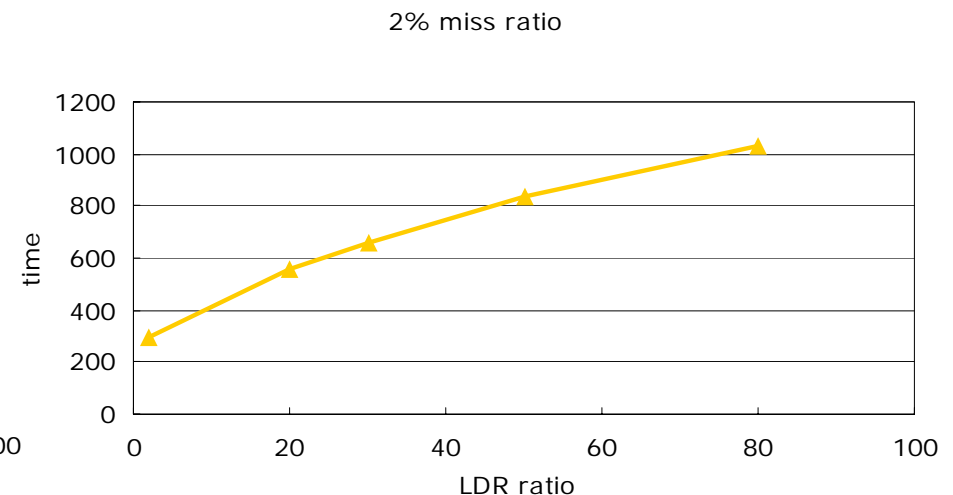
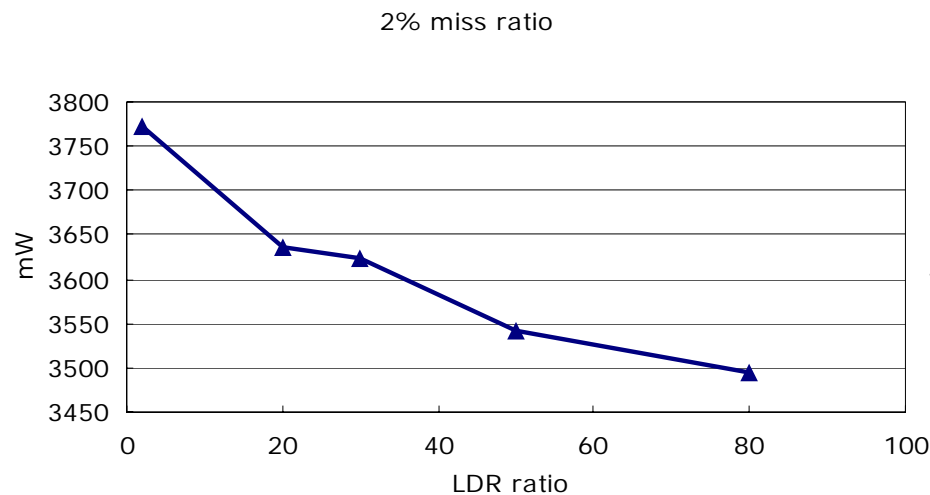
- exploit CPU stall time

- during execution of memory-bound code regions



Device Characterization

- purpose
 - obtain a power & execution time profile
- the power consumption of the core components and the execution time depend on
 - CPU frequency, bus frequency, memory latency, instruction mix



Device Characterization

- execute
 - varying instruction mixes
 - at all frequencies
- data obtained is approximated by several linear functions

frequency	LDR ratio			
	2%	20%	80%	
0				
cache miss ratio	1%	2.665	2.621	2.547
	2%	2.496	2.447	2.428
	3%	2.582	2.504	2.457
	5%	2.562	2.515	2.431
	10%	2.522	2.490	2.429

