Dynamic Power Management Using Data Buffers

Le Cai* and Yung-Hsiang Lu
School of Electrical and Computer Engineering, Purdue University
{lc, yunglu}@purdue.edu

Abstract

This paper presents a method to reduce energy consumption by inserting data buffers. The method determines whether power can be reduced by inserting a buffer between two components and periodically turning off one of them. This method calculates the length of the period and the required buffer size to achieve the optimal energy savings. Our approach can be applied to any applications whose data arrival and departure rates are different and known in advance.

1. Introduction

Energy conservation is one of the most important design goals for electronic systems. Many methods have been proposed for reducing the energy consumption of individual components, such as processors, wireless network interface cards (WNICs), or hard disk drives (HDDs). These methods predict the periods when a component is idle or under-utilized. The component is turned off (sometimes called shut down) or scaled down the performance to reduce energy consumption during the periods. This is called energy management or dynamic power management [1, 2]. Some methods use scheduling to explicitly create idle or under-utilized periods [10, 12, 14]. Even though existing methods have demonstrated various degrees of energy savings, they are difficult to extend to two interacting components with different performance levels. When these components have producer-consumer relationship, data buffers can be inserted to achieve power savings [9, 12, 18].

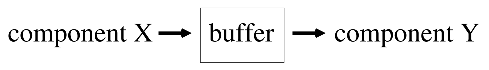

Figure 1. A buffer is inserted between two interacting components.

Figure 1 illustrates a model of two interacting components, X and Y. Component X produces data for component Y to process. A buffer is inserted between these two components. This model can be applied to many scenarios. For example, a processor generates data and stores the data on an HDD. In this example, X is the processor and Y is the HDD. Another example is a processor sending data through a WNIC; X is the processor and Y is the WNIC. A symmetric example is a WNIC receives data for the processor to analyze; in this example, X is the WNIC and Y is the processor. This model can be extended to more than two components. Suppose in the last example, the processor writes the analysis results to an HDD, there are three components: the WNIC, the processor, and the HDD.

Inserting a buffer may reduce energy consumption if Y can be turned off when data are accumulating in the buffer. Before the buffer is full, Y is turned on and removes all data in the buffer. Several factors affect the energy savings: (a) the rate X generates data, also called the *arrival rate* for the buffer, (b) the rate Y removes data, also called the *departure rate*, (c) the energy to turn on and off Y, (d) the power consumed by the buffer, and (e) the buffer size. Inserting buffers also introduces delay, as explained in Section 4.5. This paper focuses on energy savings and considers applications where delay can be tolerated.

This paper presents a new method for energy reduction using buffers. It assumes constant arrival and departure rates. It considers static and dynamic energy consumed by all components. The method calculates the length of the period when Y should be turned off to achieve the optimal energy savings. The method also calculates the necessary buffer

* Le Cai is supported by Purdue Research Foundation.

size to achieve the optimal energy savings. Because the buffer consumes energy, our method determines whether inserting buffers can actually reduce the total energy consumption. Our method is essentially a scheduling solution. It is deadlock-free because there is no circular waiting. This paper has the following contributions: (a) It presents a general strategy for power management with buffers. (b) It includes the static energy consumed by the buffers. (c) It calculates the optimal buffer sizes. (d) It provides the conditions when inserting buffers can reduce energy. (e) It complements performance scaling because it may be impossible to scale Y's performance and match it to X's performance.

2. Related Work

Various techniques have been developed to reduce the energy consumption of electronic systems. Energy management [1, 2] dynamically changes a component's power state based on the run-time requirements. Energy management detects the periods when a component is idle or under-utilized. The component can be turned off when it is idle, or scaled down its performance when it is under-utilized. Because changing a component's state has overhead — additional energy and delay — energy management is beneficial only if the saved energy can compensate the overhead. Several surveys are available about different techniques in energy management [1, 2, 13].

Most existing methods focus on individual components, for example, processors [9, 10, 11, 14, 17] and HDDs [3, 4, 5, 7, 8, 19]. To reduce the energy of processors, these methods detect when the processors are under-utilized and scale down the processors' performance by reducing the frequencies and, sometimes, voltages. The differences among existing methods are their approaches to detect these under-utilized periods. Some methods use scheduling techniques to explicitly create under-utilized periods. Managing HDDs' energy is different because most HDDs have only one power state that can read or write data. Hence, these methods have to predict when the HDDs are idle [3, 4, 5, 7, 8, 19]. The prediction methods can be further divided into three types: timeout, predictive, and stochastic [13].

