Exploring the Probabilistic Design Space of Multimedia Systems

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

In this paper, we propose the novel concept of probabilistic design for multimedia systems and a methodology to quickly explore such design space at an early design stage. The probabilistic design is motivated by the challenge of how to design, but not over-design, multimedia embedded systems while systematically incorporating such application's performance requirements, uncertainties in execution time, and tolerance for reasonable execution failures. Our goal is to bridge the gap between real-time analysis and embedded software implementation for rapid and economic (multimedia) system prototyping. Our method takes advantage of multimedia system's unique features mentioned above to relax the rigid hardware requirements for software implementation and eventually avoid over-designing the system.

1. Introduction

Real-time multimedia embedded systems including battery-operated PDAs etc., have become more and more popular in our life. 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 the repetitive processing on periodically arriving inputs, such as voice samples or video frames, and the tolerance to occasional deadline (determined by the throughput requirement of the input data streams) misses without being noticed by human visual and auditory systems. For example, in packet audio applications, loss rates between 1% - 10% can be tolerated [2]. Furthermore, in many multimedia DSP applications, although the execution time of a task can vary dramatically due to a number of factors such as cache miss(es) or conditional branches, it is possible to obtain the execution time distribution for each task by knowing (e.g., by sampling technique) detailed timing information about the system or by profiling the target hardware [15].

Prior design space exploration methods for hardware-software codesign of embedded systems, e.g., [4, 6, 12], guarantee no deadline missing by considering worst case execution time (WCET) of each task. As the multimedia systems can tolerate some violations of timing constraints, these methods will often lead to over-designed systems that deliver higher performance than necessary at the cost of expensive hardware, higher energy consumption, and other system resources.

There are plenty of studies on the estimation of soft real-time system's probabilistic performance when the application's computation time can be varied [7, 11, 15]. However, their goals are to improve system's performance or to provide probabilistic performance guarantees. Our recent work discusses offline and on-line scheduling strategies to reduce system's energy consumption by taking advantage of multimedia application's tolerance to deadline misses [8]. To the best of our knowledge, there is no reported effort on systematically incorporating application's performance requirements, uncertainties in execution time, and tolerance for reasonable execution failures to guide rapid and economic prototyping of real-time embedded systems.

In this paper, we study the problem of how to integrate such tolerance to deadline misses into the design of multimedia systems. We propose the novel concept of probabilistic design for multimedia systems and a methodology to quickly explore such design space at an early design stage. Given the execution time distribution of each task and the tolerance to deadline misses (measured by the quantitative completion ratio), we have developed a set of algorithms to estimate the probabilistic timing performance and to manage system resources in such a way that the system achieves the required completion ratio probabilistically with a reduced amount of system resources. This method relaxes the rigid hardware requirements for software implementation to meet the WCET and eventually avoids over-designing the multimedia system. We will use system's energy consumption, one of the most critical resources for multimedia embedded systems, as an example to demonstrate how our approach can lead to significant energy-efficient designs.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 gives the overview of our probabilistic design space exploration methodology. Probabilistic timing performance estimation is discussed in Section 4. Section 5 introduces our system resource management approaches. Experiment results for our resource management techniques to reduce the energy consumption are given in Section 6. Finally, we conclude the paper in Section 7.

2. Related Work

The most relevant work are on design space exploration and performance analysis, probabilistic performance estimation and scheduling techniques for low power.

An integrated hardware-software codesign system should support design space exploration with optimization [5]. There are several works on performance analysis for design space exploration based on monoprocessor architecture. In PMOSS [4], the authors present a methodology for rapid analysis, synthesis and optimization of embedded systems by providing modularity. Henkel and Ernst [6] have presented high-level estimation techniques for the hardware effort and hardware/software communication time. They claim that the proposed techniques are well suited for fast design space exploration. In the LYCOS system [12], the authors use profiling techniques and evaluations of low-level execution time for hardware, software and communication to estimate the system performance. For the rapid prototyping of hardware-software codesigns, Chatha and Vemuri [3] introduce their performance evaluation tool to provide fast and accurate performance estimates based on profiling and scheduling. However, all of the above works specify deadline as one of the design constraints that has to be met.

