B#: A Battery Emulator and Power-Profiling Instrument

Chulsung Park, Jinfeng Liu, and Pai H. Chou
University of California, Irvine

B# (B sharp) is a programmable power supply that emulates battery behavior. It measures current load, calls a battery simulation program to compute voltage in real time, and controls a linear regulator to mimic a battery's voltage output. The instrument enables validation of battery-aware power optimization techniques with accurate, controllable, reproducible results.

As consumers demand ever more functionality and smaller form factors in portable devices such as PDAs and celluar phones, batteries are quickly becoming a limiting factor. Recently, researchers have started developing battery-aware power management techniques that exploit the nonideal features of batteries to maximize their effective lifetimes. Specifically, varying the discharge patterns can dramatically affect battery life. To validate these techniques, researchers can use either real batteries or battery simulation.

The most obvious way to validate the results is to measure the power with actual batteries, but this approach has several drawbacks. If nonrechargeable batteries are used, they must be replaced and disposed of after each experiment, making this approach expensive and unfriendly to the environment.

Rechargeable batteries produce less waste, but the results might not be reproducible. Real batteries experience effects that cause the charge capacity to vary. In the rate-capacity effect, for example, a higher current draw than the rated current makes a battery appear less charged (with lower voltage and less total energy output) for a period of time.[1] In the rate-recovery effect or relaxation effect, the battery can recover from its lost efficiency if it has a chance to rest during periods of reduced load.[2]

In addition, temperature and aging can affect charge capacity. Lansburg, Cocciantelli, and Vigerstol reported that available charge capacity could vary from 73% to 103% of rated capacity after six or seven months of battery use, depending on discharge rate, temperature, and storage conditions.[3] Therefore, to avoid misleading results, researchers must consider all these factors when conducting experiments involving batteries.

To achieve full reproducibility, researchers have turned to simulation and have proposed and implemented several simulators. Dualfoil is an electrochemical model that solves partial differential equations in Fortran.[1] It outputs the battery's voltage and temperature in response to the power consumer's discharge current load. It models rate-capacity and rate-recovery effects, but it is computationally intensive and does not capture aging effects. Researchers have also proposed models based on Spice[4] and discrete-time VHDL,[5] and they are faster though less accurate. Researchers have developed stochastic models to capture these effects,[6] but they are designed to estimate battery lifetime rather than the voltage response.

A battery simulator must be driven by an actual load. Normally, the input to a battery simulator is a synthetic or actual current profile collected by running a system. Because there is no feedback to the system, such an approach is good for validating open-loop or battery-friendly power managers rather than battery-aware ones. An alternative approach is to close the feedback loop by simulating the power manager to drive the battery simulator. However, this requires online estimation of the rest of the power-managed system's power consumption. Detailed, accurate, fast power estimation remains a challenge.

We take a different approach: We propose a battery emulator called B# (pronounced B sharp) for experiments with battery-aware designs. B# is an intelligent power supply that mimics a battery's behavior by running a battery simulation program in real time. It senses the current load and responds by controlling the output voltage as an actual battery would. This lets researchers conduct in situ experiments on battery-aware designs without

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

having to use actual batteries. Our approach combines the speed and accuracy of measurement-based approaches with the flexibility and reproducibility of simulation-based approaches. The only other battery emulator we are aware of is the unpublished work on the Penn State University Battery Simulator.[7] Developed for testing electric vehicles, the PSU-BS can output up to 600 W with a response time of 170 ms. However, very limited information on this project is publicly available.

B# has potentially more uses than emulating batteries. In the emulator's profiling mode, the user can collect power profiles through real-time measurements and save the data to a file. B# complements cycle-accurate energy-monitoring tools for systemwide power consumption without a centralized clock.[8] In training mode, the user can connect an actual battery for calibrating a simulation model. B# can also play back a recorded stream of voltage values as they are collected from other sources, such as a solar panel over the course of a day under various weather conditions. We've prototyped our design and tested it on PDAs, achieving high accuracy and fast response time.

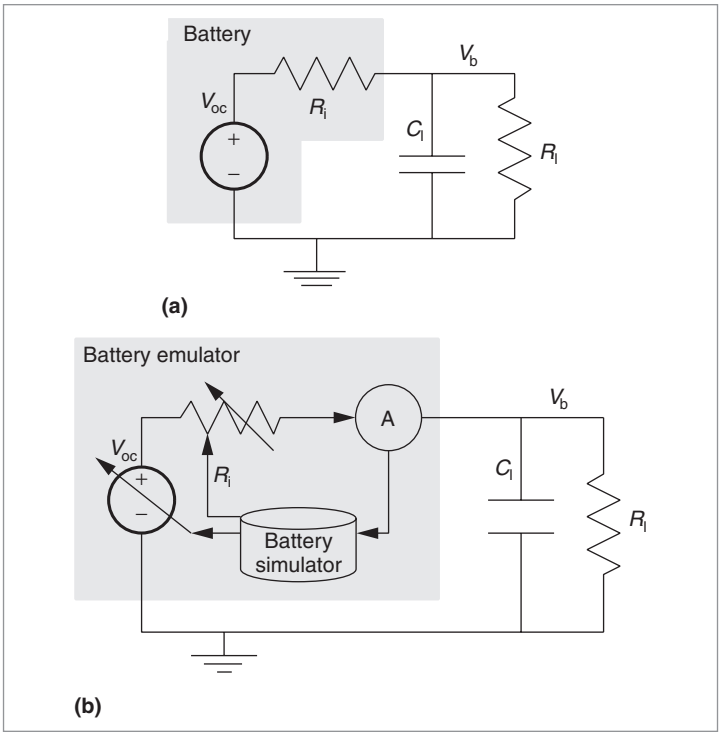


Figure 1. Circuit models of a battery with a load (a), and a battery emulator with a load (b).