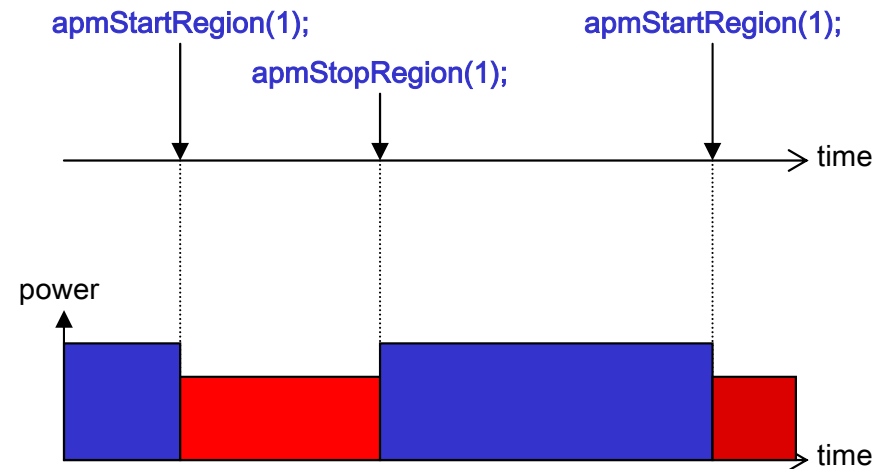
- power & execution time estimation formulae

$$P_r = P(f, instr, acc, miss)$$

$$T_r = T(f, instr, acc, miss)$$

APM Regions

- APM regions
 - potentially memory-bound code regions
- `apmStartRegion`
 - start measuring
 - # of instructions executed
 - # of cache accesses
 - # of cache misses
 - switch to optimal frequency
- `apmStopRegion`
 - stop measuring
 - switch back to original frequency
 - calculate optimal frequency



Energy Calculation

- Calculating the energy for a region
 - using the measured execution time and performance counter values
 - $E_r = P(f_{new}, instr, acc, miss) \cdot T(f_{new}, instr, acc, miss) + E_{switch}(f_{old}, f_{new})$
- recalculate whenever the values change significantly
- compute a time correction factor
 - used to improve the accuracy of the following calculations
 - $t_{corr} = \frac{t_{computed}}{t_{measured}}$

