User-perceived Latency Driven Voltage Scaling for Interactive Applications [*]

Le Yan
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
lyan@princeton.edu

Lin Zhong
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
lzhong@princeton.edu

Niraj K. Jha
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
jha@princeton.edu

ABSTRACT

Power has become a critical concern for battery-driven computing systems, on which many applications that are run are interactive. System-level voltage scaling techniques, such as dynamic voltage scaling (DVS) and adaptive body biasing (ABB), have been shown to reduce energy consumption effectively. Previous works on DVS and ABB exploit low CPU utilization of the processor to drive voltage scaling. This has become inadequate for modern interactive applications involving high CPU usage. In this work, we target computer responsiveness during voltage scaling to exploit more opportunities for energy reduction. Instead of CPU utilization, we use the user-perceived latency, the delay between user input and computer response, to drive voltage scaling. Considering the tradeoff between energy consumption and computer responsiveness during voltage scaling not only reduces energy consumption effectively, but also ensures good computer responsiveness for interactive applications. Experimental results show that for the 70nm technology, during the execution of seven commonly-used interactive applications, the energy consumption of the processor using user-perceived latency driven DVS is reduced by an average of 37.3%, and the user-perceived latency by an average of 18.3%, compared to CPU utilization driven DVS. If both DVS and ABB are performed simultaneously based on the user-perceived latency, then the energy consumption is reduced by another 38.9% compared to when DVS is performed alone, while maintaining a similar computer responsiveness level. We have implemented user-perceived latency driven voltage scaling under Linux with X Window system. However, the methodology is extensible to other operating systems as well.

Categories and Subject Descriptors: D.4.1 [Operating System]: Process Management - scheduling.
General Terms: Algorithms.
Keywords: Adaptive body biasing, computer responsiveness, dynamic voltage scaling, power consumption.

1. INTRODUCTION

Power has become a limiting factor for battery-driven computing systems. Dynamic and leakage power are two main sources of power consumption. DVS, supported by various processors, *e.g.* Intel XScale [1], exploits the quadratic dependence of dynamic power consumption on the supply voltage [2,3]. It dynamically adjusts the clock frequency along with the supply voltage of the processor to reduce power consumption. Leakage power consumption is due to the subthreshold leakage current and junction leakage current. ABB is a technique that adjusts the threshold voltage by varying the body bias voltage at run-time to reduce leakage power consumption [4,5]. Since leakage power is becoming increasingly important as technology size shrinks, optimizing dynamic and leakage power consumption simultaneously has been demonstrated to be important [6–8]. Combined DVS and ABB techniques have been proposed to consider the tradeoff between supply voltage and body bias voltage to manage both dynamic and leakage power consumption for a multiply-accumulate unit [6], a single voltage-scalable processor [7], and distributed embedded systems [8]. Many applications executed on battery-driven systems are interactive, involving human-computer interaction. Previous works have used CPU utilization to drive voltage scaling. This was effective in the past when interactive applications were primarily textual and involved low CPU usage. Modern interactive applications, ranging from video games to sophisticated simulation and virtual reality environments, are compute-intensive. Voltage scaling driven by CPU utilization has become inadequate.

Computer responsiveness is an important consideration for interactive applications. User attention has become the most precious computing resource [9]. An immediate visual causality between user input and computer response is expected. The delay between the "cause," *e.g.*, pressing the mouse button, and the "effect," *e.g.*, a window popping up, is called the user-perceived latency. Human beings generally do not feel any delay if the user-perceived latency is below the human-perceptual threshold (a value of 50-100ms is commonly used [10,11]). In this work, we take computer responsiveness into account during voltage scaling for interactive applications to reduce energy consumption. By keeping track of user-perceived latencies for past user inputs, we predict the user-perceived latency for the upcoming user input and use it to drive voltage scaling. We employ two system-level voltage scaling techniques, DVS alone and combined DVS and ABB. Trading off computer responsiveness for energy consumption enables voltage scaling to reduce energy consumption more aggressively, while still ensuring good computer responsiveness for interactive applications.

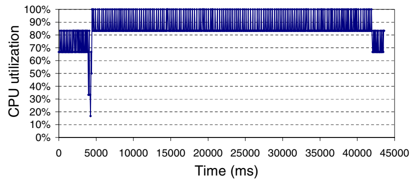
The rest of the paper is organized as follows. In Section 2, we use a motivational example to illustrate the importance of considering computer responsiveness during voltage scaling. In Section 3, we first detail how to track and predict user-perceived latency. We then discuss the tradeoff between energy consumption and computer responsiveness. We finally present the voltage scaling algorithm that is driven by user-perceived latency. We present experimental results in Section 4 and conclude in Section 5.

