

VOLTAGE SCALING FOR ENERGY MINIMIZATION WITH QOS CONSTRAINTS

Ali Manzak and Chaitali Chakrabarti

Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287-5706
manzak@asu.edu, chaitali@asu.edu

ABSTRACT

In this paper we propose variable voltage scheduling algorithms that minimize energy while satisfying the quality of service requirements. We consider the case when multiple applications are running on a single processor equipped with a limited sized buffer and each application has a different computational load and timing constraint. We use the Lagrange multiplier method to theoretically determine the relation between the application voltages such that the energy is minimum, and then develop iterative algorithms to satisfy the relation. The iterative algorithms find the minimum energy solution with polynomial time complexity for both the off-line case and the on-line case. We show the effect of buffer size and application deadline times on the ability of the system to reduce energy. Furthermore, we consider the effect of discharge current on battery life and show that the voltage assignment for maximum battery capacity is very similar to the voltage assignment for minimum energy.

1. INTRODUCTION

Energy minimization is an important performance metric in a world full of portable devices running on batteries. Substantial energy minimization can be achieved without sacrificing performance by lowering the supply voltage (and thus slowing the clock), instead of idling, when the computational load of an application is low. Thus determining the supply voltage of the processor as it executes applications with different computational loads such that the energy consumption is minimum is indeed an important problem.

There are several on-line and off-line algorithms for the variable voltage scheduling problem [1]-[7]. The scheduling problem for the case when there are no dependencies between the applications is addressed in [1] and solved using integer linear programming. A two phase scheduling algorithm is presented for non-preemptive hard real-time applications in [2], where in the first phase, all applications are scheduled at the nominal voltage and in the second phase, the voltages of each application are adjusted till no further reduction in power is obtained. The algorithm in [3] gives a minimum energy schedule for off-line preemptive scheduling and uses an average rate heuristic for the on-line version. A fixed priority scheduling scheme has been developed in [4] that exploits the variation in the applications' execution times in the assignment of the supply voltage. The assignment is done on an application by application basis. Greater energy savings can be obtained if the slack during the execution of single application can be exploited [5].

This feature has also been used in [6] where past history is used to predict the workload (and thus the supply voltage) during the execution of a single application. In our previous work in voltage scheduling [7], we first developed a relation between the application voltages that minimizes energy consumption and then used this relation to distribute the available slack. We considered different types of configurations - periodic, aperiodic, on-line, off-line, etc.

In this paper, we consider a system where multiple applications are being executed by a single processor and the supply voltage and frequency of the processor can be dynamically changed based on the workload. Thus if the workload is lower, the resulting slack is traded off for a reduction in the supply voltage (and thus energy). The processor is equipped with a buffer to better exploit the slack time in a multiple application environment. Our aim here is to determine the supply voltage of the processor during the execution of each application such that the quality of service requirements (application deadline time, total computation time) are satisfied and the total energy is minimized. Our procedure consists of first determining (theoretically) the optimal voltage assignment for minimum energy and then developing on-line and off-line algorithms for its implementation. Furthermore, we consider the effect of timing constraints (applications' deadline times) and buffer size on the system energy. A similar problem has been addressed in [8], where computation requirement of the system changes with the service time. A dynamic programming based method has been used to determine the speed of the processor. We also consider the case when the processor runs on battery.

The rest of the paper is organized as follows. Section 2 describes the energy minimization and the battery capacity maximization formulation given the QoS requirement. Section 3 describes the on-line and off-line algorithms to implement the energy minimization procedure. Section 4 describes the effect of buffer size and application deadlines on the minimum energy solution. Section 5 concludes the paper.

2. ENERGY MINIMIZATION

2.1. System Configuration

In Fig 1, the basic architecture for a dynamic voltage scaling system is shown. The input data is first stored in the buffer and then sent to the processor core. The application scheduling algorithm specifies the operating voltage and frequency based on the application load. The DC/DC

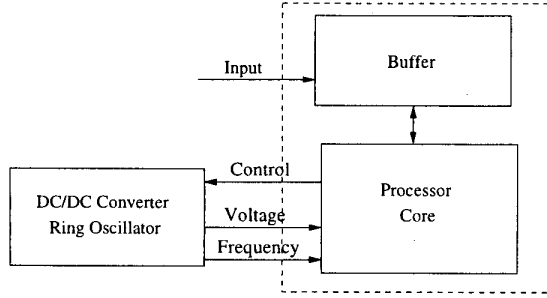


Figure 1: System architecture

converter has a sensitivity of ΔV and can provide voltages in the range $V_{min} \leq V \leq V_{max}$. The oscillator provides a clock frequency that is related to voltage V by $1/f = k' C_L \frac{V}{(V - V_t)^2}$, where V_t is the threshold voltage, C_L is the load capacitance and k' is a device parameter.

