Transactions Briefs

Nonideal Battery and Main Memory Effects on CPU Speed-Setting for Low Power

Thomas L. Martin and Daniel P. Siewiorek

***Abstract*—This paper explores the system-level power-performance tradeoffs of dynamically varying CPU speed. Previous work in CPU speed-setting considered only the power of the CPU and only CPUs that vary supply voltage with frequency. This work takes a broader approach, considering total system power, battery capacity, and main memory bandwidth. The results, which are up to a factor of four less than ideal, show that all three must be considered when setting the CPU speed.**

***Index Terms*—Battery modeling, low power design, low power dissipation, system level, tradeoffs, variable voltage.**

I. Introduction

TECHNICAL developments in recent years have enabled designers to build hand-held and wearable computers. In many of these computers, the CPU consumes a substantial fraction of the total power, making the CPU a prime target for power savings. One of the major ways to trade power for performance is to vary the CPU speed when the CPU's performance is greater than required, called CPU speed-setting. This paper reports the battery discharge measurements of an application on a real system as the CPU frequency is varied. It provides experimental evidence that the basic assumptions of the previous work in CPU speed-setting and variable-voltage operation of VLSI circuits do not hold at the system level. The paper examines the power, performance, and battery life of the Itsy, a StrongARM SA-1100-based hand-held computer from Compaq's Western Research Lab [1], while running an MPEG video player over a range of CPU frequencies. The experimental results are up to nearly a factor of four less than what would be expected given ideal assumptions about the system's battery capacity and performance.

The remainder of this paper is organized as follows. Section II describes the related work. Section III describes the three major factors in CPU speed-setting from a system perspective. Section IV details the experimental method. Section V presents the results of the experiments, showing the impact of the factors described in Section III. Finally, Section VI presents the conclusions and describes future work in the area.

II. Related Work

The two areas of research most closely related to this paper are power-performance metrics that include nonideal battery behavior and operating system policies for CPU speed-setting.

Manuscript received May 12, 2000; revised September 1, 2000. This work was supported by NSF Contract EIA-9901321, DARPA under Contract N660019928918-N306A, and an NSF Graduate Fellowship.

T. L. Martin is with the Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: tlmartin@ece.uah.edu).

D. P. Siewiorek is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Publisher Item Identifier S 1063-8210(01)00809-5.

A. Battery-Power-Performance Metrics

To the best of our knowledge, we were the first to combine models of electronic system power consumption, system performance, and nonideal battery behavior for exploring power-performance tradeoffs [2]. There we introduced the idea that metrics for low power operation of battery-powered electronics should include the energy source and the energy consumers and showed that the number of *computations completed per discharge* adequately reflected both the battery's energy capacity as well as the energy per operation of the electronics. We then used Peukert's empirical model [3] of battery capacity to derive the *work ratio*, the normalized computations completed per discharge, for the case of changing the CPU speed in a fixed-voltage system.

In the area of mobile computer networking, Zorzi and Rao modeled the energy consumption of wireless protocols and noted that the loss of battery capacity would have an impact on them [4].

Pedram *et al.* [5] and Pedram and Wu [6], [7] combined analytical models of battery behavior under continuous loads with models of a VLSI circuit operating from a single V_{dd} to show that the optimal value of V_{dd} depends on the nonideal battery behavior as well as the energy per operation. They defined a metric called the *Battery-Discharge delay* (*BD-delay*) product that is the inverse of the computations per discharge above. In [6] they used a phenomenological SPICE model to show that battery life depends on both the average and distribution of the power consumption when loads are intermittent.

In [8], we studied similar intermittent behavior using a one-dimensional finite difference model for rechargeable Li-ion batteries based upon first principles [9]. We examined the effect of peak power on battery capacity from the point of view of the operating system, showing that peak power more closely predicts the capacity than average power, and concluding that it is more effective to reduce power when a system is active than when it is idle. Once the peak power is below a certain level that depends on the battery chemistry and design, however, the battery can be treated as ideal, effectively delivering its full capacity, and average power is adequate for predicting battery life. A similar point was made by Doyle *et al.* [9], but from the point of view of the battery designer.

