Energy Reduction Techniques for Multimedia Applications with Tolerance to Deadline Misses *

Shaoxiong Hua, Gang Qu, and Shuvra S. Bhattacharyya
Electrical and Computer Engineering Department and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA
{shua, gangqu, ssb}@eng.umd.edu

ABSTRACT

Many embedded systems such as PDAs require processing of the given applications with rigid power budget. However, they are able to tolerate occasional failures due to the imperfect human visual/auditory systems. The problem we address in this paper is how to utilize such tolerance to reduce multimedia system's energy consumption for providing guaranteed quality of service at the user level in terms of completion ratio. We explore a range of offline and on-line strategies that take this tolerance into account in conjunction with the modest non-determinism in application's execution time. First, we give a simple best-effort approach that achieves the maximum completion ratio; then we propose an enhanced on-line best-effort energy minimization (BEEM) approach and a hybrid offline/on-line minimum-effort (O^2ME) approach. We prove that BEEM maintains the maximum completion ratio while consuming the provably least amount of energy and O^2ME guarantees the required completion ratio statistically. We apply both approaches to a variety of benchmark task graphs, most from popular DSP applications. Simulation results show that significant energy savings (38% for BEEM and 54% for O^2ME, both over the simple best-effort approach) can be achieved while meeting the required completion ratio requirements.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: Computer-Aided Design; I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling

General Terms

algorithms, performance, design

Keywords

soft real-time scheduling, low-power design, voltage scaling, on-line algorithm

*A portion of this work is supported by the National Science Foundation under Grant No. 9734275.

DAC 2003, June 2–6, 2003, Anaheim, California, USA.

1. INTRODUCTION

Low energy consumption has emerged as one of the most important design objectives for many modern embedded systems particularly the battery-operated PDAs. These systems require the processing of signal, image, and video data streams in a timely fashion to the end user's satisfaction. Such applications are often characterized by 1) the repetitive processing on periodically arriving inputs, such as voice samples or video frames; 2) the uncertain execution time of each input processing; and 3) the tolerance to occasional deadline misses without being noticed by human visual and auditory systems. The deadline can be (implicitly) determined by the throughput requirement of the input data streams. For example, a 1%-10% loss rate can be tolerated in packet audio applications[4], while tolerance for losses in low bit-rate voice applications may be significantly lower[11].

We can exploit these characteristics, particularly the tolerance to deadline misses, to create slacks when streamlining the embedded processing associated with such applications. More specifically, when the embedded processing does not interact with a lossy communication channel, or when the channel quality is high compared to the tolerable rate of missed deadlines, we are presented with the opportunity of dropping some tasks to introduce slacks in the application for reduced cost or power consumption.

Note that the slacks here are fundamentally different from the ones that have been exploited for power/latency minimization. Previously, designs based on worst case execution time (WCET) guarantee the perfect completion ratio. Slack occurs, out of designer's control, when real-time execution time is less than WCET. We create slack *intentionally* (e.g., by dropping samples) in order to minimize system's energy consumption. This becomes possible in multimedia applications that do not require the completion of each data sample. However, identifying the best trade-off between performance and power is hard and requires intensive design space explorations[8, 12].

For many DSP applications, task's execution information beyond WCET is also available. Examples include the execution times in the best case, with cache miss(es), when pipeline stalls, or when different conditional branch happens. More important, it is possible to obtain the probabilities at which these events occur by knowing (e.g., by sampling technique) detailed timing information about the system or by simulation on the target hardware[18]. Given such information, on-line and offline schedulers could make better decisions towards more energy savings.

In this paper, we develop systematic techniques to lever-

age the information on application's deadline miss tolerance and statistical task execution time to perform efficient dynamic voltage scaling for energy reduction. This departs us from the conservative view of over-designing embedded systems in order to meet deadlines all the time and under WCET. Our result is an algorithmic framework that integrates considerations of iterative on-line/offline task scheduling, dynamic voltage scaling, non-deterministic computation times, and quality of service considerations (in terms of deadline miss rate or task completion ratio), and provides for robust and power-optimized execution of multimedia applications. The highlights of our work are two scheduling algorithms: **BEEM** that achieves the highest completion ratio with the provably minimum energy consumption; and **O^{2}ME** that can further reduce energy consumption by taking advantage of the tolerance to deadline misses.