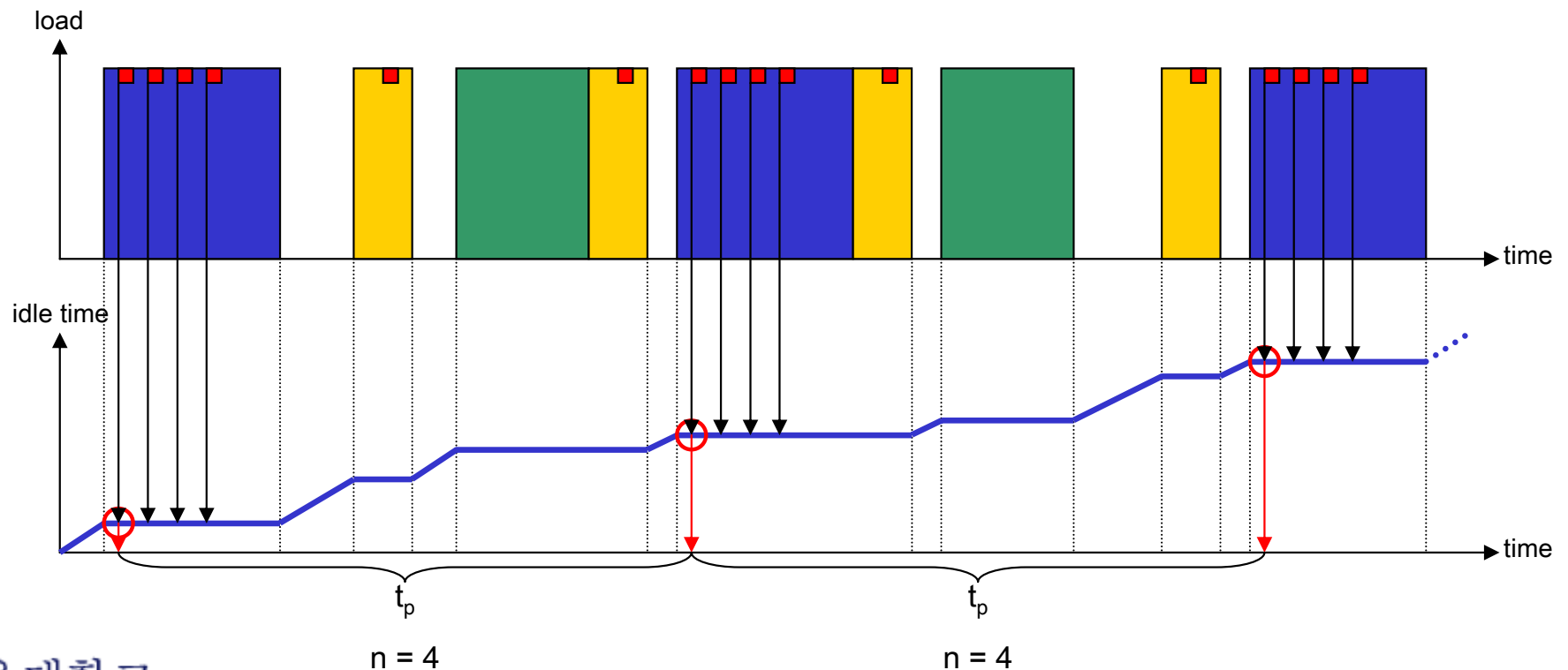


Region Placement

- APM calls should be placed around 'interesting' code regions
 - hot-spots
 - loop nests
- placement can be done by
 - static compiler analysis
 - profiling
 - the programmer

Detecting Periodic Behavior

- to exploit idle time
- only regions of the highest priority process are candidates
- relies on the idle time reported by the OS
- only one periodic region



Energy Calculation

- Calculating the energy for a periodic region
 - using the measured execution time and performance counter values

- $$E_p = n \cdot E_r(f_r, instr_r, acc_r, miss_r) + P(f_p, instr_p, acc_p, miss_p) \cdot T(f_p, instr_p, acc_p, miss_p) + P_{idle}(f_p) \cdot (t_p - n \cdot T_r - T(f_p, instr_p, acc_p, miss_p))$$