Using buffers to reduce energy has been proposed in [9, 12, 18]. Im et al. [9] use buffers to exploit the slack time when multimedia applications do not use the worst-case execution time. When the processor has slack time, the processor fills the buffers so that the processor's frequency and voltage can be scaled down. Lu et al. [12] use buffers to smoothen the variations of MPEG frames' execution time. Their method constructs a graph whose vertexes represent the buffers' status and the processor's frequency. They find a sequence of the vertexes to minimize the sum of the frequencies. Qiu et al. [18] use a buffer and model the arrival of requests as Markov processes; their method provides guaranteed quality of service with lower power consumption.

Our approach differs from existing methods in four aspects: First, our method is a strategy for energy management with buffers. Second, it considers the energy consumed by all components, including the source (X), the destination (Y), and the buffer. Third, our method calculates the buffer size for optimal power savings. Determining the optimal buffer size for reducing energy is similar to determining the optimal warehouse size for maximizing profits [6]. Fourth, it provides the conditions when adding buffers can reduce energy; no existing method provides similar guidance whether inserting buffers can reduce energy.

3. Problem Statement

Our method considers the following parameters:

1. α: arrival rate. X generates α MBs of data every second.
2. β: departure rate. Y removes β MBs from the buffer every second. To prevent buffer overflow, the departure rate has to be greater than or equal to the arrival rate, i.e. $\beta \geq \alpha$. We assume that α and β are constants.
3. e_X: energy consumed by X to generate each MB of data. X also consumes static power p_X even when X does not generate any data.
4. e_Y: energy consumed by Y to remove each MB of data. Y consumes static power p_Y even when Y does not remove any data.
5. e_b: energy consumed by the buffer to store each MB of data; this includes the energy to write data (by X) and to read data (by Y). The buffer also consumes static (leakage) power p_b for each MB of capacity; this power depends on the total size of the buffer and is independent of the amount of data stored in the buffer.
6. e_d: dynamic energy for generating, storing, and removing one MB, $e_d = e_X + e_Y + e_b$.

7. k: energy to turn on and off Y.

Our method reduces the static power of component Y by turning it off periodically. We will answer the following questions in the next section: (a) Should Y be turned on/off periodically to reduce the total energy consumption? What are the conditions to reduce energy by turning off Y periodically? (b) What is the length of a period? (c) What is the size of the buffer? (d) How are these answers affected by the values of the parameters?

4. Energy Reduction

4.1. Problem Formulation

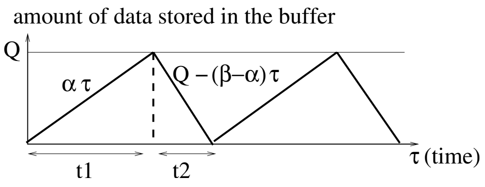

Figure 2. The amount of data stored in the buffer changes periodically.

Figure 2 shows the amount of data stored in the buffer if Y is turned off periodically. During $[0, t_1]$, Y is turned off; data accumulate in the buffer at rate α. At t_1, Y is turned on and it starts removing the data from the buffer. At this moment, Q MBs of data are stored in the buffer. This is the required buffer size. Because X continues generating data, the amount of data in the buffer decreases at rate $\beta - \alpha$. The buffer becomes empty after t_2 and Y is turned off to save energy. Y is turned on again when Q MBs of data are stored in the buffer. Y is turned on every $t_1 + t_2$; this is the length of a period, represented by t.

The total energy consumed in one period is the energy consumed by X, the buffer, and Y. Each component consumes both static energy and dynamic energy. The static energy is consumed even when no data are processed; hence, it is independent of α. The dynamic energy occurs for generating, storing, or removing data so it depends on α.

1. X: static $= p_X \times t$, dynamic $= e_X \times \alpha \times t$.
2. buffer: static $= p_b \times Q \times t$, dynamic $= e_b \times \alpha \times t$.
3. Y: static $= p_Y \times t_2$, dynamic $= e_Y \times \alpha \times t$. Y also consumes energy k every period for being turned on and off. Y consumes static power during t_2 because Y is turned off during t_1.