2. RELATED WORK

Although leakage power dissipation is gaining more and more attention recently, dynamic power still dominates in most embedded systems. Power is proportional to the square of the supply voltage, therefore, reducing supply voltage can result in great power saving. Tiwari et al. analyzed the power dissipation in high-performance CPU, discussed various power reduction techniques, and concluded that voltage scaling is the most effective and most promising technique[19]. Early research on voltage scaling is on systems that have multiple supply voltages available at the same time. This is due to various design and implementation concerns of having voltage changing arbitrarily at run-time. However, such ideal variable voltage systems can be more energy-efficient because of the flexibility of choosing proper operating voltages[14]. Hong et al. developed a design methodology for low power core-based system-on-chip using hardware that is capable of changing voltages dynamically[7]. Burd et al. reported the implementation of microprocessor that can adjust its operating speed dynamically[5].

Scheduling techniques have been used at many design levels for the purpose of power reduction. Raje and Sarrafzadeh showed, at behavioral level, how to apply multiple voltages to function units to reduce power while satisfying the timing constraints[16]. Johnson and Roy designed an algorithm that determines the optimal voltages along the datapaths[10]. At system level, task scheduling techniques have been exploited to process real-time applications with hard deadlines. Shin and Choi used fixed priority scheduling method to achieve power reduction by exploiting slack times in real-time systems[17]. Chen and Sarrafzadeh developed a provably good lower bound algorithm based on maximal-weighted-independent-set for the power consumption of dual supply voltage systems[6]. More recently, Quan and Hu proposed a fixed-priority scheduling policy for real-time systems[15].

The above scheduling schemes are designed to meet application's deadline at all time. In real-time system society, there exist several approaches on how to maximize completion ratio or system utility when the system is overloaded. For example, Baruah et al. showed that in general any online scheduling algorithm can perform arbitrarily worse than an offline algorithm[3]. Aydin et al. studied the problem of how to maximize system's reward where they proved that the problem is NP-hard for convex reward functions and gave an optimal algorithm for concave reward functions[2].

The unique feature in the embedded multimedia system we address here is its tolerance to certain level of deadline misses. We propose scheduling algorithms that seek to minimize system's energy consumption under the constraint of user-level quality of service in terms of completion ratio.

3. PROBLEM FORMULATION

We consider the *task graph* $G = (V, E)$ for a given application. Each vertex in the graph represents one computation and directed edges represent the data dependencies between vertices. For each vertex v_i, we associate it with a finite set of possible execution time $\{t_{i1}, t_{i2}, \cdots, t_{ik_i}\}$ and the set of probabilities $\{p_{i1}, p_{i2}, \cdots, p_{ik_i} | \sum_{l=1}^{k_i} p_{il} = 1\}$ that such execution time will occur at run-time. That is, with probability p_{ij}, vertex v_i requires an execution time of t_{ij}. Assuming that $t_{i1} < t_{i2} < \cdots < t_{ik_i}$, we define $P_{il} = \sum_{j=1}^{l} p_{ij}$, which measures the probability that the computation at vertex v_i is no longer than t_{il}. The *completion time* of the task graph G under a fixed execution order $< v_1 v_2 \cdots v_n >$, is the sum of each vertex's run-time execution time e_i: $C(< v_1 v_2 \cdots v_n >) = \sum_{i=1}^{n} e_i$.

The *deadline* constraint $\mathcal{M}$ specifies the maximum time allowed to complete the application, which is executed periodically with its deadline $\mathcal{M}$ as the period. We say that an iteration is *successfully completed* if $C(< v_1 v_2 \cdots v_n >) \leq \mathcal{M}$. Closely related to $\mathcal{M}$ is a real-valued completion ratio constraint $\mathcal{Q} \in [0, 1]$, which gives the minimum completion ratio that the system has to maintain over a sufficiently large number of iterations. Alternatively, $\mathcal{Q}$ can be interpreted as a guarantee on the probability with which an arbitrary iteration can be successfully completed.

