Energy-Aware Cosynthesis of Real-Time Multimedia Applications on MPSoCs Using Heterogeneous Scheduling Policies

MINYOUNG KIM, SUDARSHAN BANERJEE, NIKIL DUTT, and
NALINI VENKATASUBRAMANIAN
University of California, Irvine

Real-time multimedia applications are increasingly being mapped onto MPSoC (multiprocessor system-on-chip) platforms containing hardware–software IPs (intellectual property), along with a library of common scheduling policies such as EDF, RM. The choice of a scheduling policy for each IP is a key decision that greatly affects the design's ability to meet real-time constraints, and also directly affects the energy consumed by the design. We present a cosynthesis framework for design space exploration that considers heterogeneous scheduling while mapping multimedia applications onto such MPSoCs. In our approach, we select a suitable scheduling policy for each IP such that system energy is minimized—our framework also includes energy-reduction techniques utilizing dynamic power management. Experimental results on a realistic multimode multimedia terminal application demonstrate that our approach enables us to select design points with up to 60.5% reduced energy for a given area constraint, while meeting all real-time requirements. More importantly, our approach generates a tradeoff space between energy and cost allowing designers to comparatively evaluate multiple system level mappings.

Categories and Subject Descriptors: C.3 [**Computer Systems Organization**]: Special-purpose and Application-based Systems—*Real-time and embedded systems*

General Terms: Design

Additional Key Words and Phrases: Real-time scheduling, cosynthesis, MPSoC, energy

ACM Reference Format:

Kim, M., Banerjee, S., Dutt, N., and Venkatasubramanian, N. 2008. Energy-aware cosynthesis of real-time multimedia applications on MPSoCs using heterogeneous scheduling policies. ACM Trans. Embedd. Comput. Syst. 7, 2, Article 9 (February 2008), 19 pages. DOI = 10.1145/1331331.1331333 http://doi.acm.org/ 10.1145/1331331.1331333

This work was partially supported by NSF Grant CNS-0615438.
Authors' address: Minyoung Kim, Sudarshan Banerjee, Nikil Dutt, and Nalini Venkatasubramanian, School of Information and Computer Sciences, University of California - Irvine, Irvine, California 92697-3425; email: {minyounk, banerjee, dutt, nalini}@ics.uci.edu. Sudarshan Banerjee is currently with Liga Systems, Sunnyvale, California 94085.

1. INTRODUCTION

Battery-powered mobile devices are increasingly becoming a key part of our daily lives. Shrinking time-to-market deadlines coupled with strict timing/energy/cost constraints have created the need for flexible computing platforms capable of supporting a variety of multimedia applications executing on such devices. MPSoCs (multiprocessor system-on-chip) present a viable platform for these applications, since they typically contain multiple heterogeneous PEs (processing elements), memories, I/O components, etc. Indeed, several commercial MPSoC platforms are now available including the TI OMAP [Helmig 2002], ST Nomadik [STMicroelectronics], NXP Nexperia [NXP], and STI Cell processor [Pham et al. 2006]. In order to effectively use these MPSoC platforms, designers need the ability to explore alternative mappings of multimedia applications that results in implementations with different energy and cost configurations. Since the scheduling policy used by the processors in the MPSoC directly affects the energy and cost, there is a critical need to explore the effects of application mapping using different processor scheduling policies.

In this work, we propose an integrated HW–SW (hardware–software) cosynthesis framework for design space exploration of multimedia applications on MPSoCs with heterogeneous real-time scheduling policies. In addition, our framework reduces system energy by dynamic power-management techniques. In Figure 1, we consider an application driver, the MMMT (multimode multimedia terminal) [Oh and Ha 2002].[1] Such an application consists of real-time tasks like the H.263 encoder with strict timing requirements—the key cosynthesis steps of *partitioning* (choice of a suitable processing element for executing each function) and *scheduling* (choice of a suitable execution order of tasks executing on a processing element) need to ensure that application timing requirements are met while optimizing for design objectives, such as minimum energy, hardware cost, etc. Given the critical nature of energy constraints for battery-powered devices, there is a rapidly growing body of work on HW–SW cosynthesis under energy constraints [Schmitz et al. 2005; Wolf 2004].

Typical work in design space exploration for such multimedia systems proposes ad-hoc scheduling policies to minimize the energy [Schmitz et al. 2005; Yang et al. 2001]. However, such an approach adds the overhead of customized scheduling tables for each individual application. In addition, such an approach is unable to take advantage of a key system feature—a typical MPSoC platform contains libraries with implementations of well-known scheduling policies, such as RM (rate monotonic), EDF (earliest deadline first) [Buttazzo 2005].

Our framework enables us to perform design space exploration while considering three critical constraints: area, energy, and performance. For a given multimode application, our approach selects a suitable PE for each task and a suitable scheduling policy for each PE, such that energy is minimized while meeting an aggregate area and timing constraints for the individual tasks. Our approach addresses resource sharing between different modes. Experimental

[1]A multimode embedded system supports multiple applications (e.g., video phone, MP3 player, VoD player) by dynamically reconfiguring system functionality.

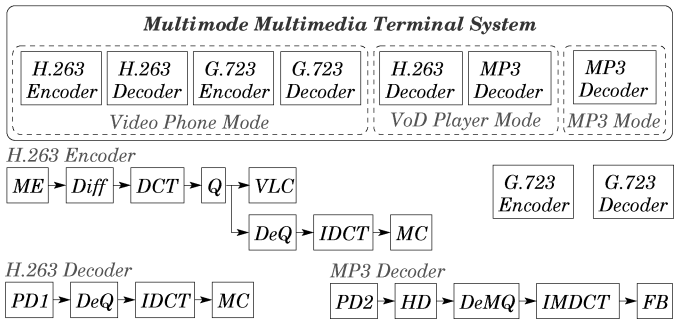


Fig. 1. Multimode multimedia terminal (MMMT) example.

results indicate that in a realistic MMMT application, our approach can identify design points with energy savings up to 60.5% compared to previous approaches. Our work also presents the designer with a tradeoff space allowing comparative evaluation of multiple mappings for energy and cost.

2. RELATED WORK

HW–SW cosynthesis for MPSoCs is a rich area of research with several lines of work that we briefly review and contrast against our approach.

In [Oh and Ha 2002], the authors provide a framework for cosynthesis of a MMMT system that is based on a nonpreemptive static scheduling technique integrated with mapping. An allocation controller considers processor utilization to select appropriate PEs and resource sharing among common functional nodes such that the schedulability constraint is satisfied and system cost is minimized.

In [Schmitz et al. 2005], the authors consider probabilistic execution of the different modes based on the observation that typical usage profile (e.g., activation profile of a mobile phone) needs to be considered for reducing the energy consumption through appropriate resource sharing between tasks. They propose a genetic algorithm that considers resource sharing, component shutdown, and mode transition issues, and further reduces the energy dissipation by introducing a transformation-based method to extend existing dynamic voltage-scaling (DVS) approaches.