Problem statement

Before designing the battery emulator, we reviewed circuit models for batteries and the performance target of the emulator. The key constraints on a battery emulator are its power circuitry and timing.

Power circuitry

Figure 1a shows a battery-powered system modeled as an equivalent circuit. The shaded region corresponds to the battery, in which V_{oc} and R_i are the open-circuit voltage and internal resistance. The unshaded region corresponds to the load with capacitance C_l and resistance R_l. The load can vary over time with activities and power management policies. If the circuit sources current I, the battery's observed voltage V_b is $V_{oc} - I \times R_i$. As the battery discharges, V_{oc} decreases while R_i increases, and both are based on the battery's state and internal temperature. The simulation model maintains the battery's state, whereas the ambient temperature and current can be measured. Figure 1b shows one possible circuit for a battery emulator based on the circuit model in Figure 1a. The emulator performs the following steps repeatedly:

- Measure the current (I) and temperature (T).
- Call the simulator to compute V_{oc} and R_i in response to I and T.
- Set the V_{oc} and R_i values.

Ambient temperature T, which we use to set the simulator's initial condition, can affect the battery's internal temperature. Most existing simulators, including Dualfoil, compute V_b directly without explicitly computing R_i or V_{oc}. An emulator controls V_{oc}, which is simply $V_b + I \times R_i$. We can implement R_i effectively by compensating the value of V_{oc} as a function of I, and it need not be a physical programmable resistor.

Timing

The battery emulator's timing constraints include the emulation period and the response time. Emulation period p is the time between successive updates of the output voltage by the emulator. Response time δ is the latency from the time the current is sampled to the time the output voltage is observed. Note that the sampling and simulation periods can be equal to or shorter than the emulation period. Also, it's not necessary that $\delta \leq p$. Both p and δ are limited by the following factors:

- load capacitance C_l,
- data acquisition time, which is limited by sampling speed and communication time to the computer; and
- computation and communication time of running one simulation iteration.

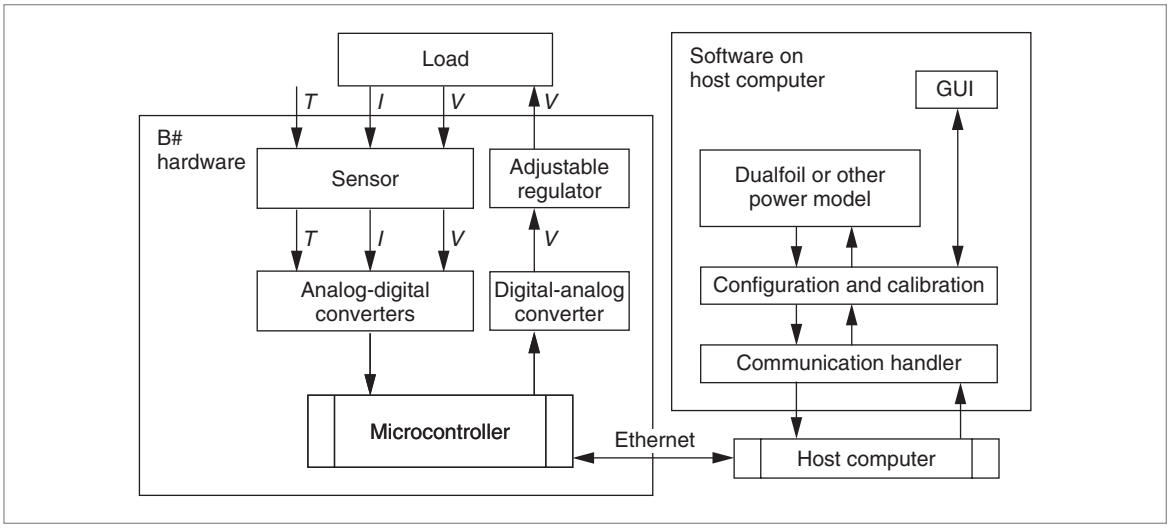


Figure 2. B# system block diagram: hardware board, connected via Ethernet to host computer running GUI and Dualfoil.