Finally, we assume that there exist multiple levels of supply voltage and corresponding threshold voltages on the system. For the simplicity of our discussion, we further assume that there is no energy/delay overhead for the system to switch its operating voltage from one level to another. To consider the energy and delay overhead for voltage switching, note that we will use no more than two different voltages for the execution of a vertex[9]. One can easily compare the energy saving from multiple voltages and the energy/delay overhead to decide whether multiple voltages should be used for each vertex. We consider the following problem:

> *For a given task graph, its deadline, and its completion ratio constraint, find a scheduling strategy for a multiple voltage system such that the energy consumed to satisfy the completion ratio constraint is minimized.*

The scheduling strategy consists of determining the execution order of vertices in the given task graph and selecting the supply voltages for the execution of each vertex. It is well-known that the problem of variable voltage task scheduling for low power is in general NP-hard[14].Our focus will be on developing offline/on-line voltage scaling algorithms to determine *when* and *at which voltage* should each task be executed in order to minimize the total energy consumption while meeting the completion ratio requirement.

4. ENERGY-DRIVEN VOLTAGE SCALING TECHNIQUES

In this section, we present three algorithms: the first one achieves the highest possible completion ratio, but may con-

sume more energy than necessary; the second one maintains this same completion ratio with the provably minimum energy consumption; the last one minimizes energy by dropping some completable tasks to avoid achieving a completion ratio higher than required.

4.1 A Naïve Best-Effort Approach

In this approach, the processor will keep on executing tasks, at the highest voltage, to the completion of current iteration or when deadline $\mathcal{M}$ is reached. In the latter, if there is any task remains unfinished, we say the current iteration is *failed*; otherwise, we have a *successful completion* or simply *completion*.

An iteration can only be processed before its deadline and the processor cannot start an iteration early[1]. The naïve best-effort approach operates the processor at the highest possible voltage, which transfers to the highest processing speed and therefore the highest possible completion ratio. Suppose that it gives us k completions over $N >> 1$ iterations, then $\frac{k}{N}$ is the highest achievable completion ratio. Since this naïve approach will not drop any iteration until the deadline $\mathcal{M}$ is reached, we know the execution time for all failed iteration will be $\mathcal{M}$. Let $\mathcal{C}_i$ be the completion time of the i-th completion, we can calculate the energy consumption for these N iterations by:

$$E = P_{ref} \cdot \left(\sum_{i=1}^{k} \mathcal{C}_i + \sum_{j=1}^{N-k} \mathcal{M} \right) \quad (1)$$

where P_{ref} is the power dissipation at the reference voltage.

The maximum completion ratio achieved by the naïve approach and its energy consumption provide us a guideline to evaluate other approaches in trading completions for energy.

4.2 On-line Best-Effort Energy Minimization

Consider the following two occasions when we could save energy over the naïve approach: first, if a completion occurs earlier than the deadline $\mathcal{M}$, we could have operated the processor at a lower voltage; second, if an iteration eventually fails at $\mathcal{M}$, we could have terminated it earlier. Apparently, we want to slow down the processing speed as much as possible and terminate as early as possible to save energy. We now discuss how to achieve the maximum energy saving without sacrificing the completion ratio provided by the naïve approach.

For a given task graph with execution order (e.g., any topological order) $v_1, v_2, \cdots, v_n$, we define vertex v_i's *latest completion time* T_{l_i} and *earliest completion time* T_{e_i}:

$$T_{l_n} = T_{e_n} = \mathcal{M} \quad (2)$$

$$T_{l_i} = T_{l_{i+1}} - t_{i+1,1} \quad (3)$$

$$T_{e_i} = T_{e_{i+1}} - t_{i+1,k_{i+1}} \quad (4)$$

where $t_{i+1,1}$ and $t_{i+1,k_{i+1}}$ are the BCET (best case execution time) and WCET of vertex v_i.

T_{l_i} represents the latest time, or a *hard deadline* to complete vertex v_i to guarantee a completion with a non-zero probability. That is, if its completion time $t > T_{l_i}$, then the current iteration can never be completed before the deadline $\mathcal{M}$. To see this, let us consider the best case when all the successors of v_i have their BCET. The completion time in this case will be:

$$\begin{aligned} t + \sum_{q=i+1}^{n} t_{q,1} &> T_{l_i} + t_{i+1,1} + \sum_{q=i+2}^{n} t_{q,1} \\ &= T_{l_{i+1}} + t_{i+2,1} + \sum_{q=i+3}^{n} t_{q,1} \\ &= \cdots \\ &= T_{l_{n-1}} + t_{n,1} = T_{l_n} = \mathcal{M} \end{aligned}$$

Similarly, we can show that any completion time $t < T_{e_i}$ will result in a completion earlier than deadline $\mathcal{M}$, which is not energy-efficient. The operating voltage for v_i can be reduced as long as v_i's completion is not after T_{e_i}, the *soft deadline*.

```
1. current time t = 0;
2. for each vertex v_i  (i = 1, 2, ··· , n)
3.   t_ij = the execution time of v_i;
4.   if (t + t_ij > T_{l_i})
5.     terminate the current iteration;
6.   if (t + t_ij < T_{e_i})
7.     scale voltage so v_i is finished at t = T_{e_i};
8.   else
9.     run v_i at the reference voltage to finish v_i at t = t + t_ij;
```

Figure 1: Best-effort energy-minimization scheduling algorithm.

Figure 1 depicts the proposed on-line best-effort energy-minimization (BEEM) scheduling algorithm after computing the soft/hard deadline pair $\{T_{e_i}, T_{l_i}\}$ for each vertex. The scheduler scales the operating voltage at run-time when the execution time of each vertex is available in order to save energy with a guaranteed best completion ratio. More specifically, we have

Theorem 1. BEEM guarantees the highest completion ratio with the minimum energy consumption.

4.3 Offline/On-line Minimum-Effort Approach

Both the naïve approach and BEEM achieve the best possible completion ratio. If the application requires lower completion ratio, a simple way to conserve energy is to shut the system down once sufficient iterations have been completed[2]. The proposed offline/on-line minimum-effort (O^2ME) algorithm leverages the available task execution time information beyond BCET and WCET to further reduce energy consumption. It consists of an offline execution time allocation phase, where each vertex v_i is assigned a pair (S_i, T_i), and an on-line voltage scaling phase as shown in Figure 2.

Let $E_{ij}(S_i)$ be the minimum energy to complete the workload t_{ij} required by vertex v_i in time S_i. On an ideal variable voltage processor, $E_{ij}(S_i)$ is the energy consumed by running at voltage V_{ideal} throughout the entire assigned slot S_i,

[1] A common reason for this is that the starting time of an iteration, which is the deadline for the previous iteration, corresponds to the arrival times of successive data samples in the input data stream.

[2] One drawback of this scheme is that there will not be any completion after the system is shut down. However, the deadline misses will be tolerant only when they are evenly distributed so human will not notice. Therefore, a practical way is to divide the entire N iterations into smaller groups, shut down the system once we have completed sufficiently many iterations in a group, and restart executing for the next group.

```
/*offline part */
1. for each vertex v_i, l_i = k_i        /* assign WCET to v_i*/
2. Q = 1;
3. while (Q > Q_0)
4. { pick v_j that has the maximum (t_{jl_j} - t_{j(l_j-1)}) / P_{jl_j} · P_{j(l_j-1)};
5.    Q = Q · P_{j(l_j-1)} / P_{jl_j};
6.    if (Q > Q_0)
7.       l_j = l_j - 1;
8. }
9. C = Σ t_{il_i};          /* calculate the completion time C */
10. if (C > M) exit;                /* Q_0 cannot be met */
11. for each vertex v_i
12.    T_i = t_{il_i}; S_i = t_{il_i} · M/C;
/* on-line part */
13. current time t = 0;
14. for each vertex v_i
15.    t_ij = the execution time of v_i;
16.    if (t_ij > T_i)    terminate the current iteration;
17.    else    scale voltage so v_i is completed at t = t + S_i;
```

Figure 2: Offline/on-line minimum-effort approach.

where V_{ideal} is the voltage level that enables the processor to accumulate the workload t_{ij} at the end of the slot[14]. On a multiple voltage processor with only a finite set of voltage levels $V_1 < V_2 < \cdots$, $E_{ij}(S_i)$ is the energy consumed by running at V_k for a certain amount of time and then switching to the next higher level V_{k+1} to complete t_{ij}, where $V_k < v_{ideal} < V_{k+1}$ and the switching point can be conveniently calculated from S_i and t_{ij}[9].

Recall that p_{ij} is the probability that vertex v_i requires execution time t_{ij} and $P_{il} = \sum_{j=1}^{l} p_{ij}$ is the probability that v_i can be completed within time t_{il}. The completion of an iteration requires the completion of all the vertices. Therefore, if we execute vertex v_i only when it has a real execution time less than or equal to T_i, then the probability that an iteration can be completed is

$$\mathcal{Q} = \prod_{i=1}^{n} P_{il_i} = \prod_{i=1}^{n} \sum_{j=1}^{l_i} p_{ij} \tag{5}$$

where l_i satisfies $t_{il_i} \leq T_i < t_{il_i+1}$. Further, if we assign $S_i (\geq T_i)$ CPU units to v_i, expected energy consumption for a completion is:

$$E_{completion} = \sum_{i=1}^{n} \sum_{j=1}^{l_i} p_{ij} \cdot E_{ij}(S_i) \tag{6}$$

An early termination at v_q implies a failed iteration with the successful execution at vertices $v_1, v_2, \cdots, v_{q-1}$. These "wasted" computation consumes energy in the amount of

$$E_{failure}(v_q) = (1 - P_{ql_q}) \sum_{i=1}^{q-1} \sum_{j=1}^{l_i} p_{ij} \cdot E_{ij}(S_i) \tag{7}$$

Hence, the expected energy consumption for an iteration is

$$E(\{(S_1, T_1), \cdots, (S_n, T_n)\}) = E_{completion} + \sum_{q=2}^{n} E_{failure}(v_q)$$

The goal of the offline part in the O^{2}ME approach is to *minimize* $E(\{(S_1, T_1), \cdots, (S_n, T_n)\})$ *by finding pairs* (S_i, T_i) *such that* $T_i \leq S_i$, $\sum_{i=1}^{n} S_i \leq \mathcal{M}$, *and* $\mathcal{Q} \geq \mathcal{Q}_0$. The first inequality guarantees that the task is completable at step 17; the second one requires total allocated CPU units not to exceed the deadline $\mathcal{M}$; and the last one enforces that the required completion ratio $\mathcal{Q}_0$ is met. Obviously, energy will be minimized only when *minimum effort* is paid. That is, the scheduler should not attempt to achieve completion ratio higher than the required $\mathcal{Q}_0$. We develop a heuristic algorithm to solve this offline problem as shown in the first part of Figure 2.

O^{2}ME distinguishes itself from the best-effort approaches in the following: 1) each vertex has a fixed execution slot while execution slot in the best-effort approach is determined by the on-line scheduler at run time. This makes O^{2}ME simple to be implemented into the RTOS. 2) O^{2}ME approach takes into account both the statistical execution time information and the completion ratio requirement. This might give us additional energy savings over the best-effort approaches. Unfortunately, it is hard to get any analytic results and we use experimental results to show O^{2}ME's energy efficiency in the next section. We summarize our study on O^{2}ME by the following

Theorem 2. O^{2}ME approach provides a statistical guarantee to the completion ratio requirement with the potential of further energy reduction over best-effort approaches.

5. EXPERIMENTAL RESULTS

We have implemented the proposed scheduling techniques and validated their energy efficiency and the guarantee to the required completion ratio over a variety of real-life benchmark task graphs extracted from popular DSP applications. They include Fast Fourier Transform of two different implementations (FFT1 and FFT2); Laplace transform (Laplace); a quadrature mirror filter bank (qmf4); the Karplus-Strong music synthesis algorithm with 10 voices (karp10); a measurement application (meas); an upside down binary tree representing the sum of products computation (sum1); and others reported in early literatures[1, 13, 20, 21].

For each task graph, we simulate 100,000 iterations. We assign each vertex three different execution times $e_0 < e_1 < e_2$ with probabilities $p_0 >> p_1 > p_2$, respectively. One can treat e_0 as the BCET, which happens much more often than other cases for most of DSP applications. We consider a processor with four different voltages ranging from 3.3V to 1.2V and neglect the overhead of voltage switching as we have discussed earlier in Section 3. The deadline $\mathcal{M}$ varies from the sum of BCET to the sum of WCET, and the required completion ratio $\mathcal{Q}$ varies from 1.0 to 0.4 with a step of 0.05. For each pair of $(\mathcal{M}, \mathcal{Q})$, we simulate the execution of each application with the naïve, BEEM, and O^{2}ME approaches respectively and track the completion ratio and energy consumption. We now report a few representative sets of results when the four voltages are set at 3.3V, 2.6V, 1.9V, and 1.2V.

We first consider the energy consumption by different approaches with the same deadline and completion ratio constraint. Table 1 gives the number of vertices in each benchmark [1, 13, 20, 21] and reports the results from three proposed energy minimization techniques. The deadlines are set to be around three times of the sum of BCET. The completion ratio requirement is fixed at 0.800. As we have mentioned earlier, both the naïve best-effort approach and BEEM take 100 iterations as a group and stop execution

				naïve	BEEM		O^2ME			
	n	$\mathcal{M}$	$\mathcal{Q}_0$	energy	energy	saving over naïve	energy	saving over naïve	saving over BEEM	$\mathcal{Q}$
FFT1	28	2280	0.800	699.69	539.02	22.96%	371.00	46.98%	31.17%	0.8043
FFT2	28	1440	0.800	441.91	259.08	41.37%	199.14	54.94%	23.14%	0.8043
Laplace	16	4320	0.800	1322.2	724.39	45.21%	571.42	56.78%	21.11%	0.8222
qmf4	14	480	0.800	147.65	72.84	50.67%	67.64	54.19%	7.13%	0.832
karp10	21	1782	0.800	547.03	382.83	30.02%	264.57	51.64%	30.89%	0.8229
meas	12	849	0.800	260.10	145.82	43.94%	115.37	55.64%	20.88%	0.8026
sum1	15	261	0.800	80.30	57.08	28.9%	39.60	50.59%	30.63%	0.8192
almu	17	216	0.800	66.03	35.89	45.64%	28.60	56.68%	20.31%	0.8132
DSC-7-7	7	39	0.800	12.18	8.19	32.77%	5.32	56.29%	34.99%	0.809
DSC-7-8	7	39	0.800	12.18	7.65	37.13%	5.18	57.43%	32.28%	0.809
average saving				/	/	37.86%	/	54.13%	25.25%	/

Table 1: Average energy consumption per iteration for naïve, BEEM, and O^2ME approaches to achieve $\mathcal{Q}_0 = 0.800$ with deadline constraints $\mathcal{M}$. (n: number of vertices in the task graph; $\mathcal{Q}$: the actual completion ratio achieved by O^2ME without forcing the process stop at $\mathcal{Q}_0$; energy is in the unit of the dissipation in one CPU unit at the reference voltage.)

once 80 iterations in the same group have been completed. BEEM provides the same completion ratio with an average of nearly **38%** energy saving. The O^2ME approach, as predicted in Theorem 2, is more energy efficient. We actually observe energy reduction of **54%** over the naïve approach and **25%** over BEEM by O^2ME. Although O^2ME does not provide any absolute guarantee on completion ratio, the last column shows that the required 0.800 completion ratio is achieved in all the cases. Note that the O^2ME approach does not count the number of completion but drops iterations if certain vertex's execution time exceeds a given amount (T_i in Figure 2). Although it may complete more iteration than necessary, we see it still manage to consume significantly less energy than BEEM. Finally, a small amount of extra energy (1.8% on average over O^2ME) could be saved if we count the number of completions by O^2ME and stop its execution at the desired completion ratio $\mathcal{Q}_0$.

Secondly, we consider the impact of different completion ratio requirements to energy consumption. Both best-effort approaches have the counting mechanism (to count the number of completions), so their energy consumption should decrease linearly with the completion ratio requirement. Figure 3 depicts this effect for the FFT1 benchmark with a fixed deadline around three times of the sum of BCET. In this case, the best achievable completion ratio by O^2ME is slightly larger than 0.85. We see that different completion ratio has much less impact on O^2ME's energy consumption, which consists of two parts: the portion on complete iterations and the portion on failed iterations as given in Equations (6) and (7). When the completion ratio requirement decreases, the first part decreases but the second part increases because more and more iterations are intentionally dropped. This becomes clear when the completion ratio is low (around 0.43), O^2ME consumes even more energy than BEEM due to the "wasted energy" on failed iterations. On the other end, both best-effort approaches are capable of reaching completion ratio very close to 1 (perfection), which O^2ME cannot achieve. This limitation of O^2ME is caused by its offline execution slot reallocation. When completion ratio requirement is high, all vertices will compete for execution slots and O^2ME cannot meet the deadline (step 10 in Figure 2). We conclude that O^2ME outperforms best-effort approaches when the completion ratio is moderately high.

Finally, we discuss deadline's impact. The naïve approach operates at the highest voltage till the required $\mathcal{Q}_0$ is reached. Therefore, its energy consumption remains constant regardless of the deadline. In BEEM, the hard/soft deadline pair (Equations 2-4) is directly related to the iteration deadline. Therefore, its energy consumption is highly dependent on the deadline. Similar to the above, deadline variations have positive impact to O^2ME, but not as dramatic as it does to BEEM. All these have been verified by simulation as shown in Figure 4 on the same FFT1 example.

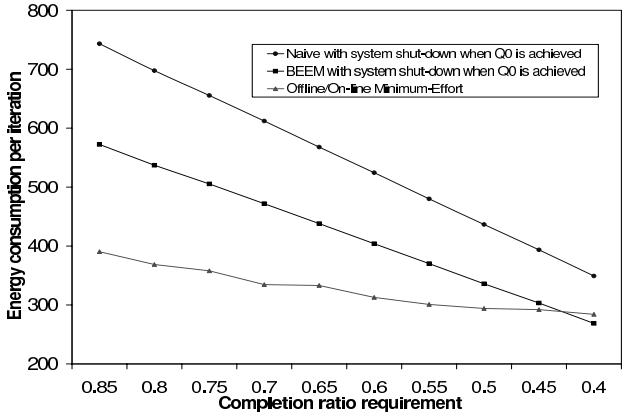

Figure 3: Completion ratio requirement's impact to the average energy consumption per iteration on benchmark FFT1.

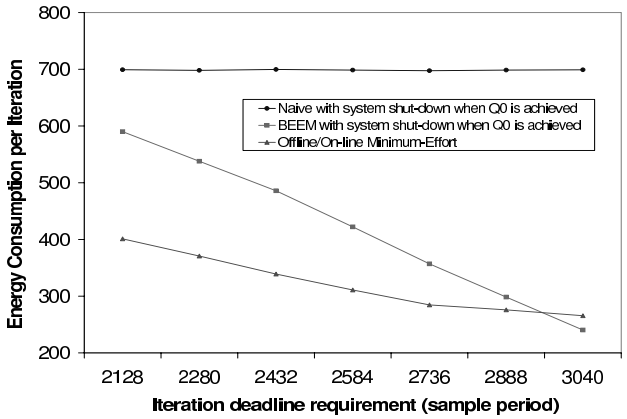

Figure 4: Deadline requirement's impact to the average energy consumption per iteration on benchmark FFT1.

6. CONCLUSIONS AND FUTURE WORK

This paper addresses on-line and offline voltage scaling techniques that incorporate completion ratio constraints and uncertainties in task execution times. Our approaches are based on the facts that many applications, particularly DSP applications, require repetitive execution on the stream of data samples and are able to tolerate certain amount of deadline misses. We consider the problem of how to convert such tolerance to energy saving.

We first give a naïve best-effort approach that provably optimizes completion ratio (or equivalently minimizes deadline misses), but does not involve any energy minimization process. We then propose a best-effort energy minimization (BEEM) approach that achieves the same completion ratio with the provably minimum energy consumption by considering execution time bounds. The last technique is a hybrid offline/on-line minimum-effort (O^2ME) approach that statistically provides the required completion ratio with reduced energy consumption. Our simulation over popular DSP applications validates the energy-efficiency of these proposed approaches. BEEM saves 38% energy over the naïve approach without missing any additional deadlines, and O^2ME can save 25% over BEEM if the maximum completion ratio is not required. BEEM's energy reduction is greatly affected by, roughly proportional to, the completion ratio and deadline. O^2ME achieves more energy saving under tight deadline and high completion ratio constraint, and is less sensitive to the change to these requirements.

Our approaches are based on a given execution order, it will be interesting to investigate different task ordering's impact to energy reduction. Although the voltage switching overhead is neglected in the paper, one can model such overhead and easily integrate it into the proposed techniques. Finally, it is our goal to build prototype multimedia systems, measure the overall energy consumption, and evaluate the quality of service provided at user level.

7. REFERENCES

[1] M.A. Al-Mouhamed. "Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs with Communication Costs", *IEEE Transaction on Software Engineering*, Vol.16, No. 12, 1990

[2] H. Aydin, F. Melhem, D. Mosse, and P.M. Alvanez. "Optimal Reward-Based Scheduling for Periodic Real-Time Tasks," *IEEE Transactions on Computer*, Vol. 50, No. 2, pp. 111-130, 2001.

[3] S. Baruah, J. Haritsa, and N. Sharma. "On-line Scheduling to Maximize Task Completions." *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 39, pp. 65-78, 2001.

[4] J. Bolot and A. Vega-Garcia. "Control Mechanisms for Packet Audio in the Internet", *Proceedings of IEEE Infocom*, 1996.

[5] T.D. Burd, T. Pering, A. Stratakos, and R. Brodersen. "A Dynamic Voltage-Scaled Microprocessor System," *IEEE International Solid-State Circuits Conference,* pp. 294-295, 466, 2000.

[6] C. Chen and M. Sarrafzadeh. "Probably Good Algorithm for Low Power Consumption with Dual Supply Voltages", *IEEE/ACM International Conference on Computer Aided Design*, pp. 76-79, 1999.

[7] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.B. Srivastava. "Power Minimization of Variable Voltage Core-Based Systems", *35th ACM/IEEE Design Automation Conference*, pp. 176-181, 1998.

[8] H. Hsieh, F. Balarin, L. Lavagno, A.L. Sangiovanni-Vincentelli. "Efficient methods for embedded system design space exploration". *37th ACM/IEEE Design Automation Conference*, pp. 607-612, 2000.

[9] T. Ishihara and H. Yasuura. "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," *International Symposium on Low Power Electronics and Design,* pp. 197-202, 1998.

[10] M.C. Johnson and K. Roy. "Scheduling and Optimal Voltage Selection for Low Power Multi-Voltage DSP Datapaths", *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, pp. 2152-2155, 1997.

[11] M. J. Karam and F. A. Tobagi. "Analysis of the Delay and Jitter of Voice Traffic Over the Internet", *Proceedings of IEEE Infocom*, 2001.

[12] R. Marculescu, A. Nandi, L. Lavagno, A.L. Sangiovanni-Vincentelli. "System-Level Power/Performance Analysis of Portable Multimedia Systems Communicating over Wireless Channels". *IEEE/ACM International Conference on Computer-Aided Design*, pp. 207-214, 2001.

[13] C.L. McCreary, et al. "A Comparison of Heuristics for Scheduling DAGs on Multiprocessors", *Proceedings of the International Parallel Processing Symposium*, 1994.

[14] G. Qu. "What is the Limit of Energy Saving by Dynamic Voltage Scaling?" *IEEE/ACM International Conference on Computer-Aided Design*, pp. 560-563, 2001.

[15] G. Quan and X. Hu. "Energy Efficient Fixed-Priority Scheduling for Real-Time Systems on Variable Voltage Processors," *38th IEEE/ACM Design Automation Conference,* pp. 828-833, 2001.

[16] S. Raje and M. Sarrafzadeh. "Variable Voltage Scheduling", *International Symposium on Low Power Electronics and Design*, pp. 9-14, 1995.

[17] Y. Shin and K. Choi. "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems", *36th ACM/IEEE Design Automation Conference*, pp. 134-139, 1999.

[18] T.S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.W.-S. Liu. "Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times", *Proc. Real-Time Technology and Applications Symposium*, pp. 164-173, 1995.

[19] V. Tiwari et al. "Reducing Power in High-Performance Microprocessors", *35th ACM/IEEE Design Automation Conference*, pp. 732-737, 1998.

[20] M. Wu and D. D. Gajski. "Hypertool: A Programming Aid for Message-Passing Systems", *IEEE Tran. on Parallel and Distributed Systems*, Vol. 1, No. 3, 1990.

[21] T. Yang and A. Gerasoulis. "DSC: Scheduling parallel tasks on an unbounded number of processors", *IEEE Tran. on Parallel and Distributed Systems*, Vol. 2, pp. 951-967, 1994