An On-line Approach for Power Minimization in QoS Sensitive Systems

Jennifer L. Wong
Computer Science Dept.
University of California, Los Angeles
Los Angeles, CA 90095
E-mail: jwong@cs.ucla.edu

Gang Qu
Electrical & Computer Engineering Dept.
University of Maryland
College Park, MD 20742
E-mail: gangqu@glue.umd.edu

Miodrag Potkonjak
Computer Science Dept.
University of California, Los Angeles
Los Angeles, CA 90095
E-mail: miodrag@cs.ucla.edu

Abstract— Majority of modern mobile systems have two common denominators: quality-of-service (QoS) requirements, such as latency and synchronization, and strict energy constraints. However, until now no synthesis techniques have been proposed for the design and efficient use of such systems. We have two main objectives: synthesis and conceptual. The synthesis goal is to introduce the first design technique for quality-of-service (QoS) low power synthesis. The conceptual objective is to develop a generic technique for the automatic development of on-line algorithms from efficient off-line algorithms using statistical techniques.

We first summarize a system of provably-optimal techniques that minimize energy consumption of stream-oriented applications under two main QoS metrics: latency and synchronization. Specifically, we study how multiple voltages can be used to simultaneously satisfy hardware requirements and minimize power consumption, while preserving the requested level of QoS in terms of latency and synchronization. The off-line algorithm is used as input to statistical software used to identify important relevant parameters of the processes, buffer occupancy rate indicators, and a way how combine them to form a fast and efficient on-line algorithm which decides which task to run at which voltage. The effectiveness of the algorithms is demonstrated on a number of standard multimedia benchmarks.

I. Introduction

In the last decade, low power synthesis and optimization techniques received a great deal of attention. A variety of techniques have been proposed for all steps of synthesis and compilation. Combination of new logic families and circuits, smaller feature sizes, more power efficient architectures, power-aware CAD tools, and low power dynamic run-time policies resulted in a dramatic increase in energy efficiency. However, the power requirements of new product generations have been constantly challenging the limits of battery capacities.

The most popular mobile low power applications, such as audio and video, are stream-oriented. The nature of these applications impose a need for addressing the QoS requirements under energy constraints. Latency and synchronization are the most relevant QoS metrics in these types of applications. Our goal is to develop a spectrum of techniques and algorithms which minimize energy consumption under the most important QoS metrics.

Specifically, we study how to use multiple voltage technologies to simultaneously satisfy hardware requirements and minimize power consumption, while preserving the requested level of QoS in terms of latency and synchronization. Our starting point is a provably optimal off-line algorithm for power minimization under QoS and buffer constraints. We identified four key properties of streaming processes (latency slack, synchronization slack, relative burstiness, and number of tasks (samples)), in addition to buffer occupancy, a crucial criteria for deciding which process to run at which voltage. We plot a 5xm-dimensional space, where m is the number of concurrent processes, and for each point in this space we use the off-line algorithm as an indicator of a correct decision with respect of task and voltage selection. This space is analyzed to reduce context switching overhead and speed up the decision process.

Our primary goal is to present competitive on-line algorithms for power minimization for streaming media applications for given hardware resource constraints: latency and synchronization constraints as well as context switching overhead. We aim to dynamically adjust the supply voltage in such a way that an incoming statistical stream of data does not overflow the buffer capacity of our processing system while expending the least amount of energy. By considering the long and short term statistics of the media streams and current buffer backlog, we decide which supply voltage to apply. Furthermore, by considering latency and synchronization constraints, we decide which task to schedule at the current moment. Finally, we use the new on-line algorithm to explore the trade-off between buffer size (cost) and energy consumptions.

II. Related Work

Our research results can be viewed in the context of four related areas: low power modeling and optimization, quality of service, on-line algorithms, and statistical techniques.

Power modeling and optimization have also been considered at different levels of the synthesis process [11]. Recently, a number of approaches have been proposed which reduce power consumption through the use of multiple voltages [5, 12], and a number of variable voltage techniques have been considered [15]. In most of these efforts, power minimization is achieved by scheduling operations on the critical path at higher voltages, which also results in faster execution. The overall energy consumption is reduced by scheduling noncritical operations at lower voltages than the critical path. Dynamic power management reduces power consumption of electronic systems by selectively shutting down idle components [2, 10].