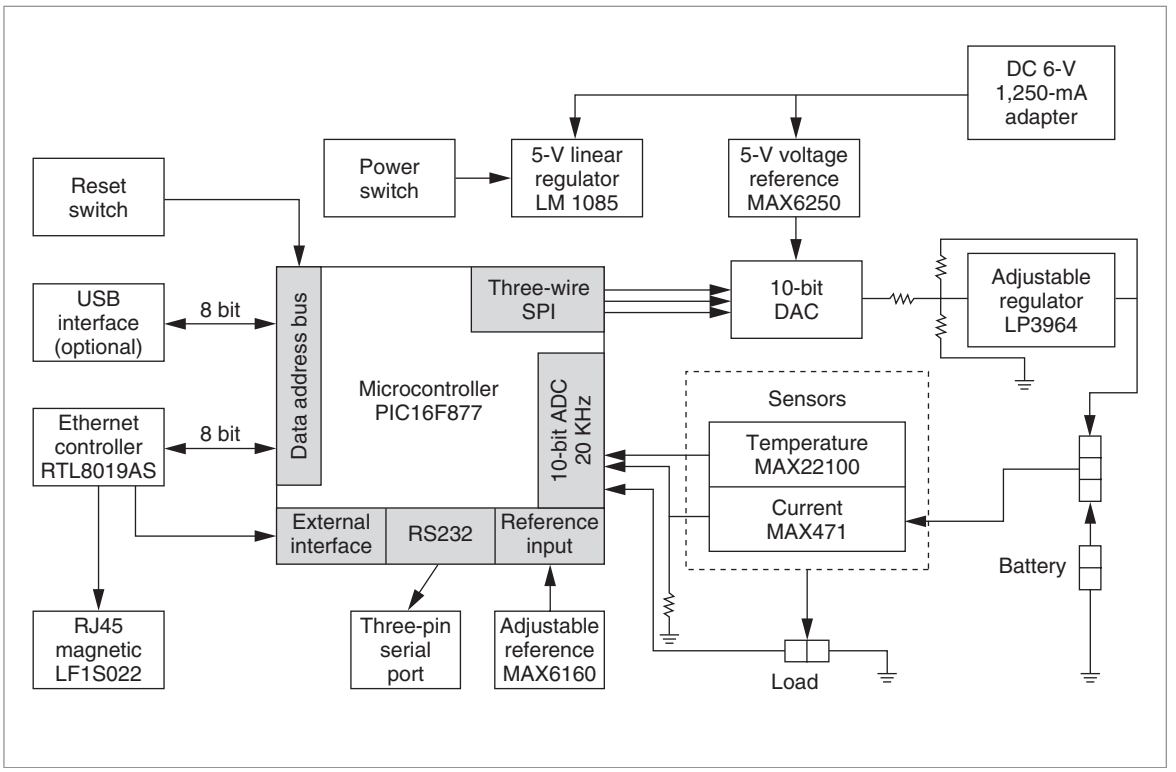


Figure 3. B# hardware block diagram.

Emulator design

Figure 2 diagrams the B# system. It consists of the B# hardware board, connected via Ethernet to a host computer running the GUI and Dualfoil. Here we describe the B# system's hardware, software, and communication protocol.

B# hardware

Figures 3 and 4 show a hardware block diagram and a photo of the B# board. We categorize the hardware into digital and analog subsystems.

Analog: power circuitry. The power circuitry imple-

ments the equivalent circuit shown in Figure 1b. It contains sensors to measure current I, temperature T, and voltage V_b, and it must adjust the output voltage in the range of interest.

We implement the I, T, and V_b sensors by sampling the voltage values of the three respective measurement circuits. We measure current by digitizing the voltage drop across a very small resistor (25 mΩ) in series with the power supply. We measure temperature by sampling an AD22100 temperature sensor IC. The PIC16F877 microcontroller already has eight built-in channels of 10-bit analog-digital converters (ADCs) with an acquisition time of 19.72 µs. We use three channels for measuring I, T, and V_b.

We implement output voltage control with a MAX5721 10-bit digital-analog converter (DAC) and an LP3964 adjustable linear regulator. The microcontroller uses a serial peripheral interface (SPI, also called a three-wire interface) to control the DAC, whose voltage then controls the adjustable linear regulator. The LP3964 has a maximum current rating of 800 mA, which we can increase by using another adjustable regulator with a higher current rating or by composing several regulators in parallel.

We don't include a passive resistor in model R_i, because the LP3964 already has its own internal resistance R_r, which is actually higher than the battery's R_i. Therefore, we implement R_i by voltage adjustment, which we discuss later.

Digital control. At the heart of the B# hardware is the microcontroller. The current B# board uses a PIC16F877 microcontroller at 20 MHz with built-in eight-channel ADCs and a universal asynchronous receiver-transmitter (UART) for serial communication. The firmware is stored in the on-chip flash memory and can be upgraded via the serial port. In addition, a RealTek RTL8019AS Ethernet controller is connected to the microcontroller via memory-mapped I/O. Ethernet is the primary high-speed link for communication with the host computer. The PIC runs a command interpreter that responds to status queries or performs system configuration and ADC calibration.

Host computer software

The host computer works with the B# hardware board by running several tasks under GUI control: B# configuration and calibration, control and data communication, real-time battery simulation, and real-time graphical display of voltage and current curves. We currently use Dualfoil for power source simulation, but we could replace it with another simulator or another power model such as a solar panel.

