Energy-Efficient Task Allocation and Scheduling for Multi-Mode MPSoCs under Lifetime Reliability Constraint

Lin Huang[†] and Qiang Xu[†‡]
[†]CUhk REliable computing laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[‡]CAS-CUHK Shenzhen Institute of Advanced Integration Technology
Email: {lhuang,qxu}@cse.cuhk.edu.hk

ABSTRACT

In this paper, we consider energy minimization for multiprocessor system-on-a-chip (MPSoC) under lifetime reliability constraint of the system, which has become a serious concern for the industry with technology scaling. As today's complex embedded systems typically have multiple execution modes, we first identify a set of "good" task allocation and schedules for each execution mode in terms of lifetime reliability and/or energy consumption, and then we introduce novel techniques to obtain an optimal combination of these single-mode solutions, which is able to minimize the energy consumption of the entire multi-mode system while satisfying given lifetime reliability constraint. Experimental results on several hypothetical MPSoC platforms with various task graphs demonstrate the effectiveness of the proposed approach.

1. INTRODUCTION

In response to today's competitive electronics market, when designing complex embedded systems, it is increasingly popular to employ pre-designed multiprocessor system-on-a-chip (MPSoC) platforms and map applications onto them to reduce design risk and achieve short time-to-market [19]. Various platforms with specific functionalities reflecting the need of the expected application domain have been developed in the industry recently, e.g., ARM PrimeXsys platform [1].

When building platform-based embedded systems, a basic issues is to conduct task allocation and scheduling for applications, in which the allocation of tasks is to effectively utilize the available processors while scheduling is to meet various requirements (e.g., timing constraints). Recently, minimizing energy consumption has become a critical task for embedded system designs, and a widely-used technique is dynamic voltage scaling (DVS) [9], by which we scale down the voltage/frequency of individual processors according to temporal performance requirements of applications. Various energy-efficient task allocation and scheduling techniques for DVS-enabled embedded systems have been presented in the literature (e.g., [11]).

Despite the significant advancement of platform-based embedded system design methodologies in recent years, only limited works have considered the lifetime reliability of the system [7, 23]. With the ever-increasing on-chip power and temperature densities, however, the wearout failures (e.g., negative bias temperature instability on PMOS transistors) have become serious concerns for the industry [3, 12, 21]. As shown in [17], the failure rates for electronic products can be quite high within its warrantee period and the main reason was traced to excessive stress on the embedded processors.

Existing energy-efficient task allocation and scheduling techniques target at reducing the overall energy consumption of the system. The lifetime of the system, however, is determined by the component with the shortest service life. