Design Space Exploration of Real-time Multi-media MPSoCs with Heterogeneous Scheduling Policies [*]

Minyoung Kim, Sudarshan Banerjee, Nikil Dutt, Nalini Venkatasubramanian
School of Information and Computer Sciences
University of California, Irvine
CA 92697-3425, USA
{minyounk, banerjee, dutt, nalini}@ics.uci.edu

ABSTRACT

Real-time multi-media applications are increasingly being mapped onto MPSoC (multi-processor 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 heterogenous 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 multi-mode multi-media 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
Keywords: Real-time Scheduling, Cosynthesis, MPSoC, Energy

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 (multi-processor system-on-chip) present a viable platform for these applications, since they typically contain multiple heterogeneous processing elements, memories, I/O components, etc. Indeed, several commercial MPSoC platforms are now available including the TI OMAP [6], ST Nomadik [16], Philips Nexperia [13], and STI Cell processor [4]. 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. Our framework additionally reduces system energy by dynamic power management techniques. In Figure 1, we consider an application driver, the MMMT (multi-mode multimedia terminal) [12] [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 [14, 17].

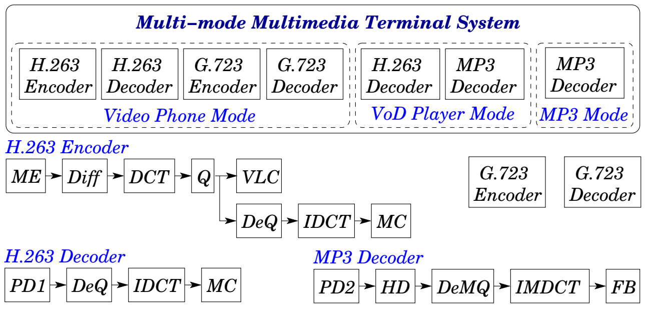


Figure 1: MMMT Example

[*]This work was partially supported by NSF Grant CNS-0615438.

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

Typical work in design space exploration for such multimedia systems proposes ad-hoc scheduling policies to minimize the energy [14]. However, such an approach adds the overhead of customized scheduling tables for each individual application. Additionally, 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).

Our framework enables us to perform design space exploration while considering three critical constraints: area, energy, performance. For a given multi-mode 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 constraint and timing constraints for the individual tasks. Experimental 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 [12], the authors provide a framework for cosynthesis of a MMMT system that is based on a non-preemptive 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 [14], 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 [15], 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.

Our goal is a cosynthesis methodology that guarantees real-time operation with low area/energy requirements. Unlike [12] and [14], 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 [15] focused on the theoretical principles that effectively allow a set of periodic tasks 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 their work for energy estimation of MPSoCs and build our dynamic power management techniques on top of the energy estimation framework for additional energy reduction.

3. PRELIMINARIES

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

3.1 Terminology and Assumptions

A multi-mode 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 pre-empted. The period of a sporadic task can be modeled as the minimum inter-arrival 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).

• *Buffering assumption*:
Typical functional blocks for audio-video streaming applications are completely pipelined – data dependencies between such blocks can be eliminated by appropriate buffering. In this work, we assume that appropriate buffering enables us to ignore data dependences and communication overhead during scheduling. Essentially, we assume that there are enough (and suitably organized) on-chip memory resources such that functional block execution dominates system performance and energy consumption – detailed considerations such as buffer sizes, buffer placement, etc., are beyond the scope of this version of our work.

• *No explicit resource sharing between modes*:
A partitioning approach that considers resource sharing between multiple modes is necessarily more complex [12]. Since our initial focus is on exploiting heterogeneity in scheduling policies for minimizing energy, we currently do not consider explicit resource sharing.