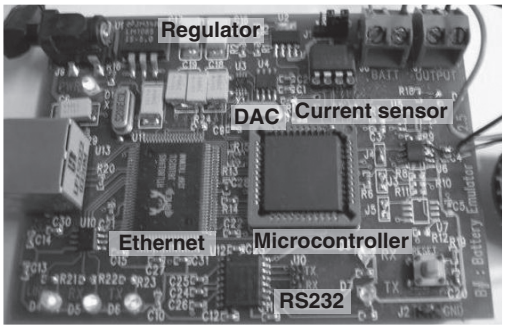


Figure 4. B# board.

Graphical user interface. The GUI supports the control and configuration of B# and Dualfoil in all stages of an experiment. First, the user can configure the simulated battery by specifying the Dualfoil parameters with additional compensation such as a DC offset. In profiling mode, the GUI saves the sampled voltage and current data stream from the B# board into a host computer file. The sampling rate can be set much higher in profiling mode than in emulation mode because B# need not respond to the current.

In both profiling and emulation modes, the GUI supports real-time on-screen display of voltage and current curves. The B# GUI can also serve as a graphical front end for running Dualfoil and displaying the power profile without connecting to B#. Then, all results from measurement, real-time emulation, and offline simulation can be superimposed on the same display.

Battery simulation and training. In the current implementation, we use Dualfoil,[1] an electrochemical simulator for lithium-ion battery cells, to model the emulated battery's behavior. Dualfoil is widely used because it is considered one of the most accurate simulators and because its Fortran source code is freely available (http://www.cchem.berkeley.edu/~jsngrp/fortran.html). The original Dualfoil program runs in batch mode through file I/O. To make it work with B#, we converted Dualfoil from Fortran to C and replaced file I/O calls with Ethernet communication calls to exchange data with the B# board in real time. Of course, other battery simulators can be plugged into the B# framework, as long as they implement the same protocol. The major

problem in adapting Dualfoil is determining a suitable simulation resolution and battery parameterization.

As an electrochemical model, Dualfoil is computationally intensive. Running Dualfoil simulations in real time can be challenging if the required temporal resolution is high. Fortunately, Dualfoil produces accurate results at about 6 to 10 iterations per second on realistic load profiles, and this is well within the performance of modern PCs. In practice, the simulation delay is 30 to 150 ms, depending on the host computer's speed and the load fluctuation. To handle execution time variations, we define emulation period p, which is the time budget for executing at least one Dualfoil iteration plus communication delay. While Dualfoil computes the voltage response in parallel, the B# board sets the voltage from the previous step and takes the next current sample. That is, the voltage response seen by the load circuit experiences a delay of one period, including an Ethernet delay of 1 to 2 ms. In the results section, we verify that this is within the response time of actual batteries and therefore has little impact on the overall accuracy of B#'s battery emulation.

As we said, a main challenge in adapting Dualfoil is the selection of battery configuration parameters. Each battery model is defined by 58 parameters.[9] Some parameters specify the physical dimensions of the cells, anode, cathode, and so forth and are easy to determine. However, most parameters are chemistry specific and not so obvious. The DLP305590 lithium polymer battery that we use is not on the list of predefined battery models in Dualfoil, so we must train our own battery model. Unfortunately, the available documentation is not sufficient for us to construct even an approximate model. Our approach is to determine the configuration parameters empirically by running Dualfoil on three load profiles:

- a constant high discharge current level at 400 mA,
- a constant low discharge current level at 100 mA, and
- a variable current between the high and low discharge levels.

It's not feasible to enumerate all 58 parameters. Based on manual exploration, we found the simulation results to be sensitive primarily to three discrete parameters, whereas the remaining 55 parameters require much manual tuning. The three integer parameters would yield $7 \times 12 \times 13 = 1{,}092$ combinations. Even if only two values are considered for each of the 55 parameters, the total number of required simulation runs would explode quickly because there would be $1{,}092 \times 2^{55}$ (that is, more than 10^{20}) combinations for each profile. Our approach is to perform a manual inspection based on several heuristics. On the absolute scale, we attempt to minimize the average-voltage error (AVE) over all sampling windows:

$$\frac{\sum_{t=0}^{N} \left| V_{\text{sim}}(t) - V_{\text{meas}}(t) \right|}{\sum_{t=0}^{N} V_{\text{meas}}(t)} \quad (1)$$

However, it is often difficult to minimize AVE over all three load profiles. We relax the matching criteria with three other heuristics:

- *AVE matching.* How closely do the AVEs of the three load profiles (Equation 1) match each other, even if individually the AVEs may be higher?
- *Proportional matching.* If $V_{\text{meas}}(t) = cV_{\text{sim}}(t)$ for some constant scaling factor c for most of the profile, it can also be a good match. As a heuristic, we attempt to minimize

$$\left| \frac{V_{\text{meas}}(\text{start})}{V_{\text{sim}}(\text{start})} - \frac{V_{\text{meas}}(\text{end})}{V_{\text{sim}}(\text{end})} \right|$$

- *Linear matching.* If $V_{\text{meas}}(t) = aV_{\text{sim}}(t) + b$ for constants a and b for most of the profile, it can also be a good candidate for calibration. As a heuristic, we attempt to minimize

$$\left| \frac{V_{\text{meas}}(\text{start})}{aV_{\text{sim}}(\text{start}) + b} - \frac{V_{\text{meas}}(\text{end})}{aV_{\text{sim}}(\text{end}) + b} \right|$$

To find a good match, we ran more than 10,000 simulations with different parameter settings and compared the simulation results with the measured battery responses according to these criteria. Of course, we could use several other objective functions, such as peak voltage error, but we found that minimizing AVE is effective for battery training. The best AVE we achieved over the three power profiles are ≤ 2%, ≤ 3%, and ≤ 1%. We later show that the set of battery configuration parameters derived this way works well for emulating the iPaq battery pack on other types of realistic load.