QoS emerged from the area of networking and real-time operating systems. The requirements for QoS include components such as bounded delay, guaranteed resolution, and synchronization. Cruz developed the most relevant QoS model which assumes periodic segmentation of time [6]. A task of varying complexity arrives during each time period. At each time period, the system serves the next first-in-first-out task. An analytical approach was presented by Rajkumar et al. which satisfied multiple streams of tasks [13]. Additional information on QoS reserach can be found in a comprehensive survey [1].

The notion of an on-line algorithm was introduced in order to define a class of algorithms for which part of the input is unknown at the beginning of the algorithm execution. A comprehensive survey on the main research areas for on-line algorithms can be found in [3].

Statistical techniques can be broadly divided in two groups: parametric and nonparametric [14]. Parametric techniques assume that the knowledge about underlying statical distribution is available (often normal distribution is assumed) and that the task is to confirm the assumption about the distribution, calculate the corresponding parameters, and establish intervals of confidence [7]. Nonparametric techniques do not make any assumption about the statistical distribution. They aim to build the conceptually and quantitatively the simplest (and therefore best) model which fits the recorded data [14].

There are a number of validation techniques, such as histograms, Chi-square tests, Kolmogorov-Smirnov test and quantile-quantile plots that can be used to validate statistical claims of the model. We opted to use resubstitution techniques because these techniques enable the validation of an arbitrary hypothesis while establishing accurate interval of confidence [8].

III. Preliminaries

In this section, we outline the used abstraction and models for power consumption, latency, synchronization and context switch overhead.

One of the main components of power consumption is the switching power. The switching power can be modeled as $P = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f$, where $\alpha \cdot C_L$ is the effective switching capacitance. Switching power is the dominating factor in power consumption. The greater throughput comes with the cost of higher voltage. The gate delay of the circuit is defined as $T = k(V_{dd}/(V_{dd} - V_t)^2)$ where k is a constant [4].

We assume the design operates using multiple voltage supplies and that the voltages change instantaneously with no overhead. These changes in voltages are assumed to happen only at the beginning or the end of a time unit. Furthermore, we assume that the voltage units are selected in such a way that the use of two consecutive voltages, $v_i \& v_i$, on two consecutive points is more effective than the use of voltages $v_i \& v_{i+1}$, due to the fact that power as a function of voltage is convex.

Latency is defined as the difference between the time when the data is processed and the time the data arrived, i.e. t_p - t_a.

TABLE I
EXAMPLE OF SYNCHRONIZATION AND LATENCY FOR TWO TASKS

Time	0	1	2	3	4
Task Arrival					
P_1	a	b	c	d	-
P_2	w	x	y	z	-
Task Processed					
P_1	-	a	-	b	c,d
P_2	w	x	-	y	z

We denote time in which a particular sample (piece of date) arrives as t_a and the time when that piece of data is completely processed as t_p. Note that latency directly impacts the memory requirements because each piece of data has to be stored in local storage for its period of latency, t_a to t_p. At the intuitive level, synchronization indicates how well two or more processes are correlated in their execution. The assumption is that for each piece of data to be processed by one of the processes there exists a corresponding piece of data in each of the other processes which need to be processed at exactly the same time to have complete synchronization. However, the majority of real life applications such as, movies playing with its video and audio processes do not have to be fully synchronized due to the imperfections of the human sensory system. Synchronization can be defined in the following way. For the sake of simplicity, we assume that there are only two processes, p_1 and p_2. We denote the tasks of the processes p_1 by p_{1i} and p_2 by p_{2j}, where $i, j = \{1, \ldots, n\}$. Perfect synchronization constraints indicate which sample (task or piece of data) of process p_1, which is denoted by p_{1i}, has to be executed at the same time as piece of data p_{2j}. Synchronization tolerance (often for the sake of brevity is solely called "synchronization") indicates the maximal amount of time by which the execution of fully synchronized samples p_{1i} and p_{2j} can maximally differ.

For example, consider the two processes shown in Table I. For each process, p_1 and p_2, there are four tasks which arrive one at each time unit. For process p_1, we have tasks 'a' through 'd', and for p_2, we have 'w' through 'z'. The latency, or the time between when a task arrives and is processed, of task 'b' is two units. It arrives at time one and is processed at time three. The synchronization between two tasks, for example 'b' and 'x' or p_{11} and p_{21}, is two. However, 'd' and 'z' or p_{13} and p_{23} is zero because both tasks are processed in the same time unit.

