ISWC'01:
Tutorial on low power design for wearable computers:
"Waste not, want not."

Tom Martin
Virginia Tech
The Bradley Dept. of Electrical and Computer Engineering
tlmartin@vt.edu

Overview

- Power and energy
- Power sources
- Hardware
- Software
- Wrap-up

Power and Energy

- Why worry about power consumption?
  - – Battery life
  - – Weight
  - – Size
  - – Heat
  - – Reliability

Power and Energy

- Relationship between power and energy
  - – Power = *d/dt* (Energy)
- Units:
  - – Power: Watts
  - – Energy: Joules
  - – 1 J = 1 W × 1 s
  - – Some equivalents:
    - 1 W = 1V × 1 A
    - 1 J = 1 V × 1 C
- "Low power" ≠ "Low energy" !
  - – E.g. a system with low power consumption but long latency

---

Power and Energy

- No "magic bullet"—the whole system affects power consumption:

Software
Hardware
Power source

- We'll start from the bottom and work our way up…

Power Sources

- Typically batteries
  - – There are alternatives: solar panels, fuel cells, and so on.
- Battery parameters:
  - – Primary (disposable) vs. secondary (rechargeable)
  - – Voltage
  - – Capacity (Either charge or energy)
  - – Chemistry

Power Sources

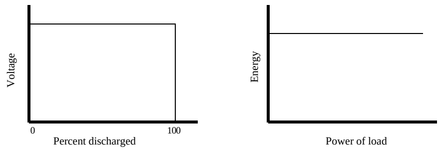
- Battery chemistry
  - Electrode materials determine voltage and theoretical capacity

Chemistry	Th'l Voltage, V	Th'l Charge Capacity, Ah/kg	Th'l Energy Capacity, Wh/kg
Lead acid	2.1	120	250
NiCd	1.3	180	240
NiMH	1.3	210	270
Li-ion	3.9	100	390

  - Realizable capacities are *much* less than the values listed
  - A reference point: TNT energy capacity is 1300Wh/kg

Power Sources

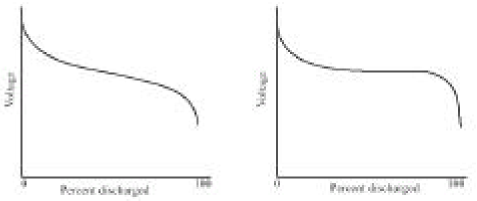
- Battery properties: Ideal



Constant voltage Constant capacity

Power Sources

- Typical battery voltage curves

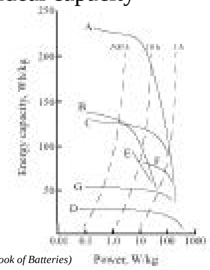


a. Sloped discharge profile b. Flat discharge profile

Power Sources

- Battery properties: Non-ideal capacity

Capacity versus power for several battery systems. A: Li/MnO$_2$ 2/3A cell; B: Zn/alkaline/MnO$_2$ AA cell; C: Li-MnO$_2$ AA cell; D: Ni-Cd AA cell; E: Zn/alkaline/MnO$_2$ AA cell; F: Li-ion AA cell; G: NiMH AA cell. A and B are primary cells, C-G are secondary cells. Dashed lines show time of discharge for reference.



(Adapted from Linden's *Handbook of Batteries)*

Power Sources

- Estimating battery life, first order:

  L = Energy capacity/Average power

- Energy capacity: $E = V_{nom} \times Crating$

- Average power:

$$P_{ave} = P_{a1}\times t_{a1} + P_{a2}\times t_{a2} + \dots P_{an}\times t_{an}$$

where P_{ai} = average power in state i,

t_{ai} = percent of time in state i

Power Sources

- Estimating battery life when capacity is non-ideal (roughly, less than 4 hours):

$$L = (C(P_{peak}) \times V_{nom})/P_{ave}$$

where $P_{peak} = max(P_{ai})$,

$C(P_{peak})$ is the capacity at the peak power

Hardware

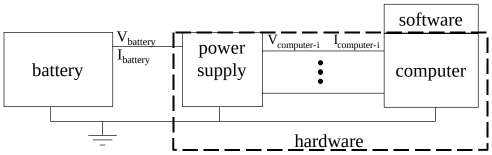
- Done with power sources, now on to hardware:

Software
Hardware
Power source

- Most hardware can't operate at the battery voltage:
  - Power supply required to produce useful voltages

Hardware

- Typical wearable system:



- If power supply were 100% efficient:

$$P_{battery} = P_{computer}$$

$$Or: \; V_{battery} I_{battery} = S\,(V_{computer-i} I_{computer-i})$$

Hardware

- Efficiency = P_{out}/P_{in}, or, in our case,

$$Efficiency = P_{computer}/P_{battery}$$

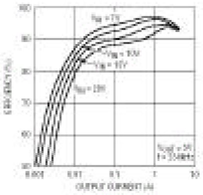
- Two types of power supply:
  - Linear regulator
    - Fewer components, usually less efficient
  - Switching regulator
    - More components, usually more efficient