3.2 Problem Statement

Given a multi-mode application and a library of candidate processing elements, our problem is to find design implementations that are Pareto-optimal in cost-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

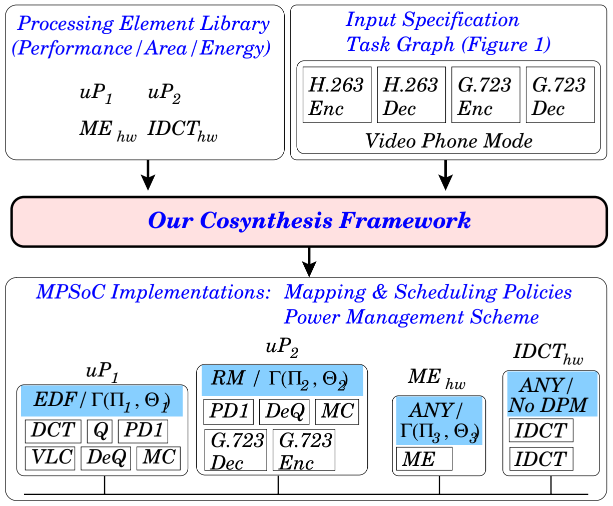


Figure 2: Cosynthesis Problem for Real-time MPSoCs with Heterogeneous Scheduling Policies

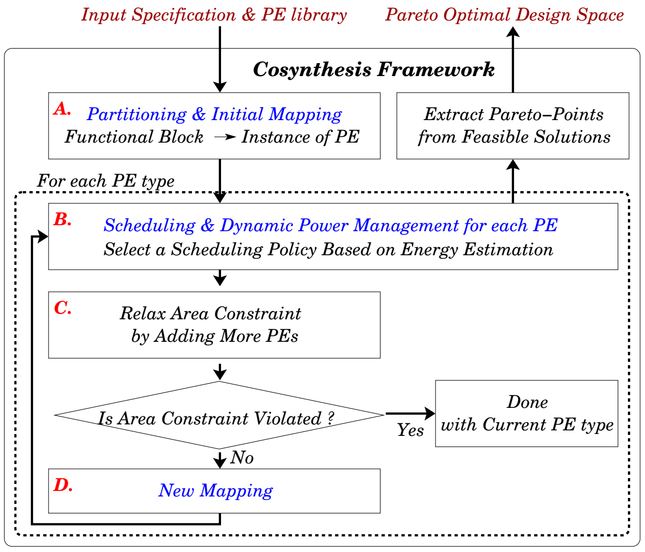
policy for each PE. As an example, in the MPSoC implementation shown in Figure 2, PE uP_1 is executing the EDF scheduling policy, and, the functional block DCT is mapped to uP_1. Our problem is described more formally below:

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

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 2 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.

[2] We can reasonably assume that power consumption is constant for PEs executing typical multimedia functions such as DCT, ME, etc [10].



Input Specification & PE library
Pareto Optimal Design Space
Cosynthesis Framework
A. Partitioning & Initial Mapping
Functional Block → Instance of PE
Extract Pareto–Points from Feasible Solutions
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

Figure 3: Flow of Our Cosynthesis Framework

4. PROPOSED APPROACH

We present an overview of our proposed cosynthesis framework in Figure 3. We first generate a partitioning considering area cost only, 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. We use the 2-approximation algorithm proposed in [1] – for EDF scheduling policy, 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 [1] for partitioning (and initial mapping) guarantees that our set of feasible solutions contains at least one design point with well-defined properties.

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

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 [9] 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 model [15] 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 4.2, 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.

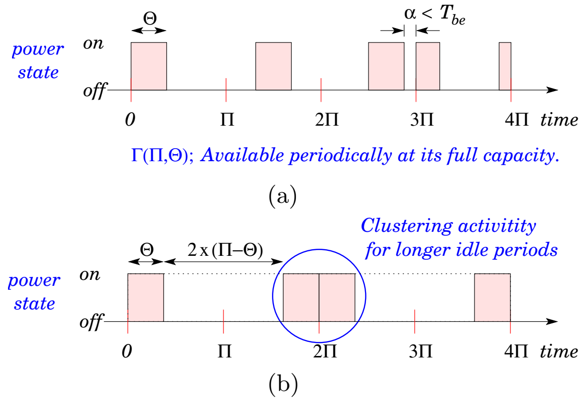
4.1 Scheduling for Energy Reduction

For a given mapping, most of the previous work provides customized static scheduling results [14, 11]. 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 lightweight 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 [15] 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 [15]. 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 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. Additionally, 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) [2] 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 4(a) and 4(b). 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 shutdown 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 4(b). After clustering, the PE can be shutdown 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 4(a), 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 $\frac{T_{be}+\alpha}{4\times\Pi}$.