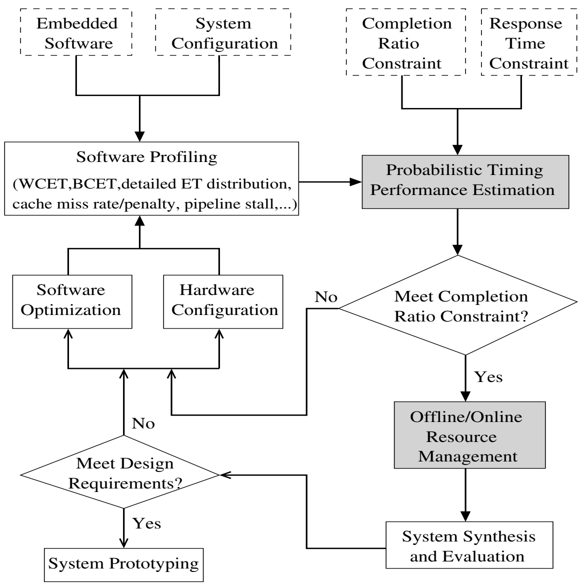
There are several papers on the probabilistic timing performance estimation for soft real-time systems design [7, 11, 15]. The general assumption is that each task's execution time can be described by a discrete probability density function that can be obtained by applying path analysis and system utilization analysis techniques [13]. In [15], the authors extend the scheduling algorithms and schedulability analysis methods developed for periodic tasks in order to provide probabilistic performance guarantee for semi-periodic tasks when the total maximum utilization of the tasks on each processor is larger than one. They describe the transform-task method that transforms each semi-periodic task into a periodic task followed by a sporadic task. The method can provide an absolute guarantee for requests with shorter computation times and a probabilistic guarantee for longer requests. In [11], a performance estimation tool that outputs the exact distribution of the processing delay of each application is introduced. It can help the designers develop multimedia networked systems requiring soft real-time guarantees in a cost efficient manner. Given that the execution time of each task is a discrete random variable, Hu *et al.* [7] propose a state-based probability metric to evaluate the overall probabilistic timing performance of the entire task set. Their experimental results show that the proposed metric reflects well the timing behavior of systems with independent and/or dependent tasks. However, their evaluation method becomes very time consuming when task has many different execution time values. Hua *et al.* [8] reduce system's average energy consumption for providing non-perfect completion ratio guarantees statistically. Some tasks are intentionally dropped according to their on-line scheduling algorithm to conserve energy.

Low power consumption is one of the most important design objectives. There are two power reduction techniques that can impact system scheduling: dynamic voltage scaling (DVS), which varies the clock frequency and supply voltage according to the workload at run-time; and dynamic power management (DPM), which aims to shut off system parts that are not currently in use. Excellent surveys on DVS and DPM can be found in [1] and [10] respectively.

3. Probabilistic Design Methodology Overview

Many design methods have been developed based on WCET to meet the timing constraints without any deadline misses. These methods are pessimistic and are suitable for developing systems in a "hard real-time" environment, where any deadline miss will be catastrophic. However, there are also many "soft real-time" systems, such as multimedia systems, which can tolerate occasional deadline misses. The above pessimistic design methods can't take advantage of this feature and will often lead to over-designed systems. In order to avoid over-designing systems, we propose the concept of "probabilistic design" where we design the system to meet the timing constraints of periodic applications statistically. That is, the system may not guarantee the completion of each execution or iteration, but it will produce sufficiently many successful completions over a large amount of iterations to meet the user-specific completion ratio. Or even better, the probability that any execution will be completed is not lower than the desired completion ratio.

Clearly, the proposed "probabilistic design" will be preferred for many embedded systems such as portable multimedia systems where high portability, low power consumption, and reasonably good performance are equally important. However, the corresponding "probabilistic design space" becomes larger than the above mentioned pessimistic design space because it includes designs that fail some iterations while still meet the desired completion ratio requirement statistically. This increases the design complexity and makes early design space exploration difficult. The "probabilistic design" will thrive only when designers can quickly explore the huge probabilistic design space.



Embedded Software
System Configuration
Completion Ratio Constraint
Response Time Constraint
Software Profiling
(WCET,BCET,detailed ET distribution, cache miss rate/penalty, pipeline stall,...)
Probabilistic Timing Performance Estimation
Software Optimization
Hardware Configuration
No
Meet Completion Ratio Constraint?
Yes
Offline/Online Resource Management
No
Meet Design Requirements?
Yes
System Prototyping
System Synthesis and Evaluation

Figure 1. Design flow in the probabilistic design space.

Figure 1 depicts our probabilistic design space exploration approach for rapid and economic (multimedia) system prototyping. We start with the popular dataflow graph representation of the embedded software, the system's performance requirements (in terms of timing and completion ratio constraints), and a pool of target system architectures to select from. We partition the application into a set of tasks and use profiling tools to collect detailed execution information of each task. Next, we estimate the system timing performance to check whether it is feasible for the current system configuration to achieve the desired performance. If not, we change the hardware configuration and/or apply software optimization techniques and update the software profiling results that will be used in the next round of system timing performance estimation. We mention that any change on the target hardware configuration and/or software optimization may affect the application's actual execution information and therefore the software profiling process needs to be re-started. This iterative design loop terminates when all the design requirements are met.