- constrained by

$$t_p \geq n \cdot T_r + T(f_p, instr_p, acc_p, miss_p)$$

- switching overhead negligible



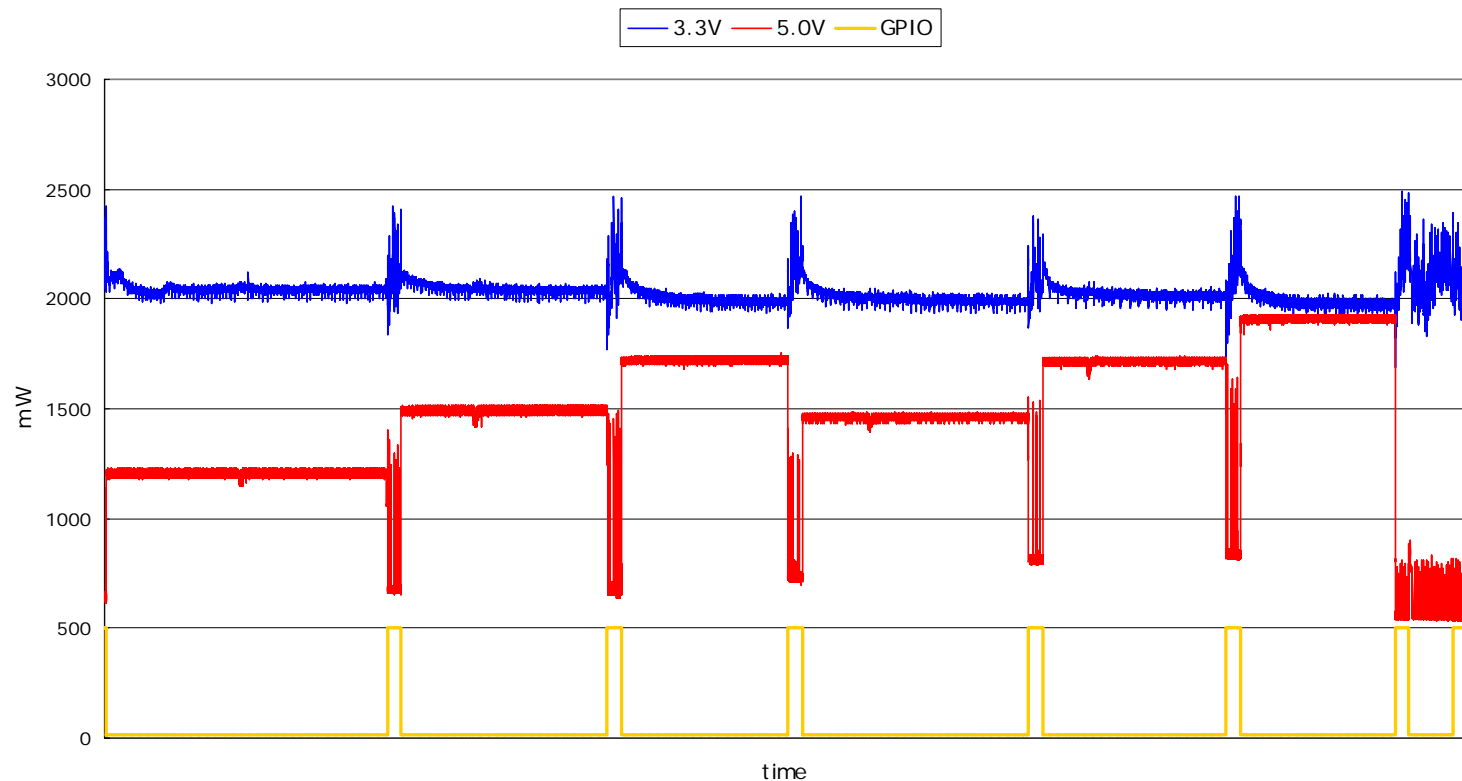
Implementation

- Hardware
 - Intel Xscale PXA255 CPU (99.5 – 398.1MHz)
 - Xscale evaluation board
 - 64MB flash memory, 64MB SDRAM
 - 320x160 pixel-wide LCD screen
 - FPGA
 - NIC, serial, IrDA, USB, PCMCIA, CF Card
 - 2 power supply lines (3.3V and 5.0V)

