Power-Aware Mapping of Probabilistic Applications onto Heterogeneous MPSoC Platforms

Andreas Schranzhofer, Jian-Jia Chen, Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
Email: {schranzhofer,jchen,thiele}@tik.ee.ethz.ch

Abstract

Multiprocessor SOC platforms have been adopted for a wide range of high performance applications. Task assignment and processing unit allocation are key steps in the design of predictable and efficient embedded systems. Provided that the probability distributions and mutual exclusion conditions for executing applications are known a priori, this paper explores the mapping of tasks onto processing units while minimizing the expected average power consumption. The underlying model considers static (leakage) and dynamic power. This study shows that deriving approximative solutions with a constant worst-case approximation factor in polynomial time is not achievable unless $\mathcal{P} = \mathcal{NP}$, even if a feasible task mapping is provided as an input. A polynomial-time heuristic algorithm is proposed that applies a multiple-step heuristic. Experimental results reveal the effectiveness of the proposed algorithm by comparing the derived solutions to the optimal ones, obtained via an integer linear program (ILP) specification.

Keywords: Power-Aware Allocation, Probabilistic Applications, MPSoC.

1. Introduction

Multiprocessor System-on-Chips (MPSoCs) typically are composed of multiple processors (processing units), memories, and a communication infrastructure. Heterogeneous MPSoCs contain different types of processing units. Therefore, system designers can take advantage of their properties when mapping tasks to specific processor types and optimize criteria such as computational performance, cost and energy consumption. A number of commercial multimedia platforms that use MPSoCs to meet performance demands are developed, e.g., NXP Nexperia and ST Nomadik [1].

Power management and power-aware computing are increasingly important design issues for both embedded systems and server systems. The power consumption influences not only the operating cost of such systems but also the lifespan, as increased power consumption results in increased peak or average temperature. The overall power consumption is mainly caused by a dynamic part due to switching activities and a static part due to leakage current, see e.g., [2].

Power-aware and energy-efficient scheduling for multiprocessor systems have been explored widely in recent years in both academics and industry, especially for real-time systems, e.g., [3]–[5], whereas [6] provides a comprehensive survey. However, only a few results have been developed for power-awareness or energy-efficiency in heterogeneous multiprocessor systems. For example, heuristics and approximation algorithms to minimize the dynamic energy consumption for dynamic voltage scaling (DVS) systems in [7]–[10].

Unfortunately, in nano-meter manufacturing, leakage current contributes significantly to the power consumption of a system e.g., [2]. Xu et al. [11] and Chen et al. [3] apply DVS techniques and the possibility to turn off processors to reduce the energy consumption in homogeneous multiprocessor systems. Rusu et al. [10] develop heuristic power-aware strategies for server farms with heterogeneous servers.

Related approaches focus on a model where tasks have to be activated within a specified time span, i.e., with an activation pattern that is known in advance. In contrast, this paper studies the concurrent execution of applications whose execution can be characterized by probability distributions rather than time spans and activation patterns. This model is common in various application domains such as multimedia processing, media terminals, mobile phones, and software defined radio (SDR) [12]–[15], just to name a few. In addition, future MPSoC platforms will not only be used by a single (static) application but by multiple concurrent applications that are independent and whose activation pattern is unknown. Nevertheless, an optimization of the hardware platform and the task mapping with respect to hardware resources, static (leakage) and dynamic power consumption is necessary.

For example, a single SDR application consists of a set of tasks and several of these applications are typically executed concurrently. In addition, the concurrency is typically

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 216008 project Predator and Taiwan National Science Council NSC-096-2917-I-564-121.

constrained by mutual exclusion conditions between some of these applications. A single SDR application often represents a signal or media processing algorithm which involves the parallel execution of tasks. Often, the corresponding execution times, power consumption, and rate characteristics have been specified or can be obtained very accurately.

Given such a set of applications, the underlying heterogeneous MPSoC hardware platform can be adapted to the characteristics of the application domain. In addition, there is a large degree of freedom in mapping the individual tasks to the allocated processing elements. In this way, it is possible to dynamically adapt to varying resource requirements and optimize the average power consumption. The mapping of tasks to the computational resources admits a fine-grained power management by switching off processing units that are not used or slowing down processing units that are not fully utilized.

Work highly related to the probabilistic application model is done by Kim et. al. [1] and Schmitz et. al. [16]. Specifically, in [1] a heuristic algorithm to reduce the dynamic energy consumption, which is related to the utilization, is proposed. This is done by adding processing elements up to an area constraint, thereby reducing the average utilization. Their application model considers probabilities of execution for modes. The power model presented in this paper considers static and dynamic power as part of the objective to be minimized. In contrast, [1] only considers dynamic power consumption in the objective and an area constraint that has to be satisfied. Schmitz et. al. [16] consider probabilistic execution of multi-mode applications. They propose a genetic algorithm and four different mutation strategies to reduce the energy dissipation.

In this paper we study the above described mapping of a set of applications onto a heterogeneous MPSoC. The hardware platform is not fixed a priori but is composed by allocating processing units from a given library of processing unit (PU) types. The mapping process creates a feasible hardware platform from this library by instantiating processing units and determines a task assignment that satisfies the computational demands of the applications while minimizing the static and dynamic power consumption. Applications are defined by their task graphs, mutual exclusion conditions and execution probabilities. *Scenarios* define sets of applications that can execute concurrently without violating exclusion conditions.

The studied problem is to derive an optimal hardware platform and an optimal task mapping, such that the average power consumption is minimized while satisfying the execution constraints of all possible scenarios.

There are several research results for energy-efficient and power-aware designs in heterogeneous multiprocessor systems with non-negligible leakage power consumption, see e..g. [10], [17]. In these approaches, static usage scenarios are assumed and probabilities of applications are not considered. As a result, the worst case that all applications could run concurrently would need to be considered. This not only results in an over-dimensioned MPSoC platform but also in a non-optimal task mapping which overestimates the *average power consumption*. It is implausible for the considered application domain, e.g., SDR systems, to assume that all applications are active all the time.

In this paper we explore the minimization of expected average power consumption for applications with probabilistic distributions. The major contributions of our paper are:

- We show that there is no polynomial-time approximation algorithm with constant approximation factor unless $\mathcal{P} = \mathcal{NP}$.
- We propose a multiple-step approach to solve the problem in polynomial time by (1) computing an initial solution which assigns the tasks to their most effective processing unit types and (2) applying a greedy heuristic algorithm to remap tasks, thereby reducing the expected average power consumption.
- Experiments show the effectiveness of the proposed algorithm in terms of expected average power consumption and computation time.

The rest of this paper is organized as follows: Section 2 introduces the models and defines the studied problem. Section 3 presents the proposed algorithm, while Section 4 provides experiments for performance evaluation. Section 5 concludes this paper.