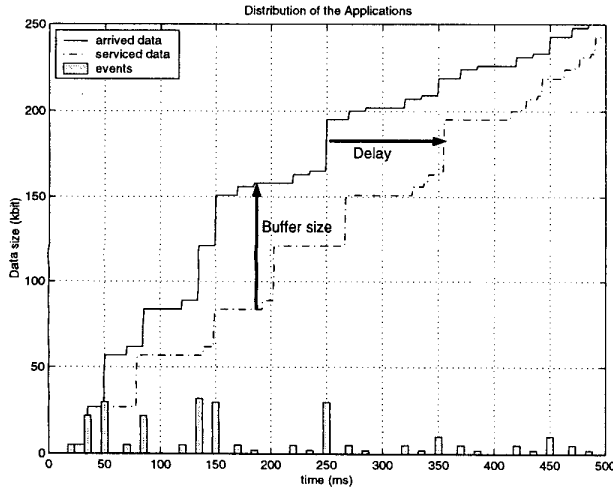


Figure 2: Arrival and service curve for the fictitious example

We consider the case when multiple applications are executed on a single resource (processor) and each application has a computation requirement that varies with time. We assume that the computational requirement is proportional to the data size. Thus if the data size is high, then the corresponding computational requirement is high.

Motivational example: Figure 2 shows the arrival times and sizes of individual events which are input to a system (fictitious example). For instance, at time $t = 250ms$, the event size is 30kbit. Figure 2 also describes the arrival and service curve for this example. The distribution of the event data is used to generate the arrival curve. Associated with each point in the arrival curve is arrival data, $a(t)$, defined as the total data that is input to the system at time t . Associated with each point in the service curve is serviced data, $s(t)$, defined as the total data that is sent to the processor for execution at time t . In the example in Figure 2, at

time $t = 180ms$, $a(t) = 157kbit$ and $s(t) = 84kbit$. Thus at time t , the system has to be able to store data of size $a(t) - s(t) = 73kbit$. This data is stored in a buffer. For feasibility, the buffer size $B \geq \max\{a(t) - s(t)\}$, for all t . We consider the buffer size to be a resource constraint. Furthermore, there is a delay associated with the buffer. This is defined as the difference between the time when data is input and when the data gets serviced. For instance in Figure 2, data j is input at time $a_j = 250ms$ and it is serviced at time $s_j = 354ms$. Thus the buffer delay is 104ms for data j . If it takes 60ms to execute data j , then the deadline of application j , $d_j \geq 414ms$ for feasible assignment (d_j is the time where execution of application j is completed).

In order to generate the service curve in Figure 2, all the applications are executed at 2V (reference voltage $V_{ref} = 3.3V$). This voltage assignment corresponds to the minimum energy solution. However this comes at the expense of a buffer size B of 80kbit and a buffer delay (maximum) of 170ms. If the given constraints, i.e., the buffer size and delay are less than these values, it would not be possible to execute the applications at 2V. As a result, the energy reduction would be smaller.

2.2. Problem Definition

The problem that we address in this paper is stated below.

- Given a set of applications, $\gamma_1, \gamma_2, \dots, \gamma_n$, each with its arrival time, size and execution time, find for each application its supply voltage such that the QoS requirements (application deadline time, total computation time) are satisfied and the total energy is minimized.

We consider applications activated at irregular intervals. We assume that the entire application is executed at the same voltage. If τ_j is the execution time of application j (at the reference voltage V_{ref}), n the number of applications and m_j the number of discrete instances of application j , then

$$T_{total} \geq \sum_{j=1}^n \tau_j = \sum_{j=1}^n k' C_L \frac{V_j}{(V_j - V_t)^2} \sum_{i=1}^{m_j} s_j(i), \quad (1)$$

where $s_j(i)$ is the number of control cycles of instance i of application j , V_j is the supply voltage of the processor when executing application j , V_t is the threshold voltage, C_L is the load capacitance and k' is a device parameter.