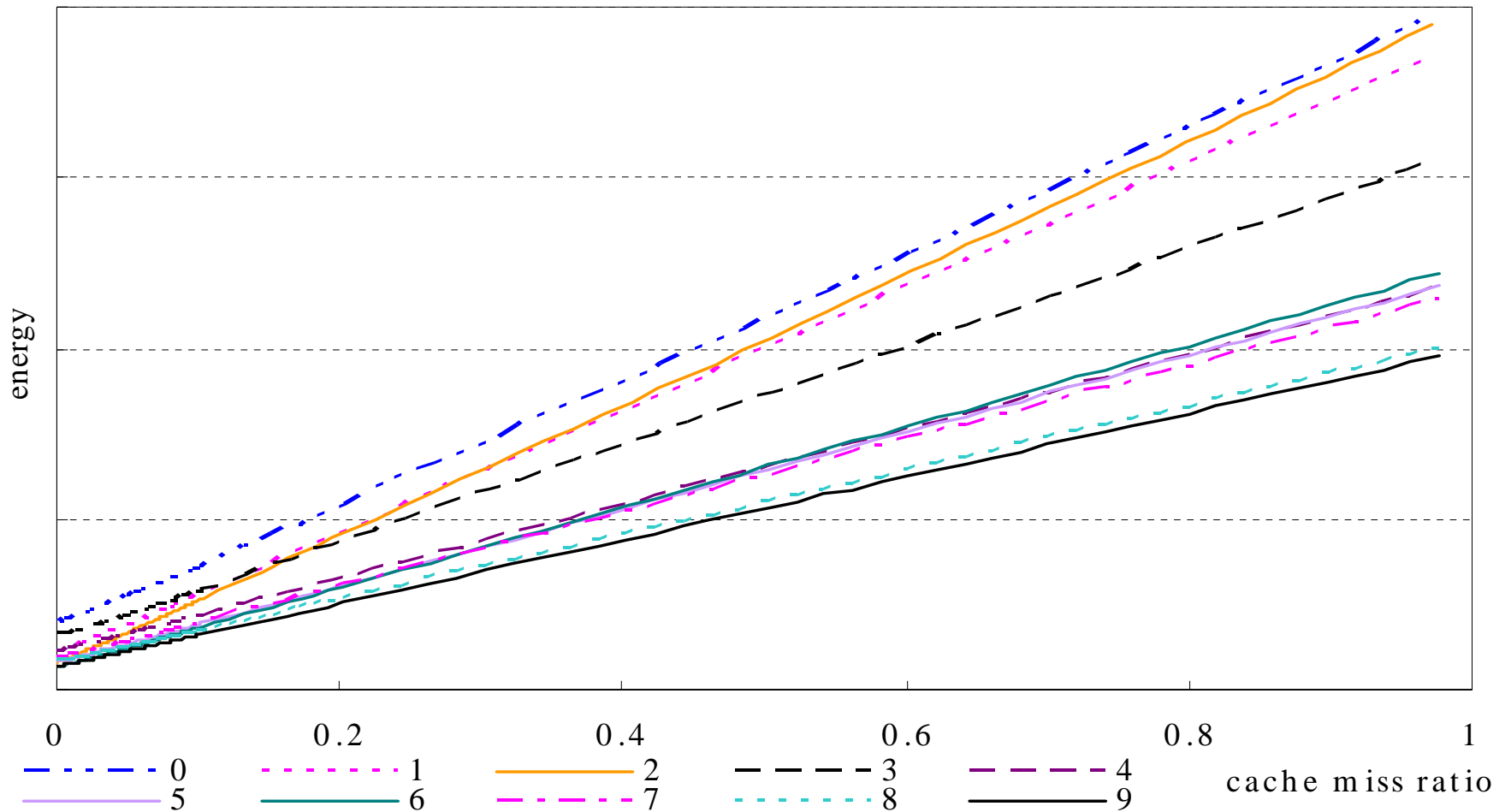
- Software
 - Microsoft Windows CE.NET 4.2
 - Microsoft eMbedded Visual C++ 4.0

Power & Time Profile

- executing 100 million instructions
- measurement performed with a high-speed analog data acquisition system
- GPIO pin used to signal the beginning and the end of a certain code section



Xscale: Fastest is Best...





Xscale: Fastest is Best...

- changing the CPU core frequency of an Xscale processor also changes the bus speed:

f_{CPU}	$f_{CPU \rightarrow MEM}$	$f_{MEM \rightarrow LCD}$	f_{SDRAM}
99.5	50.0	99.5	99.5
199.1	50.0	99.5	99.5
298.6	50.0	99.5	99.5
132.7	66.0	132.7	66.0
199.1	99.5	99.5	99.5
298.6	99.5	99.5	99.5
398.1	99.5	99.5	99.5
265.4	132.7	132.7	66.0
331.8	165.9	165.9	83.0
398.1	196	99.5	99.5

