**Power and Energy Management**

Ulrich (Uli) Kremer  
Ricardo Bianchini  
Department of Computer Science  
Rutgers University

---

**Why Power/Energy Management?**

- Prolong battery life
- Reduce heat dissipation  
  - packaging costs and cooling  
  - reliability

![Pentium III processor vs. Transmeta's Crusoe TM5400](image)

Without cooling: 105°C (221°F)  
vs. 42°C (111°F)  
running DVD application (Source: Transmeta’s Crusoe processor white paper)

---

EEL/DARK Light Seminar, Spring 2003  
EEL Laboratory
HW/OS/Compiler Optimizations

optimizations

- try to improve quality of code (may fail in some cases)
- optimizing compilers typically consists of multiple passes
- different optimization objectives:
  - execution time reduction
  - reduction in resource requirements (memory, registers)
  - (peak) power and energy reduction

criteria for effectiveness of optimizations

- safety - program semantics must be preserved
- opportunity - how often can it be applied?
- profitability - how much improvement?

Power vs. Energy

power: activity level at a given point in time
energy: total amount of activity

same energy, different (peak) power

optimizing for (peak) power == optimizing for energy?
**Power vs. Energy**

**power**: activity level at a given point in time  
**energy**: total amount of activity

same energy, different (peak) power

**optimizing for (peak) power == optimizing for energy?**

**ANSWER**: Not necessarily! Example: re-schedule activities

---

**Power/Energy vs. Performance**

**performance**: overall program execution time

**optimizing for power/energy == optimizing for performance?**
Power/Energy vs. Performance

**performance**: overall program execution time

**optimizing for power/energy == optimizing for performance?**

**ANSWER**: (a) Mostly Yes, at least for traditional optimizations that reduce overall computation and memory activity

- redundancy elimination (CSE, PRE, dead code elimination)
- strength reduction (e.g.: replace $2*a$ with $a+a$), loop invariant code motion
- memory hierarchy (locality) optimizations (register allocation, loop interchange, loop distribution, blocking for cache)

**ANSWER**: (b) Not really, in particular for optimizations that exploit tradeoff between power/energy usage and performance

- loop invariant code motion, aggressive speculation
- blocking for cache
- DFVS, resource hibernation, remote task execution, QoR
Power/Energy vs. Performance

Which code is better in terms of power/energy and which in terms of performance?

(A)  
```
for (i=0, i<10; i++) {
    a = b* 2;
    c[i] = d[i] + 2.0;
}
```

(B)  
```
a = b* 2;
for (i=0, i<10; i++) {
    c[i] = d[i] + 2.0;
}
```

**Answer:** It depends
- simple RISC architecture: (B)
- VLIW or superscalar architecture with empty "slots": (A)
Power/Energy vs. Performance

You can run, but you cannot hide
- pushing instructions onto the non-critical execution path ("hiding") does not necessarily reduce energy
- higher threshold for profitability of speculation

You cannot beat hardware
- if an operation is implemented in hardware, that’s the best you can do (e.g.: floating-point unit)
- need to be able to disable hardware if not in use

Keep the overall picture in mind
- performance is measured for the entire program
- power/energy should also be measured for the entire system, in addition to optimized system component

Why Compiler and not OS/Hardware?

Compiler advantages:
- low or no overhead (power/energy and performance) at program execution time
- may know about “future” program behavior through aggressive, whole program analysis.
- can better identify profitability of high overhead optimizations based on large context analysis.
- can reshape program behavior through code transformations and thereby enable optimizations.

Compiler disadvantages:
- insufficient information about runtime program behavior may lead to code of poor quality
**Why Compiler and not OS/Hardware?**

Possible Scenarios:

- **single user environment:** compiler-directed power and energy management is directly "executed" by underlying OS/HW.
- **multiple user environment:** power/energy application profile is used by OS to make scheduling decisions.

---

**Syllabus**

- **Web site:**
  
  http://www.cs.rutgers.edu/~uli/PowerEnergy/spring2003

- **Research papers:** our reading list will be constructed on the fly; possible topics (incomplete)
  - resource hibernation
  - remote task execution
  - dynamic voltage and frequency scaling
  - QoS tradeoffs
  - load concentration (servers)

- **Will cover techniques from hardware, OS, and compiler, and possible interactions between them**

- **Students give one 30 minutes presentation**