B. CPU Speed-Setting Policies

Weiser *et al.* first pointed out that it is feasible to build systems that can use dynamic speed-setting, varying CPU voltage and frequency on-the-fly, to reduce energy consumption [10]. The speed would be set by the operating system to reduce the energy consumption of the CPU while still meeting the performance requirements of the user. Weiser *et al.* and Govil *et al.* [11] used trace-driven simulation to study several dynamic speed-setting policies that slowed the CPU when it was mostly idle and sped it up when it was mostly active. Pering and Brodersen applied the concept of real-time scheduling to dynamic speed-setting [12].

All of the previous work in CPU speed-setting makes the following assumptions.

- Performance is proportional to clock frequency f.
- Power is proportional to fCV^2, where C is capacitance and V is voltage.

Given these two assumptions, the previous work shows that if V is held constant while f is changed, then the energy per operation is constant for all frequencies. Therefore, the previous work concluded

that reducing the CPU frequency will not save energy if the voltage is held constant while the frequency is changed. For speed-setting to be useful, both the CPU voltage and frequency must be decreased [10]. (The voltage determines the maximum frequency, so it is not possible to reduce the voltage without also reducing the frequency.) If both are reduced by a factor of s, then the power of the CPU will change by a factor of s^3, the time per operation will change by a factor of $1/s$, and consequently the energy per operation will change by a factor of s^2. Thus running as slowly as possible will minimize the energy per operation. Kuroda *et al.* implemented a CPU based upon varying both voltage and frequency [13], and several companies have recently announced or introduced CPUs that vary both [14], [15].

The policies have two other features in common: The only variable used to determine the CPU speed is the idle time, and the lower bound on the range of speeds is the CPUs minimum operating frequency.

This paper will demonstrate that a practical policy must consider not only idle time, but three system-level effects—total system power, nonideal battery capacity, and nonideal memory behavior—when trying to maximize computations per discharge.

The next section details each of these three factors.

III. System Factors in Speed Setting

While the focus of the low power design community has been on reducing the energy per operation of electronic devices, users of battery-powered systems desire to complete as much work as possible before the batteries are fully discharged. In this paper, we measure "work" by counting the computations completed per battery discharge, or more simply, the *computations per discharge* [2]. After a mobile computer meets the performance expectations of a user, the next major consideration should be to maximize the number of computations per discharge.

The computation per discharge is essentially the battery capacity divided by the energy per operation for a given computation. From a system perspective, there are three major factors in setting the CPU speed to increase the computations per discharge: 1) battery capacity as a function of system power; 2) system power as a function of the CPU frequency; and 3) application performance as a function of the CPU frequency. The relationship between these factors and the number of computations per discharge is given by

$$\text{Computations per Discharge} = \frac{\text{BatteryCapacity}(\text{SystemPower}(f))}{\text{SystemPower}(f)} \times \text{Performance}(f)$$

where f is the CPU clock frequency, the battery capacity is given in Watt-hours, the power in Watts, and the performance in iterations per hour.

A. Nonideal Battery Capacity

The first factor, battery capacity, is typically assumed to be constant, in which case maximizing computations per discharge is the same as minimizing the energy per operation. In practice, however, battery capacity decreases as the load power increases, typically by as much as 20–40% over a useful range of load power [3]. When battery capacity is not constant, the operating point that minimizes the energy per operation may not maximize the computations per discharge, because less total energy may be available from the battery at that operating point.

B. Total System Power

The second factor in setting the CPU speed is the power of the system as a function of frequency. This factor, together with the performance as a function of frequency, determines the CPU frequency that results in the minimum energy per operation. The previous work considered only the power of a single VLSI circuit, the CPU, operating at one value of V_{dd}. In practice, however, systems are made up of several subsystems, often with multiple V_{dd}s and operating frequencies. The total system power determines the battery life, not just the power of the CPU. If a system were to be constructed using a variable-voltage CPU, with CPU power proportional to s^3 as described in Section II, one would expect that there would be other terms in the function of system power versus frequency due to subsystems that operate at a fixed voltage, changes in efficiency of the power supply over the range of loads, and other reasons. The system power versus frequency could then be fitted to a polynomial of the form $a_3 s^3 + a_1 s + a_0$, where s is the normalized frequency, $a_3 s^3$ represents the power of the subsystems that can vary both voltage and frequency, $a_1 s$ represents the subsystems that operate at a fixed voltage but can vary frequency, and a_0 represents the subsystems that operate at fixed voltage and fixed frequency.