2. System Model

In this section, the hardware as well as the application models with the underlying assumptions and terminology are introduced. The hardware model describes processing units selected from a set of processing unit types. Processing units are described by their computational resources and by their power consumption. The application model considers multiple concurrently executing applications, composed of tasks. Mutually exclusive applications form application groups. Table 1 gives an overview of regularly used variables. A formal problem statement and a complexity analysis of the optimal solution is given at the end of the section.

2.1. Hardware Model

The hardware platform is based on a set ($\mathcal{P}$) of available processing unit (PU) types. A PU type $p \in \mathcal{P}$ is characterized by its *available computational resources per time unit* λ, e.g., measured in terms of operations or execution cycles per time unit, and its static and dynamic *power consumption*: σ and δ, respectively. For example, if a task has a *computational demand* of γ on a specific resource type (again measured in operations or execution cycles per time unit) and it is mapped on a resource instance of that

Processing Unit		Application/Scenario		Task	
$\mathcal{P}$	Library of PU types	$\mathcal{S}$	Set of Scenarios	$\mathcal{T}$	Set of Tasks
p_j	Processing unit type j	S_s	Scenario s	$\gamma_{i,j}$	computational resource demand of task t_i on PU type p_j
$\hat{p}_{j,k}$	instance k of PU type p_j	χ_s	Probability of Scenario S_s	$u_{i,j}$	utilization of task t_i on PU type p_j
σ_j	static power consumption of PU type p_j	A_m	Application m	ψ_i	Probability of task t_i
δ_j	dynamic power consumption of PU type p_j	$\hat{\chi}_m$	Probability of application A_m	$M_{i,j,k}$	binary variable to indicate mapping of task t_i to $\hat{p}_{j,k}$
λ_j	computational resources of PU type p_j	$Z_{s,j,k}$	binary variable to indicate Scenario S_s on $\hat{p}_{j,k}$	$\theta_{i,s}$	binary variable to indicate task t_i to Scenario S_s assignment

Table 1. Overview of regularly used variables.

type with *available computational resources* of λ, then the corresponding utilization of the resource instance is $\frac{\gamma}{\lambda}$.

Static power σ consumption describes the leakage power consumed by a processing unit type, independent of whether tasks are executed or not. Dynamic power consumption δ describes the additional power consumed by a processing unit type depending on its utilization.

As an example, suppose that a PU type p_j has a utilization of α, i.e., tasks are executing for an α fraction of the time span τ. Then, $\alpha\delta_j + \sigma_j$ is the *average power consumption* of the processing unit. The corresponding energy consumption for τ time units is $\tau \cdot (\alpha\delta_j + \sigma_j)$.

Given the available PU Types, a concrete hardware platform is constructed by instantiating processing units with unit types from $\mathcal{P}$. Any number $k \in \mathbb{N}$ of instances is allowed for each PU type. Constructing a feasible hardware platform is part of the mapping problem.

2.2. Application and Scenario Specification

An application A is described as a *task graph*. A task t_i in the given task set $\mathcal{T}$ is described by its computational resource demand $\gamma_{i,j}$ per time unit on a given PU type $p_j \in \mathcal{P}$. The utilization $u_{i,j}$ of a task t_i mapped onto PU type p_j is given as

$$u_{i,j} = \frac{\gamma_{i,j}}{\lambda_j}. \tag{1}$$

By definition, the total resource demands of tasks mapped on a processing unit cannot exceed the available computational resources. As a result, the total utilization of tasks mapped on a processing unit is constrained not to exceed 100%. For each task t_i, we assume that there is at least one PU type p_j with utilization $u_{i,j} \leq 100\%$; otherwise, there is no feasible solution for completing the task in time.

Mutually exclusive applications that cannot execute concurrently form an application group $\hat{\mathcal{A}}$. As a result, applications in different application groups can execute concurrently. If an application has no exclusion condition with any other applications, it forms a separate application group by itself.

An application A_i is characterized by its probability of execution $\hat{\chi}_i$, i.e., the probability that an application executes

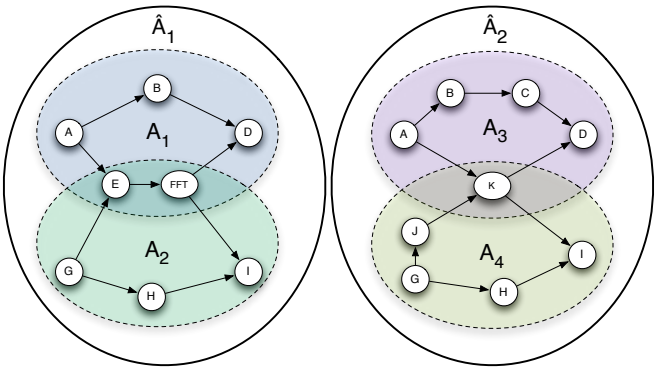

Figure 1. Application groups with two applications each and shared tasks.

on the hardware platform at a certain time instance is given by $\hat{\chi}_i$. These probabilities are either given by the system designer or can be obtained by proper profiling of the whole system. As has been mentioned already, this model is common for many application domains where time spans and activation patterns of applications are unknown statically.

Fig. 1 shows an example of two different application groups, each consisting of two applications with shared tasks. Such an exclusion condition is common in SDR (Software Defined Radio) applications. Radio communication systems such as Wireless Local Area Network (WLAN) and Digital Video Broadcast Handhelds (DVB-H) require tasks such as Fast Fourier Transform (FFT) and decode [14], [18] that are shared by multiple applications. This situation is depicted for sync/send and receive in Table 2.

The above concept of application groups result in a number of possible combinations of concurrently active applications. These combinations form a set of *Scenarios* $\mathcal{S}$, i.e., a single scenario consists of a set of applications that can execute concurrently. Figure 2 shows the scenarios resulting from Fig. 1. Obviously, the scenarios for application groups $\hat{\mathcal{A}}_1, \ldots, \hat{\mathcal{A}}_K$ can be computed by the Cartesian product $\hat{\mathcal{A}}_\mathbf{1} \times \hat{\mathcal{A}}_\mathbf{2} \times \cdots \times \hat{\mathcal{A}}_\mathbf{K}$. Conclusively a scenario can be composed by at most one application from each application

Application groups	$\hat{\mathcal{A}}_1$ (DVB-H)		$\hat{\mathcal{A}}_2$ (WLAN)	
Applications	A_1 (**sync**)	A_2 (**receive**)	A_3 (**send**)	A_4 (**receive**)
Tasks	FFT_1 phase_shifting_1 noise_detection_1 $\text{frequency\_shifting}_1$ $\vdots$	FFT_1 phase_shifting_1 $\text{payload\_decoding}_1$ video_decoding_1 $\vdots$	FFT_2 signal_generation encrypting encoding $\text{frequency\_shifting}_2$ $\vdots$	FFT_2 phase_shifting_2 decrypting $\text{payload\_decoding}_2$ $\text{frequency\_shifting}_2$ $\vdots$
Probability of applications	$\hat{\chi}_1$	$\hat{\chi}_2$	$\hat{\chi}_3$	$\hat{\chi}_4$

Table 2. An example for different applications in DVB-H and WLAN.