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

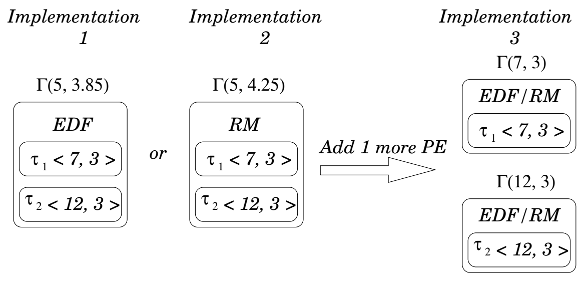
Figure 4: Energy Reduction by (a) Periodic Resource Model and (b) Clustering of Idle Periods

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.

4.2 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.



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

Figure 5: Energy Reduction by Adding More PEs

As an example, consider the workload shown in Figure 5, $W = \{\tau_1, \tau_2\}$ 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 5. 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 multi-mode multimedia terminal (MMMT) application (Figure 1) discussed previously in Section 1. In our second 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 6(a) and 6(b) show the results of design space exploration for the video-phone mode and the complete MMMT application, respectively. In each figure, the X-axis represents the relative area cost compared to that of [12] 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 [12]. We assumed energy consumption to be proportional to the area cost [3, 7, 5]. We assumed that the usage profile is identical for each mode, i.e., 33% [3]. We also set the scheduling period of the top level scheduler as $0.5 \times min(periods\ of\ blocks)$.

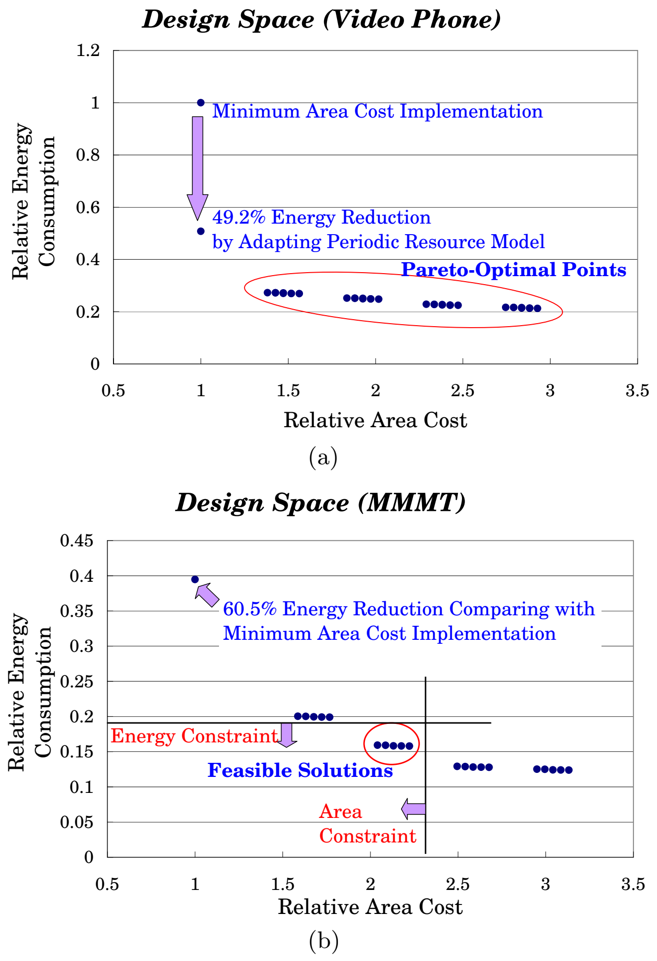


Figure 6: Design Space Exploration (Area Cost - Energy Consumption) (a) Video Phone, (b) MMMT

In Figure 6(a), we see the energy reduction obtained by periodic resource model and dynamic power management. Compared to the minimum area solution from [12], our approach generates 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), 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 application (Figure 6(b)) 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

[3]More detailed experiments with different usage profiles are available in [8].