In [Shin and Lee 2003, 2004], the authors propose a resource model to characterize a periodic resource allocation and present exact schedulability conditions under EDF and RM scheduling policies. Their technique essentially abstracts the timing requirements for a set of periodic tasks under the two scheduling policies such that the entire task set can be treated as a single periodic task. Based on their abstraction, they introduce a composition method such that for a hierarchy of schedulers, the timing requirements of each parent scheduler can be determined from the timing requirements of its children, i.e., the timing requirement of the parent scheduler is satisfied, if, and only if, the timing requirements of its child schedulers are satisfied. More recently, the authors in [Matic and Henzinger 2005] have extended the resource model to address data dependences between tasks. They consider two semantics for dataflow constraints, LET (logical execution time) and RTW (real-time workshop), that differ in data propagation between tasks. They demonstrate that LET semantics allow for better composability with tighter resource bounds in the abstraction hierarchy, including shared as well as distributed resources.

Our goal is a cosynthesis methodology that guarantees real-time operation with low area/energy requirements. Unlike [Oh and Ha 2002] and [Schmitz et al. 2005; Yang et al. 2001], we do not propose new scheduling algorithms—we instead exploit the existing heterogeneity in MPSoCs by choosing from a library of well-known scheduling policies, such as EDF and RM. This allows us to enable extensive design space exploration where we can evaluate multiple tradeoff points. The authors in [Pop et al. 2006] focused on communication protocols in the context of schedulability analysis, but they did not consider energy aspects in their work. On a similar note, the authors in [Shin and Lee 2003] and [Matic and Henzinger 2005] focused on the theoretical principles that effectively allow a set of periodic tasks with data dependences and shared as well as distributed resources to be treated as a single periodic task to meet timing constraints, but they do not consider energy issues. Our work is different in that we adapt previous theoretical work for energy estimation of MPSoCs and build our dynamic power-management techniques on top of the energy estimation framework for additional energy-reduction. It includes considerations of resource sharing between different application modes with corresponding benefits in energy consumption based on usage profile.

3. PRELIMINARIES

In this section, we define the terminology, the key assumptions underlying our approach, and follow-up with a formal problem statement.

3.1 Terminology and Assumptions

A multimode application consists of a set of modes, where a mode is a set of concurrent tasks. Each task in turn consists of a set of functional blocks. A task may be executed in various modes, but, it has a separate period and deadline in each mode. We illustrate these ideas using the MMMT example in Figure 1 consisting of three modes: VoD player, MP3 player, video phone. The video phone mode consists of the H.263 encoder and decoder tasks along with G.723 encoder and decoder tasks. The H.263 decoder task consists of four functional blocks (PD1, DeQ, IDCT, MC) that can be shared between the VoD player mode and video phone mode.

We next state our key assumptions:

- *Preemptive periodic tasks with block-level granularity*. Given that our cosynthesis framework targets multimedia applications, we assume that individual tasks are periodic and can be preempted. The period of a sporadic task can be modeled as the minimum interarrival time between successive requests. Partitioning (and scheduling) is done at the functional block granularity, i.e., the basic unit for partitioning is a function (such as the DCT in H.263 encoder task).

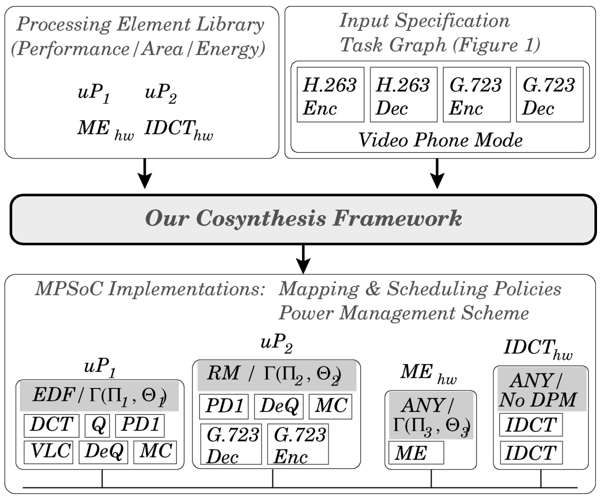

Processing Element Library (Performance/Area/Energy)
uP_1 uP_2
ME_{hw} $IDCT_{hw}$
Input Specification Task Graph (Figure 1)
H.263 Enc
H.263 Dec
G.723 Enc
G.723 Dec
Video Phone Mode
Our Cosynthesis Framework
MPSoC Implementations: Mapping & Scheduling Policies Power Management Scheme
uP_1
EDF / $\Gamma(\Pi_1, \Theta_1)$
DCT Q PD1
VLC DeQ MC
uP_2
RM / $\Gamma(\Pi_2, \Theta_2)$
PD1 DeQ MC
G.723 Dec
G.723 Enc
ME_{hw}
ANY / $\Gamma(\Pi_3, \Theta_3)$
ME
$IDCT_{hw}$
ANY / No DPM
IDCT
IDCT

Fig. 2. Cosynthesis problem for real-time MPSoCs with heterogeneous scheduling policies.

- *Data propagation assumption*. In this work, we follow LET (logical execution time) semantics ([Matic and Henzinger 2005]) for data propagation between tasks, i.e., we assume that the output data of a task is available at the relative deadline defined by the task period (instead of when the task completes execution). As explained in [Matic and Henzinger 2005], LET semantics increases latency, but allows for better composability with tighter abstractions that minimize lower level resource requirements. In this version of our work, LET semantics enable us to effectively ignore data dependences (and corresponding communication overhead) during scheduling by providing a tight abstraction that addresses intertask buffering resource concerns.

3.2 Problem Statement

Given a multimode application and a library of candidate processing elements, our problem is to find design implementations that are pareto-optimal in cost and energy, while ensuring that timing requirements are also met; an implementation refers to choosing a suitable processing element from the library for each functional block and a scheduling policy for each PE. As an example, in the MPSoC implementation shown in Figure 2, PE uP_1 is executing the EDF scheduling policy and the functional block DCT is mapped to uP_1. Our problem is described more formally below:

Input—Library Specification: The library of candidate PEs, L, consists of m different types of processing elements, i.e., $\mathrm{L} = \{pe_1, pe_2, \ldots, pe_m\}$. Each processing element pe_j has two parameters $\{a_j, p_j\}$, where a_j and p_j are the area and the average power consumption of the PE, respectively.[2]

[2]We can reasonably assume that power consumption is constant for PEs executing typical multimedia functions, such as DCT, ME, etc. [Marculescu et al. 2001].

Input—Application Specification: The application consists of a set of periodic real-time tasks $T = \{\tau_1, \tau_2, \ldots\}$. Each task τ_k has two parameters, $\{d_k, p_k\}$, where d_k and p_k are the deadline and the period, respectively. Given that each task is composed of a set of functional blocks, the complete application consists of n functional blocks $F = \{f_1, f_2, \ldots, f_n\}$. Each function f_i has a set of parameters $\{wcet_i^1, wcet_i^2, \ldots, wcet_i^m\}$, where $wcet_i^j$ is the worst-case execution time of function f_i on the processing element pe_j.