A context switch is the time overhead which is incurred by a multitasking kernel when it decides to process different tasks. The amount of context switching time dramatically depends on the processor. Context switching time for a typical DSP processor is fairly low, around ten cycles, while for a RISC processor it is much higher, approximately 100 cycles. In our experimentation, we used ten cycles.

IV. Off-line Optimal Algorithm

The basis for the on-line algorithm is an optimal off-line algorithm [16]. In this section, we summarize the problem formulation of the off-line QoS low power synthesis problem and present an overview of the optimal off-line algorithm. Informally, we summarize the problem as follows. Given a processor that can operate at multiple supply voltages, the goal is to service multiple processes, each comprising of a number of discrete tasks that arrive at periodic time intervals, in such a way that minimal energy is expelled and a given amount of storage (memory) is not exceeded while meeting various QoS requirements.

More formally, a process consists of a sequence of tasks. With each task, t_i, we associate

- a_i: The arrival time, the time when a task is generated from the process and makes the CPU request;
- p_i: The time needed to complete this task at the nominal voltage, v_{ref};
- s_i: The storage demand which is the minimal amount of memory to store this task on its arrival.

Tasks have latency and synchronization constraints superimposed. Latency (or deadline) d_i is the given amount of time which task t_i has to be served after its arrival to satisfy QoS requirements. We say that task t_i from one process and task t_j from another are k-synchronized if the difference between their finishing times is within k CPU units. We denote this by $syn(t_i, t_j) \leq k$.

The variable voltage processor has multiple supply voltages among which it can switch. The processing speed of the processor varies with the supply voltage, and therefore so will the actual execution time of a task to receive its required amount of service. Suppose a task needs one CPU unit at the nominal voltage v_{ref}, then the execution time to accumulate the same amount of processing at voltage v_{dd} is given by[4]:

$$\frac{(v_{ref} - v_t)^2}{v_{ref}} \cdot \frac{v_{dd}}{(v_{dd} - v_t)^2} \qquad (1)$$

where v_t is the threshold voltage. We consider only the dominating switching power which is proportional to the square of the supply voltage.

Given n processes $\tau^1, \tau^2, \cdots, \tau^n$, each τ^k consists of a sequence of tasks $t_1^k, t_2^k, \cdots$. A *schedule* is a set consisting of the starting time, finishing time, and the voltage level for each task. A schedule is *feasible* if the processor starts each task after its arrival, finishes it before the latency constraint, and satisfies all the synchronization requirements. The quality of a schedule is measured by its energy consumption and the memory requirement. Since these two metrics are non-comparable to each other, we introduce the concept of competitiveness. We say two schedules are *competitive* if neither outperforms the other in both energy consumption and memory requirement. We formulate the problem as:

On a processor with multiple voltages, for a given set of processes, find all the feasible competitive schedules.

We adopt the following assumptions:

1. Tasks in the same process have to be executed and completed in the first-in-first-out (FIFO) fashion;
2. A task's processing demand, p_i, is proportional to its storage demand, s_i;
3. The memory occupied by a task can be partially freed, but only at the end of a CPU unit;
4. The processor can instantaneously switch the supply voltage, but only at the beginning of each CPU unit.

The optimal off-line algorithm finds all competitive schedules with the minimum energy consumption and memory requirement for multiple processes using a dynamic programming-based technique. For multiple processes, a schedule must specify for each CPU unit, which process to be executed and at what voltage level. Once the competitive schedules have been found a simple backtracking technique can be used to retrieve the actual schedule (i.e., the voltage for each CPU time unit).

The optimal off-line algorithm consists of three phases: state configuration, energy consumption calculation, and schedule determination. In the first phase, state configuration, a kxm dimensional table is built, where m is the number of processes and k is the number of voltage levels. This table holds the minimal memory requirement for each of the different schedules.

This table also computes the total memory requirement for each schedule, which corresponds to the value in the schedules final state. In the energy consumption calculation phase, the total energy consumption for each final state is calculated. Energy consumption is path-independent, therefore no matter which schedule was used to reach the final state it will consume the same amount of energy. Therefore for each final state, the memory requirement and energy consumption have been calculated. The last phase finds the schedules for each competitive final state. This is achieved by using backtracking.