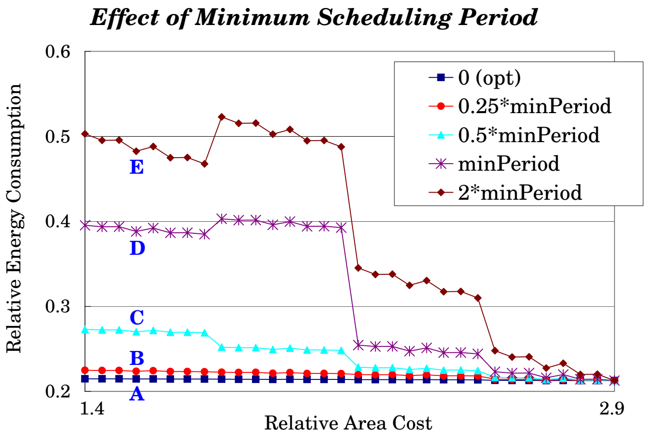


Figure 7: Effect of Minimum Scheduling Granularity

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 6(b), 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 6(a) and 6(b) represent Pareto Optimal solutions that are reasonably close in area, energy. As explained in 4.2, 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.

• *Execution time of approach:* The execution time of our approach is very reasonable – the entire set of Pareto-Optimal points for the MMMT application as shown in 6(b) was obtained in 6.55 seconds on a 1.8 GHz Pentium processor running Linux.

5.2 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 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 7.

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 due to the periodic resource model abstraction leads to fewer opportunities for energy reduction for 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. 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. More specifically, we plan to relax our *ideal* buffering assumptions and focus on integrating realistic memory organization (and communication) aspects from commercial platforms into our scheduling formulation.

7. REFERENCES

[1] S. K. Baruah. Cost Efficient Synthesis of Real-Time Systems upon Heterogeneous Multiprocessor Platforms. In *WPDRTS '04*, page 120b.

[2] L. Benini, A. Bogliolo, and G. D. Micheli. A Survey of Design Techniques for System-level Dynamic Power Management. *IEEE TVLSI*, 8(3):299–316, 2000.

[3] T. Bijlsma, P. T. Wolkotte, and G. J. Smit. An Optimal Architecture for a DDC. In *RAW '06*.

[4] D. Pham et al. Key Features of the Design Methodology Enabling a Multi-core SoC Implementation of a First-Generation CELL Processor. In *ASP-DAC '06*, pages 871–878.

[5] K. Flautner, D. Flynn, D. Roberts, and D. I. Patel. IEM926: An Energy Efficient SoC with Dynamic Voltage Scaling. In *DATE '04*, pages 324–329.

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

[7] S. Hill. The ARM10 Family of Advanced Embedded Mocroprocessor Cores. In *HotChips '01*.

[8] M. Kim, S. Banerjee, N. Dutt, and N. Venkatasubramanian. Scheduling Policy Selection for Cosynthesis of Real-time Multimedia applications onto Heterogeneous MPSoCs. CECS Technical Report, UC Irvine, May 2006.

[9] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM*, 20(1):46–61, 1973.

[10] R. Marculescu, A. Nandi, L. Lavagno, and A. L. Sangiovanni-Vincentelli. System-Level Power/Performance Analysis of Portable Multimedia Systems Communicating over Wireless Channels. In *ICCAD '01*, pages 207–214.

[11] H. Oh and S. Ha. A Static Scheduling Heuristic for Heterogeneous Processors. In *EuroPar '96*, pages 573–577.

[12] H. Oh and S. Ha. Hardware-software Cosynthesis of Multi-mode Multi-task Embedded Systems with Real-time Constraints. In *CODES '02*, pages 133–138.

[13] Philips. Nexperia Processor http://www.semiconductors.philips.com/products/nexperia/.

[14] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Cosynthesis of Energy-efficient Multimode Embedded Systems with Consideration of Mode-execution Probabilities. *IEEE TCAD*, 24(2):153–169, 2005.

[15] I. Shin and I. Lee. Periodic Resource Model for Compositional Real-Time Guarantees. In *RTSS '03*, pages 2–13.

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

[17] W. Wolf. The Future of Multiprocessor Systems-on-chips. In *DAC '04*, pages 681–685.