2. MOTIVATION

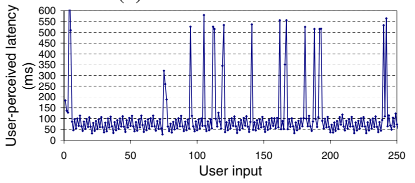
This section presents a motivational example to illustrate the importance of considering computer responsiveness during voltage scaling.

Example 1: Consider the racing game, *TuxRacer*, running on an IBM Thinkpad laptop having two performance levels with different frequency/supply voltage settings: *high* (1.8GHz/1.3V) and *low* (1.2GHz/1.2V). The CPU utilization driven DVS is interval-based (50ms in this example). If the current CPU utilization is less than or equal to 66.7%, the processor scales down its frequency/supply voltage and runs at the *low* performance level. Otherwise, it runs at the *high* performance level. Since *TuxRacer* involves high CPU us-

[*]Acknowledgments: This work was supported by NSF under contract no. CCF-0428446.



(a) CPU utilization



(b) User-perceived latency

Figure 1: Traces for the execution of *TuxRacer*

age (above 80%) after the racing starts (from 5s onwards) as shown in Figure 1(a), there is no opportunity for scaling down the frequency/supply voltage for energy reduction. The user-perceived latency driven DVS policy decreases the frequency/supply voltage of the processor if the current user-perceived latency is below the human-perceptual threshold (100ms in this example). As seen from the user-perceived latency trace shown in Figure 1(b), the user-perceived latency driven DVS policy has many opportunities for voltage scaling. For 58.1% of the total execution time for *TuxRacer*, the processor runs at the *low* performance level under user-perceived latency driven DVS, compared to only 14.2% under CPU utilization driven DVS. This yields an energy reduction of 15.3% with respect to CPU utilization driven DVS. Obviously, user-perceived latency driven DVS provides more opportunities for energy reduction than the CPU utilization driven one. ■

3. USER-PERCEIVED LATENCY DRIVEN VOLTAGE SCALING

In this section, we first describe how to track and predict the user-perceived latency for user inputs. We then discuss the trade-off between energy consumption and computer responsiveness. Finally, we detail the user-perceived latency driven voltage scaling algorithm for interactive applications.

3.1 Tracking and prediction of user-perceived latency

The time it takes a computer to execute code to respond to a user input is called the user-perceived latency. It is an indication of the level of computer responsiveness. A user input is typically handled under Linux/X Window system using the following steps.

1. When the user presses a key or button, an interrupt is generated.
2. The X server reacts to the interrupt by generating an X event and sending it to the corresponding X client.
3. The X client processes the X event and sends a graphics request to the X server.
4. The X server generates and sends updated display data to the monitor, and then sends a reply to the X client.

In our work, the measurement of the user-perceived latency consists of steps (2)-(4). Based on our experience, the contribution of step (1) is negligible compared to that of steps (2)-(4). When a keyboard/mouse event is generated, the X server records its birth time, relative to the X server start time. We modified the X server slightly so that the time is recorded relative to the start of the day instead. Most X clients call *XNextEvent*() directly or indirectly for fetching the new X events sent by the X server. If there is no new event, *XNextEvent*() blocks, waiting for the next user input. Its restart marks the end of processing for the previous event. We added bookkeeping code to record the difference between an X event's birth time and the time when the X client calls *XNextEvent*() again. The difference is the user-perceived latency used in this work.

Given the history of the user-perceived latency for past user inputs, two prediction mechanisms are employed to predict the value of the user-perceived latency for the upcoming user input in this work: *PAST* and *AVG(w)*. *PAST* predicts that the user-perceived latency of the next user input will be the same as that of the previous one, $UPL_i = M_{i-1}$, where UPL_i and M_i denote the predicted and measured values of the user-perceived latency for the i^{th} user input, respectively. For the first user input, we assume $UPL_1 = M_1$. *AVG(w)* predicts that the user-perceived latency of the next user input will be similar to those of the previous ones. It calculates an exponentially moving average of past user-perceived latencies given by Equation (1):

$$UPL_i = \frac{wUPL_{i-1} + M_{i-1}}{w+1} \tag{1}$$

where w is a decay factor. The value of w impacts the number of voltage transitions. Smaller values of w imply the supply and body bias voltages may be adjusted very frequently. The transition time overhead increases the user-perceived latency. The transition energy overhead increases the total energy consumption. Larger values of w imply a reduced number of transitions. However, the processor may continuously run at the higher (lower) performance level before the supply and body bias voltages are changed to either reduce energy consumption or improve computer responsiveness. We will discuss this later in Section 4.