This method saves Y's static energy during t_1 but it also introduces overhead: (a) energy k to turn on and off Y every period and (b) the energy consumed by the buffer. This additional energy is $k + p_bQt + e_b\alpha t$ per period; therefore, the power overhead is $\frac{k+p_bQt+e_b\alpha t}{t} = \frac{k}{t} + p_bQ + e_b\alpha$.

The values of t_1 and t_2 follow the relationship $Q = \alpha t_1 = (\beta - \alpha)t_2$. Since $t_1 + t_2 = t$, we can obtain $t_1 = \frac{(\beta-\alpha)t}{\beta}$, $t_2 = \frac{\alpha t}{\beta}$, and $Q = \frac{(\beta-\alpha)\alpha t}{\beta}$. The total energy in each period is $p_Xt + e_X\alpha t + p_bQt + e_b\alpha t + p_Yt_2 + e_Y\alpha t + k$. This can be simplified as $p_Xt + p_Yt_2 + e_d\alpha t + p_b\frac{(\beta-\alpha)\alpha t}{\beta}t + k$. Our goal is to minimize the average power, namely $\frac{(p_X+p_Y\frac{\alpha}{\beta})t+e_d\alpha t+p_b\frac{(\beta-\alpha)\alpha t}{\beta}t+k}{t}$. We can separate the formula to two parts. The first part is independent of t and the second part depends on t:

$$\begin{array}{ll} \text{(a)} & \{p_X + \frac{\alpha}{\beta}p_Y + e_d\alpha\}+ \\ \text{(b)} & \{p_b\frac{(\beta-\alpha)\alpha}{\beta}t + \frac{k}{t}\} \end{array} \tag{1}$$

4.2. Period Length

In (1), the first part is independent of t. Hence, we focus on the second part to find the period to make average power the minimum.

$$\min\{p_bt\frac{(\beta-\alpha)\alpha}{\beta} + \frac{k}{t}\} \tag{2}$$

This formula has two terms. The first term grows as the length of a period increases. It represents the static power consumed by the buffer. When the period is longer, more data are stored in the buffer. Thus, the buffer has to be larger and consumes more static power. The second term decreases as the length of a period increases. This term represents the amortized power for turning on and off Y. We find the minimum power by taking the first derivative of (2). The minimum power consumption occurs when

$$t = \sqrt{\frac{k\beta}{p_b\alpha(\beta-\alpha)}} \tag{3}$$

If we replace t in Formula (1), we can obtain the minimum power as

$$p_X + \frac{\alpha}{\beta}p_Y + \alpha e_d + 2\sqrt{kp_b\alpha(1-\frac{\alpha}{\beta})} \tag{4}$$

The buffer size is $Q = \frac{(\beta-\alpha)\alpha t}{\beta}$, or

$$Q = \sqrt{\frac{k\alpha}{p_b}(1 - \frac{\alpha}{\beta})} \tag{5}$$

4.3. Analysis

According to formula (3) when the energy overhead k increases, t increases to amortize the overhead. The optimal value of t is proportional to the square root of k. As t increases, Y is turned on/off less frequently and the overhead occurs less frequently. On the other hand, when p_b increases, t becomes shorter to make Q smaller so that the buffer consumes less static power. The dynamic energy (e_d) has no effect on calculating the optimal length t.

The buffer size Q depends on both α and $(1-\frac{\alpha}{\beta})$. If β is a constant, Q is the maximum when $\alpha = \frac{\beta}{2}$. When $\alpha < \frac{\beta}{2}$, Q decreases as α becomes smaller. This can be explained in the following way: as α becomes smaller, data accumulate in the buffer slowly so Q can be smaller. When $\frac{\beta}{2} < \alpha < \beta$, however, Q decreases as α becomes larger. This is because t_1 becomes smaller so Y has to be turned on sooner.

When $0 < \alpha < \frac{\beta}{2}$, the power consumption shown in (4) decreases as α decreases because both Y's static power and the power overhead decrease. When β increases, $\frac{\alpha}{\beta}p_Y$ decreases but $1 - \frac{\alpha}{\beta}$ increases. Therefore, the power may increase or decrease according to the ratio of p_Y and $\sqrt{kp_b\alpha}$. When β is very large, t_2 is very short so Y's static power ($\frac{\alpha}{\beta}p_Y$) is negligible. The power in (4) can be approximated by $\lim_{\beta\to\infty}\{p_X + \frac{\alpha}{\beta}p_Y + \alpha e_d + 2\sqrt{kp_b\alpha(1-\frac{\alpha}{\beta})}\} = p_X + \alpha e_d + 2\sqrt{kp_b\alpha}$.