Host-board protocol. During battery emulation, the B# board and the host computer exchange measured

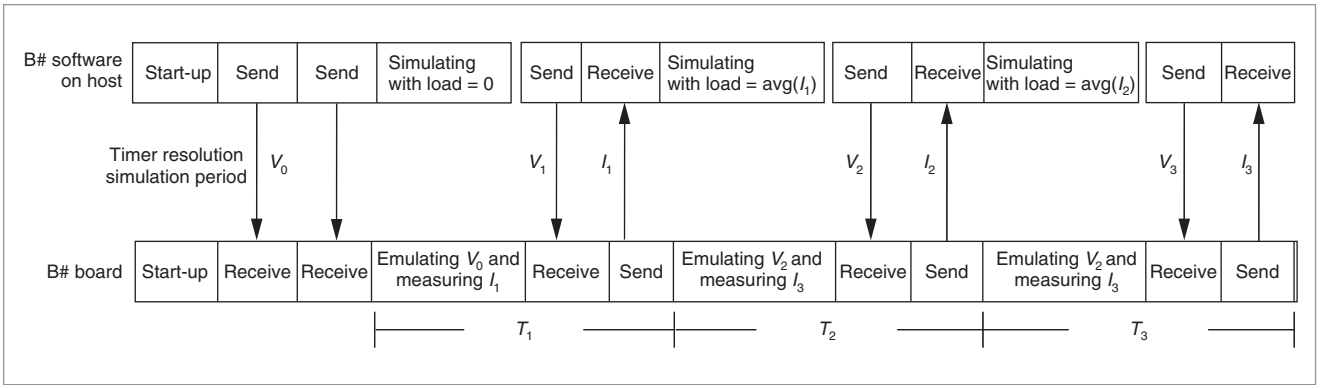


Figure 5. Scheduling of B# hardware, software, and communication.

current load and simulated voltage response once every emulation period. We define a protocol between the host and the B# board not only to exchange data, but also to ensure that both sides can perform their tasks in addition to communicating under real-time constraints. Recall that the B# board must perform current measurement and voltage setting, whereas the host must run Dualfoil simulation and interpretation of current input and voltage output.

Figure 5 shows the scheduled activities on the B# board and the host computer and their communication over time. On start-up, the host computer sends commands to the B# board to set the board's timer resolution and the emulation period, which must be multiples of the timer resolution. The host sets the B# board to an initial open-circuit voltage and starts Dualfoil simulation with no current load. Then, the B# board starts measuring the circuit's current load on each timer interrupt. After finishing one Dualfoil simulation iteration, the host computer sends the recently simulated voltage values to the B# board and receives the measured current values from the B# board. At this time, the first emulation period ends and the next emulation period starts.

The host computer runs another Dualfoil simulation iteration with the averaged load current, while the B# board continues sampling current and emulating voltage on each timer tick. If the simulator outputs a different voltage response from the previous emulation period, the host computer linearly interpolates this voltage change over time by sending a series of (timer tick, voltage) pairs to the B# board. The B# board then sets the voltage accordingly on those timer interrupts. The user can configure the emulation period, which can be shortened to 30 ms on a fast host computer. In our experience, the energy error caused by the delay in voltage response is negligible.

Evaluation

We evaluated the current implementation of B# through measurements with handheld devices and real batteries. Here we provide an overview of the experimental setup, explain our methodology, and present the results for the emulator's response time, internal resistance, and overall accuracy.

Experimental setup

Our experimental setup consisted of a pair of B# boards and the load. One B# board works as the battery emulator and the other as the power profiler that records the voltage and current to a file. We calibrated both boards with a digital multimeter before the experiments. We used this setup to test the following handheld systems as the load:

- iPaq 3650, with a 206-MHz StrongARM SA-1110 CPU, 32-Mbyte RAM, 16-Mbyte ROM, and a 320 × 240, 12-bit thin-film transistor (TFT) display;
- Palm Tungsten C, with a 400-MHz XScale PXA255 CPU, 64-Mbyte RAM, a built-in 802.11b wireless interface, and a 320 × 320, 16-bit TFT backlit display.

Figure 6 shows the experimental setup for evaluating B# with the iPaq, and the setup for the Tungsten is similar. The reference battery is the DLP305590 lithium polymer battery from Danionics. Its physical dimensions are 3.1 mm (T) × 55 mm (W) × 90 mm (L), and it weighs 31 ± 2 g. Its output is rated for 1,000 mA, 3.0 to 4.2 V. This is a battery pack for the iPaq, but we used it to power the Tungsten, too.