Two theorems that summarize the key results of the optimal off-line algorithm are the optimality theorem and the complexity theorem [16]. The optimality theorem states that the dynamic programming based algorithm is provably optimal regardless of the number of tasks and their arrival distribution. The essence of the complexity theorem is that the runtime is polynomial, and is $O(X^{mk})$.

Optimality Theorem: The optimal off-line algorithm finds all the feasible competitive schedules.

Complexity Theorem: If we need X CPU units to service all the processes at the reference voltage, the runtime of the proposed algorithm is $O(X^{mk})$.

V. On-line Heuristics

In this section, we present the on-line algorithm for power minimization under the QoS constraints, synchronization and latency. We begin with the introduction of the overall flow for the creation of the on-line algorithm. We present the components of the on-line approach. Finally, we address the task and voltage selection process for the on-line algorithm.

We have multiple on-line streaming processes, with tasks which arrive at periodic time intervals. For each task of each process, we have memory and CPU requirements. Each of

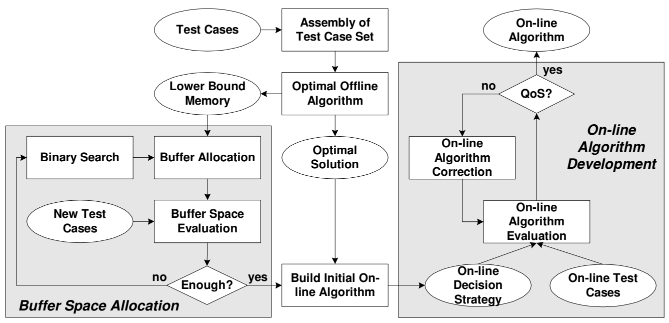
Test Cases
Assembly of Test Case Set
Lower Bound Memory
Optimal Offline Algorithm
Binary Search
Buffer Allocation
Optimal Solution
New Test Cases
Buffer Space Evaluation
no
Enough?
yes
Build Initial On-line Algorithm
Buffer Space Allocation
On-line Algorithm
yes
no
QoS?
On-line Algorithm Correction
On-line Algorithm Development
On-line Algorithm Evaluation
On-line Decision Strategy
On-line Test Cases

Fig. 1. Overall Flow for the creation of the on-line algorithm.

these tasks have a given latency constraint, and on some subset of these tasks additional synchronization constraints are imposed. We are given multiple supply voltage levels in which to execute these tasks. The goal of the on-line algorithm is to decide which task from the stream processes to execute at each time interval and at which voltage in such a way that all latency and synchronization constraints are satisfied. Additionally, at no point of time the requirements for storage should exceed the memory size (buffer space).

In order to solve the overall problem, we must answer the following three questions: (i) how much buffer space is needed, (ii) which task to execute and (iii) which voltage to apply. The answer to each of these questions is determined by our synthesis and on-line scheduling approach which is presented in Figure 1. The on-line approach uses the optimal off-line algorithm to determine its decision mechanism.

The on-line approach begins with the assembly of a diverse set of test cases. The test set should be as diverse as possible, while not excessively large. The off-line optimal algorithm provides a lower bound on the memory requirement for the system along with the optimal QoS solution for the test set. The lower bound memory requirement is used to determine the proper buffer allocation size for the on-line algorithm. In this phase, a binary search on the size of the buffer is conducted. Each iteration tests the new buffer size on a new set of test cases, until the buffer space allocated is large enough to handle all considered cases.

Next, the buffer size and the optimal solutions are used to build the on-line algorithm. The details of the creation of the initial algorithm are presented later in this section. The initial on-line algorithm builds a statistical model from the optimal off-line solutions and creates an on-line decision strategy, which is used in order to select the proper task and voltage in which to execute in each situation. The decision strategy is then evaluated on a set of on-line test cases. If the decision strategy does not provide the level of QoS specified, then modification of not only the statistical model and the decision strategy, but also the allocated buffer space is conducted. We continue to make modifications until the desired level QoS is reached. The final on-line algorithm consists of the buffer allocation size and the decision strategy.