2.3. Formulation for energy minimization

The energy of application j is given by $\alpha_j C_L V_j^2 S_j$, where α_j is the average switching activity of application j , $S_j = \sum_{i=1}^{m_j} s_j(i)$ is the number of control cycles required to execute application j , and the terms C_L and V_j , are the same as defined before. The total energy consumption is thus

$$E_{total} = \sum_{j=1}^n \alpha_j C_L V_j^2 S_j$$

If all the applications have to be computed in time T_{total} , then our aim is to minimize E_{total} subject to the constraint $T_{total} = \text{constant}$. We use the Lagrange multiplier method to determine the supply voltage of each application. Note that while the total number of control cycles

(S_j) does not change with the service time, the time to execute a control cycle changes with the voltage.

We find E_{total} is minimum when the following condition is satisfied.

$$\frac{\alpha_1 V_1 (V_1 - V_t)^3}{V_1 + V_t} = \dots = \frac{\alpha_n V_n (V_n - V_t)^3}{V_n + V_t}$$

For $V_j > 3V_t$, this is approximated to

$$\alpha_1 (V_1 - V_t)^3 = \dots = \alpha_n (V_n - V_t)^3 \quad (2)$$

Thus, to minimize energy, the application voltages have to be chosen according to eqn.(2). Note that if the switching activities of the applications are similar, then all the applications are assigned to the same voltage. The voltages are different only if the switching activities are different.

Each application can consist of several instances. If the switching activity of each instance is known, then eqn.(2) can be extended by considering each instance to be a different application. Here too, if the switching activities of the instances of an application are similar, then all the instances of the application are assigned to the same voltage.

2.4. Formulation for maximum battery capacity

In battery operated devices, a more important performance metric is maximizing battery capacitance. Recent studies have shown that the total amount of energy provided by actual batteries is not a constant, but depends on the rate of discharge of the battery [9]. According to an empiric equation known as Peukert's formula, $C = x/I^\beta$, where C is the total energy that can be drawn from a battery (also known as the battery capacitance), x is a technology dependent constant (a characteristic of the particular type of battery used), I is the average discharge current, and β is a technology-dependent fitting factor. For typical NiCd batteries, for instance, β ranges between 0.1 and 0.3. Thus if we decrease the discharge current, we can actually increase the total amount of energy that is provided by the battery.

In order to estimate the current that is drawn from the battery, we assume for the sake of simplification that the transistors are operating in the saturation region and that the discharge current is the saturation current. For a switching activity α , the current that is drawn from the battery

$$I \propto \alpha (V_{dd} - V_t)^2.$$

Since $C = x/I^\beta$, maximizing C is equal to minimizing I^β . Using the Lagrange multiplier method for the case when $T_{total} = constant$, the battery capacitance is maximized when the following condition is true.

$$\alpha_1^\beta (V_1 - V_t)^{2\beta+1} = \dots = \alpha_n^\beta (V_n - V_t)^{2\beta+1} \quad (3)$$

Note that if $\beta = 1$, this equation is identical to eqn.(2). Since β is a small number less than 1, switching activity plays a very minor role. Thus for battery operated devices, it is reasonable to assign the same voltage to all the applications. The error introduced as a result of this assignment is quite small.

From the above analysis we conclude that if the switching activities of the applications are similar, minimizing energy also maximizes battery capacitance. In the following sections, we present the algorithms and results for energy minimization. The results for battery capacity maximization are very close and have not been included.