Hardware

- Linear regulator efficiency:
  - For linear regulators, $I_{in} = I_{out}$
  - Then Efficiency = $V_{out}I_{out}/V_{in}I_{in} = V_{out}/V_{in}$
  - But V_{out} £ $V_{in} - V_{dropout}$
  - Assume $V_{dropout}$ = 1V, $V_{in} = V_{bat}$ = 3.5V, V_{out} = 2.5V
  - Then Efficiency = 2.5V/3.5V = 71%
  - If V_{ba} = 4.5V, Efficiency = 2.5V/4.5V = 56%
  - Average efficiency is approx. 64%
  - For every 1 W in the computer, you need 1 W/0.64 = 1.6 W from the battery…

Hardware

- Switching regulator efficiency:



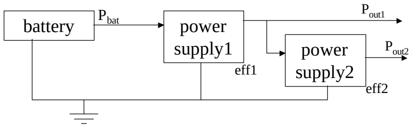
(From Linear Technology Databook)

- Typical efficiency > 90% over most of range, so every 1W for computer requires 1W/0.9 = 1.1W from battery

---

Hardware

- Some implications: Cascading supplies



- $Pbat = (P_{out1} + P_{out2}/eff2)/eff1$
  $= P_{out1}/eff1 + P_{out2}/(eff1 \times eff2)$
- If P_{out2} dominates, then overall efficiency will be close to $(eff1 \times eff2)$

Hardware

- Power management: Putting idle subsystems into low power modes
- Power-performance trade-offs: Decreasing performance (and thus power) while active
- Difficult to call these "hardware" – they involve aspects of both hardware and software

---

Hardware

- Power management: Cost of turning subsystem back on vs. power saved while it is off.
- Example: IBM Microdrive
  - – 0.45W spinning
  - – 0.07W standby
  - – 0.7W startup (2-3 seconds)
  - – Then drive must be able to shut off for 3.1-4.7 seconds to break even…
    - And this doesn't consider the power consumed by the rest of the system during those 2-3 seconds…

Hardware

- Overall savings depends on fraction of total system power and fraction of time subsystem is idle
- E.g. If subsystem is 10% of total power, then if we can shut it off all the time, the maximum savings is 10%…

Hardware

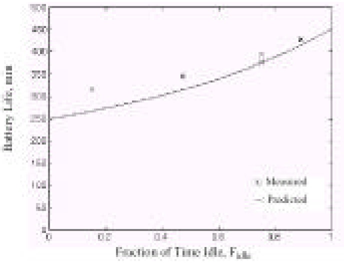
- The equivalent of Amdahl's Law for power consumption:

$$BatteryLife = \frac{Energy}{(1 - F_{idle}) \times P_{active} + F_{idle} \times P_{idle}}$$

Where $Energy$ is the (constant) battery capacity,
F_{idle} is the fraction of time idle,
P_{idle} is the power while idle, and
P_{active} is the power while active

Hardware

- Experimental example:



Hardware

- Dynamic power management: Benini, et. al. "A Survey of Design Techniques for System-Level Dynamic Power Management", IEEE Trans. on VLSI, June 200
- Queueing theory approach to power management:

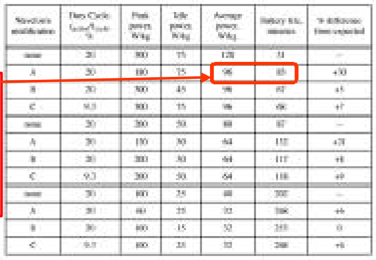
Fig. 9. Markov model of a power-managed system and its environment.

Hardware

- Non-ideal battery properties show that power-performance trade-offs are desirable:

Hardware

- Simulation results:



Same average power, but much greater battery life

Hardware

- Power performance trade-offs
  - CPU clock frequency
  - Wireless transmit power
  - Main memory bandwidth

- But lower power isn't necessarily more energy efficient, nor does it necessarily lead to more computations completed per battery life …

Hardware

- CPU clock frequency: Most previous work on this problem makes the following assumptions:
- Power α *fCV*^{2}
- Execution time α *1/f*

⇒ Energy per operation is

a) **constant** if *V* is kept constant

b) **proportional to *s*^{2}** if both *V*, *f* are varied by a factor of *s*

Hardware

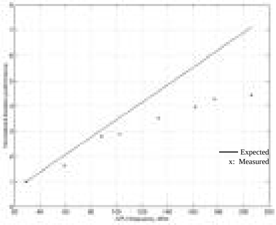
- Problems with those assumptions:
  - The CPU isn't the only power consumer
  - Performance doesn't necessarily scale with frequency
  - Implicitly assumes battery capacity is constant
- Must account for all three