For example, consider the power versus frequency of our experimental platform, shown in Fig. 1. Even if it were to have a variable-voltage CPU, the total system power would be nearly linear with respect to frequency. If the SA-1100 were a variable-voltage CPU, only its 1.5-V supply would be able to vary. The 3.3-V supply powers the CPUs I/O pins and must remain fixed. Fig 1 shows that if only the 1.5-V CPU voltage were allowed to vary, then the power of the system would be dominated by the 3.3-V supply, which supplies the CPU I/O pins and other subsystems (e.g., memory, display, and interface ports). The system power would be more linear than cubic. Consequently, even if system performance and battery capacity were ideal, the assumption that power is proportional to s^3 would be invalid, and the goal of "running as slowly as possible" this assumption leads to would be invalid as well.

It can be shown that the lower order terms mean that the slowest CPU frequency may not result in the minimum energy per operation. (A full derivation is available in [16, chap. 5].) If there are lower order terms, there will be a point where decreasing the CPU frequency will increase the energy per operation, contrary to the assumptions described in the previous work.

C. Nonideal Performance Speedup

The final factor in setting the CPU speed is the performance as a function of frequency. The typical assumption is that performance is proportional to frequency [2], [10], [11]. This ignores the memory subsystem. For example, in applications with a high cache miss ratio, performance can be limited by memory bandwidth rather than CPU frequency. Once an application becomes limited by memory bandwidth, increasing the CPU frequency will have little effect on the application performance and will increase the energy per operation. This nonideal speedup could be avoided by designing the memory subsystem such that the speed of the memory is matched to the CPUs maximum frequency. However, using faster memories may increase system cost unacceptably, especially in price-sensitive systems such as PDAs and notebook computers. In addition, it may be desirable to implement a CPU speed-setting scheme on an existing hardware platform that cannot be redesigned. Finally, acknowledging that the memory subsystem can have a large effect on CPU speed-setting will show system designers that it is an area of prime concern.

In the following sections, we study each of these three factors using a system based on a fixed-voltage CPU.

IV. Method

The experimental results reported in this paper were gathered using the Itsy, a Linux-based hand-held computer from Compaq's Western Research Laboratory [1], based on the StrongARM SA-1100 [17]. We added two calls to the Linux kernel for changing the CPU frequency

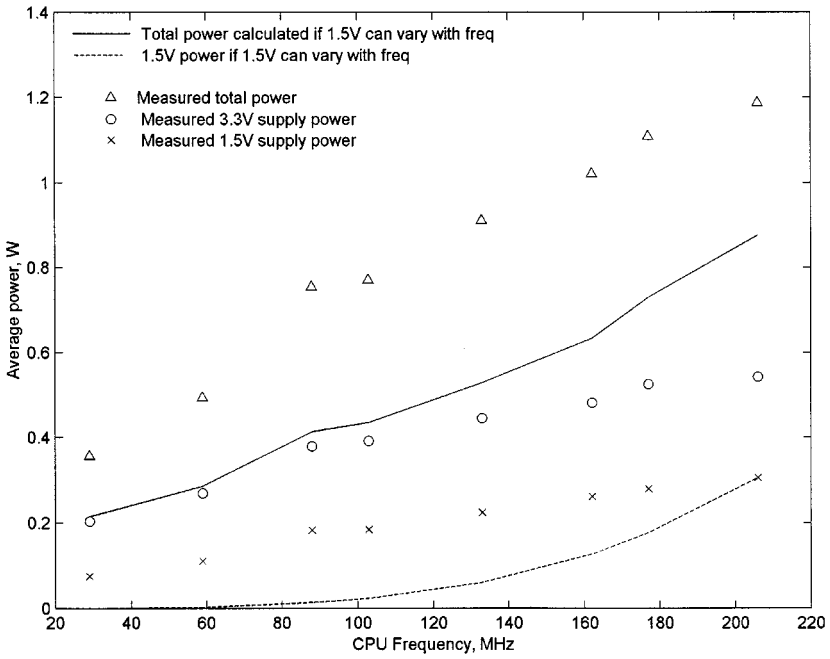

Fig. 1. Measured system power and estimated system power if 1.5 V CPU supply could scale with frequency.

based upon the clocking scheme for the StrongARM, which is shown in Fig. 2. The combination of the two calls allows the CPU core frequency to vary from 29.5 to 206 MHz. Full details of the calls are in [16].