4.4. Management Decision

We have addressed questions (b) - (d) listed in Section 3. Now, we answer whether energy can be saved by turning Y off periodically. The answer depends on two factors: the delay to turn on/off Y and the average power if Y is kept on continuously. The delay to turn on a component can range from a few milliseconds for a processor to several seconds for an HDD. Energy management should be considered when t_1 is longer than the delay. It t_1 is too short, a larger buffer is needed and it consumes more power. The second factor considers the power when Y is always on and consumes the static power continuously. Since the buffer becomes unnecessary, the average power is

$$p_X + p_Y + (e_X + e_Y)\alpha \tag{6}$$

We compare the difference between this power and the power in (4). Energy management reduces the average power if (6) > (4):

$$\begin{aligned}&\{p_X + p_Y + (e_X + e_Y)\alpha\} - \{2\sqrt{kp_b\alpha(1-\tfrac{\alpha}{\beta})} + \\ &\quad p_X + \tfrac{\alpha}{\beta}p_Y + (e_X + e_b + e_Y)\alpha\} > 0 \\ &\Rightarrow (1-\tfrac{\alpha}{\beta})p_Y - (\alpha e_b + 2\sqrt{kp_b\alpha(1-\tfrac{\alpha}{\beta})}) > 0\end{aligned} \tag{7}$$

In summary, energy management can save energy if t_1 is longer than the delay and inequality (7) is true. The left-hand side of inequality (7) is the average power savings using buffers. Since Y is turned on only during t_2, buffer insertion reduces its static power by $(1-\frac{\alpha}{\beta})p_Y = (1-\frac{t_2}{t})p_Y$. The power overhead is $\frac{k}{t} + p_bQ + e_b\alpha = \alpha e_b + 2\sqrt{kp_b\alpha(1-\frac{\alpha}{\beta})}$. The total power savings are the difference of Y's reduced static power consumption and the power overhead.

4.5. Delay

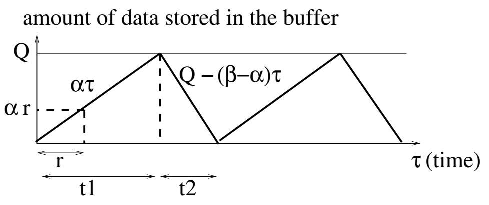

Figure 3. Delay calculation.

Buffer insertion introduces delay because data are buffered before component Y receives them. The delay depends on the time when the data enter the buffer. Suppose the buffer is first-in-first-out. In Figure 3, when one MB data enter the buffer at time r, αr MB data have already entered the buffer. This is true even when $r > t_1$; however, if $r > t_1$ some data have been removed from the buffer. This MB data which enter the buffer at r have to wait until all the earlier data are removed from the buffer. It takes $\frac{\alpha r}{\beta}$ time to remove the earlier data after Y is turned on. Hence, this MB is removed from the buffer at $t_1 + \frac{\alpha r}{\beta}$: the delay is $t_1 + \frac{\alpha r}{\beta} - r$. Remember $t_1 + t_2 = t$, $t_1 = \frac{(\beta-\alpha)t}{\beta}$, and $t_2 = \frac{\alpha t}{\beta}$. We can calculate the average delay for all data within one period by using the following formula for $0 \le r \le t$:

$$\frac{\int_0^t (t_1 + \frac{\alpha r}{\beta} - r) dr}{t} = \frac{t_1}{2} \quad (8)$$

The average delay is equivalent to half of Y's off time.

4.6. Extensions

This section discusses how to extend our method for two different scenarios: (a) multiple performance levels and (b) arrival rate > departure rate. Suppose Y has two performance levels corresponding to two different departure rates, β_1 and β_2. We can find the performance level which causes lower power consumption. At different performance levels, the static power of Y may be different. The energy to change power states may also be different. Let p_{Y1} and p_{Y2} be the static power for the two performance levels. Let k_1 and k_2 be the energy to turn on/off Y. We use (4) to determine the average power in the two performance levels: at β_1, the power is $p_X + \frac{\alpha}{\beta_1} p_{Y1} + \alpha e_d + 2\sqrt{k_1 p_b \alpha (1 - \frac{\alpha}{\beta_1})}$; at β_2, the power is $p_X + \frac{\alpha}{\beta_2} p_{Y2} + \alpha e_d + 2\sqrt{k_2 p_b \alpha (1 - \frac{\alpha}{\beta_2})}$. The difference is

$$\alpha\left(\frac{p_{Y1}}{\beta_1} - \frac{p_{Y2}}{\beta_2}\right) + 2\sqrt{p_b \alpha}\left(\sqrt{k_1 - \frac{k_1 \alpha}{\beta_1}} - \sqrt{k_2 - \frac{k_2 \alpha}{\beta_2}}\right) \quad (9)$$