Problem Objective: For the given application specification {T, F} and the library specification L, the goal of our cosynthesis framework is to obtain multiple pareto-optimal area-energy tradeoff points. Each design point is defined by the following three components:

1. *Mapping*: $\forall i, [f_i \rightarrow pe_j^l]$
   For each functional block f_i, *mapping* defines the type of processing element pe_j on which f_i is to be implemented and the instance pe_j^l of the PE (if more than one instance of pe_j is present in the implementation).
2. *Scheduling*: $\forall j, \forall l, [S_j^l \rightarrow \{EDF, RM\}]$
   For each instance pe_j^l of processing element pe_j, *scheduling* defines the scheduling policy executing on it.
3. *Power Management Policy*: $\forall j, \forall l, [PM_j^l \rightarrow \{(\Pi, \Theta)\}$
   For each instance pe_j^l of processing element pe_j, *power management* defines *if and when* the PE can be shut down for energy reduction. This component is defined more precisely in the next section.

4. PROPOSED APPROACH

We present an overview of our proposed cosynthesis framework in Figure 3. We first generate set of initial design points with partitioning and mapping, as shown in Box *A*. In our context, partitioning refers to the selection of a suitable PE type for each functional block, while mapping refers to additionally deciding the number of instances of a PE type and the assignment of the function to an instance of a PE. In our prior work [Kim et al. 2006], we use the two-approximation algorithm proposed in [Baruah 2004] as initial partitioning method to generate a single initial design point. However, in this extended version, we propose a KLFM (Kernighan–Lin/Fiduccia–Mattheyses) [Kernighan and Lin 1970; Fiduccia and Mattheyses 1982] based approach that generates multiple such partition, while considering area and energy aspects of a multimode system.

The partitioning step selects the type of processing element for every functional block in the application. Successive steps shown in Boxes *B*-*D* are individually applied to each processing element type.

First, we select a mapping for the given partitioning—note that initial mapping in Box *A* is required for a specific PE type only if the utilization [Liu and Layland 1973] of the PE type is greater than 1. Corresponding to a selected mapping, we next need to select a suitable scheduling policy that will give us the best energy profile. For this step shown in Box *B*, we use the periodic resource

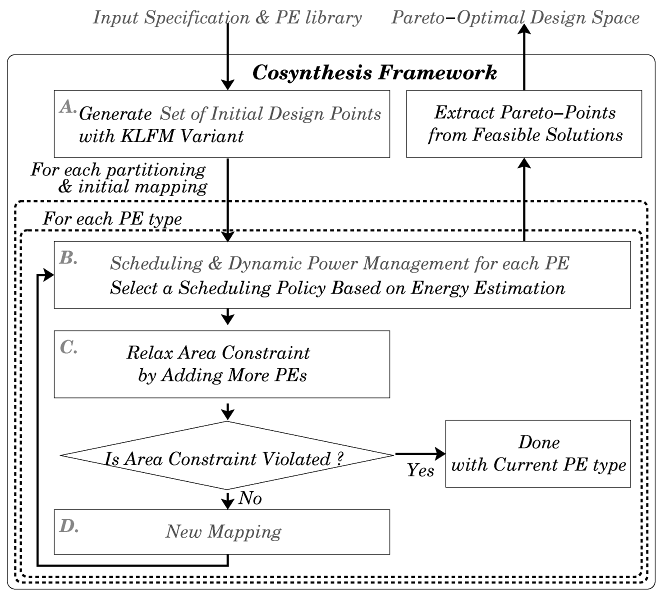

Input Specification & PE library
Pareto–Optimal Design Space
Cosynthesis Framework
A. Generate Set of Initial Design Points with KLFM Variant
Extract Pareto–Points from Feasible Solutions
For each partitioning & initial mapping
For each PE type
B. Scheduling & Dynamic Power Management for each PE
Select a Scheduling Policy Based on Energy Estimation
C. Relax Area Constraint by Adding More PEs
Is Area Constraint Violated ?
Yes
Done with Current PE type
No
D. New Mapping

Fig. 3. Flow of our cosynthesis framework.

model [Shin and Lee 2003] for schedulability test and choose the scheduling policy that maximizes energy-reduction. It is possible that schedulability test may fail even if the utilization of a PE instance is less than 1, in that case, we add one more instance of the PE, generate a new mapping, and check again for schedulability.

Next, we attempt to improve the system energy consumption by adding one more instance of the current processing element type. Adding more instances necessitates a change in mapping, as shown in Box D, and, of course, this step is feasible only if the area constraint is not violated. In Section 4.3, we will explain in detail how an additional instance of a PE can lead to energy-reduction for the system, along with how we modify the mapping to include the new instance.

We next present our detailed approach.

4.1 Mapping Based on Multiway KLFM

As described in Section 3.2, we define *mapping* as determining the type of processing element pe_j on which each functional block f_i is to be implemented, along with the instance pe_j^l of the PE (if more than one instance of pe_j is present in the implementation).

Our approach is based on *multiway KLFM* (Kernighan–Lin/Fiduccia–Mattheyses) [Kernighan and Lin 1970; Fiduccia and Mattheyses 1982] partitioning principles. We essentially attempt to place *objects* (f_i) into *buckets* (pe_j) until the *terminating condition* is satisfied. To find a mapping that gives the best implementation point, we propose a penalty function that includes area and energy considerations.

Algorithm 1 Design Space Exploration Based on Multi-way KLFM

```
Inputs:
    Objects: each functional block $f_i$
    Buckets: types of processing element $pe_j$
    Terminating Condition: area constraint
Output:
    Mapping: $\forall i$, $[f_i \rightarrow pe_j^l]$, where $pe_j^l$ represents the instance of $pe_j$
    Scheduling: $\forall j, \forall l$, $[S_j^l \rightarrow \{EDF, RM\}]$ for each $pe_j^l$
    Power Management: $\forall j, \forall l$, $[PM_j^l \rightarrow \{(\Pi, \Theta)\}$ for each $pe_j^l$
Procedure:
    01: Create initial partitioning & mapping by two-approximation;
    02: Initialize areaConst = area constraint, bestMapping = currentMapping;
    03: while (area improves)
    04:     Unlock all nodes;
    05:     while (valid moves exist)
    06:         if (current area cost < areaConst)
    07:             Do detailed Design Space Exploration
                        (with scheduling & energy reduction);
    08:             Update areaConst = current area cost;
    09:         endif;
    10:         Find best node & best move based on penalty function f;
    11:         Move best node & lock moved node;
    12:         if(currentMapping is better than bestMapping)
    13:             bestMapping = currentMapping;
    14:         endif;
    15:         Update bucket data structures;
    16:     endwhile;
    17:     currentMapping = bestMapping;
    18: endwhile;
```

The advantages of a strategy based on multiway KLFM are twofold:

1. The penalty function described later enables us to simultaneously consider multidimensional design criteria (energy consumption and area cost of an implementation) from the early design stage.
2. In addition, the penalty function considers resource sharing among different modes in the form of mode execution *probability*.