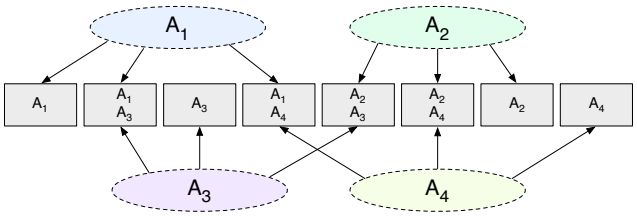

Figure 2. scenarios derived from application groups and applications, where the rectangles indicate scenarios.

group. The probability of executing a Scenario $S_s \in \mathcal{S}$ is denoted as χ_s and can be computed as the product of its applications probabilities, i.e., $\chi_s = \Pi_{A_i \in S_s} \hat{\chi}_i$.

The probability of execution ψ_i of a task t_i depends on the probabilities of those scenarios, it is present in and computes as:

$$\psi_i = \sum_{S_s \in \mathcal{S}} \chi_s \theta_{i,s}. \tag{2}$$

The binary variable $\theta_{i,s}$ indicates whether a task t_i is present in scenario S_s, i.e., $\theta_{i,s} = 1$ if task t_i is in one of the applications $A \in S_s$ and $\theta_{i,s} = 0$ otherwise.

2.3. Problem Definition

The problem explored in this paper is to find a mapping of tasks in $\mathcal{T}$ onto a hardware platform which consists of processing units from a given set of PU types $\mathcal{P}$. The selection of used processing unit types and the corresponding number of processing units is part of the problem. As described above, applications are characterized by their probabilities of execution. Therefore, the objective is to minimize the *average expected power consumption*. Once the time span of the system execution is known, the expected average energy consumption can be computed.

Beside selecting the optimal number of processing units, the mapping needs to determine the binding of a task t_i to an allocated processing unit $\hat{p}_{j,k}$ of PU type $p_j \in \mathcal{P}$. A task is mapped onto a PU $\hat{p}_{j,k}$ while respecting the maximum utilization constraint for all possible scenarios.

We assume that the number of possible instances k per PU type p_j is limited to be no more than F_j and for each task t_i there is at least one PU type p_j on which it can be executed, i.e., $u_{i,j} \leq 1$. Hence, there exists a feasible solution to the mapping problem and any feasible solution will at most use $|\mathcal{T}|$ instances of PU type $p_j \in \mathcal{P}$. Therefore, we consider $F_j \leq |\mathcal{T}|$.

The binary variables $M_{i,j,k}$ indicate which processing units to map task t_i. Let $M_{i,j,k} = 1$ if task t_i is assigned to PU $\hat{p}_{j,k}$ and $M_{i,j,k} = 0$ otherwise. Once we have $M_{i,j,k} = 1$, task t_i consumes a portion of the dynamic power δ_j on PU $\hat{p}_{j,k}$. This portion is related to the tasks utilization $u_{i,j}$ and its probability of execution ψ_i.

Furthermore, the binary variable Z indicates which processing units are involved in which scenarios, i.e., whether they need to execute at least one task in a specific scenario. We define $Z_{s,j,k} = 1$ if there exists a task t_i mapped to PU $\hat{p}_{j,k}$ (i.e., $M_{i,j,k} = 1$) such that $\theta_{i,s} = 1$ (i.e., it is present in scenario S_s) and $Z_{s,j,k} = 0$ otherwise. Once we have $Z_{s,j,k} = 1$, static power σ_j is consumed on $\hat{p}_{j,k}$ whenever scenario S_s is executed, i.e., with probability χ_s. The total utilization of the tasks in Scenario S_s mapped onto PU $\hat{p}_{j,k}$ is constrained to be no more than 100% when $Z_{s,j,k} = 1$, or 0% when $Z_{s,j,k} = 0$.

As a result of the above discussion, the expected average power consumption for a mapping described by M and Z can be determined as

$$\sum_{S_s \in \mathcal{S}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} \chi_s \sigma_j Z_{s,j,k} + \sum_{t_i \in \mathcal{T}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} u_{i,j} \delta_j \psi_i M_{i,j,k}, \tag{3}$$

where the first and second terms in Equation 3 represent the static and the dynamic power consumptions, respectively. The optimization problem can then be phrased by the following integer linear programming (ILP):

$$\min \sum_{S_s \in \mathcal{S}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} \chi_s \sigma_j Z_{s,j,k} + \sum_{t_i \in \mathcal{T}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} u_{i,j} \delta_j \psi_i M_{i,j,k} \tag{4a}$$

s.t.

$$\sum_{t_i \in \mathcal{T}} \theta_{i,s} M_{i,j,k} u_{i,j} \leq Z_{s,j,k}, \quad \begin{matrix} \forall p_j \in \mathcal{P}, S_s \in \mathcal{S}, \\ \forall k = 1, 2, \ldots, F_j \end{matrix} \tag{4b}$$

$$\sum_{p_j \in P} \sum_{k=1}^{F_j} M_{i,j,k} = 1, \, \forall t_i \in \mathcal{T}, \tag{4c}$$

$$M_{i,j,k} \in \{0,1\}, \, \forall t_i \in \mathcal{T}, p_j \in \mathcal{P}, k = 1, 2, \ldots, F_j, \tag{4d}$$

$$Z_{s,j,k} \in \{0,1\}, \, \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, 2, \ldots, F_j, \tag{4e}$$

where (4b) guarantees that no scenario violates the utilization constraints, and (4c) specifies that a task is mapped on exactly one processing unit.

We denote the studied problem as the *power-aware scenario-mapping* problem. Note that dynamic power management (DPM) is not adopted in the studied problem since the necessary timing information of applications is not contained in the problem definition. Extensions to adopt dynamic voltage scaling (DVS) can be integrated into our approaches, but are omitted due to space limitation.