We can use this formula to determine which performance level will cause lower power consumption based on the values of α, k, p_b, p_{Y1}, p_{Y2}, β_1, and β_2. If this difference is positive, β_2 should be used because it causes lower power consumption. Otherwise, β_1 should be used.

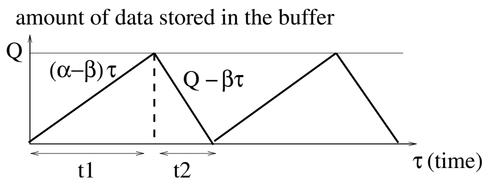

Figure 4. The amount of data in the buffer when $\alpha > \beta$ and X is turned off periodically.

Our method can be used to a symmetrical situation when $\alpha > \beta$. In this case, X is turned off periodically to save energy. Figure 4 shows the amount of data stored in the buffer. During t_1, X is turned on to generate data; the data accumulate at rate $\alpha - \beta$. Then, X is turned off during t_2 when Y continues removing the data from the buffer. From the figure, we obtain $Q = (\alpha - \beta)t_1 = \beta t_2$. Suppose k is the energy to turn on/off X. The minimum power occurs if we exchange α and β in (2) - (5).

5. Experiments

We use an Integrated Development Platform (IDP) from Accelent for our experiments because it has several test points for measuring the power of individual components. IDP has (a) an Intel 400MHz XScale processor, (b) 64MB SDRAM memory from Micron, and (c) an IBM 1GB microdrive using the PCMCIA interface. We use them as X , buffer, and Y respectively. We measure the power by a data acquisition card from National Instrument.

e_X	0.030 J/MB	e_Y	1.338 J/MB
e_b	0.006 J/MB	p_X	0.166 W
p_Y	0.125 W	p_b	0.012 W/MB
k	0.151 J	e_d	1.374 J/MB

Table 1. Parameters.

5.1. Parameters

We use three methods to obtain the parameters: (a) direct measurement, (b) calculation based on measurement, and (c) calculation using components' data sheets. First, we directly measure the static power of the XScale processor (X) and the microdrive (Y). They correspond to p_X and p_Y. Second, we calculate k, e_X, e_Y from the measurement results. The value of k corresponds to the energy consumed by Y to finish two operations in PCMCIA: `suspend` and `resume`. These two operations turn off and turn on the microdrive; together, they consume 0.151 J. In order to obtain the values of e_X and e_Y, we measure the power of X and Y with different values of α and β. The values of e_X and e_Y are obtained by subtracting the static power of X and Y and dividing the differences by α and β. Third, we use Micron's power calculator [15, 16] to calculate the power consumed by the SDRAM. This can calculate SDRAM's power consumption of different sizes. Experimental measurement cannot obtain the same information because SDRAM is always installed with a multiple of 2^n MBs (n is an integer). Table 1 lists the values of all parameters.

5.2. Power Savings

The power savings depend on the values of α and β. Figure 5 shows the percentage of power savings for different α and β. The percentage is the ratio of the left-hand side of inequality (7) and the static power of Y, namely $\frac{(1-\frac{\alpha}{\beta})p_Y-(\alpha e_b+2\sqrt{kp_b\alpha(1-\frac{\alpha}{\beta})})}{p_Y}$. The range of β is from 0.1 MB/S to 0.7 MB/S because 0.7 MB/S is the microdrive's maximum speed in IDP (this is lower than the maximum bandwidth in IBM's product specification). When β is a constant, power savings decrease as α grows. This is because the buffer consumes more power and Y has to be on longer. When α remains a constant, a larger β saves more power. The reason is that increasing β allows Y to remain off longer. When $\alpha = 0.33$ MB/S and $\beta = 0.7$ MB/S, the power savings are 0.029 W. Because the static power of Y is 0.125 W, the percentage is $\frac{0.029}{0.125} = 23\%$ (point A in Figure 5). The required buffer is 1.5 MBs. When $\alpha = 0.3$ MB/S, and β increases from 0.4 MB/S to 0.7 MB/S, the percentage increases from 5% to 28% (point B and C). This figure also indicates that, for a given value of β, no power is saved when α is sufficiently large. In other words, the inequality of (7) is not always true. This relationship between α and power savings depends on the values of k, p_Y,e_b, p_b, and β.