A sketch of our overall approach is presented in Algorithm 1. It is an iterative algorithm that begins with an initial partition and successively modifies it to improve the solution quality. In each step, the algorithm moves one node that causes the greatest improvement, or the least penalty. We lock down a node after it is moved and never move a locked node within the same iteration. The algorithm continues moving nodes until no unlocked node can be moved. Once all moves are exhausted, we use the best solution from previous iteration as the initial partition for subsequent runs (*line 17* of Algorithm 1). This allows us to climb out of local optimum, since it excludes bad intermediate moves, potentially finding a better partitioning in the following iteration that takes the best intermediate state as its starting point. After it moves back to the best intermediate state, it unlocks all nodes and continues to the next iteration. Once the algorithm fails to find a better intermediate state between checks, it finishes with the last chosen state.

Our approach differs from the basic KLFM algorithm in three ways:

1. We choose an initial partition using the two-approximation strategy from [Baruah 2004], instead of using completely random methods (*line 1*). This approach guarantees an implementation with area cost no more than a constant amount greater than twice the cost of an optimal implementation. While any exact or heuristic approach may be used for partitioning, using [Baruah 2004] for initial partitioning (and initial mapping) ensures that our set of feasible solutions contains at least one design point with well-defined properties.
2. The main search loop is augmented with a more complex cost metric, i.e., penalty function including energy and area cost, as well as mode probability (*line 10*). Best node (unlocked node in each partition that gives the smallest penalty when moved to other partition) is determined by the penalty function *f* in Eq. (1).

$$f = weight_1 \times E + weight_2 \times (C - area\ constraint) \qquad (1)$$

   where E and C represent the energy consumption and the area cost of a design, respectively, i.e.,

$$E = \Sigma_{i=0}^{numOfModes}\{\text{probability}[mode_i] \times \text{energy consumption of } mode_i\}$$
$$C = \Sigma_{j=0}^{numOfPEs}\{Max_i(\text{number of } pe_j \text{ in } mode_i \times \text{area cost of } pe_j)\}.$$

   Since we separate mapping and scheduling for fast exploration, the estimation of energy consumption does not include the scheduling information. Therefore, the mapping stage of our framework uses the approximation of energy consumption as (*utilization* × *energy consumption of a PE*). By using this penalty function, our cosynthesis framework can simultaneously consider multidimensional design criteria from the early design stage. In addition, time-multiplexed execution of multimode system is considered to enable resource sharing among different modes through the notion of *mode execution probability* in our penalty function *f*.
3. We carry out detailed exploration (line 7 and illustrated as Box *B–D* of Figure 3) with the current mapping when area cost is less than the area constraint (*areaConst* in *line 6*). Figure 4 explicitly illustrates our strategy for design space exploration based on KLFM principles. In the beginning of (*i-1*)-th iteration, the area constraint a_1 is met after one move. Therefore, we carry out design space exploration with scheduling and energy reduction from the intermediate design point dse_1. Next, we decrease the area constraint (*areaConst*) to the area cost of dse_1 (a_2). After two such moves, we reach the second design point (dse_2), where current area cost (a_3) is less than *areaConst*. At the end of each iteration, the algorithm backtracks to the point with minimum area in the sequence of moves from the current iteration as shown in the horizontal arrow from dse_3 in Figure 4. By generating multiple DSE points rather than only keeping the best result (dse_4),

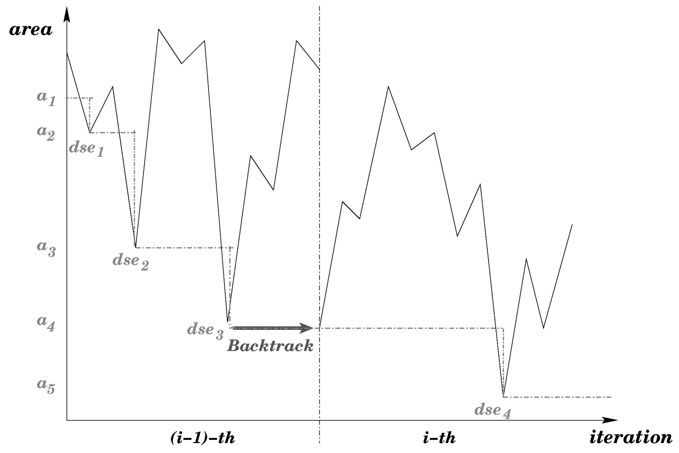


Fig. 4. Example of DSE based on variant of KLFM.

we provide more pareto-optimal design points that may not be discovered by adding more PEs.

4.2 Scheduling for Energy Reduction

For a given mapping, most of the previous work provides customized static scheduling results [Schmitz et al. 2005; Yang et al. 2001; Oh and Ha 1996]. Given that such static scheduling approaches do not scale well with rapidly increasing design complexity, we focus on utilizing generic real-time OS scheduling policies such as EDF and RM with an appropriate resource model. A suitable resource model additionally enables us to support a light-weight global power-management scheme through the encapsulation of each PE's scheduling information with a generic interface. More specifically, we use the PRM (periodic resource model) proposed in [Shin and Lee 2003] for schedulability test and encapsulating detailed scheduling information. It should be pointed out that our framework is not limited to periodic resource model. We choose PRM, since it is well-suited for dynamic power-management as well as applicable to both fixed priority (RM) and dynamic priority (EDF) real-time scheduling policies.

For ease of understanding of our energy reduction technique, we briefly outline the key aspects of the PRM as proposed in [Shin and Lee 2003]. Its goal is to provide compositional hard real-time guarantees in a hierarchy of schedulers to support resource sharing under different scheduling policies, such as EDF and RM. A scheduling model can be characterized by $M(W, \Gamma, A)$, where W, Γ, A represent the workload, the resource model, and a scheduling policy, respectively. One key property of such a model is that given any two components of the model (W, Γ), inferences can be made about the third component (A). In our framework, W is the set of periodic tasks, $A \in \{EDF, RM\}$, and we use the periodic resource model $\Gamma(\Pi, \Theta)$ to characterize a resource allocation of Θ time units out of every Π time units. We essentially use the PRM property that given W and A, we can generate solutions for the periodic resource model $\Gamma(\Pi, \Theta)$ that makes the model $M(W, \Gamma, A)$ schedulable. The obtained periodic