Test cases

The iPaq or the Tungsten acts as the load to B#. To quantify emulation accuracy, we developed several test

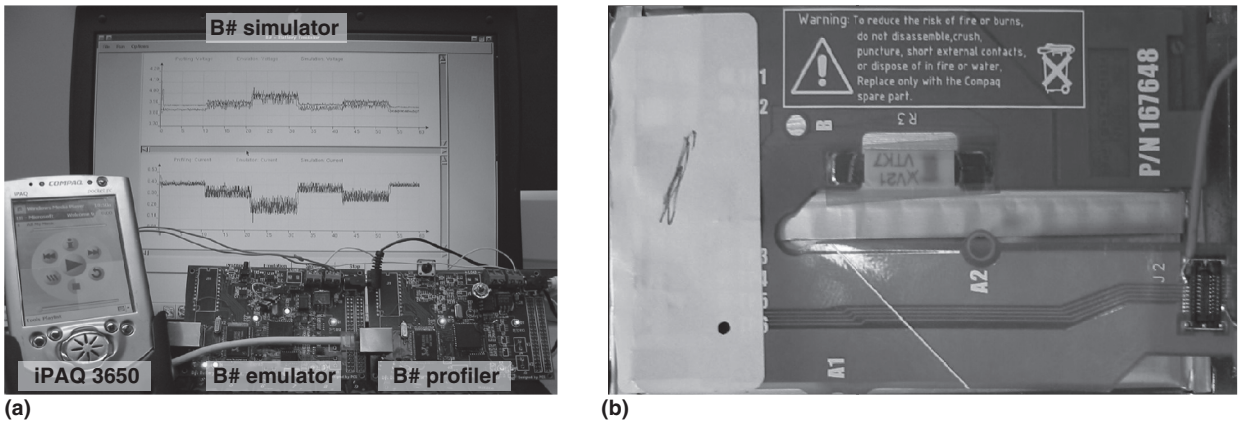


Figure 6. Experimental setup (a): one B# board to emulate the battery for the iPaq, and one B# board as a power profiler to measure voltage and current. On the right is the reference battery (b).

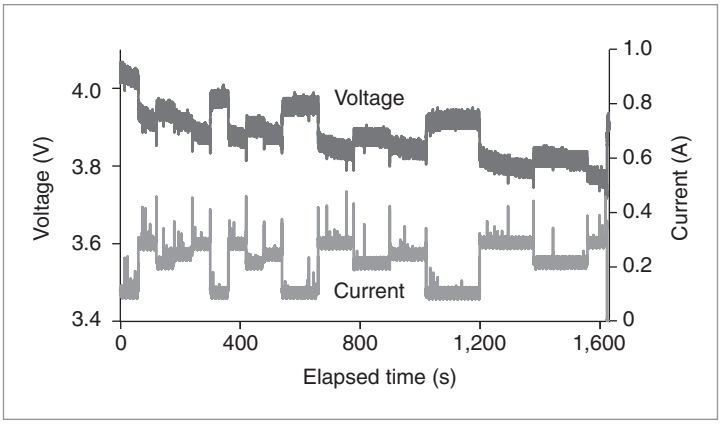


Figure 7. Profile of 27 minutes of voltage and current data collected from an iPaq running on its own battery. On battery depletion, the voltage goes back up, but the current goes to 0.

cases. In addition to constant-high and constant-low current used during calibration, we also developed scripts to vary the load on the iPaq or Tungsten. Using scripts ensures reproducibility of the load profile.

Test case: backlight and multimedia. For the iPaq, we developed scripts to control combinations of backlight settings and video/audio activities. These settings and activities consume high power and thus enable B# to exercise the widest dynamic range. We ran these experiments once on the iPaq's own battery and once with B# in place of the battery. Figure 7 shows one such power profile generated with the actual battery. We let the iPaq run until it reached its cutoff voltage around 3.7 V and shut itself off. The iPaq's cutoff voltage is higher than the lithium-ion battery's cutoff voltage to prevent deep discharge, which can permanently damage the battery.

Test case: wireless communication. Wireless communication is also a major power consumer in many electronic systems. For the iPaq 3650 to use a wireless interface such as an 802.11b card, we would have to use the expansion pack, which a separate battery pack powers. This would require multiple B# systems emulating multiple battery packs in parallel. Although there is no inherent difficulty in carrying out such an experiment, we were more interested in testing B# with a wider variety of system-level load profiles. In this case, therefore, we used the Tungsten C, which has an integrated 802.11b interface and can be powered entirely by one battery pack and one B#. The Tungsten also uses a 1500-mAh lithium-ion polymer battery, which is similar in type and capacity to the iPaq's. To exercise the wireless card, we scripted the Tungsten's Web browser via the HTTP refresh directive to periodically "pull" and render a series of Web pages containing six image files totaling 7 Mbytes in size.