The code chosen for the experiments shown here was the Itsy's MPEG video player. The justification for this choice is twofold. First, we expect a video player to be one of the typical applications for a mobile computer. Second, we wanted code that adequately exercised the memory hierarchy and operating system [18], [19]. Measuring the energy of an application that fits into the cache or that uses no operating system resources would not be representative of a mobile computer capable of running a variety of applications.

In the battery life measurements, the MPEG player ran in a continuous loop without any idle time. While this does not reflect the behavior of mobile systems in actual use, it allows us to estimate the computations per discharge in actual use because of the peak-power limited behavior of the batteries as described above and in [8].

V. Results

This section describes the measurements of performance versus frequency, power versus frequency, and computations per discharge of the Itsy running the MPEG player.

Fig. 3 shows the normalized performance versus frequency for the MPEG loop. The measured performance is nearly ideal until above 100 MHz, where there is a breakpoint. From 100 MHz to the maximum frequency of 206 MHz, the measured performance diverges from ideal. Fig. 4 shows the normalized performance versus the CPU frequency for the two functions with the most execution time in the MPEG player. The routine *mpeg_j_rev_dct* has the expected performance speed-up over nearly the entire range of CPU frequencies. But the other routine, *GrayDitherImage*, shows little increase in performance after 133 MHz, where the main memory bandwidth becomes limited by the speed of the memory chips. So, as the CPU frequency increases above 133 MHz, accesses to main memory take more CPU cycles. Because the performance of *GrayDitherImage* tracks the memory bandwidth rather than the CPU frequency, it limits the performance speedup of the program overall.

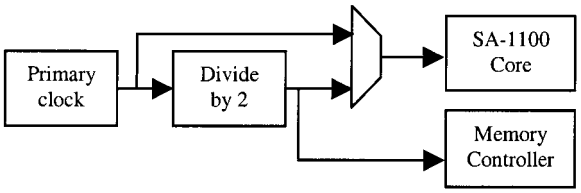

Fig. 2. Clocking scheme for the StrongARM SA-1100.

Using the measured rather than ideal performance, the computations per discharge differs from the ideal computations per discharge by 40% at the CPUs maximum frequency. This prediction was confirmed with actual battery discharges while running the MPEG player, using Sanyo UF-510 and UF-310 Li-ion cells [20], shown in Fig. 5. These cells are nearly ideal over the range of the power of the Itsy. Consequently, the measured computations per discharge agree with the expected curve quite well, although the measured points begin to be less than expected at the higher frequencies due to loss of battery capacity.

The results in Fig. 6, collected using the AAA alkaline cells that the Itsy was initially designed to use, diverge from the expected behavior even more dramatically, as the power of the system lies in the nonideal range of the battery's capacity. Because of the loss of battery capacity, the computations per discharge decreases at higher CPU frequencies even though the energy used per iteration remains nearly constant.

Fig. 6 shows that assuming ideal behavior for performance versus frequency and battery capacity versus power can be mistaken. The measured computations per discharge at the maximum CPU frequency are nearly a factor of four less than the computations per discharge predicted by ideal behavior for both factors. Even using the measured performance versus frequency, the computations per discharge decreased as the clock frequency increased, rather than increasing as predicted by assuming an ideal battery capacity. Contrary to the conclusions of the previous work, even a fixed-voltage CPU can increase its computations per discharge by running at less than its maximum frequency.