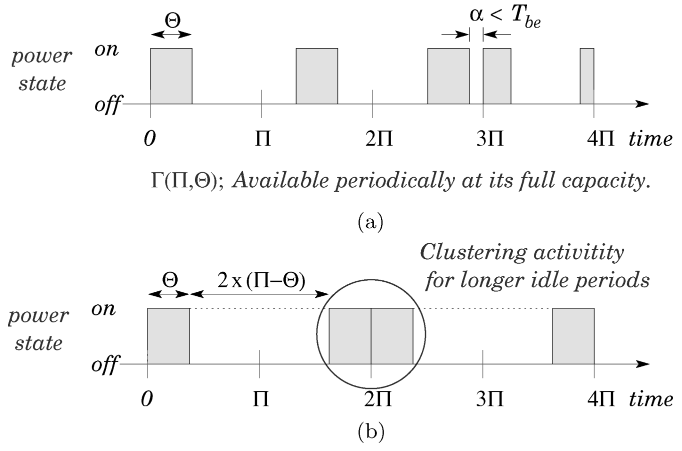
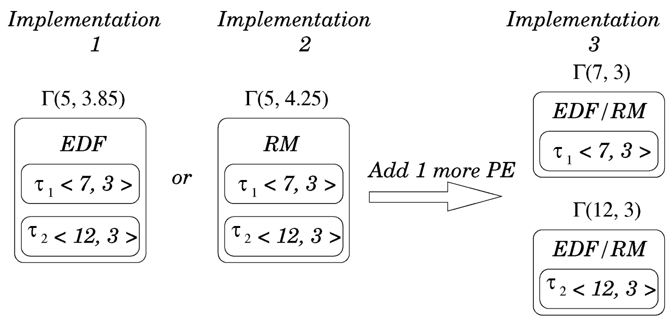

Θ
α < T_{be}
power state
on
off
0
Π
2Π
3Π
4Π
time
Γ(Π,Θ); Available periodically at its full capacity.
(a)
Clustering activitity for longer idle periods
2 x (Π–Θ)
(b)

Fig. 5. Energy reduction by (a) periodic resource model and (b) clustering of idle periods.

resource $\Gamma(\Pi, \Theta)$ can be used as a generic interface between individual PEs and the run time manager. Thus, the burden of complex scheduling is distributed through abstraction and compositional real-time guarantees. In addition, this abstraction (where at most Θ time units of resource are required every Π time units) enables simplified dynamic power-management, as discussed next.

Dynamic power-management (DPM) [Benini et al. 2000] is a widely used strategy for reducing system energy consumption. The key idea underlying all DPM-based approaches is to put a device into a low-power (and low performance) state to save energy when the device is not serving any request during a suitably long time period determined by the shutdown and wakeup overhead of the device. This device-dependent parameter is typically referred to as the *breakeven time* (T_{be}), the minimum idle time of a device that compensates for state transition overheads. As discussed earlier, we can abstract the workload as a periodic resource model that requires only Θ time units of resource every Π time units. Assuming that a PE is active (executing the functions mapped to the PE) for the first Θ time units per Π period, it is idle for the remaining $(\Pi - \Theta)$ time units. Therefore, it is beneficial to shut down the PE if T_{be} is less than $(\Pi - \Theta)$. Ideally T_{be} is zero, leading to an energy reduction ratio of $(1 - \frac{\Theta}{\Pi})$.

One key aspect of the PRM that enables additional energy reduction by DPM is shown in Figures 5a and 5b. Our discussions so far indicate that the workload requirement of Θ time units of resource every Π time units can be met by positioning Θ anywhere in an interval of length Π. However, if α, the idle-time between two successive active periods, is less than the breakeven time, it is not beneficial to shut down the PE. Based on the PRM abstraction, we can reduce the effect of the state transition overhead by clustering the idle periods as shown in Figure 5b. After clustering, the PE can be shut down if $T_{be} \leq 2 \times (\Pi - \Theta)$. Considering the time interval $[0, 4 \times \Pi]$, clustering leads to two long idle periods, resulting in an energy reduction ratio of $1 - \frac{4\times\Theta+2\times T_{be}}{4\times\Pi}$. Without clustering, there are three shorter idle periods as shown in Figure 5a,



Implementation	*1*	*2*	*3*
Number of PEs (Area Cost)	*1*	*1*	*2*
Utilization (Energy Consumption)	*0.77*	*0.85*	*0.68*

Fig. 6. Energy reduction by adding more PEs.

resulting in an energy reduction ratio of $1 - \frac{4\times\Theta+3\times T_{be}+\alpha}{4\times\Pi}$. Comparing the two approaches, clustering results in additional energy reduction by a ratio of $\frac{T_{be}+\alpha}{4\times\Pi}$.

Next, we consider the scheduling overhead of the top-level scheduler. From an implementation viewpoint, system-level power-management (coordinated by process scheduling techniques in the operating system) can improve energy profile drastically. For instance, each task can notify the scheduler about the usage periods of resources and the scheduler can rearrange the execution order of tasks to cluster idle periods of devices leading to additional energy reduction, as discussed earlier. Without any scheduling overhead, we can repeatedly shut down and wake up a PE for an infinitesimally short period (if we assume $T_{be} = 0$, for simplicity). That is, for smaller scheduling periods, finer-grain power-management becomes possible. However, this increases the context switching overhead, and for suitably small periods Π, the top-level scheduler will spend most of its time generating control signals to the PEs. In the experimental section, we study, in detail, the issues related to choice of a suitable Π.

4.3 Design Space Exploration

The primary goal of our framework is to enable design space exploration where a designer can evaluate various design alternatives in the context of multiple design objectives, such as minimum hardware-cost design, minimum energy design, etc. We, therefore, provide multiple pareto-optimal points from which a designer can select a suitable operating point. In our framework, we enable design space exploration by adding more PEs, as shown in Box *C* of Figure 3.

The PRM allows encapsulation of detailed scheduling information at the cost of utilization—this opens up the possibility of reducing system energy by adding more PEs. As an example, consider the workload shown in Figure 6, $W = \{\tau_1, \tau_2\}$ with the following parameters for some PE type: $\{\langle period_1, wcet_1\rangle, \langle period_2, wcet_2\rangle\} = \{\langle 7, 3\rangle, \langle 12, 3\rangle\}$, i.e., period of task τ_1 is 7, worst-case execution time of τ_1 on the PE is 3, etc. For the EDF scheduling policy, the periodic resource capacity bound for resource period $\Pi = 5$ is 0.77

with $\Theta = 3.85$. Given that the original utilization is 0.68 ($\frac{3}{7} + \frac{3}{12}$), the periodic resource model has an overhead of 9%. For RM scheduling policy the resource capacity bound is $\Theta = 4.25$, i.e., the overhead is 17%.

However, this overhead is eliminated if there is enough area slack allowing the design to include another instance of the PE, as shown in *Implementation 3* of Figure 6. Thus, if we consider the area of the PE to be 10 units, and the energy consumption to also be 10 units, we have three alternative implementations with $\{area, energy, scheduling\ policy\}$:

Implementation 1: $\{10, 7.7, EDF\}$ with 1 PE
Implementation 2: $\{10, 8.5, RM\}$ with 1 PE
Implementation 3: $\{20, \mathbf{6.8}, EDForRM\}$ with 2 PEs

The above example demonstrates the potential of reducing energy by adding more PEs. When we add a new PE, the mapping of functional blocks also needs to be modified (as shown in Box *D* of Figure 3)—in our current implementation, we exhaustively explore all feasible mappings. Given a set of feasible solutions, our framework extracts the pareto-optimal points by analyzing the tradeoff between area cost and energy consumption.