Internal resistance

This experiment validated the way we model the battery's internal resistance by voltage compensation. Recall that the battery and the B# linear regulator have internal resistances R_i and R_r, respectively. R_i and R_r form a voltage divider with load R_l. However, R_i and R_r differ in several ways. Real lithium-ion batteries have an R_i of about 0.2 Ω on a full charge, rising to about 0.7 Ω

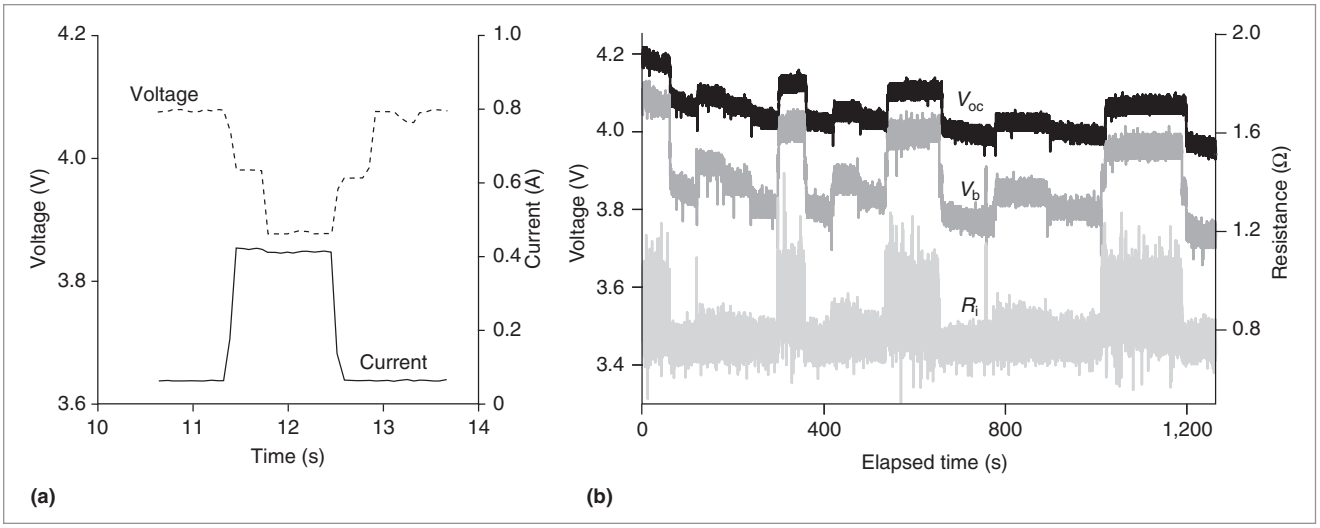


Figure 8. Measurement of battery's internal resistance when nearly fully charged—about 0.253 Ω (a). Internal resistance R_r of the B# linear regulator, computed from [V_{oc}'(set) – V_b'(measured)] / I for 21 minutes (b).

near depletion.[10] We validated this by measurement, as Figure 8a shows:

$$R_i = (V_1 - V_0)/(I_1 - I_0) = 0.253\ \Omega$$

On the other hand, the linear regulator has a variable internal resistance of $0.7\ \Omega \le R_r \le 1.3\ \Omega$, as Figure 8b shows.

We can conclude that adding a programmable resistor to model R_i would not be a feasible approach, for several reasons. First, programmable resistors do not come in such small value increments. Second, because the linear regulator's internal resistance is already higher than the battery's, adding a passive resistor in series will not reduce the resistance. As a result, we must implement R_i by adding a $\Delta V(I)$ voltage to V_{oc}.

Response time

To determine the lithium-ion battery's response time, we applied the pulse-shaped current load, as Figure 9 shows, causing a voltage difference in V_b. The current starts to increase at 11 s and drops just before 12.5 s. We determine response time and latency by averaging the respective rise and fall times:

$$T_{response} = (T_r + T_f)/2 = 0.423\ \text{s}$$
$$T_{latency} = (T_{PH} + T_{PL})/2 = 0.248\ \text{s}$$

where the subscript PH stands for propagation delay, high; and PL stands for propagation delay, low. The sampling period and response time are therefore well within the iPaq's DLP305590 battery's response time.

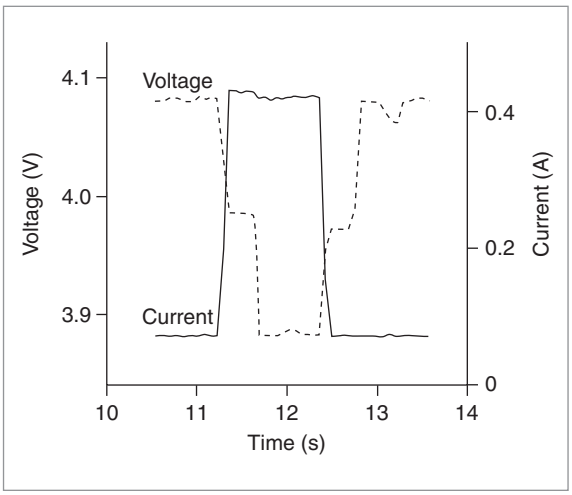


Figure 9. Measured response time of lithium-ion battery to changes in current load.

Emulation accuracy

We collected the power profiles on the actual battery and repeated the same workload with B#. Table 1 summarizes the accuracy results for energy and voltage. Two sources of inaccuracy are the Dualfoil simulator and the B# board. The calibrated Dualfoil is within 0.08% of overall energy accuracy. For all practical purposes, Dualfoil is an exact match, even though the simulation resolution is only about 8 to 10 Hz.

Given the almost perfect software model, we calibrated the R_r of the B# with a digital multimeter within 0.009 V, and this was as close as we could get with the DAC's limited resolution. The cutoff voltages differ by

Table 1. Energy and voltage accuracy results. (Note that V_{start} is not open-circuit voltage.)