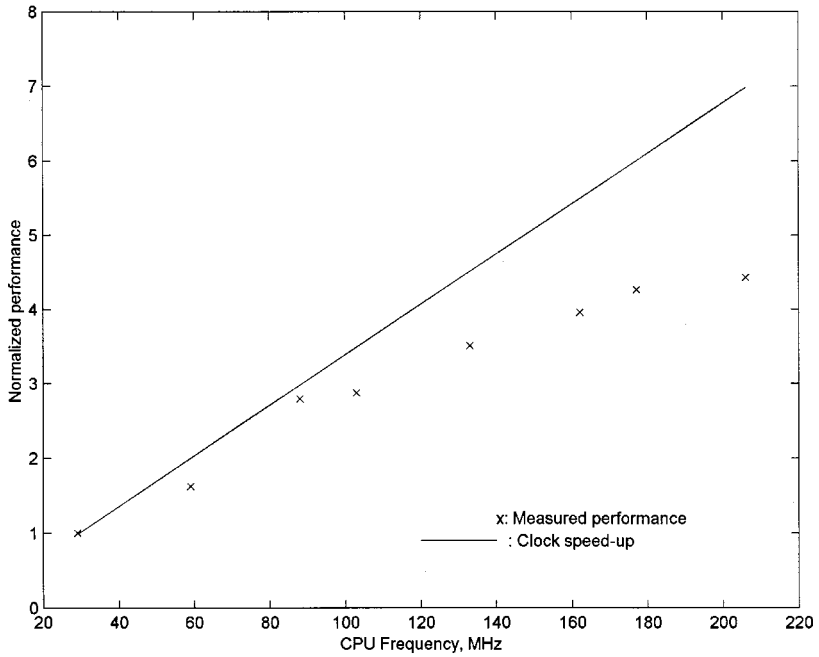

Fig. 3. Performance versus frequency of MPEG loop.

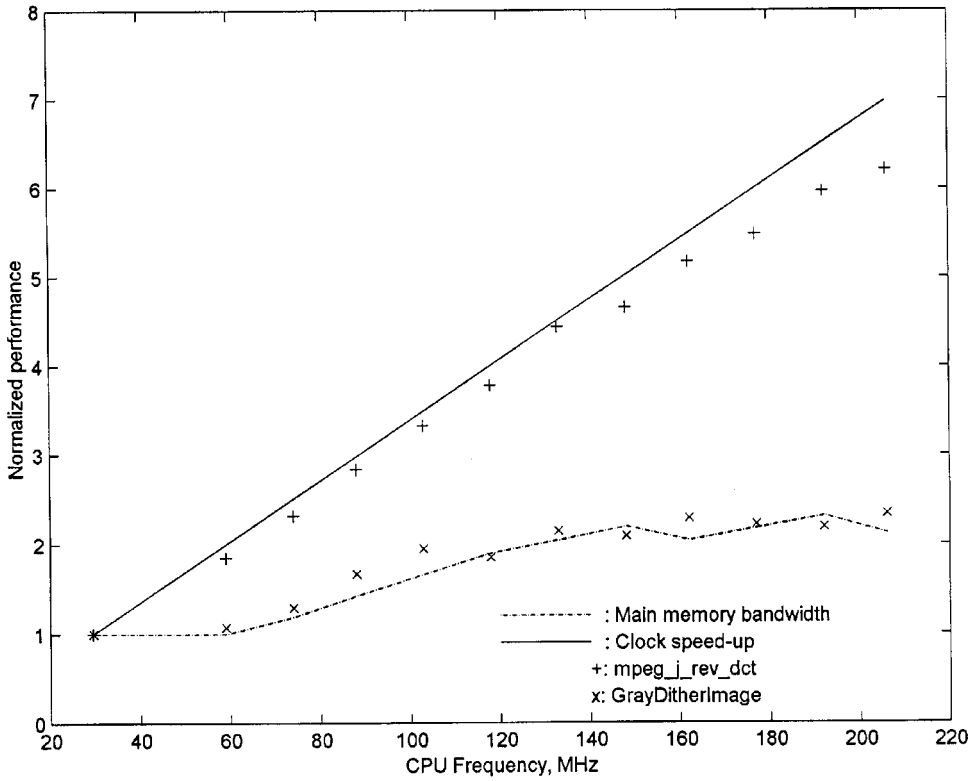

Fig. 4. Performance versus frequency for two of the MPEG player functions, with expected speed-up and measured main memory bandwidth.

VI. Conclusion

A realistic CPU speed-setting policy for a general-purpose system must consider system power, battery capacity and memory bandwidth. Using battery-discharge experiments with a real system, this paper has shown that there are regions of operation where ideal assumptions about any of these factors will lead to a decrease in the number of computations performed per discharge cycle. Further experimental results and analytical treatments of the problem are available in [16]. The results shed light on the necessary ingredients for a successful speed-setting policy.