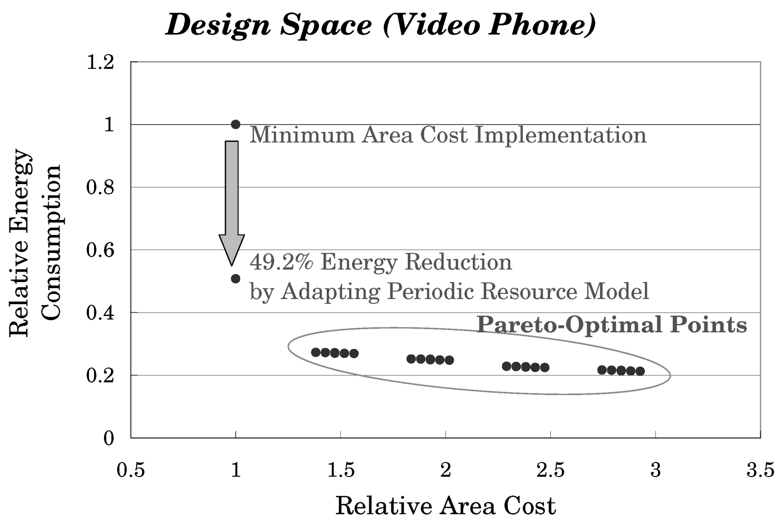
5. EXPERIMENTS

We evaluated the effectiveness of our framework by carrying out a variety of experiments. Our first set of experiments focused on design space exploration for the multimode multimedia terminal (MMMT) application (Figure 1), discussed previously in Section 1. In the next set of experiments, we focused on multimode execution of the MMMT system and explored effectiveness of resource sharing in reducing energy consumption for different usage profiles. In our final set of experiments, we focused on the effect of top-level scheduling granularity in the context of the video phone application.

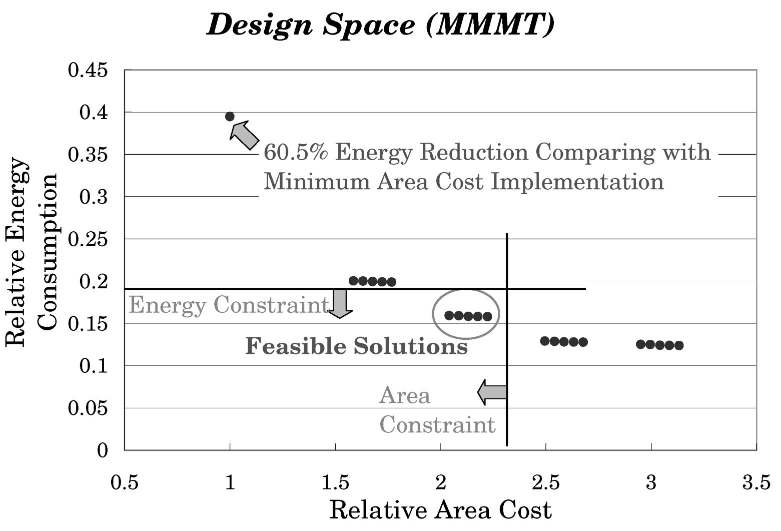
5.1 Design Space Exploration for MMMT

Figures 7 and 8 show the results of design space exploration with partitioning generated by two-approximation and modified KLFM, respectively. In each figure, the x axis represents the relative area cost compared to that of [Oh and Ha 2002], while the y axis represents the relative energy consumption compared to an approach without power-management. For these experiments, we used the PE library from [Oh and Ha 2002]. We assumed energy consumption to be proportional to the area cost [Bijlsma et al. 2006; Hill 2001; Flautner et al. 2004]. We assumed that the usage profile is identical for each mode, i.e., 33% for this set of experiments. We also set the scheduling period of the top-level scheduler as $0.5 \times min(periods\ of\ \ blocks)$, and the same weight values for *energy* and *area cost* in Eq. (1).

The initial partitioning (Box *A* of Figure 3) has effect on the solution quality. We first present design space from initial partitioning by two-approximation that only considers area cost. In Figure 7a, we see the energy reduction obtained by periodic resource model and dynamic power-management. Compared to the minimum area solution from [Oh and Ha 2002], our approach generates



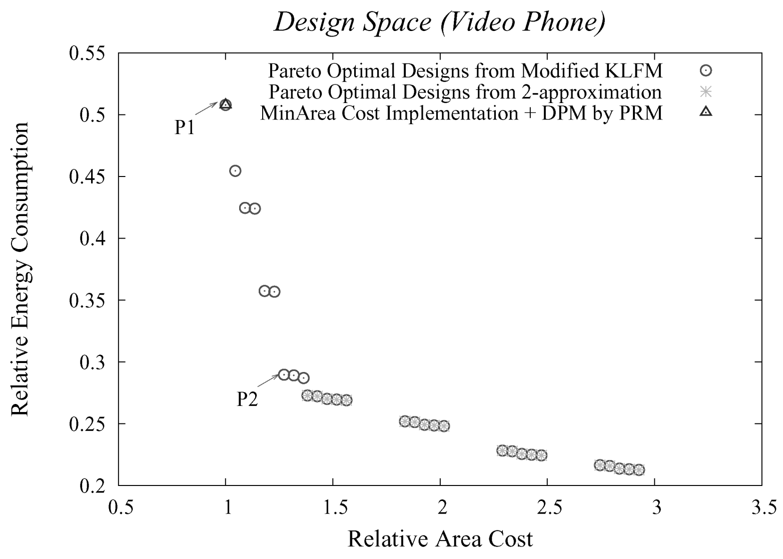
(a)



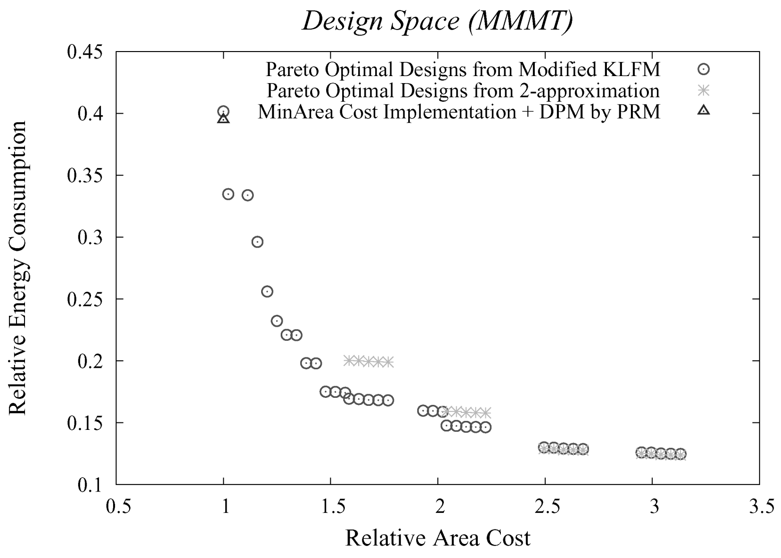
(b)