The initial on-line algorithm is created using the pseudocode shown in Figure 2. In the first step, we identify the relevant properties for the QoS requirement. For example, in

```
Build On-Line Algorithm() {
1.    Identify Relevant Parameters;
2.    Evaluate Relevant Parameters;
3.    Build Statistical Model;
4.    Develop Decision Strategy;
5.    Validate Model;}
```

Fig. 2. Procedure for development of on-line algorithm.

the case of latency and synchronization, we define properties such as average latency, maximum synchronization delay, and buffer occupancy. We evaluate the relevance of these properties in terms of the off-line optimal algorithm. We eliminate all properties which show little relevance to the outcome of the optimal off-line solutions. Following this step, both the optimal off-line solutions and the relevant properties are used to build the statistical model. We build a nxm-dimensional space, where n is the number of properties and m is the number of processes. The resolution of each property is specified, and for each subspace we determine the statistical values for task selection. Each subspace contains the percentage of time the optimal off-line algorithm selected each of the tasks under the defined property conditions. In a similar way, the statistical values for all situations and each voltage level is calculated.

Before we evaluate the effectiveness of the model, we have to develop the on-line decision strategy. The strategy is responsible for making the decision as to which task and which voltage to select according to the particular combination of property values. This entire strategy is reliant on the context switch time or penalty. For each subspace, in the task selection statistical model, the decision strategy must decide with task to select based on the values in the subspace and the context switch penalty. If there was no penalty for context switching, then for each situation we would select the statistically strongest task from the statistical model. However, if the context switch penalty is extremely high, we would like to continue to run the tasks of the currently selected process as long as possible. In the moderate case, the proper time to switch between processes needs to be defined, and therefore we propose different points in the statistical model to switch between processes. The final step is to evaluate each of these proposed points to determine the proper switching point in the model. Once the proper switching points has been defined, we compact the nxm-dimensional table by combining subspaces with the same task/process selected to execute. Statistically, the subspaces should not be interleaved, and therefore we shall have continuous subspaces. For each decision the on-line algorithm determines which subspace the properties fall into, and select the assigned task/process to execute. The same process is applied to determining the voltage selection decision strategy. This on-line decision strategy is then passed on to the final stage of the overall on-line approach.

In order to better illustrate the algorithm and to make our ideas more tangible, we use a small example shown in Figure 3. For the sake of clarity and brevity, we consider only two processes, A and B, and two properties, latency of process A and synchronization lag between processes A and B. The lag is positive if process A is in front of process B in terms of its

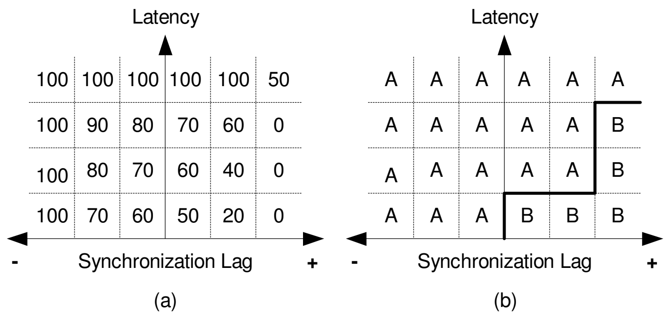

Fig. 3. Example of the statistical model and decision strategy of the on-line algorithm.

execution. Figure 3(a) shows numbers that can be obtained in principle by running the optimal off-line algorithm on a large set of examples. Each defined region of the space contains the percentage of cases in which the optimal off-line algorithm selected task/process A to execute. For example, the top left corner contains the value 100% and indicates that in all cases, process A was selected. The intuition is that process A has very high latency and is lagging behind synchronization with process B.

These numbers are used by the on-line algorithm development process to create the decision table shown in Figure 3(b). Note that the decision strategy must take into account the context switching penalty, therefore not making the mapping from Figure 3(a) to Figure 3(b) straightforward. Specifically, the value 40% from Figure 3(a) was mapped to the decision that process A should be executed in order to reduce context switching overhead.

For example, if the latency of task/process A is high and the synchronization of A is behind B, we should run task A with a likelihood of 100%. However, in the case of task A with high positive synchronization lag, the likelihood of running task A is very low, despite the level of latency of task A.

The on-line algorithm builds a statistical model and an on-line decision strategy based on the nxm-dimensional space defined by the properties. The goal is to select properties which provide strong indication of which task should be run at which voltage. We have defined the following five properties. For each property, we state the typical reasonings and examples of why that property should be included in the decision strategy of a high quality on-line algorithm.

Latency. If the latency of multiple tasks of a process are close to their maximum allowed latency, this process should be selected. Additionally, a higher voltage should be run to ensure each of the tasks meet their latency requirements. However, if the latency for all tasks/processes are at lower levels, then the task of the current process should be executed to eliminate a context switching penalty.