The desirable features of a system that allows the OS to set the speed include hints from applications, cache performance registers in the CPU like those found in high-end systems and battery information

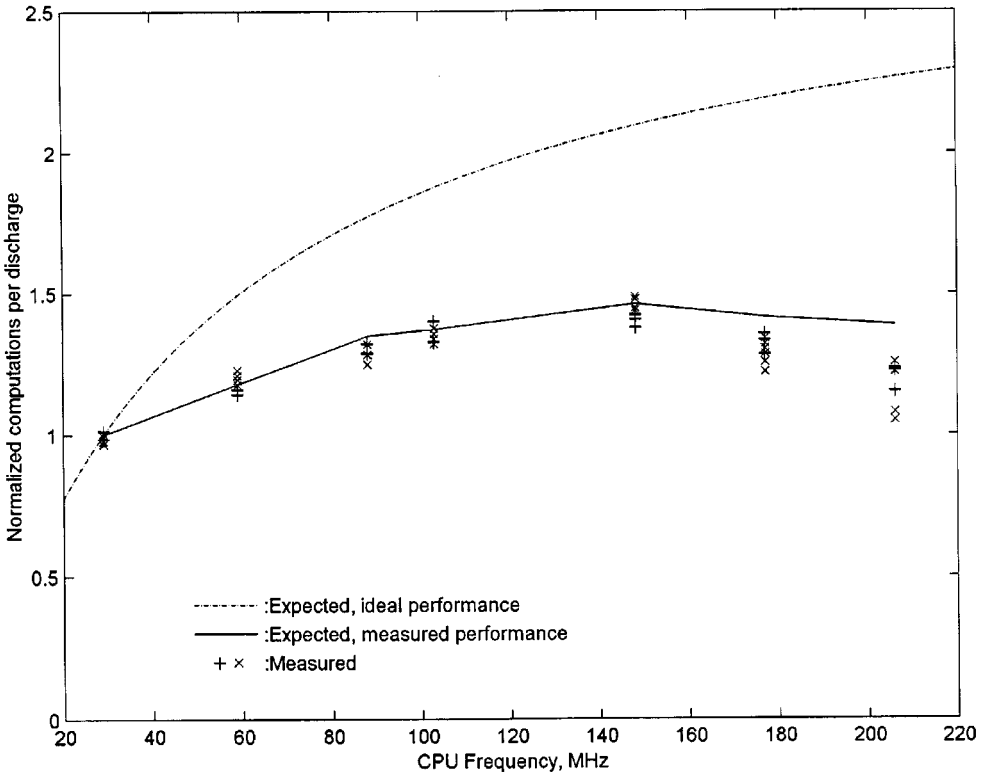

Fig. 5. Expected and measured results with Li-ion batteries, Sanyo UF-510 and UF-310.

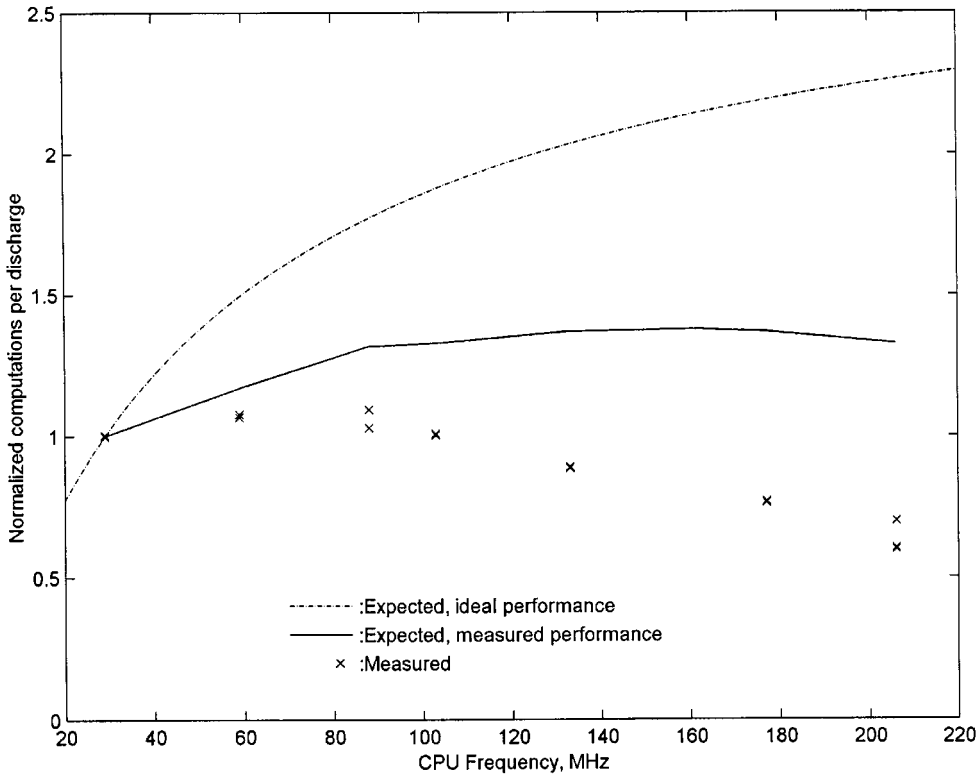

Fig. 6. Expected and measured results with AAA batteries.

using battery "gas gauge" integrated circuits [21]. Whereas ideally a policy would have the CPUs minimum operating speed as its lower bound, in practice the lower bound should be the speed that maximizes the computations per discharge. We described a method for determining the lower bound and how it should be used by a policy in [22].

Although we have focused on only the MPEG player application, the characteristic upon which the results depend—performance speedup limited by main memory bandwidth—is likely to be shared by other applications expected to be typical in a mobile environment. Cache interference of operating system and application code could lead to more main memory bandwidth limitations than would be found in the applications alone [18], [19]. Similar performance behavior has been observed in other applications running on the Itsy [23]. As part of our future work, we intend to study other typical applications. However, the results show that there exist real applications with nonideal performance behavior that impacts the choice of CPU speed.

In addition, we have assumed that only the memory hierarchy is nonideal. Other subsystems, such as networking or secondary storage, could have a similar effect. However, these other subsystems can be treated in the manner we have described here.

In conclusion, CPU speed-setting for mobile computers is a system-level problem. The results described in this paper show that, to properly balance performance and power consumption, the entire computer system must be taken into consideration.

ACKNOWLEDGMENT

The authors would like to thank Compaq's Western Research Laboratory for its generous donation of Itsies without which the results with the Itsy would not have been possible. They would also like to thank M. Doyle for giving them access to his Li-ion cell model.

REFERENCES

[1] M. Viredaz, "The Itsy pocket computer version 1.5 user's manual," Compaq Western Research Lab. Tech. Note TN-54, July 1998.
[2] T. Martin and D. Siewiorek, "A power metric for mobile systems," in *Proc. 1996 Int. Symp. Low-Power Electronics Design*, Aug. 1996, pp. 37–42.
[3] D. Linden, *Handbook of Batteries*, 2nd ed. New York: McGraw-Hill, 1995.
[4] M. Zorzi and R. Rao, "Error control and energy consumption in communications for nomadic computing," *IEEE Trans. Comput.*, vol. 46, pp. 279–289, Mar. 1997.
[5] M. Pedram, C.-Y. Tsui, and Q. Wu, "An integrated battery-hardware model for portable electronics," in *Proc. Asia South Pacific Design Automation Conf.*, Feb. 1999, pp. 109–112.
[6] M. Pedram and Q. Wu, "Design considerations for battery-powered electronics," in *Proc. 36th Design Automation Conf.*, June 1999, pp. 861–866.
[7] ——, "Battery-powered digital CMOS design," in *Proc. Design Automation Test Europe*, Mar. 1999, pp. 72–76.
[8] T. Martin and D. Siewiorek, "Non-ideal battery properties and low power operation in wearable computing," in *Proc. 2nd Int. Symp. Wearable Computers*, San Francisco, CA, Oct. 1999, pp. 101–106.
[9] M. Doyle, J. Newman, A. Gozdz, C. Schmutz, and J. Tarascon, "Comparison of modeling predictions with experimental data from plastic lithium ion cells," *J. Electrochemical Soc.*, vol. 143, no. 6, pp. 1891–1903, June 1996.
[10] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. 1st USENIX Symp. Operating Systems Design Implementation*, Nov. 1994, pp. 13–23.
[11] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low power CPU," in *Proc. 1st ACM Int. Conf. Mobile Computing Networking*, 1995, pp. 13–25.
[12] T. Pering and R. Brodersen, "Energy efficient voltage scheduling for real-time operating systems," in *Proc. 4th IEEE Real-Time Technol. Applicat. Symp., Works-In-Progress Session*, June 1998.
[13] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, "Variable supply-voltage scheme for low-power high-speed CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 33, pp. 454–462, Mar. 1998.
[14] Transmeta Corp. (2000, Jan.) LongRun Technol.. [Online]. Available: http://www.transmeta.com/crusoe/lowpower/longrun.html
[15] Intel Corporation, Mobile Pentium III Processor and 440BX AGPset Performance Brief, Apr. 2000.
[16] T. Martin, "Balancing batteries, power and performance: System issues in CPU speed-setting for mobile computing," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1999.
[17] Digital Equipment Corporation, Digital Semiconductor SA-1100 Microprocessor: Technical Reference Manual, revision EC-R5MTC-TE, Mar. 1998.
[18] A. Agarwal, *Analysis of Cache Performance for Operating Systems and Multiprogramming*. Boston, MA: Kluwer, 1989.
[19] R. Uhlig, D. Nagle, T. Mudge, S. Sechrest, and J. Emer, "Instruction fetching: Coping with code bloat," in *Proc. 22nd Int. Symp. Computer Architecture*, July 1995, pp. 345–356.
[20] Sanyo Corp., UF812248 and UF612248 Data Sheets, Apr. 1998.
[21] Benchmarq Microelectronics, Inc., *1998 Data Book*.
[22] T. L. Martin, D. P. Siewiorek, and J. M. Warren, "A CPU speed-setting policy that accounts for nonideal memory and battery properties," in *Proc. 39th Power Sources Conf.*, Maple Hill, NJ, June 12–15, 2000, pp. 502–505.
[23] J. Warren, "Interaction between architecture performance and power consumption in mobile systems," Master's dissertation, Carnegie Inst. Technol., Carnegie Mellon Univ., Pittsburgh, PA, May 2000.

Reducing Power Consumption of Turbo-Code Decoder Using Adaptive Iteration with Variable Supply Voltage

Oliver Yuk-Hang Leung, Chi-Ying Tsui, and Roger Shu-Kwan Cheng

***Abstract*—Turbo-code becomes popular for the next generation wireless communication systems because of its remarkable coding performance. One of the problems for decoding turbo-code in the receiver is the complexity and the high power consumption since multiple iterations of Soft Output Viterbi Algorithm (SOVA) or Maximum *a posteriori* (MAP) decoding have to be carried out to decode a data frame. To reduce the complexity of the turbo-code decoder, adaptive iteration based on cyclic redundancy checking (CRC) and output convergence approaches has been proposed to reduce the average number of iterations required for decoding a data frame. This results in a system that has variable workload since the amount of computation required for decoding each data frame is different. In this work, we propose a dynamic voltage scaling approach to further reduce the power consumption. Different from other variable workload systems, the workload here is not known at the time when the data is being decoded. Thus, optimum voltage assignment is not feasible. We propose several heuristic algorithms to assign supply voltage for different decoding iterations. Simulation results show that significant reduction of power consumption is achieved comparing with the system using fixed supply voltage.**

***Index Terms*—Communication, digital, low-power design, system-level, VLSI.**

I. INTRODUCTION

In recent years, a novel class of parallel-concatenated recursive systematic convolutional (RSC) codes known as turbo-code was introduced [1], [2]. It has drawn a lot of attention because of its remarkable performance. Turbo-code consists of multiple RSC component codes

Manuscript received May 12, 2000; revised September 8, 2000. This work was supported in part by Hong Kong RGC CERG under Grant HKUST6217/98E.

O. Y.-H. Leung is with Perception Digital, Clear Water Bay, Hong Kong (e-mail: oliver@perceptiondigital.com).

C.-Y. Tsui and R. S.-K. Cheng are with the Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: eetsui@ee.ust.hk).

Publisher Item Identifier S 1063-8210(01)00805-8.