2.5. Example and Results

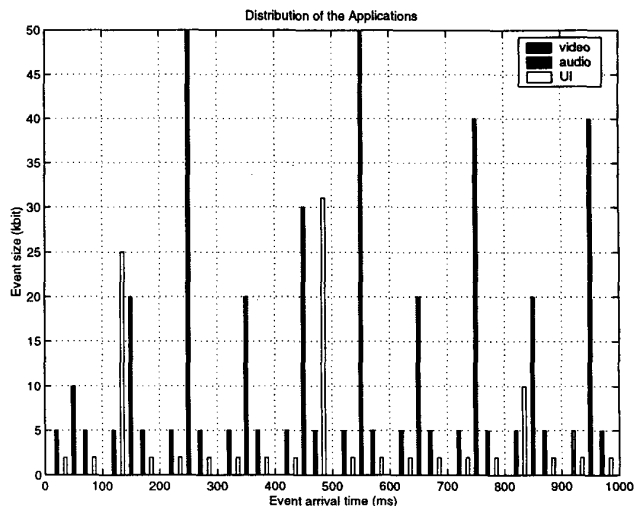


Figure 3: AVUI: Distribution of audio, video and UI events

We explain our procedure with an example, AVUI, where three applications, namely, video, audio and UI are being run simultaneously on the processor. Each application has different switching activity, period and execution time. Figure 3 shows the audio, video and UI signals changing periodically with periods of 100ms, 100ms and 50ms respectively. The average normalized switching activity of audio signals $\alpha_a = 1$, video signals $\alpha_v = 1.5$, and UI signals $\alpha_{UI} = 2$. The total size of the video, audio and UI signals are 300kbit, 100kbit and 100kbit respectively. So, the processor should be able to operate on 500kbit data in the given 1s time slot. We assume that the processor requires 1ms to execute on 1kbit data at $V_{ref} = 3.3V$. Thus to execute the 500kbit data only 500ms is required and the processor is 50% utilized. The available slack is now utilized to reduce the energy consumption.

Assume that the buffer size is unlimited and applications are soft (e.g. no hard arrival and deadline constraints). First, consider the most simplified case, where all the switching activities of all signals are the same. Then the minimum energy solution can be calculated using eqn.(2). By exploiting the available slack, the processor voltage can be reduced from 3.3V to 2V. The corresponding energy $E_{new} = 0.364E_{3.3}$.

Now if the average switching activities of each application are taken into account ($\alpha_a = 1$, $\alpha_v = 1.5$, $\alpha_{UI} = 2$), the processor voltages are 2V, 2.2V, 1.8V respectively for the video, audio and UI applications. The corresponding energy is $E_{new} = 0.36\bar{E}_{3.3}$, where $\bar{E}_{3.3}$ is the energy when the processor voltage is 3.3V and switching activities are taken into account. If we ignore the effect of switching activities while assigning voltages, the energy consumption is $E_{new} = 0.367\bar{E}_{3.3}$ which is 2% higher than the optimal solution. If we assume that the switching activity ratio between applications is at most 2, then ignoring the switching activity results in an error that is less than 3% for energy

minimization. However, if the switching activity ratio between instances of an application and between applications is small (< 2), the error introduced by ignoring the effect of switching activity in the voltage calculation (eqn.2) is quite small.

Another observation is that the variations on execution time of each application do not affect the voltage assignment for the minimum energy solution. Finally, the energy reduction is the highest when the buffer size is unlimited and applications are soft. In Section 3, we consider the effect of buffer size and application deadline on the energy reduction procedure.

2.6. Slack adjustment for minimum energy

In this section we describe how the Lagrange results can be used to optimally distribute the slack among the applications. Let V_k be the voltage of application k . Then, if the Lagrange relation (eqn.2) is to be satisfied, the voltage of application j , V_j is related to V_k by

$$V_j = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3}(V_k - V_t) + V_t,$$

Now if the voltage of application k is decreased by ΔV , where ΔV is the converter sensitivity, the voltage of the remaining applications have to be adjusted. The new voltage of application j is

$$V_{j_{new}} = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3}(V_k - \Delta V - V_t) + V_t$$

The change in voltage of application j , $\Delta V_j = V_j - V_{j_{new}}$ is a function of ΔV .

$$\Delta V_j = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3} \Delta V \quad (4)$$

From eqn.(4), we see that the maximum voltage change occurs for the application with the minimum α value. Let that application be referred to as application m . Then the application voltages are changed with respect to application m . This ensures that the voltage change for the remaining applications will not be more than ΔV . Thus

$$V_j = \left(\frac{\alpha_{min}}{\alpha_j}\right)^{1/3}(V_m - V_t) + V_t \quad (5)$$

Starting voltage of application m : The application voltages are iteratively adjusted so that the Lagrange relation is satisfied. In the first iteration, application m is set to V_{start} . V_{start} is higher than V_{max} to ensure that in the beginning no application is assigned a voltage lower than V_{max} . V_{start} is calculated assuming that the application with the highest α value is assigned V_{max} .

$$V_{start} = \left(\frac{\alpha_{max}}{\alpha_{min}}\right)^{1/3}(V_{max} - V_t) + V_t \quad (6)$$

Number of iterations: In the iterative procedure, in each step, the voltage of application m decreases by ΔV ; $V_m(k) = V_{start} - k\Delta V$. Thus in the k th iteration, the voltage of the application j ($1 \leq j \leq n$) is

$$V_j(k) = \left(\frac{\alpha_{min}}{\alpha_j}\right)^{1/3}(V_{start} - k\Delta V - V_t) + V_t \quad (7)$$

The maximum number of iterations is given by $k_{max} = \frac{1}{\Delta V}(V_{start} - V_{min}) = \frac{1}{\Delta V} \left(\left(\frac{\alpha_{max}}{\alpha_{min}} \right)^{1/3} (V_{max} - V_t) + V_t - V_{min} \right)$

By doing a binary search, the number of iterations can be reduced to $\log(k_{max})$.