- memory-bound code regions run much slower
- in our experimental environment, fastest is always best
→ we can't save energy in memory-bound code regions



mpeg2dec

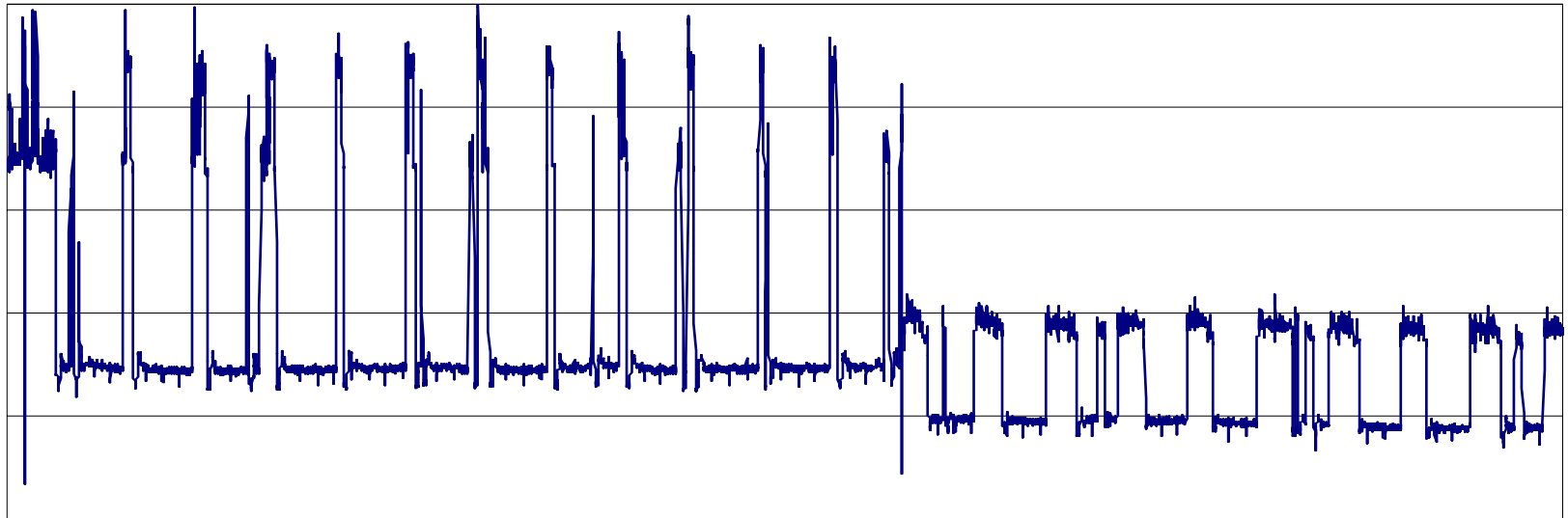
- reference MPEG decoder by the MPEG group
- part of MediaBench

- Windows CE port
 - no output
 - fixed framerate, i.e., periodic decoding

- APM regions
 - one APM region, using profiling
 - # regions per period varies

mpeg2dec

- overall power consumption



- energy savings

video size	Standard PM		APM		Power savings (%)
	avg. power [W]	avg. jitter [ms]	avg. power [W]	avg. jitter [ms]	
80x60	2.873	1.10	2.644	1.29	8.0
160x120	3.189	1.10	2,955	2.06	7.3
256x192	3.507	1.12	3.365	1.25	4.0