- First, CPU as part of system power: $P = S + fCV^2$
- Then energy per operation $= P/f = S/f + CV^2$
  - Only constant if $S/f << CV^2$

Hardware

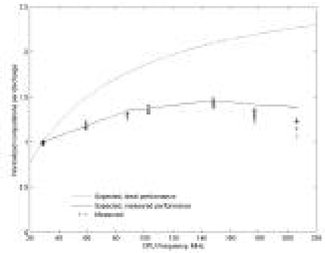
•Second, memory bandwidth affects performance

Overall performance of loop at different CPU frequencies:



Hardware

Normalized computations per battery life:



Hardware

- Other major problem: Predicting work load
- Lots of people have looked at this:
  - Weiser, et. al. "Scheduling for Reduced CPU Energy," Proceedings of the 1st USENIX Symposium on Operating Systems Design and Implementation, 1994
  - Govil et. al. "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," 1st ACM International Conference on Mobile Computing and Networking, 1995
  - Neufeld, et. al."Policies for Dynamic Clock Scheduling," Fourth Symposium on Operating Systems Design and Implementation, 2000.
- And there are no good answers ..yet

Software

- Done with hardware, now on to software:

Software
Hardware
Power source

- General consensus is hardware savings are easier to achieve, but software savings could be orders of magnitude greater…

Software

- Main areas of power savings in software
  - Compilers
  - Operating system
  - Application
- Less previous work on power savings in software than in hardware

Software

- Compilers: Tiwari et.al. "Instruction level ower analysis and optimization of software," Journal of VLSI Signal Processing, 1996.
  - Measured power of long loops of an instruction, added power info to compiler optimization
  - Main finding is that fast software is more energy efficient
    - Choice of instructions is more important than order
  - Average power while active was about the same or greater, but execution time was decreased.
    - Overall energy decrease

Software

- Compilation: Simunic et.al. "Energy –Efficient Design of Battery-Powered Embedded Systems," ISLPED'99
- Energy vs. compiler and source code opt'ns for SA-1100 SmartBadge, using cycle-accurate CPU simulator
- Compiler optimizations mostly < 1%
- Source code optimizations could be significant:
  - Unsigned int division vs. signed: 15% more energy efficent
  - Table lookup vs. switch statement: 50% more efficient
  - Integer variable vs. char: 18% more efficient
  - Function call overhead was significant—passing args in registers vs. passing them on stack: 70-90% more efficient
- For large example, employing all optimizations saved 32%
  - Time savings was 35%…

Software

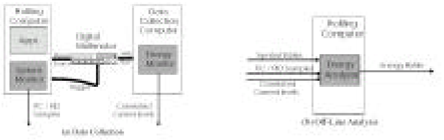
- Operating systems:
  - – Two main specifications:
    - Advanced Power Management (APM)
    - Advanced Configuration and Power Interface (ACPI)
  - – APM is being phased out, ACPI is its replacement
  - – ACPI provides for both power management and power-performance reduction
    - Hardware/BIOS provides tables and low-level functions for device control
    - OS provides means to read those tables, functions; uses them in drivers and kernel
    - Global states, Device power states, Performance states…

Software

- OS and Application software: Lorch and Smith, "Software Strategies for Portable Computer Energy Management," IEEE Personal Communications, June 1998.
- Categorizes software control issues:
  - – Transition: When should a component switch modes?
  - – Load-change: How can a component's functionality needs be modified so it can be put in low power modes more often?
  - – Adaptation: How can software permit novel, power-saving uses of components?
- Other papers by Lorch study power problems of MacOS
  - – Threads active while not doing any work…

Software

- Flinn et. al. "Powerscope: A tool for profiling the energy usage of mobile applications," IEEE Workshop on Mobile Computing Systems and Applications, 1999



- Allows per process and per function energy measurement

Wrap-up

- Battery life: Is it really what you want to measure?
  - – Better measure: Work per battery life
- Suppose 50% decrease in CPU frequency gives:
  - 50% decrease in performance,
  - 20% decrease in power:
- Then work per battery life = 0.5/0.8 = 0.63 of original…

Wrap-up

- Find out where power is going
- Things to keep in mind while measuring power:
  - – Sampling rate of meter
  - – Latency of trigger
    - I.e. you can't measure a 100us function call if trigger latency is 1ms…

Wrapping up

- Stupid notebook tricks--Windows login and my laptop:
  - – Windows, logged in, screen off: 11.5 W
  - – Windows, logged in, screen bright: 17.0 W
  - – Windows, logged in, screen dim: 14.1 W
  - – Windows, *at login prompt*, screen dim: 27.4 W
  - – Linux, screen dim: 15.1 W

Wrap-up

- Ruminations on the future:
  - Computing will consume negligible power
  - Communication will consume some power
  - Human interface will be the big hog

- Closing thought:

“Energy is eternal delight.”

–*William Blake, the Marriage of Heaven and Hell.*