2.4. Hardness

It is not difficult to see that the problem is $\mathcal{NP}$-hard in a strong sense, since the special case with one scenario and one processing unit type is the bin packing problem. Moreover, when there is a limitation of the allocated units, i.e., $F_j < |\mathcal{T}|$, deriving a feasible solution for Equation (4) is a $\mathcal{NP}$-complete problem since the bin packing problem is its special case. Therefore, unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial-time algorithm for deriving a feasible solution for any input instance that allows for a feasible mapping when $F_j < |\mathcal{T}|$. Moreover, even if the architecture can be synthesized without any cost restriction, deriving an optimal solution for Equation (4) is still $\mathcal{NP}$-hard in a strong sense, and there does not exist any polynomial-time approximation algorithm to have a constant *approximation factor* unless $\mathcal{P} = \mathcal{NP}$. An algorithm is said to be with an approximation factor β if the objective function of its derived solution is at most β times the optimal objective solution for any input instance.

Theorem 1: Even when there is only one scenario, there does not exist any polynomial-time approximation algorithm with a constant approximation factor for the power-aware scenario-mapping problem, unless $\mathcal{P} = \mathcal{NP}$.

Proof: This theorem is proved by an L-reduction [19] from the set cover problem, which does not admit any polynomial-time approximation algorithm with a constant approximation factor unless $\mathcal{NP} = \mathcal{P}$. Given a universe $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ of m elements, a collection $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_n\}$ of sub-collections of $\mathcal{E}$, and the cost $C_i > 0$ for each sub-collection $\mathcal{S}_i$, the set cover problem is to pickup a minimum-cost sub-collection of $\mathcal{S}$ that covers all elements in $\mathcal{E}$.

The L-reduction is done as follows: For each sub-collection $\mathcal{W}_j$, we create a processing unit type p_j with static power consumption σ_j equal to C_j. The dynamic power consumption δ_j on each processing unit p_j is a constant L. For each element e_i in $\mathcal{E}$, we create a task t_i. If e_i is in $\mathcal{W}_j$, let $u_{i,j}$ be $1/|\mathcal{E}|$. By restricting to the special case with one scenario, all the constructed tasks are present in one scenario with 100% probability.

For an optimal solution of the set cover problem with cost C^*, there is a feasible solution of the reduced input instance of the power-aware scenario-mapping problem with $C^* + L$ expected average power consumption. For a feasible solution with C expected average power consumption for the reduced input instance of the power-aware scenario-mapping problem, there exists a solution for the set cover problem with cost no more than $C - L$. As a result, when $L \ll C^*$, if there is a polynomial-time β-approximation algorithm for power-aware scenario-mapping problem, the set cover problem will admit a polynomial-time β-approximation algorithm. The contradiction is reached. □

Corollary 1: There does not exist any polynomial-time approximation algorithm with a constant approximation factor for the power-aware scenario-mapping problem, unless $\mathcal{P} = \mathcal{NP}$.

According to the $\mathcal{NP}$-completeness of the derivation of feasible solutions for the power-aware scenario-mapping problem when F_j is less than $|\mathcal{T}|$, for the rest of this paper, we focus our study on the case that F_j is not specified, i.e., $F_j = |\mathcal{T}|$. If F_j is specified, our remapping algorithm in Section 3 can be revised to try to fit the required demand, but there is no feasibility guarantee.

3. Multi-Step Algorithm

As it is difficult to derive solutions with worst-case guarantees in polynomial time, this section presents an efficient multi-step heuristics to derive approximative solutions for the power-aware scenario-mapping problem. We first derive initial solutions based on linear programming relaxation, and then perform task remapping to improve the solutions.

3.1. Initial Solutions

To derive a feasible initial solution, we can first relax the integral constraints in Equation (4d) and Equation (4e) so that $M_{i,j,k}$ and $Z_{s,j,k}$ can be any fractional variable. That

is,

$$\min \sum_{S_s \in \mathcal{S}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} \chi_s \sigma_j Z_{s,j,k} + \sum_{t_i \in \mathcal{T}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} u_{i,j} \delta_j \psi_i M_{i,j,k} \tag{5a}$$

s.t.

$$\sum_{t_i \in \mathcal{T}} \theta_{i,s} M_{i,j,k} u_{i,j} \leq Z_{s,j,k}, \quad \begin{matrix} \forall p_j \in \mathcal{P}, S_s \in \mathcal{S}, \\ \forall k = 1, 2, \ldots, F_j \end{matrix} \tag{5b}$$

$$\sum_{p_j \in P} \sum_{k=1}^{F_j} M_{i,j,k} = 1, \; \forall t_i \in \mathcal{T}, \tag{5c}$$

$$M_{i,j,k} \geq 0, \; \forall t_i \in \mathcal{T}, p_j \in \mathcal{P}, k = 1, 2, \ldots, F_j, \tag{5d}$$

$$Z_{s,j,k} \geq 0, \; \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, 2, \ldots, F_j. \tag{5e}$$

In contrast to Equation (4b), Equation (5b) is not upper-bounded anymore, as $Z_{s,j,k}$ can take any positive value. Moreover, in Equation (5), we can use $\hat{M}_{i,j} = \sum_{k=1}^{F_j} M_{i,j,k}$. As a consequence of the unboundedness of $Z_{s,j,k}$ tasks are assigned to PU types, rather than to instances thereof.

The optimal solution for the relaxed problem is equivalent to the following linear program:

$$\min \sum_{p_j \in \mathcal{P}} \sum_{t_i \in \mathcal{T}} \psi_i u_{i,j} (\delta_j + \sigma_j) \hat{M}_{i,j} \tag{6a}$$

s.t.

$$\sum_{p_j \in \mathcal{P}} \hat{M}_{i,j} = 1, \; \forall t_i \in \mathcal{T}, \tag{6b}$$

$$\hat{M}_{i,j} \geq 0, \; \forall t_i \in \mathcal{T}, p_j \in \mathcal{P}, \tag{6c}$$

where the variable $\hat{M}_{i,j}$ indicates the portion of task t_i that is assigned to execute on PU type p_j. By applying the extreme point theory [20], it is clear that there exists an optimal solution for Equation (6) which maps every task $t_i \in \mathcal{T}$ to the PU type $p_j \in \mathcal{P}$ that has the minimum static and dynamic power consumption ($u_{i,j}(\delta_j + \sigma_j)$).