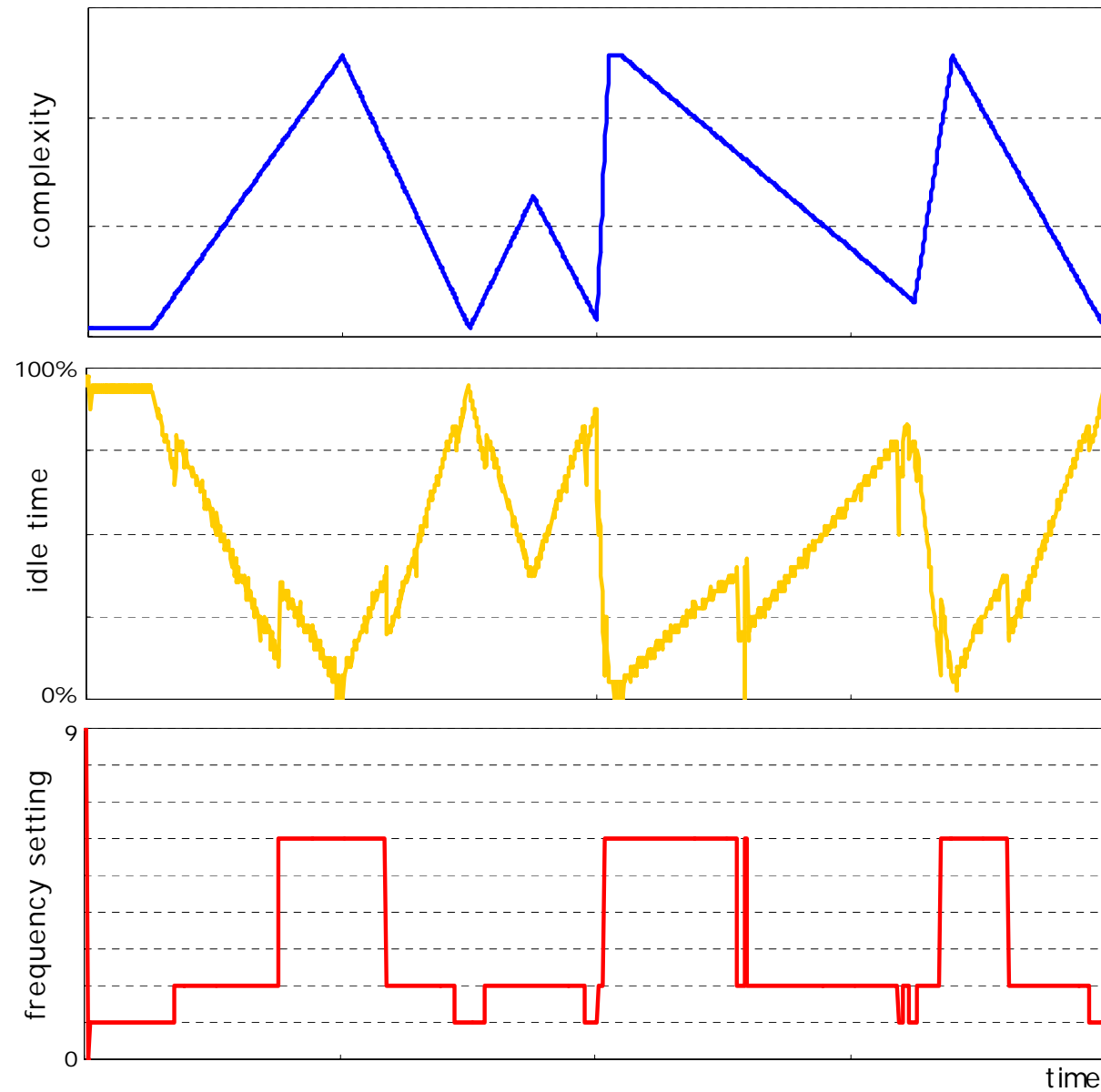
3dview

- synthetic benchmark
- simulates a 3D world viewer
- load varies over time

- Windows CE port
 - no output
 - 25 frames/second

- APM regions
 - one APM region, placed by hand
 - constant # regions per period

3dview

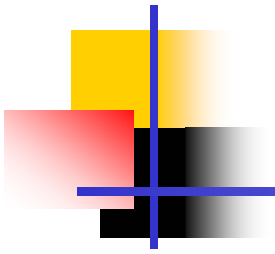




Conclusions & Future Work

- application-specific and adaptive power management technique
 - based on a execution time/power profile of the target device
 - computation performed online
 - can exploit both periodic behavior and memory-bound code
 - without modifying the scheduler

- implement on Intel Celeron/AMD Athlon Mobile



Questions?