Fig. 7. Design space exploration (area cost - energy consumption) from two-approximation (a) video phone; (b) MMMT.

a cosynthesis solution with 49.2% energy reduction for the same area cost. As mentioned earlier, each feasible solution includes mapping, scheduling policy, and dynamic power-management scheme. For the solution with minimum area cost, three PEs are selected: PE0(uP), IDCT, DCT. Out of the 14 functional blocks in video phone mode, 11 blocks are mapped onto PE0(uP) and the two IDCT blocks and one DCT block are mapped onto dedicated hardware. The best energy profile is obtained with EDF scheduling policy for PE0 with periodic resource model of $\Gamma(\Pi, \Theta) = (12, 11)$. The energy reduction ratio for the complete



(a)



(b)

Fig. 8. Design space exploration (area cost - energy consumption) from modified KLFM (a) video phone; (b) MMMT.

application (Figure 7b) is much higher, 60.5%, since the video phone mode is the most computationally intensive mode in the application.

The minimum energy design point is just one of the multiple pareto-optimal points that enable designers to choose appropriate solutions, based on their design objectives. As an example, if the area and energy constraints are 19% and 2.3 times of the value for the minimum area implementation point shown in

Figure 7b, the designers can select any one of the five feasible solutions, based on other decision criteria. It is important to note that the clustered points in 7a and 7b represent pareto-optimal solutions that are reasonably close in area and energy. As explained in Section 4.3, our strategy for design space exploration is to add one instance of a PE in each iteration leading to the possibility of energy reduction—some PEs such as DCT, DeQ, etc., have a relatively low area-power overhead resulting in closely clustered solutions, while other PEs such as Microprocessor, ME, etc., have high overhead leading to solutions that are significantly different.

Modified multiway KLFM significantly improves design space exploration, as shown in Figure 8. For tight area constraints, the KLFM strategy results in design points with significantly reduced energy consumption. Consider design point p_1 corresponding to the minimum area cost implementation. Design point p_2 generated by our KLFM based strategy with relative area cost of 1.318 has significantly reduced energy consumption (43.1% less).

- *Execution time of approach*—The execution time of our approach is very reasonable. The entire set of pareto-optimal points for the MMMT application with 28 functional blocks, 3 modes, and 13 PE types as shown in Figure 8b was obtained in 145.43 s on a 1.8-GHz Pentium processor running Linux.

5.2 Resource Sharing in Multimode Execution

Next, we validate our strategy of resource sharing between different modes based on Eq. (1) in Section 4.1. As an example, if the IDCT functional block in H.263 decoder task (Figure 1) is selected to be implemented on dedicated hardware $IDCT_{hw}$ in video phone mode, it might be beneficial to map the IDCT block in VoD player mode on $IDCT_{hw}$. Assume that the current mapping for MMMT contains PEs $\{P_1, IDCT_{hw}, ME_{hw}\}$ and all functional blocks in video phone mode are *locked*. In this situation, KLFM attempts to pick the best node and move based on Eq. (1), which leads to implementing IDCT block on existing PE $IDCT_{hw}$. Otherwise, implementation of FB block on PE FB_{hw} is selected as the best move with least penalty if we only consider VoD player mode. Thus, multimode resource sharing results in a solution with less area cost (PEs $\{P_1, IDCT_{hw}, ME_{hw}\}$) compared to a minimum area implementation without time-multiplexed resource sharing (PEs $\{P_1, IDCT_{hw}, ME_{hw}, FB_{hw}\}$).

We also examine the effect of different usage profiles, based on the key observation that operational modes can execute during uneven amounts of time. Figure 9 presents summary of our experiments. Consider the set of design points corresponding to higher probability (80%) of executing the most energy-intensive mode, i.e., the video phone mode. The energy reduction with such usage profile is expectedly less, compared to usage where the video phone mode has much lower probability (10%).

5.3 Effect of Scheduling Granularity

In this set of experiments, we study the effect of scheduling granularity of the top-level scheduler. We generated all possible mappings from the initial partitioning—for each mapping, we estimated the highest energy

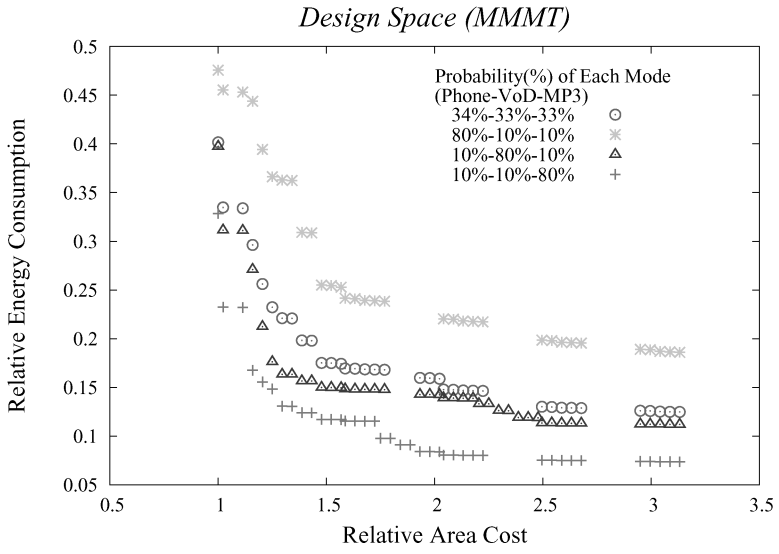


Fig. 9. Effect of mode probability.

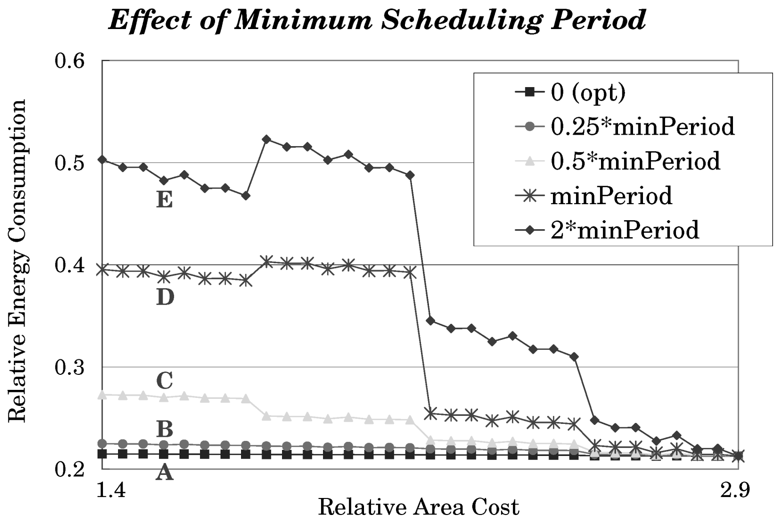


Fig. 10. Effect of minimum scheduling granularity.

reduction ratio for different scheduling granularities. We varied the minimum scheduling period from *zero* (the optimal case with no overhead) to $2 \times min(periods\ of\ blocks)$. The results are presented in Figure 10.