Algorithm 1 presents the pseudo-code of the procedures to derive an initial solution for the power-aware scenario-mapping problem. Step 1 and Step 2 in Algorithm 1 derive an assignment of tasks to PU types, where Steps 3 to 17 allocate processing unit instances for tasks as follows:

After assigning tasks to PU types, the actual number of instances k per PU type p_j has to be computed. Suppose that the set of tasks, that are mapped onto a specific PU type p_j are denoted as $\mathcal{T}_j \subseteq \mathcal{T}$. For each PU type, we order the tasks in $\mathcal{T}_j$ from high utilization to low utilization. Starting with task $t_i \in \mathcal{T}_j$ with the highest utilization $u_{i,j}$, tasks are assigned to an instance k of processing unit type p_j, denoted $\hat{p}_{j,k}$. A task assignment is feasible, if the additional utilization $u_{i,j}$ on target PU $\hat{p}_{j,k}$ does not violate the utilization constraint for any Scenario S_s that task t_i is assigned to (compare to constraint 4b). If no feasible assignment exists, an additional instance of the corresponding PU type p_j is created, and the task is mapped onto this new instance. This process is repeated for each task and each PU type p_j until all tasks in $\mathcal{T}$ are assigned to an instance of a processing unit. In case there are already multiple instances of a PU type available, tasks can be assigned to any concrete instance, as long as the utilization constraint is not violated.

Algorithm 1 Initial Solution

Input: $\mathcal{T}, \mathcal{P}, \mathcal{S}$

1: $\mathcal{T}_j \leftarrow \emptyset, \forall p_j \in \mathcal{P}$;
2: for each $t_i \in \mathcal{T}$, find $p_{j^*} \leftarrow \underset{p_j \in \mathcal{P}}{\operatorname{argmin}} u_{i,j}(\delta_j + \sigma_j)$ and $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{t_i\}$;
3: **for** each $\mathcal{T}_j \neq \emptyset$ **do**
4: order tasks in $\mathcal{T}_j$, e.g., decreasing on the utilization;
5: $U_{s,j,k} \leftarrow 0, \forall S_s \in \mathcal{S}$ and $k = 1, 2, \ldots, |\mathcal{T}|$;
6: **for** each task t_i in $\mathcal{T}_j$ **do**
7: let $\mathcal{S}_{t_i}$ be the set of scenarios that t_i is associated with;
8: **if** there exists an index k with $U_{s,j,k} > 0$ for some $S_s \in \mathcal{S}_{t_i}$ and $U_{s,j,k} + u_{i,j} \leq 1$ for all $S_s \in \mathcal{S}_{t_i}$ **then**
9: assign task t_i to the k-th allocated PU of p_j;
10: $U_{s,j,k} \leftarrow U_{s,j,k} + u_{i,j}$ for all $S_s \in \mathcal{S}_i$;
11: **else**
12: let k^* be the smallest k with $U_{s,j,k} = 0, \forall S_s \in \mathcal{S}$;
13: allocate the k^*-th unit of p_j and assign task t_i to it;
14: $U_{s,j,k^*} \leftarrow U_{s,j,k^*} + u_{i,j}$ for all $S_s \in \mathcal{S}_i$;
15: **end if**
16: **end for**
17: **end for**

Clearly, the derived solution is feasible for the power-aware scenario-mapping problem, since each task is assigned onto one processing unit and the total utilization of tasks in a scenario mapped onto a processing unit is guaranteed to be no more than 100%. The time complexity is $O(|\mathcal{P}||\mathcal{T}|^2|\mathcal{S}|)$.

3.2. Task Remapping

In Equation (6) tasks are mapped to their most power efficient PU type. This may distribute the tasks over a large amount of PU types and instances thereof, which results in those PUs to be low utilized. According to our power model, a PU consumes static power, once it is switched on. Distributing tasks over a large amount of low utilized PUs leads to a high contribution of static power consumption to the expected average power consumption. Equation (6) disregards that fact, and thus might underestimate the objective.

We propose an approach to improve the solution iteratively by applying a multiple-step procedure. Given an initial solution, derived from Algorithm 1, we iteratively improve the solution by considering the remapping of tasks. Task remapping is done by considering sets of tasks that are assigned to a Scenario $\mathcal{S}_s$ on a specific PU $\hat{p}_{j,k}$. Let $\mathcal{T}_{s,j,k}$ be the set of tasks assigned on PU $\hat{p}_{j,k}$ in Scenario $\mathcal{S}_s$. The remapping procedure attempts to remap all the tasks in $\mathcal{T}_{s,j,k}$ to other PUs, in order to reduce the expected average power consumption. To reduce the time complexity for remapping,

Algorithm 2 Remapping

Input: $\mathcal{T}_{s,j,k}, \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \ldots, f_j$;
1: **while** true **do**
2: **for each** non-empty set $\mathcal{T}_{s^*,j^*,k^*}$ **do**
3: apply Algorithm 3 to remap all the tasks in $\mathcal{T}_{s^*,j^*,k^*}$, and let $\mathcal{T}_{s,j,k}^{s^*,j^*,k^*}, \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \ldots, f_j$ be the solution;
4: let Δ_{s^*,j^*,k^*} be its reduced expected average power consumption if the remapping (in Step 3) is successful;
5: **end for**
6: **if** there is no successful remapping (in Step 3) **then**
7: break;
8: **else**
9: let $s^\dagger, j^\dagger, k^\dagger$ be the indexes s^*, j^*, k^* with the minimum Δ_{s^*,j^*,k^*};
10: $\mathcal{T}_{s,j,k} \leftarrow \mathcal{T}_{s,j,k}^{s^\dagger,j^\dagger,k^\dagger}, \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \ldots, f_j$;
11: **end if**
12: **end while**
13: **return** the task sets $\mathcal{T}_{s,j,k}, \forall S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \ldots, f_j$;

only PUs are considered as valid target units that already host all Scenarios $\mathcal{S}'_s$ to which the tasks in $\mathcal{T}_{s,j,k}$ are assigned to. Tasks in $\mathcal{T}_{s,j,k}$ can only be mapped to PU $\hat{p}_{j',k'}$ if the following condition applies: The sets of tasks $\mathcal{T}_{s',j',k'}$ for Scenarios $\mathcal{S}_{s'}$, where $\theta_{i,s'} = 1\ \forall t_i \in \mathcal{T}_{s,j,k}$, are non-empty sets. Among all task sets $\mathcal{T}_{s,j,k} \in \mathcal{T}$ we choose to remap the set of tasks $\mathcal{T}_{s^*,j^*,k^*}$ first, that yields the highest reduction of expected average power consumption. This remapping step iterates until no further performance gain can be achieved.

The pseudo-code for remapping is presented in Algorithm 2, where *the detail of Step 3 will be presented in Algorithm 3* later. To reduce the time complexity for remapping, we only consider non-empty sets $\mathcal{T}_{s,j,k}$. In Algorithm 2, we use f_j to denote the number of allocated units of PU type p_j in the initial solution.

We now present how to determine the highest expected average power reduction of task set $\mathcal{T}_{s,j,k}$, i.e., the implementation of Step 3 in Algorithm 2. Suppose that $U_{s,j,k}$ is the utilization of tasks in $\mathcal{T}_{s,j,k}$, i.e., $U_{s,j,k} = \sum_{t_i \in \mathcal{T}_{s,j,k}} u_{i,j}$. Furthermore, once tasks assigned to Scenario $\mathcal{S}_s$ are mapped onto PU $\hat{p}_{j,k}$, we set $Z^\dagger_{s,j,k}$ as 1, while $Z^\dagger_{s,j,k}$ is 0 otherwise.