	Backlight and multimedia			802.11b		
Test method	**Energy (J)**	**V_{start}**	**V_{end}**	**Energy (J)**	**V_{start}**	**V_{end}**
Actual battery	743.6	3.885	3.801	822.53	3.928	3.837
Simulation (Dualfoil)	743.0	3.854	3.785	823.747	3.927	3.833
Difference	–0.6	–0.031	–0.016	1.217	–0.001	–0.004
Error (%)	–0.08	–0.798	–0.421	0.148	–0.102	–0.104
Emulation (B#)	753.2	3.894	3.858	822.798	3.932	3.832
Difference	9.6	0.009	0.057	0.268	0.004	0.005
Error (%)	1.294	0.232	1.500	0.033	0.102	0.111

only 0.057 V, which is also practically exact. In terms of the total energy output, B# is accurate within 1.294%.

Our ongoing work deals with hardware enhancements and new models for other power sources, including both battery and nonbattery sources. We are extending the B# board to emulate recharging. The Dualfoil simulator can model recharge, but the B# hardware requires enhancements to draw power from the charger. We are also evaluating models for other types of batteries such as NiCd and Lithium MnO_2 to replace Dualfoil with our own battery model. We are validating the results of using B# to emulate solar panels. The main difference between batteries and solar panels is that solar panels have a much wider dynamic range of internal resistance and output power. This has enabled researchers to experiment with solar-aware power management techniques without having to wait for weather conditions and geographical locations to be exactly right. All these enhancements will make B# an indispensable instrument for experimenting with power-aware designs with far more controllability. ■

Acknowledgments

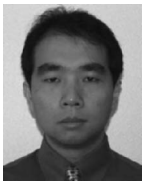
This work was sponsored in part by the National Science Foundation under grant CCR-0205712, the DARPA PAC/C program under subcontract 4500942474 with Rockwell/Collins, and the Printronix Fellowship. We thank Jae Park and Kien Pham for their assistance, and Bruce Tromberg for making laboratory space available for the first prototyping of the B# instrument.

■ References

1. M. Doyle, T.F. Fuller, and J.S. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell," *J. Electrochemical Soc.*, vol. 140, no. 6, June 1993, pp. 1526-1533.
2. T.F. Fuller, M. Doyle, and J.S. Newman, "Relaxation Phenomena in Lithium-Ion Insertion Cells," *J. Electrochemical Soc.*, vol. 141, no. 4, Apr. 1994, pp. 982-990.
3. S. Lansburg, J.-M. Cocciantelli, and O. Vigerstol, "Performance of Ni-Cd Batteries after Five Years of Deployments in Telecom Networks Worldwide," *Proc. 24th Ann. Int'l Telecommunications Energy Conf.* (INTELEC 02), IEEE Press, 2002, pp. 251-258.
4. S. Gold, "A PSPICE Macromodel for Lithium-Ion Batteries," *Proc. 12th Ann. Battery Conf. Applications and Advances*, IEEE Press, 1997, pp. 215-222.
5. L. Benini et al., "A Discrete-Time Battery Model for High-Level Power Estimation," *Proc. Design, Automation and Test in Europe* (DATE 00), IEEE CS Press, 2000, pp. 35-39.
6. D. Panigrahi et al., "Battery Life Estimation for Mobile Embedded Systems," *Proc. 14th Int'l Conf. VLSI Design*, IEEE CS Press, 2001, pp. 55-63.
7. M.M. Mench, Penn State University Battery Simulator (PSU-BS), http://mtrl1.me.psu.edu/mtrl/BatSimFac.htm, 2000.
8. D. Shin et al., "Energy-Monitoring Tool for Low-Power Embedded Programs," *IEEE Design & Test*, vol. 19, no. 4, July-Aug. 2002, pp. 7-17.
9. K.E. Thomas, M. Doyle, and J. Newman, "Introduction to Dualfoil.f," http://www.cchem.berkeley.edu/~jsngrp/dualfoilfaq.pdf, Oct. 2002.
10. D. Linden and T. Reddy, *Handbook of Batteries*, McGraw-Hill, 2001.

Chulsung Park is pursuing a PhD in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. His research interests include wireless sensor net-

works and low-power embedded-system designs. Park has a BS in electrical engineering from Seoul National University. He is a student member of the IEEE.



Jinfeng Liu is a PhD candidate in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. His research interests include distributed embedded systems and low-power designs. Liu has a BS in electronic engineering from Tsinghua University, China, and an MS in computer science from the Chinese Academy of Sciences. He is a student member of the IEEE and the ACM.

Pai H. Chou is an assistant professor at the University of California, Irvine. His research interests include hardware-software codesign of embedded systems, low-power designs, wireless embedded sensing systems, and medical devices. Chou has a BA from the University of California, Berkeley, and an MS and a PhD in computer science and engineering, both from the University of Washington. He is a member of the IEEE and the ACM, and a recipient of the NSF Career award.

■ Direct questions and comments about this article to Chulsung Park, Center for Embedded Computer Systems, University of California, Irvine, CA 92697-2625; chulsung@uci.edu.