Relative Burstiness. The recent burstiness of a process, or rapid arrival of tasks for a process, can play an important role in voltage and task selection. If a task has shown recent burstiness, we should consider the execution of the task/process due to the likelihood that this task will continue to be bursty, therefore consuming more buffer space and extending the latency of each of the tasks if they are not run.

Number of Tasks. The number of tasks which a process has waiting also plays a key in the task and voltage selection process. If the current selected process has more tasks than the other processes, the tasks of the current process should continue to be selected. This eliminates context switching penalties which are not necessary. Low voltage should be applied if all the processes have very few tasks.

Synchronization. When the synchronization for any task/process is nearing the maximum allowed level for QoS this task should be selected. If the synchronization of the currently selected task/process is high, the algorithm should continue to run this process as long as possible to avoid context switching. For voltage selection, if the selected task is close to the maximum allowed synchronization level, a higher voltage should be applied.

Buffer Occupancy. Buffer occupancy is an indiction of the current demand of the processes as a whole. This property looks at the percentage of the entire buffer in which each process occupies. If the buffer is near capacity, the processes with higher buffer occupancy should be selected.

VI. Experimental Results

We used six streaming applications [9] to establish the effectiveness of the approach: IJG JPEG encoder and decoder, MSG MPEG encoder and decoder, CCITT G.721 encoder, and PGP encryption and description module. We use 4 CPU units for the latency constraints and for synchronization 8 CPU units. The goals of the our experimentation and results analysis was to answer the following questions: Are multiple voltages useful? How many voltages are needed? What is the relative quality of the on-line algorithm with comparison to the optimal off-line scheme? How much benefit one can obtain for on-line algorithms when the goal is to minimize design costs (specifically buffer storage size) under energy consumption constraints?

The first three questions are addressed using the experiments displayed in Table II. For different number of processes the table shows the normalized energy requirements when the optimal off-line and on-line algorithms are applied under the same memory requirement. All the values are normalized to the single voltage (3.3V) case found using the off-line algorithm. We use the off-line values as the lower bound. The table shows the results after applying the optimal off-line algorithm, the on-line algorithm and the percentage by which the off-line and the on-line algorithm differ for both the 2-voltage and 3-voltage cases. On average the on-line algorithm requires an overhead of 25%. However, it saves energy over the off-line single voltage case with 36.5% savings for 2-voltages, and 44.4% savings for 3-voltages. Therefore we see that significant savings in energy can be achieved by applying multiple voltages. However, since the savings for the 3-voltage systems over the 2-voltage systems is relatively low compared to the saving of the 2-voltage systems over the single voltage case, it is apparent that the usage of more than three voltages brings diminishing returns. Finally, we see that although the on-line algorithm is not capable of completely matching the performance of the off-line algorithm it nevertheless brings very significant improve-

TABLE II
ENERGY CONSUMPTION BY THE ON-LINE HEURISTICS

#	2-voltages (3.3V, 1.8V)			3-voltages (3.3V, 1.8V, 1.0V)		
Processes	off-line	on-line	%	off-line	on-line	%
2	0.647	0.714	10.4%	0.544	0.587	7.9%
3	0.582	0.669	14.9%	0.506	0.553	9.3%
4	0.524	0.648	23.7%	0.47	0.57	21.3%
5	0.471	0.655	39.1%	0.473	0.521	10.1%
6	0.443	0.64	44.5%	0.415	0.531	28.0%
8	0.428	0.573	33.9%	0.401	0.583	45.4%
10	0.421	0.552	31.1%	0.392	0.552	40.8%
Average	0.502	0.635	28.2%	0.457	0.556	23.3%
Median	0.471	0.648	31.1%	0.47	0.553	21.3%

TABLE III
MEMORY USED BY THE ON-LINE HEURISTICS

#	2-voltages (3.3V, 1.8V)			3-voltages (3.3V, 1.8V, 1.0V)		
Processes	off-line	on-line	%	off-line	on-line	%
2	0.351	0.392	11.7%	0.304	0.314	3.3%
3	0.342	0.359	5.0%	0.286	0.306	7.0%
4	0.322	0.347	7.8%	0.256	0.288	12.5%
5	0.301	0.32	6.3%	0.231	0.265	14.7%
6	0.278	0.315	13.3%	0.224	0.258	15.2%
8	0.289	0.303	4.8%	0.218	0.247	13.3%
10	0.254	0.283	11.4%	0.212	0.232	9.4%
Average	0.305	0.331	8.6%	0.247	0.272	10.8%
Median	0.301	0.32	7.8%	0.231	0.265	12.5%

ments over the single voltage case, and that this difference has only limited additional potential for further energy reduction.