A set of tasks $\mathcal{T}_{s^*,j^*,k^*}$ can only be remapped to a processing unit $\hat{p}_{j',k'}$ if $Z^\dagger_{s^*,j',k'}$ is 1. This allows to satisfy the previously defined condition, that tasks can only be remapped if all the scenarios they are assigned to are hosted on the target processing unit.

The best remapping of task set $\mathcal{T}_{s^*,j^*,k^*}$ is to find an optimal solution for the following integer linear programming:

$$\min \sum_{S_s \in \mathcal{S}} \sum_{p_j \in \mathcal{P}} \sum_{k=1}^{F_j} \lambda_s \delta_j \left(U_{s,j,k} + \sum_{t_i \in \mathcal{T}_{s^*,j^*,k^*}} M_{i,j,k} u_{i,j} \right) \tag{7a}$$

s.t.

$$U_{s,j,k} + \sum_{t_i \in \mathcal{T}_{s^*,j^*,k^*}} M_{i,j,k} u_{i,j} \leq Z^\dagger_{s,j,k}, \quad \begin{array}{l} \forall p_j \in \mathcal{P}, S_s \in \mathcal{S}, \\ \forall k = 1, 2, \ldots, F_j \end{array} \tag{7b}$$

$$\sum_{p_j \in P} \sum_{k=1, k \neq k^*}^{F_j} M_{i,j,k} = 1,\ \forall t_i \in \mathcal{T}_{s^*,j^*,k^*}, \tag{7c}$$

$$M_{i,j,k} \in \{0, 1\},\ \forall t_i \in \mathcal{T}_{s^*,j^*,k^*}, p_j \in \mathcal{P}, k = 1, 2, \ldots, F_j. \tag{7d}$$

Note that $Z^\dagger_{i,j,k}$ is not a variable in the programming in Equation (7), but is derived from the initial mapping.

As the number of tasks in task set $\mathcal{T}_{s^*,j^*,k^*}$ is significantly reduced compared to $\mathcal{T}$, it is possible to derive optimal solutions of Equation (7). However, the ILP in Equation (7) has to be executed many times (in $O(|\mathcal{P}||\mathcal{S}|)$) to determine which remapping is the best so far. Once the resulting task set is remapped, the ILP has to execute again, to find the next set of tasks for remapping. As a consequence, applying an ILP solver to exactly solve Equation (7) with high complexity is impractical. This paper presents how to perform task remapping by applying a heuristic approach as shown in Algorithm 3.

Algorithm 3 has as an input the set of tasks $\mathcal{T}_{s^*,j^*,k^*}$ that shall be remapped. In Step 2, task t_{i^*} with the lowest utilization $u_{i^*,j}$ is retrieved. Furthermore $\mathcal{S}_{i^*}$ is defined to be the scenarios task t_{i^*} is present in.

The next steps perform the search for a processing unit $\hat{p}_{j',k'}$ where the utilization $u_{i^*,j'}$ of task t_{i^*} is minimized and there exists an instance k' such that the utilization constraints for all Scenarios $\mathcal{S}_{i^*}$ are satisfied. Furthermore the target and the origin PU cannot be the same unit ($\hat{p}_{j',k'} \neq \hat{p}_{j,k}$). If such a PU cannot be found (Step 5), remapping fails and the algorithm continues with the next task in the input task set. Otherwise (Step 7) the remapping is performed and task t_{i^*} is removed from the set of tasks to be remapped. This process repeats, as long as there are task in the input task set.

The time complexity of Algorithm 3 is in $O(|\mathcal{S}||\mathcal{P}||\mathcal{T}||\mathcal{T}_{s^*,j^*,k^*}|)$ and the overall time complexity for task remapping in Algorithm 2 is $O(|\mathcal{T}|^4|\mathcal{S}|^3|\mathcal{P}|^3)$.

4. Performance Evaluation

This section provides the simulation results by means of realistic SDR (software defined radio) applications. Specifically, Wireless Local Area Network (WLAN), as described in [14], Digital Video Broadcast - Handhelds (DVB-H), as

Application groups	$\hat{\mathcal{A}}_1$ (DVB-H)		$\hat{\mathcal{A}}_2$ (WLAN)		$\hat{\mathcal{A}}_3$ (UWB)	
Applications	A_1 **(sync)**	A_2 **(receive)**	A_3 **(send)**	A_4 **(receive)**	A_5 **(send)**	A_6 **(receive)**
Number of tasks	4	7	10	23	3	5
Number of shared tasks	0	0	10	10	1	1
$\gamma_{i,j}$	random variables with $0 \leq \gamma_{i,j} \leq 0.5$					
χ	determined by profiling [21]					

Table 3. Simulation setup for WLAN, DVB-H, and UWB applications.

Algorithm 3 Remapping of One Task Set

Input: $\mathcal{T}_{s,j,k}, S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \cdots, f_j$; $\mathcal{T}_{s^*,j^*,k^*}$ for remapping;

1: **while** $\mathcal{T}_{s^*,j^*,k^*} \neq \emptyset$ **do**
2: $t_{i^*} \leftarrow \underset{t_i \in \mathcal{T}_{s^*,j^*,k^*}}{\operatorname{argmin}} u_{i,j^*}$;
3: let $\mathcal{S}_{i^*}$ be the set of the scenarios that task t_{i^*} is associated with;
4: among the non-empty task sets $\mathcal{T}_{s,j,k}$ $((j,k) \neq (j^*,k^*))$ with $S_s \in \mathcal{S}_{i^*}$ and $u_{i^*,j} + \sum_{t_i \in \mathcal{T}_{s,j,k}} u_{i,j} \leq 1$ for all S_s in $\mathcal{S}_{i^*}$, let (j',k') be the indexes with the minimum $u_{i^*,j'}$ and satisfied constraint $U_{s,j',k'}$;
5: **if** indexes (j',k') do not exist **then**
6: **return** remapping fails;
7: **else**
8: $\mathcal{T}_{s',j',k'} \leftarrow \mathcal{T}_{s',j',k'} \cup \{t_{i^*}\}, \forall S_{s'} \in \mathcal{S}_{i^*}$;
9: $\mathcal{T}_{s',j^*,k^*} \leftarrow \mathcal{T}_{s',j^*,k^*} \setminus \{t_{i^*}\}, \forall S_{s'} \in \mathcal{S}_{i^*}$;
10: **end if**
11: **end while**
12: **return** task sets $\mathcal{T}_{s,j,k}, S_s \in \mathcal{S}, p_j \in \mathcal{P}, k = 1, \cdots, f_j$;

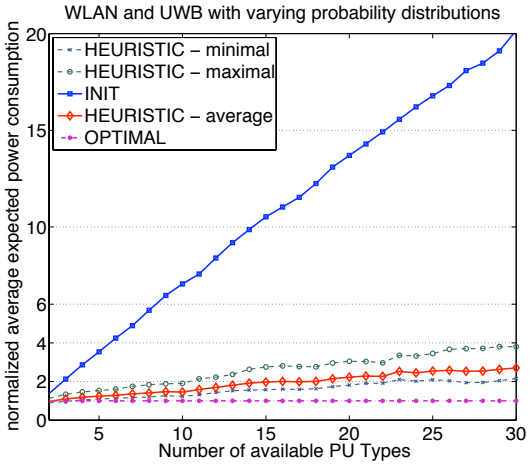

Figure 3. Simulation results for WLAN/UWB.

variables	values
σ_j	random $0 \leq \sigma_j \leq 1.0$
δ_j	random $0 \leq \delta_j \leq \sigma_j$
λ_j	random $0 \leq \lambda_j \leq 3.0$

Table 4. Parameters for generating the processing unit types.

described in [18], and Ultra Wideband (UWB), as described in [15], are adopted in the simulations.

The simulation setup is characterized in Table 3. For each application, a set of tasks is extracted and the required computational resource demand $\gamma_{i,j}$ of task t_i on processing unit type p_j is generated.

We simulate systems with two application groups (either WLAN and DVB-H or WLAN and UWB), and two applications in each application group (send/sync and receive). The PU types library $\mathcal{P}$ which is used in the initial mapping process to construct a feasible hardware platform is generated using random variables (see Table 4).

Experiments are executed for different PU type library sizes, reaching from 2 available types to 30. The generated libraries and application representations for the simulations can be downloaded from our web page [21]. The applications and application groups probabilities' have been varied, and experiments where executed with 6 different distributions. For each distinct library size we created 500 system instances. These 500 instances where simulated with different probability distributions, resulting in 3000 realizations per PU types library size. Each instance uses a different set of application parameters and a different library of PU types $\mathcal{P}$ to construct the hardware platform.

The initial mapping, see Algorithm 1, denoted as 'INIT', the heuristic mapping process, see Algorithm 2, and the optimal solution, see Equation 4, denoted as 'OPTIMAL' are evaluated. The heuristic mapping performance is evaluated as an average, maximum and minimum case. The average case (denoted 'HEURISTIC - average') is the average performance computed from all realizations simulated for a specific PU types library size. The maximum (denoted 'HEURISTIC - maximal') characterizes the worst performance for a specific PU types library size and the minimum (denoted 'HEURISTIC - minimum') characterizes the best performance from all realizations for a given library size.

For each simulation, the time consumed to compute the results is evaluated. For the optimal case, we terminate the ILP after a reasonable amount of time and do not consider those experiments for our results.

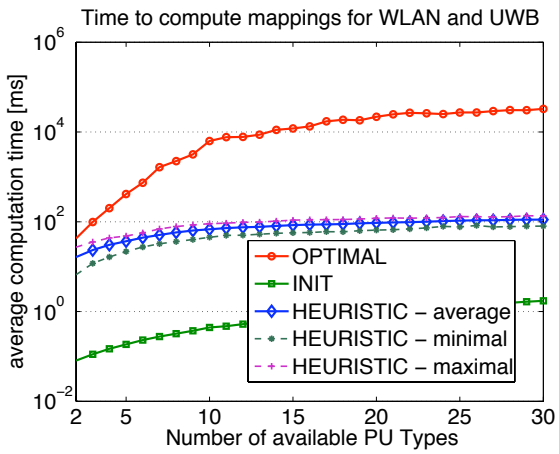

Figure 4. Time complexity for WLAN/UWB.

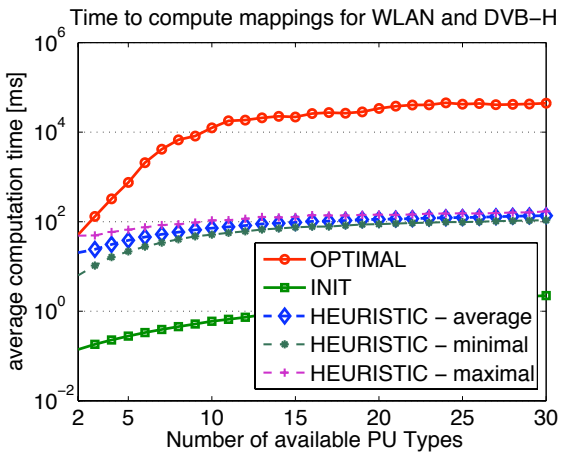

Figure 6. Time complexity for WLAN/DVB-H.

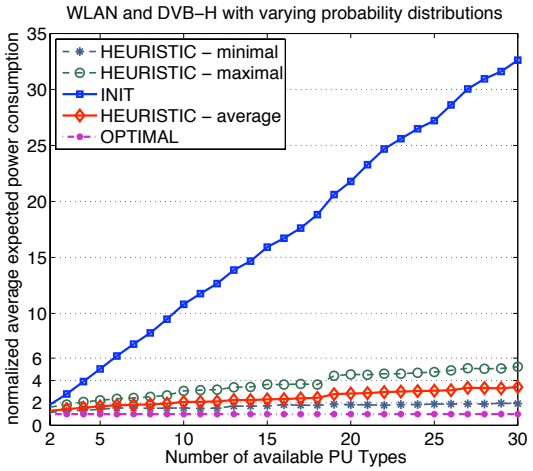

Figure 5. Simulation results for WLAN/DVB-H.

4.1. Simulation results

Figure 3 represents the simulation results for systems with WLAN and UWB applications. With increasing PU types library size the performance of the initial mapping is deteriorating. The heuristic performance stays below a factor of 4 even for the maximal case and a large number of PU types. The average heuristics' performance is below 2 for PU type library size smaller than 15 PU types and below 3 for libraries of less than 30 PU types. The performance degradation is very slow, compared to the initial mapping. This shows the effectiveness of the proposed multi-step heuristic.

Figure 4 represents the timing evaluation of the remapping process for systems with WLAN and UWB applications. Already at 2 available PU types the heuristic mapping process is faster than the optimal. At a library size of 5 PU types, the heuristic process is already one magnitude and at a library size of 10 PU types 2 magnitudes faster than the optimal algorithm. It can be noted that at a library size of about 25, the time to compute the heuristic remapping process stays constant and the gap between minimal and maximal time elapsed to compute the results for the heuristic approach diminishes.