3. ALGORITHMS

In the previous section, we considered the case when the system has unlimited buffer size and the applications have no deadline constraints. In a real system, applications have deadline constraints and the buffer size is not unlimited and should be considered as a resource constraint. In this section we develop on-line and off-line scheduling algorithms that minimize energy when buffer size is limited and each application has a deadline constraint.

3.1. Off-line Algorithm

Let us assume that the characteristics of input data, namely, data size ($data_j$), execution time (e_j), deadline (d_j), arrival time (a_j) or period (p_j) are known, and that all applications have to be executed in time T_{total} . The minimum energy solution is then given by eqn.(2). A modified version of EDF (Earliest Deadline First) algorithm can then be used to find the minimum energy solution with buffer constraint. The procedure is as follows.

In each iteration k , the voltage of the critical application (or application m) is decreased, the voltages of the other applications adjusted, and the service time of the application j , s_j , compared with its deadline d_j . Additionally, at service time of application j , $t = s_j$, the total data that is sent to the processor, $s(t)$, and the total data that has arrived at the buffer, $a(t)$, are calculated. The buffer size constraint $B \geq (a(t) - s(t))$ is checked. If there is a violation in the assignment $V_j(k)$, then the previous voltage value ($V_j(k-1)$) is the optimal voltage value. Furthermore, since applications with earlier deadline than application j could have caused deadline violation of application j , all these applications are assigned voltages corresponding to iteration $k-1$, $V_l = V_l(k-1)$ for $l = i$ to j (i is the application with the largest deadline where $a_i > s_{i-1}$). The algorithm continues until voltages for all the applications are determined.

Sort applications with respect to deadline

for $k = 1$ to k_{max}

 update $V_m(k)$, $\tau_m(k)$ for the critical application

 for each unscheduled application

 update $V_j(k)$, $\tau_j(k)$ using eqn.(7)

$s_j(k) = \max\{s_{j-1}(k), a_j\} + \tau_j(k)$

 update $a(t)$, $s(t)$ where $t = s_j(k)$

 if ($s_j(k) > d_j$) or ($B < (a(t) - s(t))$)

 Assign $V_l(k-1)$ to application l , $i \leq l \leq j$.

Complexity: The worst case complexity of the algorithm is $O(n \log n + nk_{max})$, where n is the number of applications and k_{max} is proportional to the number of voltage levels ($k_{max} = \frac{1}{\Delta V}(V_{start} - V_{min})$). The $n \log n$ part is due to sorting n applications based on the deadlines, and the nk_{max} part is due to finding the optimal voltages of the applications in k_{max} iterations.

3.2. On-line Algorithm

A more practical scenario is when there is limited information about future input data, such as arrival time or period. Since the maximum data size ($maxdata_j$) is known, we use

$maxdata_j$ to calculate $a(t)$ for periodic applications. We apply the off-line algorithm on the applications inside the buffer. When the processor finishes execution of a particular application, the algorithm assigns voltages to the applications inside the buffer based on the maximum data size of the future input data.

if at time t_0 processor is idle and buffer is not empty
Sort applications with respect to deadline

for $k = 1$ to k_{max}
 update $V_m(k), \tau_m(k)$ for the critical application
 for each unscheduled application
 update $V_j(k), \tau_j(k)$ using eqn.(7)
 $s_j(k) = \max\{s_{j-1}(k), a_j\} + \tau_j(k)$
 $a(t) = a(t_0) + \sum_{u=1}^j maxdata_u \lceil \frac{s_j(k)-t_0}{p_u} \rceil$
 update $s(t)$ where $t = s_j(k)$
 if $(s_j(k) > d_j)$ or $(B < (a(t) - s(t)))$
 Assign $V_j(k-1)$ to application j .