We first consider the two extremes indicated by the set of points A, E. For the set of points (E) with large scheduling period, there is much less scope for energy reduction with few PEs, since Θ will be saturated to Π to guarantee the timing requirements. However, as more PEs are added, the energy profile shows dramatic improvement. At the other extreme, for the set of points (A) with no scheduling overhead, Π can be a very small number—this implies that execution of the functional blocks will be fragmented and the ratio of $\frac{\Theta}{\Pi}$ will be saturated to the sum of the utilizations of the mapped tasks. In such a scenario, DPM will not provide additional energy reduction, even if more PEs are added. The remaining set of points *B—D* simply confirm that as scheduling period increases, the loss in utilization because of the periodic resource model abstraction leads to fewer opportunities for energy reduction with tighter area constraints.

6. CONCLUSION

In this paper, we proposed a generic and scalable cosynthesis framework for mapping modern real-time multimedia applications onto MPSoCs with library of heterogeneous scheduling policies. In our framework, a suitable scheduling policy is chosen for each PE (processing element)—our choice of a suitable resource model additionally enables energy reduction by a light-weight system-level dynamic power-management strategy. It includes resource-sharing considerations between different modes for multimode applications. Our experiments on design space exploration for a realistic multimedia application demonstrate the capability of our framework to generate multiple pareto-optimal design points with energy-cost tradeoffs—a design point with 60.5% energy reduction for a given area constraint also demonstrates the effectiveness of our power-management strategy. In future work, we will extend our proposed framework to consider shared system resources, such as memory, I/O, etc.

REFERENCES

Baruah, S. K. 2004. Cost efficient synthesis of real-time systems upon heterogeneous multiprocessor platforms. In *WPDRTS '04: Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems*. 120b.

Benini, L., Bogliolo, A., and Micheli, G. D. 2000. A survey of design techniques for system-level dynamic power management. *IEEE Trans. Very Large Scale Integr. Syst. 8,* 3, 299–316.

Bijlsma, T., Wolkotte, P. T., and Smit, G. J. 2006. An optimal architecture for a DDC. In *RAW '06: Proceedings of the 20th International Parallel & Distributed Processing Symposium Reconfigurable Architectures Workshop*.

Buttazzo, G. C. 2005. Rate monotonic vs. EDF: Judgment day. *Real-Time Syst. 29,* 1, 5–26.

Fiduccia, C. M. and Mattheyses, R. M. 1982. A linear time heuristic for improved network partitions. In *DAC '04: Proceedings of the 19th Annual Conference on Design Automation*. 241–247.

Flautner, K., Flynn, D., Roberts, D., and Patel, D. I. 2004. IEM926: An energy efficient SoC with dynamic voltage scaling. In *DATE '04: Proceedings of the Design, Automation and Test in Europe Conference and Exposition*. 324–329.

Helmig, J. Texas Instruments, 2002. Developing Core Software Technologies for TI's OMAP Platform.

Hill, S. 2001. The ARM10 family of advanced embedded microprocessor cores. In *HotChips '01: A Symposium on High Performance Chips*.

Kernighan, B. W. and Lin, S. 1970. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J. 49,* 2, 291–308.

Kim, M., Banerjee, S., Dutt, N., and Venkatasubramanian, N. 2006. Design space exploration of real-time multimedia MPSoCs with heterogeneous scheduling policies. In *CODES+ISSS '06: Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis*. 16–21.

Liu, C. L. and Layland, J. W. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM 20,* 1, 46–61.

Marculescu, R., Nandi, A., Lavagno, L., and Sangiovanni-Vincentelli, A. L. 2001. System-level power/performance analysis of portable multimedia systems communicating over wireless channels. In *ICCAD '01: Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*. 207–214.

Matic, S. and Henzinger, T. A. 2005. Trading end-to-end latency for composability. In *RTSS '05: Proceedings of the 26th IEEE International Real-Time Systems Symposium*. 99–110.

NXP. Nexperia Processor http://www.nxp.com/products/nexperia/home.

Oh, H. and Ha, S. 1996. A static scheduling heuristic for heterogeneous processors. In *Euro-Par '96: Proceedings of the Second International Euro-Par Conference on Parallel Processing-Volume II*. 573–577.

Oh, H. and Ha, S. 2002. Hardware-software cosynthesis of multimode multi-task embedded systems with real-time constraints. In *CODES '02: Proceedings of the 10th International Symposium on Hardware/software Codesign*. 133–138.

Pham, D., Anderson, H.-W., Behnen, E., Bolliger, M., Gupta, S., Hofstee, P., Harvey, P., Johns, C., Kahle, J., Kameyama, A., Keaty, J., Le, B., Lee, S., Nguyen, T., Petrovick, J., Pham, M., Pille, J., Posluszny, S., Riley, M., Verock, J., Warnock, J., Weitzel, S., and Wendel, D. 2006. Key features of the design methodology enabling a multi-core soc implementation of a first-generation cell processor. In *Proceedings of the Conference on Asia South Pacific Design Automation (ASP-DAC'06)*. 871–878.

Pop, P., Eles, P., Peng, Z., and Pop, T. 2006. Analysis and optimization of distributed real-time embedded systems. *ACM Trans. Des. Autom. Electron. Syst. 11,* 3, 593–625.

Schmitz, M. T., Al-Hashimi, B. M., and Eles, P. 2005. Cosynthesis of energy-efficient multimode embedded systems with consideration of mode-execution probabilities. *IEEE Trans. on CAD of Integrated Circuits and Systems 24,* 2, 153–169.

Shin, I. and Lee, I. 2003. Periodic resource model for compositional real-time guarantees. In *RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium*. 2–13.

Shin, I. and Lee, I. 2004. Compositional real-time scheduling framework. In *RTSS '04: Proceedings of the 25th IEEE International Real-Time Systems Symposium*. 57–67.

STMicroelectronics. ST Nomadik Multimedia Processor http://www.st.com/nomadik.

Wolf, W. 2004. The future of multiprocessor systems-on-chips. In *DAC '04: Proceedings of the 41st Annual Conference on Design Automation*. 681–685.

Yang, P., Wong, C., Marchal, P., Catthoor, F., Desmet, D., Verkest, D., and Lauwereins, R. 2001. Energy-aware runtime scheduling for embedded-multiprocessor SOCs. *IEEE Des. Test 18,* 5, 46–58.

Received December 2006; accepted May 2007