3.2 Energy consumption vs. computer responsiveness

Figure 2 shows the normalized energy consumption with respect to the normalized user-perceived latency for *TuxRacer*. The normalized values are relative to the processor running at 14GHz. The parameters for the energy consumption model are adapted from [7] for a 70nm technology. As the user-perceived latency increases, the energy consumption of the processor decreases. This means that sacrificing computer responsiveness can provide energy savings. Therefore, to reduce energy consumption, while maintaining good computer responsiveness for interactive applications, we require the user-perceived latency to be less than the human-perceptual threshold. This ensures the user will not perceive the delay incurred by scaling down the performance level. If the predicted user-perceived latency is less than the human-perceptual threshold, which means that computer responsiveness is acceptable, the processor can drop its performance level to reduce the energy consumption until the predicted user-perceived latency becomes larger than or equal to the human-perceptual threshold. On the other hand, if the predicted user-perceived latency is more than the human-perceptual threshold, which means that computer responsiveness is unacceptable, the processor can boost its performance level to improve computer responsiveness, until the user-perceived latency drops to a value lower than the human-perceptual threshold.

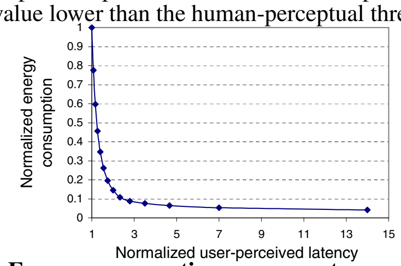


Figure 2: Energy consumption vs. computer responsiveness

3.3 Voltage scaling algorithm

Algorithm 1 gives the pseudo-code for the user-perceived latency driven voltage scaling algorithm. We assume the supply voltage and body bias voltage can be scaled in parallel in a continuous fashion. We use $[K_{min}, K_{max}]$, instead of a single value, to represent the allowed range of the human-perceptual threshold K, where K_{min} and K_{max} are the minimum and maximum allowed value of K, respectively. This avoids oscillation during voltage scaling.

For each user input, the value of user-perceived latency for the upcoming user input is predicted using either *PAST* or *AVG(w)*, as discussed in Section 3.1. If the predicted user-perceived latency UPL falls in the range $[K_{min}, K_{max}]$, the frequency f, supply voltage V_{dd}, and body bias voltage V_{bs} remain unchanged in order to maintain the current computer responsiveness level. Otherwise, if UPL is less than K_{min}, there exist opportunities for voltage scaling to reduce energy consumption. The following steps are repeated until UPL increases beyond K_{max}. We first save the current values of f, V_{dd}, and V_{bs}. Then f is decreased by step Δf and the corresponding V_{dd} and V_{bs} are updated using DVS alone or combined

DVS and ABB [8]. The predicted value of *UPL* is also updated based on the frequency scaling ratio. If f, V_{dd} or V_{bs} drops below its allowed minimum level, or the new *UPL* is above K_{max}, the latest voltage scaling changes get invalidated and f, V_{dd}, and V_{bs} are switched back to the old values. If *UPL* is more than K_{max}, which means the user may perceive time-lags, we boost the performance level to improve computer responsiveness. The following steps are repeated until *UPL* is no longer more than K_{max}. We first save the current values of f, V_{dd}, and V_{bs}. Then f is increased by Δf and the corresponding V_{dd}, V_{bs}, and *UPL* are updated. If f, V_{dd} or V_{bs} becomes larger than its allowed maximum level, the latest voltage scaling changes get invalidated and f, V_{dd}, and V_{bs} are switched back to the old values. Finally, the algorithm returns V_{dd} and V_{bs}.