Complexity: In each cycle, the application with the earliest deadline is scheduled with complexity $O(l(\log l + k_{max}))$. This is because there are l applications inside the buffer and sorting them has a complexity of $O(l(\log l))$. Finding the optimal voltages of the applications has $O(lk_{max})$ complexity. The overall complexity of the algorithm is $O(n^2 k_{max})$, where n is the total number of applications.

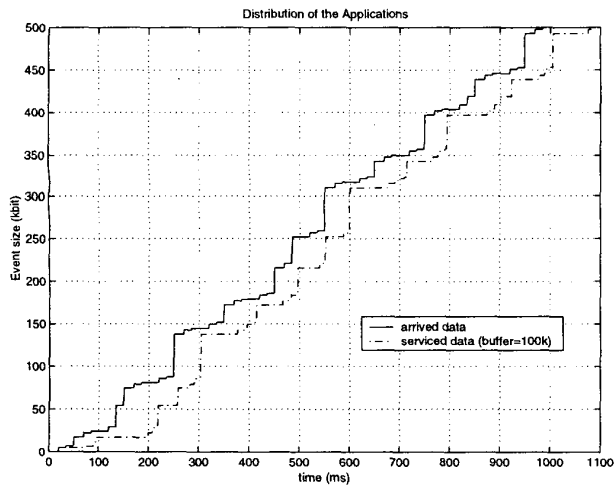


Figure 4: Arrival and service curve for the AVUI example for buffer size of 100kbits and no deadline constraint.

3.3. Optimality of the Algorithms

The proposed off-line algorithm gives optimal results since the characteristic of the input data is known. The on-line algorithm optimally chooses the voltages for the applications inside the buffer. Thus the optimality of the on-line algorithm is a function of the goodness of the future data estimates.

4. EFFECT OF BUFFER SIZE AND DEADLINE CONSTRAINTS

4.1. Buffer size constraints

In this section we show how buffer size affects the amount of energy that can be minimized during on-line scheduling. The service curve for buffer size of 100kbits for the AVUI example in Figure 3 has been shown in Figure 4. The energy for this configuration is $0.42E_{3.3}$, where $E_{3.3}$ is the energy at 3.3V.

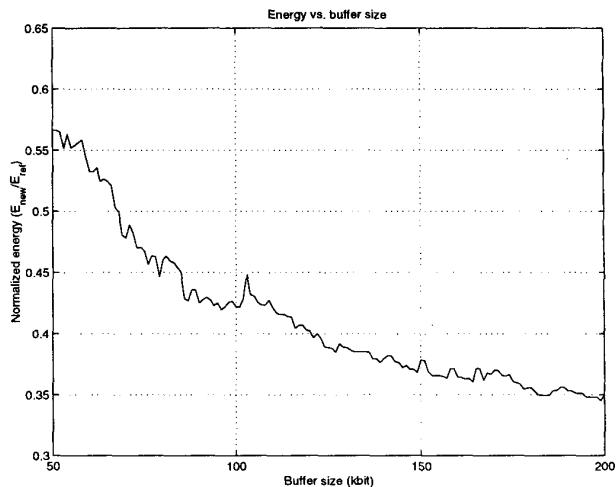


Figure 5: Energy vs. buffer size for the AVUI example without deadline constraint.

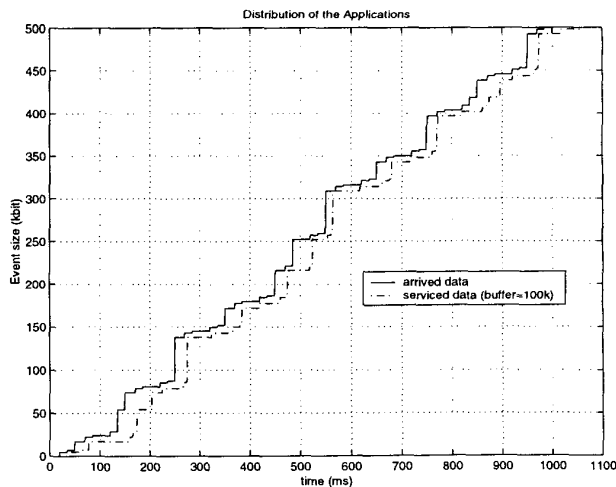


Figure 6: Arrival and service curve for the AVUI example for buffer size of 100kbits and deadline constraint 70ms