Once the completion ratio constraint can be met, we move on to the phase of offline/on-line resource management. This is the key step in the proposed probabilistic design where we 1) allocate minimum system resources to each task to make the desired completion ratio probabilistically achievable, and 2) develop real time schedulers to manage the resources at run time such that the required completion ratio can be achieved probabilistically. Finally, we conduct system synthesis, simulation, and evaluation before prototyping the system.

4. Estimating the Probabilistic Timing Performance

In order to determine whether a given system implementation can meet the desired completion ratio constraint, we need to estimate the system's probabilistic timing performance. Specifically, we calculate the upper bound of the completion ratio that the system can achieve to help us in exploring the probabilistic design space.

We consider the *task graph* $G = (V, E)$ for a given application. Each vertex in the graph represents one task computation and directed edges represent the data dependencies between vertices. We adopt the assumption that the execution time of each vertex can be described by a discrete probability density function [7, 15]. Specifically, for each vertex v_i, we associate with a finite set of possible execution time $\{t_{i1}, t_{i2}, \cdots, t_{ik_i}\}$ (under a reference system configuration) 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}. Such statistics on task's execution time can be obtained by profiling tools.

The *completion time* of the task graph G (or equivalently the given application) 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. The application (or its task graph) will be executed periodically with its deadline $\mathcal{M}$ as the period. We say that an iteration is *successfully completed* if $C(< v_1 v_2 \cdots v_n >) \leq \mathcal{M}$. The performance requirement is measured by a real-valued completion ratio $\mathcal{Q}_0 \in [0, 1]$, which is the minimum ratio of completions that the system has to maintain over a sufficiently large number of iterations. Let k be the number of successfully completed iterations over a total of $N >> 1$ iterations, the actual completion ratio can be denoted by $\mathcal{Q} = \frac{k}{N}$. We say that the completion ratio constraint is *achievable* if $\mathcal{Q} \geq \mathcal{Q}_0$.

For a given system configuration, let t'_{ij} be the time to execute task v_i that requires an execution time t_{ij} under the reference configuration, we have a completion if the completion time is less than the deadline, that is, $\sum_{i=1}^{n} t'_{ij} \leq \mathcal{M}$. The probability that this occurs is $\prod_{i=1}^{n} p_{ij}$. Therefore, we have

Theorem 1. The maximum achievable completion ratio is given by:

$$\mathcal{Q}^{max} = \sum_{\sum_{i=1}^{n} t'_{ij} \leq \mathcal{M}} \prod_{i=1}^{n} p_{ij} \qquad (1)$$

where the sum is taken over the execution time combinations that meet the deadline constraint $\mathcal{M}$ and the product computes the probability each such combination happens.

This is similar to the state-based feasibility probability defined in [7]. $\mathcal{Q}^{max}$ helps us to quickly explore the probabilistic design space. Specifically, if $\mathcal{Q}^{max} < \mathcal{Q}_0$, which means that the completion ratio requirement is not achievable under current system, we can make the early and correct decision to reconfigure the hardware or optimize the software implementation rather than further investigating the current system configuration.

The drawback of this estimation is that Equation (1) is computational expensive particularly when there are many tasks and each task has multiple execution times. For example, a task graph with 50 vertices and each vertex has only the best, average, and worst case execution time yields 3^{50} different execution time combinations! Due to the importance of determining whether the required $\mathcal{Q}_0$ is achievable in designing fast probabilistic design space exploration techniques, we have developed the following heuristic.

Assuming that the task's execution times are ordered such that $t_{i1} < t_{i2} < \cdots < t_{ik_i}$, we define the prefix sum of the occurrence probability

$$P_{il} = \sum_{j=1}^{l} p_{ij} \tag{2}$$

which measures the probability that the computation at vertex v_i is not longer than t_{il}. If we allocate time t_{il_i} to task v_i and drop the iteration if its actual execution time is longer, then we achieve a completion ratio

$$\mathcal{Q} = \prod_{i=1}^{n} P_{il_i} = \prod_{i=1}^{n} \sum_{j=1}^{l_i} p_{ij} \tag{3}$$

We use greedy approach to estimate whether completion ratio $\mathcal{Q}_0$ can be achieved within deadline $\mathcal{M}$. First, we assign each vertex its WCET. This yields $\mathcal{Q} = 1$ but the completion time $\sum_{i=1}^{n} t_{ik_i}$ will most likely exceeds the deadline constraint. From Equation (3), if we cut the time slot of vertex v_i from t_{il_i} to $t_{i(l_i-1)}$, the completion ratio will be reduced by the factor of $\frac{P_{i(l_i-1)}}{P_{il_i}}$. We iteratively cut the time slot of vertex v_j that yields the largest $(t_{jl_j} - t_{j(l_j-1)}) \cdot \frac{P_{j(l_j-1)}}{P_{jl_j}}$ as long as it gives a completion ratio larger than $\mathcal{Q}_0$. This greedy selection approach frees more assigned time slot at the minimum level of completion ratio reduction. When we cannot reduce the completion ratio any further and the total assigned time $\sum_{i=1}^{n} t_{ik_i}$ is still larger than the deadline $\mathcal{M}$, our heuristic concludes that the completion ratio cannot be guaranteed even though $\mathcal{Q}^{max}$ may still be larger than $\mathcal{Q}_0$.

5. Managing System Resource under Probabilistic Performance Constraint

When $\mathcal{Q}^{max} \geq \mathcal{Q}_0$, it becomes theoretically possible to deliver the probabilistic performance guarantee (in terms of completion ratio) with the current system configuration. The resource management phase in our design space exploration aims to reduce the design cost. It includes: 1) determining the minimum system resource required to provide the probabilistic performance guarantees; and 2) developing on-line scheduling algorithms to guide the system to achieve such guarantees at run time with the determined minimum resource.

As energy consumption has emerged as one of the most important concerns in the design of embedded systems particularly for the battery-operated portable systems, we consider in this section energy as the resource to manage and summarize our newly developed offline/on-line energy reduction techniques with completion ratio guarantees. We achieve the energy saving by the dynamic voltage scaling method on multiple supply voltage and multiple threshold voltage system, which has been identified by the International Technology Roadmap for Semiconductors (ITRS) as the trend of future systems [18]. Specifically, we consider the following problem:

For a given task graph, its deadline, its completion ratio constraint and the task execution time distribution, find a scheduling strategy for a multiple voltage system such that the resource (e.g., 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 voltage for the execution of each vertex.

5.1. Dynamic Voltage Scaling Systems

Dynamic power, which is the dominant source of power dissipation in CMOS circuit, is proportional to $\alpha C_L v_{dd}^2 f_{clock}$, where αC_L is the effective switched capacitance. Reducing the supply voltage can result in substantial power and energy saving. Roughly speaking, system's power dissipation is halved if we reduce V_{dd} by 30% without changing any other system parameters. However, this saving comes at the cost of reduced throughput, slower system clock frequency, or higher gate delay. The gate delay is proportional to $\frac{v_{dd}}{(v_{dd}-v_{th})^\beta}$ where v_{th} is the threshold voltage and $\beta \in (1.0, 2.0]$ is a technology dependent constant. Dynamic voltage scaling is a technique that varies system's operating voltage and clock frequency based on the computation load to provide desired performance with the minimum energy consumption. It has been demonstrated as one of the most effective low power system design techniques and has been supported by many modern microprocessors. Examples include Transmeta's Crusoe, AMD's K-6, Intel's XScale and Pentium III and IV, and some DSPs developed in Bell Labs.

The latest ITRS predicts that "power dissipation for high performance microprocessor will exceed the package limits by 25X in 15 years" and as a result, future systems will feature "multiple V_{dd} and multiple V_t being used on a single chip to reduce power while maintaining performance". Low power design is of particular interest for the soft real-time multimedia systems and we assume that our system has multiple voltages available on the chip such that the system can switch from one level to another at run time. To consider the energy and delay overhead associated with multiple-voltage systems, note that we use no more than two different voltages during the execution of each vertex [9], we can easily compare the energy saving from multiple voltages and the energy/delay overhead it introduced to decide whether multiple voltage should be used on this vertex. Therefore, we assume that voltage scaling occurs simultaneously without any energy and delay overhead.

5.2. Energy Reduction Techniques with Completion Ratio Guarantees

In order to reduce the energy consumption, we need to take advantage of the information we have obtained such as the execution time distribution of each task, the deadline $\mathcal{M}$ and the completion ratio constraint $\mathcal{Q}_0$. In this section, we first introduce a naïve approach and an on-line best-effort energy minimization approach, both of which can achieve the maximum completion ratio $\mathcal{Q}^{max}$. We then give a summary of our newly developed offline and on-line scheduling policies that reduce the system energy consumption while satisfying the required completion ratio constraint.

Consider the scenario that the system keeps on executing tasks with the highest speed. An iteration is either completed or failed at the deadline $\mathcal{M}$. For any iteration with execution time combination that contributes to $\mathcal{Q}^{max}$ in Equation (1), we will have a completion. Clearly, the maximum completion ratio $\mathcal{Q}^{max}$ is reached and we refer this strategy as the naïve best-effort approach. It meets the required $\mathcal{Q}_0(\leq \mathcal{Q}^{max})$ and the system's energy consumption under such scheduler provides us a guideline in evaluating other energy-efficient approaches.

With the flexibility of switching voltages at run time, the system can maintain the same completion ratio at reduced energy consumption. Specifically, we can define task v_i's *latest completion time* T_{l_i} and *earliest completion time* T_{e_i} recursively based on its best and worst case execution times. Let t_{ij} be the execution time of vertex v_i, we have developed an on-line best-effort energy-minimization (BEEM) scheduling algorithm that makes decision at time t as follows:

- if $t + t_{ij} > T_{l_i}$, terminate the current iteration;
- if $t + t_{ij} < T_{e_i}$, scale voltage so v_i is complete at time T_{e_i};
- else execute v_i at the reference voltage to its completion at time $t + t_{ij}$.

One can prove that

Theorem 2 [8]. BEEM guarantees the highest completion ratio $\mathcal{Q}^{max}$ with the minimum energy consumption.

Note that the best-effort approaches achieve $\mathcal{Q}^{max}$, the best possible completion ratio. When the required completion ratio $\mathcal{Q}_0$ is lower, carefully designed scheduling method can further reduce energy consumption. For example, by taking advantage of the information on the task's execution time that we obtained earlier from software profiling tools.

```
/* I: determine if $\mathcal{Q}_0$ is met within M. */
find a topological order of the vertices;
for each vertex $v_i$
  $t_{il_i} = t_{ik_i}, l_i = k_i$;      /* Assign WCET to each vertex */
$\mathcal{Q} = 1$;
while ($\mathcal{Q} > \mathcal{Q}_0$)
{ pick a vertex $v_j$ that has maximum
        $\frac{t_{jl_j} - t_{j(l_j-1)}}{P_{jl_j}} \cdot P_{j(l_j-1)}$ among all of vertices;
  update $\mathcal{Q}$ by $\mathcal{Q} = \mathcal{Q} \cdot \frac{P_{j(l_j-1)}}{P_{jl_j}}$;
  if ($\mathcal{Q} > \mathcal{Q}_0$)
    update $t_{jl_j}$ by $t_{jl_j-1}$ and $l_j$ by $l_j - 1$;
}
calculate the completion time $\mathcal{C}$ based on $t_{il_i}$;
if ($\mathcal{C} > \mathcal{M}$)
    $\mathcal{Q}_0$ is not met, exit.
/* II: set the time pair for each vertex to reduce energy. */
$ScaleFactor(v_i) = \frac{\mathcal{M}}{\mathcal{C}}$;
for each vertex $v_i$
  $T_{si} = t_{il_i}$;
  $T_{ei} = T_{si} \cdot ScaleFactor(v_i)$;
/* III: On-line voltage scaling for energy reduction. */
current time $t = 0$;
for each vertex $v_i$ $(i = 1, 2, \cdots, n)$
  $t_{ij}$ = the execution time of $v_i$ in the jth iteration;
  if ($t_{ij} > T_{si}$)
    terminate the current iteration;
  else
    scale voltage so $v_i$ is completed at $t = t + T_{ei}$;
```

Figure 2. The offline/on-line minimum-effort (O^{2}ME) algorithm for energy reduction with completion ratio guarantees.

Figure 2 depicts the proposed offline/on-line minimum-effort (O^{2}ME) algorithm. It consists of an offline execution

	n	$\mathcal{M}$	$\mathcal{Q}_0$	$E_{naïve}$	E_{BEEM}	E_{O^2ME}	saving over $E_{naïve}$	saving over E_{BEEM}	$\mathcal{Q}$
FFT1	28	3800	0.900	855.79	487.46	348.57	59.27%	28.49%	0.9075
FFT2	28	2400	0.900	540.50	239.84	198.25	63.32%	17.34%	0.9075
Laplace	16	7200	0.900	1615.7	635.50	540.40	66.55%	14.97%	0.9127
qmf4	14	800	0.900	180.92	66.49	61.68	65.91%	7.23%	0.9088
karp10	21	2970	0.900	669.57	341.16	258.72	61.36%	24.17%	0.9049
meas	12	1415	0.900	318.02	117.99	102.05	67.91%	13.51%	0.9132
sum1	15	435	0.900	98.31	52.25	35.87	63.51%	31.35%	0.9026
almu	17	360	0.900	80.60	31.16	26.88	66.65%	13.74%	0.9047
DSC-7-7	7	65	0.900	15.05	7.01	5.04	66.51%	28.10%	0.9162
DSC-7-8	7	65	0.900	15.05	6.35	4.91	67.38%	22.68%	0.9162

Table 1. O^{2}ME approach's energy efficiency over the two best-effort approaches. Energy is the average energy consumption per iteration and measured in the unit of the system's energy dissipation in one CPU unit at the reference voltage. $\mathcal{Q}_0 = 0.900$ is the required completion ratio and $\mathcal{Q}$ is the actual completion ratio achieved by O^{2}ME. Both naïve and BEEM approaches achieve exactly $\mathcal{Q}_0 = 0.900$ by stopping the process when it is reached. n is the number of vertices in each task graph and $\mathcal{M}$ is the deadline.

time allocation phase and an on-line voltage scaling phase. The goal of the offline part is to assign pair (T_{si}, T_{ei}) to each vertex v_i to guide the on-line voltage scaling. The system allocates time T_{ei} for the execution of vertex v_i and will use this entire slot (by scaling down voltage and speed) to execute v_i only if its execution time is less than $T_{si}(\leq T_{ei})$.

O^{2}ME trades completions for energy saving by terminating an iteration once it encounters a vertex v_i that has execution time longer than its pre-determined value T_{si}. The selection of T_{si}'s guarantees that the desired completion ratio will be met probabilistically. Stretching v_i's execution time to T_{ei} reduces the energy consumption to complete task v_i by voltage scaling. The early termination provides energy saving from not attempting to achieve completion ratio higher than the required $\mathcal{Q}_0$. Because of this unique characteristic, the approach is referred as *minimum-effort* algorithm.

Determining the pairs (T_{si}, T_{ei}) in the offline part is a challenging problem. First, we want to have T_{si} short to have more early terminations for energy reduction. However, we need to make sure that the corresponding completion ratio (given in Equation (3)) is not lower than $\mathcal{Q}_0$. Second, once T_{si} is determined, we want to assign v_i longer execution slot T_{ei} for voltage scaling. But the sum of T_{ei}'s cannot exceed the deadline $\mathcal{M}$. Finally, $T_{ei} \geq T_{si}$ must be satisfied to ensure that v_i is completable, once the on-line scheduler decides to execute it, within its assigned slot T_{ei}. O^{2}ME uses the greedy heuristic, which we discussed for probabilistic performance estimation, to solve this problem as shown in steps I and II in Figure 2. In the next section we validate that O^{2}ME provides a statistical guarantee to the completion ratio requirement with reduced energy consumption over the best-effort approaches.

6. Experimental Results

The proposed probabilistic design method relaxes the rigid hardware requirements (to meet the WCET scenario) for software implementation and eventually avoids over-designing the system. It is a promising innovation beyond conventional design methods based on WCET when perfect completion ratio is not vital, in which case we can effectively trade the performance for other more important system resources such energy consumption. We are currently working on prototyping multimedia systems to investigate how the probabilistic timing performance requirement can help us in more economic designs. In this section, we report our preliminary results on the validation of the proposed offline/on-line minimum-effort algorithm's energy efficiency under the probabilistic completion ratio constraint.

6.1 Experiment Setup

Our experiments are on a variety of benchmark task graphs extracted from real-life DSP applications [14, 16, 17]. They include two different implementations of the Fast Fourier Transform (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 other task graphs reported in early literatures.

For each benchmark task graph, we assign each vertex three possible execution time $e_0 < e_1 < e_2$ with probabilities $p_0 >> p_1 > p_2$, respectively. e_0 is the BCET that occurs frequently in most DSP applications. The deadline $\mathcal{M}$ varies from the sum of BCET to the sum of WCET. We implement our O^{2}ME algorithm to achieve a completion ratio $\mathcal{Q}_0$ from 0.4 to 1.0. We consider a processor with four different voltages ranging from 1.2V to 3.3V and simulate 100,000 iterations for each setting (deadline $\mathcal{M}$, completion ratio $\mathcal{Q}_0$, and voltage levels).

To compare the energy efficiency of O^{2}ME, we also implement both the naïve and BEEM approaches. Since the latter two achieve Q^{max} which may be higher than the required completion ratio $\mathcal{Q}_0$, to get a fair comparison, we force them to reach $\mathcal{Q}_0$ exactly in the following way. We group 100 consecutive iterations and stop the execution once $\lceil 100 \cdot \mathcal{Q}_0 \rceil$ iterations are completed in the same group. This makes the $\lceil 100000 \cdot \mathcal{Q}_0 \rceil$ completions relatively evenly distributed through the entire 100,000 iterations.

6.2 Results and Discussion

We now report a few representative sets of results for the following setup: $\mathcal{Q}_0 = 0.900$ (if not specified otherwise), $\mathcal{M}$ equals to five times of the sum of each task's BCET (if not specified otherwise), and the four voltages are 3.3V, 2.6V, 1.9V, and 1.2V. The first is on O^2ME algorithm's energy efficiency over the naïve best-effort approach and BEEM algorithm. The other two are on the impact of different completion ratio requirement and deadline to energy consumption of the proposed methods.

Table 1 reports the average energy consumption per iteration by the three different approaches. The second column gives the number of vertices in each benchmark [14, 16, 17]. The next two columns list the deadline $\mathcal{M}$ and completion ratio requirement $\mathcal{Q}_0$. $E_{naïve}$ and E_{BEEM} columns show the average energy consumption per iteration (total energy consumption over the 100,000 iterations) when the exact $\mathcal{Q}_0$ is enforced as we have discussed earlier. The next three columns are on O^2ME algorithm's energy efficiency and the last column reports the actual completion ratio $\mathcal{Q}$ it achieves. From the table we see that O^2ME algorithm consistently save significant portion of energy (an average of 64.84%) over the naïve approach. It also outperforms the BEEM algorithm, the provably most energy efficient best-effort approach. In the latter case, we observe an average 20.16% energy reduction with a relatively large variance. This is because BEEM algorithm reduces energy to the minimum level to provide the same completion ratio achieved by the naïve approach. The energy efficiency of BEEM depends on the characteristics of the application such as the execution time distribution and the deadline constraint. Although O^{2}ME does not provide any absolute guarantee on completion ratio, we see from the last column that the required 0.900 completion ratio is met in all the cases.

Figure 3 reveals how completion ratio requirement $\mathcal{Q}_0$ may impact the energy consumption for the proposed techniques on the benchmark karp10. Regradless of which technique we use, the system's total energy consumption over the 100,000 iterations consists of two parts: the energy on completed iterations which is useful and the part on eventually failed iterations which is wasted. For both best-effort approaches, the counting mechanism is used to count the number of completions hence to shut down the system once the required $\mathcal{Q}_0$ is met. This avoids "wasting" energy on unnecessary executions. As we can see from the figure, their energy consumption decreases linearly with $\mathcal{Q}_0$. On the other hand, the proposed O^{2}ME does not have such counting mechanism and we do not expect its energy consumption decreases linearly. Small $\mathcal{Q}_0$ implies that we can terminate more iterations at an early time (because T_{si} becomes smaller) and we can allocate longer time slot T_{ei} to save energy on processing each task (see Figure 2). As a result, we find that the completion ratio $\mathcal{Q}_0$ has less dramatic impact on O^{2}ME's energy consumption. Interestly, we also find that O^{2}ME's energy efficiency over BEEM also decreases as $\mathcal{Q}_0$ is reduced. This is because that O^{2}ME always executes early tasks until it finds a task with execution time longer than T_{si} to terminate the iteration, while BEEM will not consume anything once the small $\mathcal{Q}_0$ is reached.

Finally, we discuss deadline's impact to energy consumption for the proposed techniques. The naïve approach operates at the highest voltage and shuts down when the required $\mathcal{Q}_0$ is met. Therefore, its energy consumption remains constant regardless of the deadline constraint. The pair (T_{l_i}, T_{e_i}) for each vertex in the BEEM algorithm is directly related to the deadline $\mathcal{M}$. In fact, the last vertex has both T_{l_i} and T_{e_i} set to $\mathcal{M}$. The pair for other vertices are defined recursively from this. When $\mathcal{M}$ increases, T_{e_i} becomes larger and the system can slow down to save energy without dropping the task. Therefore, BEEM's energy

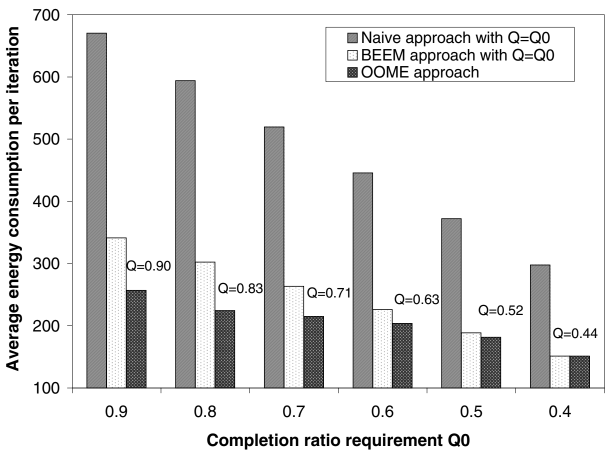


Figure 3. Different completion ratio requirement $\mathcal{Q}_0$'s impact to the average energy consumption per iteration on the karp10 benchmark.

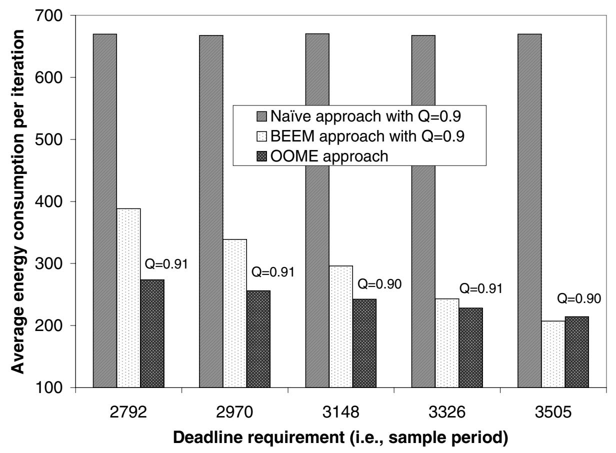


Figure 4. Different deadline requirement's impact to the average energy consumption per iteration on benchmark karp10.

consumption highly depends on the deadline. Similar to the above analysis of completion ratio's impact to O^{2}ME's energy consumption, we see that deadline also has a positive impact to O^{2}ME, but not as dramatic as it does to BEEM (see Figure 4).

7. Conclusions

This paper presents the novel concept of probabilistic design for multimedia systems and a methodology to fast explore such design space at the early design stage in order to achieve rapid and economic (multimedia) system prototyping. By taking advantage of multimedia DSP application's unique features, namely application's performance requirements, uncertainties in execution time, and tolerance for reasonable execution failures, our method systematically relaxes the rigid hardware requirements for software implementation and eventually avoids over-designing the system. As an example, we show how to design multimedia systems with reduced resource (energy consumption in our case) while providing the desired performance (completion ratio) probabilistically. Experimental results show that our proposed method achieves better designs with significant energy (resource) savings. Our ongoing work includes applying the proposed probabilistic design method to build prototype multimedia systems, measuring the overall energy consumption, and evaluating the systems performance at user level.

8 Acknowledgments

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

References

[1] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. on VLSI Systems*, 8(3):299–316, 2000.

[2] J. Bolot and A. Vega-Garcia. Control mechanisms for packet audio in the internet. *Proceedings of IEEE Infocom*, 1996.

[3] K. S. Chatha and R. Vemuri. Performance evaluation tool for rapid prototyping of hardware-software codesigns. *9th International Workshop on Rapid System Prototyping*, pages 218–224, June 1998.

[4] H. J. Eikerling, W. Hardt, J. Gerlach, and W. Rosenstiel. A methodology for rapid analysis and optimization of embedded systems. *International IEEE Symposium and Workshop on ECBS*, pages 252–259, March 1996.

[5] R. Ernst. Codesign of embedded systems: Status and trends. *IEEE Design & Test of Computers*, 15(2):45–54, 1998.

[6] J. Henkel and R. Ernst. High-level estimation techniques for usage in hardware/software co-design. *Aisa and South Pacific Automation Conference*, pages 353–360, 1998.

[7] X. Hu, T. Zhou, and E. H.-M. Sha. Estimating probabilistic timing performance for real-time embedded systems. *IEEE Trans. on VLSI systems*, 9(6):833–844, December 2001.

[8] S. Hua, G. Qu, and S. S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. *40th ACM/IEEE Design Automation Conferece*, June 2003.

[9] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *International Symposium on Low Power Electronics and Design*, pages 197–202, 1998.

[10] N. K. Jha. Low power system scheduling and synthesis. *IEEE/ACM International Conference on Computer Aided Design*, pages 259–263, 2001.

[11] A. Kalavade and P. Moghe. A tool for performance estimation of networked embedded end-systems. *Proc. Design Automation Conference*, pages 257–262, June 1998.

[12] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen, and A. E. Haxthausen. LYCOS: the lyngby co-synthesis system. *Journal for Design Automation of Embedded Systems*, 2(2):195–235, March 1997.

[13] S. Malik, M. Martonosi, and Y. S. Li. Static timing analysis of embedded software. *Design Automation Conference*, pages 147–152, June 1997.

[14] C. L. McCreary et al. A comparison of heuristics for scheduling dags on multiprocessors. *Proceedings of the International parallel Processing Symposium*, 1994.

[15] 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*, pages 164–173, 1995.

[16] M. Wu and D. D. Gajski. Hypertool: A programming aid for message-passing systems. *IEEE Trans. on Parallel and Distributed Systems*, 3(1), 1990.

[17] T. Yang and A. Gerasoulis. DSC: Scheduling parallel tasks on an unbounded number of processors. *IEEE Trans. on Parallel and Distributed Systems*, 2:951–967, 1994.

[18] International technology roadmap for semiconductors. *http://public.itrs.net*, 2001.