Algorithm 1 Voltage scaling algorithm driven by user-perceived latency

```
 1: Predict $UPL$ based on the history of user-perceived latency;
 2: if $K_{min} \le UPL \le K_{max}$ then
 3:     return $V_{dd}$ and $V_{bs}$;
 4: else
 5:     if $UPL < K_{min}$ then
 6:         while $UPL < K_{max}$ do
 7:             save current $f$, $V_{dd}$, and $V_{bs}$;
 8:             $f = f - \Delta f$;
 9:             update $V_{dd}$, $V_{bs}$, and $UPL$;
10:             if $f < f_{min} \,||\, V_{dd} < V_{ddmin} \,||\, V_{bs} < V_{bsmin} \,||\, UPL > K_{max}$ then
11:                 restore $f$, $V_{dd}$, and $V_{bs}$;
12:                 break;
13:             end if
14:         end while
15:     else
16:         while $UPL > K_{max}$ do
17:             save current $f$, $V_{dd}$, and $V_{bs}$;
18:             $f = f + \Delta f$;
19:             update $V_{dd}$, $V_{bs}$, and $UPL$;
20:             if $f > f_{max} \,||\, V_{dd} > V_{ddmax} \,||\, V_{bs} > V_{bsmax}$ then
21:                 restore $f$, $V_{dd}$, and $V_{bs}$;
22:                 break;
23:             end if
24:         end while
25:     end if
26: end if
27: return $V_{dd}$ and $V_{bs}$;
```

4. EXPERIMENTAL RESULTS

In this section, we present experimental results to evaluate the effectiveness of our user-perceived latency driven voltage scaling method. We then discuss the impact of the decay factor w used in the *AVG(w)* prediction mechanism.

We applied our voltage scaling technique to seven commonly-used applications, (1) *TuxRacer*, a video racing game; (2) *GpsDrive*, a navigation system; (3) *Xpdf*, a PDF viewer; (4) *Mines*, a small mining game; (5) *KCalc*, a calculator; (6) *KPaint*, a simple drawing application; (7) *KEdit*, a text editor, ranging from modern interactive applications to simple textual ones. The CPU utilization and user-perceived latency traces were collected on a IBM Thinkpad laptop. The constants are adapted from [7] for the 70nm technology. We chose $7GHz \le f \le 14GHz$, $\Delta f = 1GHz$, $0.5V \le V_{dd} \le 1.4V$, and $-1V \le V_{bs} \le 0V$. When the supply voltage changes from V_{dd1} to V_{dd2} and body bias voltage from V_{bs1} to V_{bs2}, the transition energy overhead is derived based on Stratakos's analysis [12]:

$$\Delta E = C_r(V_{dd2}^2 - V_{dd1}^2) + C_s(V_{bs2}^2 - V_{bs1}^2) \qquad (2)$$

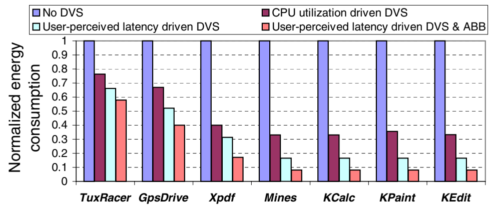
where C_r is the capacitance of the power rail, and C_s is the total capacitance of the substrate and wells of the device. The transition time overhead is given by:

$$\Delta t = max(\delta_{V_{dd}}|V_{dd2} - V_{dd1}|, \delta_{V_{bs}}|V_{bs2} - V_{bs1}|) \qquad (3)$$

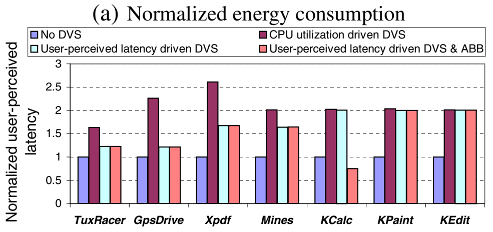
where $\delta_{V_{dd}}$ and $\delta_{V_{bs}}$ are the times needed to increase/decrease supply voltage and body bias voltage by 1V, respectively.

4.1 Effectiveness

We first discuss the effectiveness of our voltage scaling technique in terms of energy consumption and computer responsiveness for interactive applications. We chose $C_r = 1\mu F$, $C_s = 1\mu F$, $\delta_{V_{dd}} = 10\mu s/V$, and $\delta_{V_{bs}} = 10\mu s/V$. The human perceptual threshold K is set to 100ms. $[K_{min}, K_{max}] = [K - \Delta K, K + \Delta K]$, where $\Delta K = 0.1K$. Figure 3 shows the normalized energy consumption and user-perceived latency for the benchmarks using different voltage scaling policies based on the *AVG(3)* prediction mechanism. It can be seen that using user-perceived latency driven DVS, the energy consumption is reduced by an average of 37.3% and the user-perceived latency is reduced by an average of 18.3% with respect to the CPU utilization driven one. This justifies the advantages of considering the tradeoff between energy consumption and computer responsiveness during voltage scaling for interactive applications. The energy consumption is further reduced by an average of 38.9% when user-perceived latency driven combined DVS and ABB is used compared to DVS alone, while maintaining the computer responsiveness at a similar level. The combined DVS and ABB approach optimizes dynamic power and leakage power simultaneously to reduce energy consumption effectively. The impact of ABB is expected to be larger as we go deeper into the nanometer regime. We obtain similar results when voltage scaling policies are based on the *PAST* prediction mechanism. User-perceived latency driven DVS yields an average energy reduction of 39.2% and an average user-perceived latency reduction of 14.6% with respect to the CPU utilization driven one. The energy consumption is further reduced by 34.0% when user-perceived latency driven combined DVS and ABB is used compared to DVS alone. *AVG(w)* will be compared against *PAST* later.



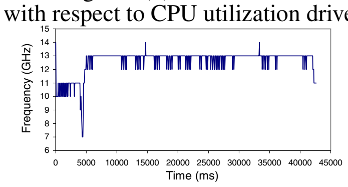
(a) Normalized energy consumption



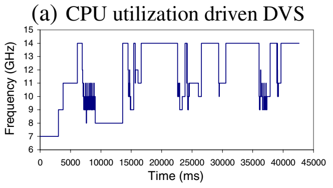
(b) Normalized user-perceived latency

Figure 3: Effectiveness of user-perceived latency driven voltage scaling

Figures 4 and 5 show the performance setting decisions for DVS driven by CPU utilization and user-perceived latency during the execution of *TuxRacer* and *KPaint*, respectively. They provide a qualitative insight into the characteristics of different voltage scaling policies. For *TuxRacer*, after the race starts (from 5s onwards), CPU utilization driven DVS maintains the processor mainly on two higher performance levels, 13GHz and 12GHz, and switches between them as can be seen from Figure 4(a). This is due to the high CPU usage of *TuxRacer*. Since CPU utilization driven DVS is unaware of the user's perception of unresponsiveness, it acts conservatively when deciding the performance levels. On the other hand, user-perceived latency driven DVS is able to predict the necessary performance level more accurately. The performance level varies from 9GHz to 14GHz after the race starts as shown in Figure 4(b). Once the user-perceived latency is predicted to be less than K_{min}, the performance level is lowered to provide energy savings. The fraction of the total time that the processor runs at 9GHz is 7.0%, compared to 0.6% for CPU utilization driven DVS. If the user-perceived latency is predicted to be above K_{max}, the performance level is boosted to improve computer responsiveness. For 48.7% of the total execution time, the processor runs at 14GHz, compared to 0.4% for CPU utilization driven DVS. User-perceived latency driven DVS responds quickly to current user perception of computer responsiveness and acts aggressively to decide the performance levels for obtaining an energy reduction. For *KPaint*, CPU utilization driven DVS boosts the performance level during execution due to high CPU usage, as seen from Figure 5(a). However, since the user-perceived latency for *KPaint* usually hovers at a low level (around 10ms), user-perceived latency driven DVS always maintains the processor at 7GHz to provide maximum energy savings, as shown in Figure 5(b), and reduces the energy consumption by 53.6% with respect to CPU utilization driven DVS.

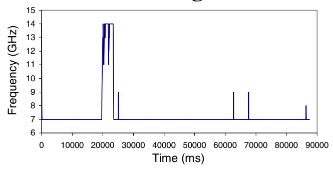


(a) CPU utilization driven DVS

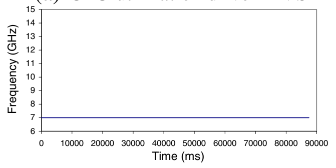


(b) User-perceived latency driven DVS

Figure 4: Performance setting decisions for *TuxRacer*



(a) CPU utilization driven DVS



(b) User-perceived latency driven DVS

Figure 5: Performance setting decisions for *KPaint*

The value of the human-perceptual threshold K in user-perceived latency driven voltage scaling is tunable. The normalized energy consumption and user-perceived latency for *TuxRacer* using different values of K during voltage scaling, based on the *AVG(3)* prediction mechanism, is shown in Figure 6. By changing the value of K, the user can achieve different levels of computer responsiveness and energy reduction. For example, if $K = 75$ms, the computer responsiveness is improved by 16.5%, while the energy consumption is increased by 33.3%, with respect to the case when $K = 100$ms.