If the buffer size is large, significantly low voltages can be selected to satisfy the minimum energy formula provided the application's timing constraints are satisfied. Figure 5

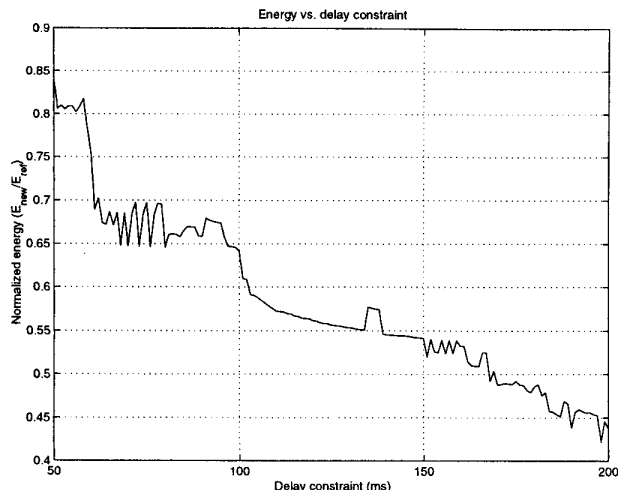


Figure 7: Energy vs. deadline constraint for the AVUI example for buffer size of 200kbits.

plots energy reduction with respect to buffer size for the AVUI example. The energy saving increases from 43% to 65% as the buffer size increases from 50kbit to 200 kbit.

From the plot we see that there are times when the energy increases even when buffer size increases. This situation occurs whenever large application instances arrive simultaneously and the buffer is relatively full. If the application is being executed at a low voltage then the newly arrived large instances may be assigned high voltages thereby increasing the energy. This case can be prevented if the voltage and frequency of the application can be changed during execution. However this would cause an increase in the overhead energy and delay (due to DC-DC converter).

4.2. Deadline constraints

Now consider the case when each instance has a deadline constraint. The energy reduction decreases further since the voltage assignments are now dictated more by the deadline constraints than the minimum energy equation. The service curve for buffer size of 100kbits and 70ms deadline constraint for each instances for the AVUI example has been shown in Figure 6. The energy for this configuration is $0.72E_{3.3}$.

Next, we demonstrate the effect of deadline constraint on the energy reduction. Figure 7 plots energy reduction with respect to deadline constraint for the AVUI example when the buffer size is 200kbits. The energy saving increases from 15% to 56% as the deadline constraint increases from 50ms to 200 ms. Compare this with the saving of 65% that is obtained when there is no deadline constraint. From Figure 7, we see that there are cases where energy increases even when the deadline constraint increases. This situation occurs when several large instances follow small instances.

5. CONCLUSIONS

In this paper we present EDF based on-line and off-line scheduling algorithms to optimally minimize energy under QoS constraints (buffer and deadline constraints). We theoretically determine the relation between the operating voltages for the minimum energy assignment using the Lagrange multiplier method and develop iterative algorithms that implement this relation. The complexity of the algorithms are $O(n \log n + nk_{max})$ for the off-line version and $O(n^2 k_{max})$ for the on-line version, where n is the number of applications and k_{max} is the number of iterations (depends on the voltage converter sensitivity).

6. REFERENCES

- [1] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processor," ISLPED 1998, pp 197-202.
- [2] I. Hong, D. Kirovski G. Qu, M. Potkonjak and M-B. Srivastava, "Power Optimization of Variable Voltage Core-Based Systems," DAC 1998, pp 176-81.
- [3] F. Yao, A. Demers and S. Shenker, "A scheduling model for reduced CPU energy," IEEE Annual Foundations of Computer Science, pp 374-82, 1995.
- [4] Y. Shin and K. Choi "Power conscious fixed priority scheduling for hard real-time systems," DAC 1999, pp 134-139.
- [5] D. Shin, J. Kim and S. Lee "Low energy intra-task voltage scheduling using static timing analysis," DAC 2001, pp 438-443.
- [6] T. Pering, T. Burd and R. Brodersen, "The simulation and evaluation of dynamic voltage scheduling algorithms," ISLPED 1998, pp. 76-81.
- [7] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," ISLPED 2001.
- [8] G.Qu and M. Potkonjak, "Energy minimization with Quality of Service," ISLPED 2000, pp 43-48.
- [9] T. L. Martin and D. P. Siewiorek, "A power metric for mobile systems," ISLPED 1996, pp. 37-42.