Table III presents the results for the dual problem evaluated using Table II. Here we evaluate how much the cost of the system, measured in terms of buffer space, can be reduced under the conditions that energy consumption is fixed. All results are normalized against the base case where storage requirements are first calculated for the set of tasks in a particular row assuming that a single voltage is used. For the case when we use 2-voltages we compare to 2.5V, and in the case for 3-voltages we use 1.8V. Again, we present the normalized results for both the 2-voltage case, in columns 2 and 3, and the 3-voltage case in columns 5 and 6 of Table III. Additionally, we present the percentage difference between the optimal off-line memory requirement and the on-line algorithm for both the 2-voltage case and the 3-voltage case in columns 4 and 7 respectively. While again we see that the on-line algorithm is not able to completely match the performance of the optimal off-line algorithm, the reduction for storage requirements are significantly larger than the energy savings. This is the consequence of the fact that energy consumption is dictated by the overall average effectiveness of on-line and off-line algorithms, while the storage requirements are primarily a function of how well these algorithms can use high voltages to reduce storage requirements during bursty periods of processes.

VII. CONCLUSION

We have developed an on-line policy for power minimization of streaming media applications under QoS requirements and hardware constraints using multiple voltages. The approach leverages the insights from the newly developed fast off-line optimal algorithm for the same problem. By exploiting statistical information and the information about buffer occupancy, we developed an on-line algorithm which performs well in a variety of load scenarios. The algorithm is flexible in the sense that it can address a variety of dual-primal QoS problem formulations as well as a variety of QoS dimensions, such as latency and synchronization.

ACKNOWLEDGEMENTS

This material is based upon work partially supported by the National Science Foundation under Grant No. ANI-0085773 (and/or other proper agency and grant). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

REFERENCES

[1] C. Aurrecoechea, et al. "A survey of QoS architectures," *Multimedia Systems*, Vol.6, No.3, pp. 138-151, 1998.

[2] L. Benini, A. Bogliolo, G.A. Paleologo, G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on CAD*, Vol.18, No.6 , pp. 813-833, 1999.

[3] A. Borodin, R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.

[4] A.P. Chandrakasan, S. Sheng, R.W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, Vol.27, No.4, pp. 473-484, 1992.

[5] J. Chang, M. Pedram, "Energy minimization using multiple supply voltages," *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 157-162, 1996.

[6] R.L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks," *IEEE Journal on Selected Areas in Communications*, Vol.13, No.6, pp. 1048-1056, 1995.

[7] M. A. Stephens, *Goodness-of-fit techniques*, New York : M. Dekker, 1986.

[8] B. Efron, R.J. Tibshirani, *An introduction to the bootstra*, Chapman & Hall, 1993.

[9] C. Lee, et al, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," *International Symposium on Microarchitecture*, pp. 330-335, 1997.

[10] Q. Qiu, M. Pedram, "Dynamic power management based on continuous-time markov decision processes," *IEEE/ACM Design Automation Conference*, pp. 555-561, 1999.

[11] J. M. Rabaey, M. Pedram, *Low power design methodologies*, Kluwer Academic Publishers, 1996.

[12] S. Raje, M. Sarrafzadeh, "Scheduling with multiple voltages," *Integration, The VLSI Journal*, Vol.23, No.1, pp. 37-59, 1997.

[13] R. Rajkumar, C. Lee, J. Lehoczky, D. Siewiorek, "A resource allocation model for QoS management," *IEEE Real-Time Systems Symposium*, pp. 298-307, 1997.

[14] R. A. Thisted, *Elements of statistical computing*, New York : Chapman and Hall, 1988.

[15] M. Weiser, B. Welch, A. Demers, S. Shenker, "Scheduling for reduced CPU energy," *USENIX Operating Systems Design and Implementation (OSDI)*, pp. 13-23, 1994.

[16] J. L. Wong, G. Qu, and M. Potkonjak, "Power minimization under QoS constraints," *International Packetvideo Workshop*, 2002.