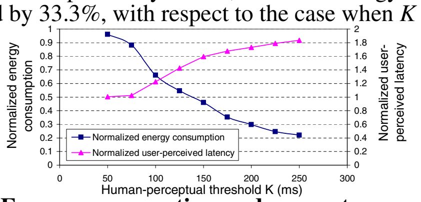


Figure 6: Energy consumption and computer responsiveness for *TuxRacer*

4.2 Impact of decay factor

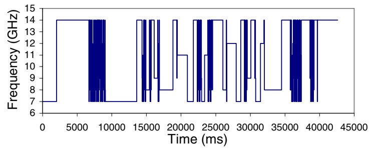
Next, we consider the impact of the decay factor w in the *AVG(w)* prediction mechanism on the user-perceived latency driven voltage scaling algorithm. As mentioned in Section 3.1, w impacts the number of voltage transitions. To demonstrate the impact of w on the transition overhead clearly, we chose $C_r = 20\mu F$, $C_s = 20\mu F$, $\delta_{V_{dd}} = 200\mu s/V$, and $\delta_{V_{bs}} = 200\mu s/V$. The human perceptual threshold K is set to 100ms. Table 1 shows the normalized energy consumption and user-perceived latency for *TuxRacer* using *AVG(w)* with different values of w. It can be seen that *AVG(3)* achieves the minimum energy consumption. Using *PAST*, equivalent to *AVG(0)*, the energy consumption is increased by 53.1% and the user-perceived latency by 40.7% with respect to no voltage scaling. This is due to the frequent voltage transitions, as seen from Figure 7(a), which incur a high transition overhead in energy and time.

5. CONCLUSIONS

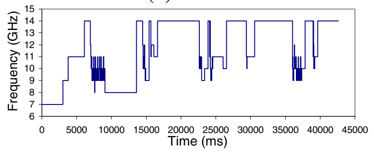
In this paper, we discussed the importance of considering computer responsiveness during voltage scaling for interactive applications. We showed how to track and predict the user-perceived latency for user inputs and utilize it, instead of CPU utilization, to drive voltage scaling. We considered the tradeoff between energy consumption and computer responsiveness to enable the voltage scaling technique to reduce energy consumption more aggressively than the CPU utilization driven one. At the same time, computer responsiveness is improved with respect to the CPU utilization driven one. Experimental results clearly demonstrated the effectiveness of user-perceived latency driven voltage scaling in reducing energy consumption, while ensuring good computer responsiveness for interactive applications. It is worth mentioning that although the implementation of user-perceived latency driven voltage scaling is based on Linux with the X Window system, the methodology is extensible to other operating systems as well.

Table 1: Impact of decay factor w

	PAST	*AVG(3)*	*AVG(7)*
Normalized energy consumption	1.22	0.71	0.73
Normalized user-perceived latency	1.41	1.23	1.17



(a) *PAST*



(b) *AVG(3)*

Figure 7: Performance setting decisions using different values of w during the execution of *TuxRacer*

6. REFERENCES

[1] Intel Xscale, `http://www.intel.com`.

[2] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. Symp. Operating System Design & Implementation*, Nov. 1994, pp. 13–23.

[3] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proc. Int. Symp. Low Power Electronics & Design*, Aug. 1998, pp. 76–81.

[4] C. H. Kim and K. Roy, "Dynamic V_{TH} scaling scheme for active leakage power reduction," in *Proc. Design, Automation & Test in Europe Conf.*, Mar. 2002, pp. 163–167.

[5] K. Nose, M. Hirabayashi, H. Kawaguchi, S. Lee, and T. Sakurai, "V_{TH}-hopping scheme for 82% power saving in low-voltage processors," in *Proc. Custom Integrated Circuits Conf.*, May 2001, pp. 93–96.

[6] M. Miyazaki, J. Kao, and A. Chandrakasan, "A 175mV multiply-accumulate unit using an adaptive supply voltage and body bias architecture," in *Proc. Int. Conf. Solid-State Circuits*, Feb. 2002, pp. 58–59.

[7] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2002, pp. 721–725.

[8] L. Yan, J. Luo, and N. K. Jha, "Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003, pp. 357–364.

[9] D. P. Siewiorek, "New frontiers of application design," *Commun. ACM*, vol. 45, no. 12, pp. 79–82, Dec. 2002.

[10] B. Schneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed. Addison Wesley Longman, 1998.

[11] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-computer Interaction*. Lawrence Erlbaum Associates, 1983.

[12] A. Stratakos, *High-efficiency Low-voltage DC-DC Conversion for Portable Applications*. Ph.D. dissertation, Univ. of California, Berkeley, 1998.