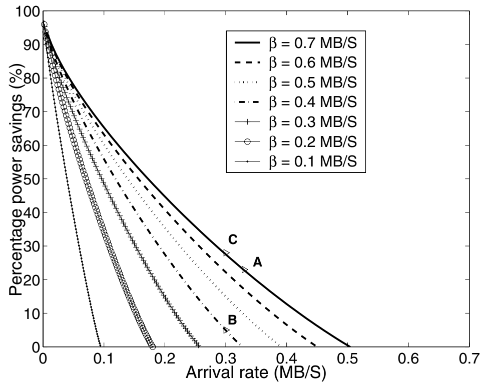

Figure 5. Power savings for different α and β

6. Conclusion

This paper presents a general method to reduce energy consumption by inserting data buffers between two components. This method considers both static and dynamic power of all components and calculates the buffer size for optimal power savings. The method can be extended to components with multiple performance levels. The experimental results show that the percentage of power savings depend on the arrival rate and the departure rate.

References

[1] L. Benini, A. Bogliolo, and G. D. Micheli. A Survey of Design Techniques for System-Level Dynamic Power Management. *IEEE Transactions on VLSI Systems*, 8(3):299–316, June 2000.

[2] L. Benini and G. D. Micheli. System-Level Power Optimization: Techniques and Tools. *ACM TODAES*, 5(2):115–192, April 2000.

[3] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. *International Conference on Supercomputing*, 86–97, 2003.

[4] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. D. Micheli. Dynamic Power Management for Nonstationary Service Requests. *IEEE Transactions on Computers*, 51(11):1345–1361, November 2002.

[5] F. Douglis, P. Krishnan, and B. Bershad. Adaptive Disk Spin-Down Policies for Mobile Computers. *Computing Systems*, 381–413, fall 1995.

[6] F. S. Hillier and G. J. Lieberman. *Introduction To Operations Research*. McGraw Hill, 1990.

[7] I. Hong and M. Potkonjak. Power Optimization in Disk-Based Real-Time Application Specific Systems. *ICCAD*, 634–637, November 1996.

[8] C.-H. Hwang and A. C.-H. Wu. A Predictive System Shutdown Method for Energy Saving of Event-driven Computation. *ACM TODAES*, 5(2):226–241, April 2000.

[9] C. Im, H. Kim, and S. Ha. Dynamic Voltage Scheduling Technique for Low-power Multimedia Applications Using Buffers. *ISLPED*, 34–39, 2001.

[10] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. *ISLPED*, 197–202, 1998.

[11] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi. Power-Aware Scheduling Under Timing Constraints for Mission-Critical Embedded Systems. *DAC*, 840–845, June 2001.

[12] Y.-H. Lu, L. Benini, and G. D. Micheli. Dynamic Frequency Scaling With Buffer Insertion for Mixed Workloads. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(11):1284–1305, November 2002.

[13] Y.-H. Lu and G. D. Micheli. Comparing System-Level Power Management Policies. *IEEE Design and Test of Computers*, 18(2):10–19, March 2001.

[14] J. Luo and N. Jha. Static and Dynamic Variable Voltage Scheduling Algorithms for Real-time Heterogeneous Distributed Embedded Systems. *ASP-DAC*, 719–726, 2002.

[15] Micron Technology Inc. SDRAM Power Calculator manual. http://download.micro.com/pdf/technotes /TN4603.pdf, May 2001.

[16] Micron Technology Inc. SDRAM data sheet. http://www.micron.com/dramds, January 2003.

[17] J. Pouwelse, K. Langendoen, and H. Sips. Energy Priority Scheduling for Variable Voltage Processors. *ISLPED*, 28–33, 2001.

[18] Q. Qiu, Q. Wu, and M. Pedram. Dynamic Power Management in A Mobile Multimedia System With Guaranteed Quality-of-service. *DAC*, 834–839, 2001.

[19] D. Ramanathan and R. Gupta. System Level Online Power Management Algorithms. *DATE*, 606–611, 2000.