Increasing the absolute number of tasks makes the problem harder, and thus the performance for systems with WLAN and DVB-H is slightly worse. Figure 5 shows the simulation results for systems with WLAN and DVB applications. The performance of the initial mapping deteriorates quickly. The heuristic approach still achieves a power performance which is less than 4 times worse than the optimal solution. In the minimal case, the heuristic approach achieves a power performance that is at most 2 times worse than the optimal. Again, the effectiveness of the multi-step heuristics is apparent, despite the fact that there does not exist any polynomial-time approximation algorithm with a constant *approximation factor*.

Figure 6 represents the timing evaluation for systems with WLAN and DVB-H applications. The increasing number of tasks, compared to the system with WLAN and UWB presented in Fig. 4, results in a larger gap between the minimum and the maximum case. As in the previous example, the heuristic approach outperforms the initial mapping by 2 magnitudes in terms of time consumption. With increasing number of available PU types, the gap between the minimal and maximal time elapsed to compute the results for the heuristic approach diminishes.

The optimal algorithm is terminated after 20 minutes. If the computation of the optimal result was terminated prematurely, the result is not included in our experiments. Prematurely terminated computations account for up to 6% for systems with WLAN and DVB-H applications. For PU library sizes below 10 this share is reduced to $\leq$ 1% and for libraries with less than 5 available PU types no

computation had to be terminated. For systems with WLAN and UWB this performance is enhanced, as the number of tasks is reduced. Prematurely terminated computations for those simulations account for up to 3% and fall below 1% for libraries with less than 10 available PU types.

As shown in the simulation results, the proposed algorithm can derive feasible solutions, and the resulting expected average power consumption of a solution is between 1.1 and 4 times of the optimal solution in average cases. The solution stays below twice of the optimal solution for libraries with less than 10 PU types. In terms of computation time a speed up of 2 magnitudes is achieved in comparison to the optimal solution for large PU type libraries. Even though we have proved the non-approximability in polynomial-time, the derived solutions are quite promising.

5. Conclusion

This paper explores the power-aware scenario-mapping problem to design embedded systems with applications/scenarios specified by their execution probabilities. We show that there is no polynomial-time approximation algorithm with a constant approximation factor unless $\mathcal{P} = \mathcal{NP}$ and provide a polynomial-time heuristic algorithm. The initial solution is based on the relaxation of the integer linear programming of the studied problem, while the multiple-step remapping procedure is developed to improve the quality of the derived solution. Taking SDR (software defined radio) applications (WLAN, DVB-H and UWB) as examples in our simulations, the results reveal the effectiveness of our multi-step heuristic. For future research, we intend to explore tasks with precedence constraints and other classes of resource competition.

References

[1] M. Kim, S. Banerjee, N. Dutt, and N. Venkatasubramanian, "Energy-aware cosynthesis of real-time multimedia applications on mpsocs using heterogeneous scheduling policies," *Trans. on Embedded Computing Sys.*, vol. 7, no. 2, pp. 1–19, 2008.

[2] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proceedings of the Design Automation Conference*, 2004, pp. 275–280.

[3] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *IEEE Real-time and Embedded Technology and Applications Symposium*, 2006, pp. 408–417.

[4] T. A. Alenawy and H. Aydin, "Energy-aware task allocation for rate monotonic scheduling," in *Proceedings of the 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'05)*, 2005, pp. 213–223.

[5] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *Proceedings of 17th International Parallel and Distributed Processing Symposium (IPDPS)*, 2003, pp. 113 – 121.

[6] J.-J. Chen and C.-F. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms," in *RTCSA*, 2007.

[7] P.-C. Chang, I.-W. Wu, J.-J. Shann, and C.-P. Chung, "ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor," in *Proc. Design Automation Conference (DAC 2008)*, 2008, pp. 776–779.

[8] J.-J. Chen and T.-W. Kuo, "Allocation cost minimization for periodic hard real-time tasks in energy-constrained DVS systems," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2006, pp. 255–260.

[9] Y. Chen, Z. Shao, Q. Zhuge, C. Xue, B. Xiao, and E. H.-M. Sha, "Minimizing energy via loop scheduling and dvs for multi-core embedded systems," in *ICPADS (2)*, 2005, pp. 2–6.

[10] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *IEEE Real Time Technology and Applications Symposium*, 2006, pp. 418–428.

[11] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé, "Energy-efficient policies for embedded clusters," in *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems(LCTES)*, 2005, pp. 1–10.

[12] Y. Lin, H. Lee, M. Woh, Y. Harel, S. A. Mahlke, T. N. Mudge, C. Chakrabarti, and K. Flautner, "Soda: A low-power architecture for software radio," in *ISCA*, 2006, pp. 89–101.

[13] Y. Lin, M. Kudlur, S. A. Mahlke, and T. N. Mudge, "Hierarchical coarse-grained stream compilation for software defined radio," in *CASES*, 2007, pp. 115–124.

[14] O. Moreira, F. Valente, and M. Bekooij, "Scheduling multiple independent hard-real-time jobs on a heterogeneous multiprocessor," in *EMSOFT '07: Proceedings of the 7th ACM & IEEE international conference on Embedded software*. New York, NY, USA: ACM, 2007, pp. 57–66.

[15] Y. Wang, A. Schranzhofer, R. van Leuken, and A. van der Veen, "A hardware platform for delay hopped transmitted reference UWB communication system prototype development," in *IEEE/ProRISC workshop on Circuits, Systems and Signal Processing*. IEEE, 2007. [Online]. Available: http://ens.ewi.tudelft.nl/pubs/yiyin07prorisc.pdf

[16] M. Schmitz, B. Al-Hashimi, and P. Eles, "Cosynthesis of energy-efficient multimode embedded systems with consideration of mode-execution probabilities," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 2, pp. 153–169, Feb. 2005.

[17] J.-J. Chen, A. Schranzhofer, and L. Thiele, "Energy minimization for periodic real-time tasks on heterogeneous processing units," in *International Parallel and Distributed Processing Symposium (IPDPS)*, 2009.

[18] L. Schwoerer, A. Kaufmann, D. Guevorkian, J. Westmeijer, and Y. Xu, "DVB (synchronized and receiving mode) summary for SDR scheduler modeling workshop: timing corresponds to 8k case," Radio Communications CTC, NOKIA Research Center, Helsinki, Finland, Tech. Rep., June 2007.

[19] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[20] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.

[21] A. Schranzhofer, J.-J. Chen, and L. Thiele, "*Experimental Data - File Archive*," 2008. [Online]. Available: http://www.tik.ee.ethz.ch/∼andrschr/RTASexperiments.tar.gz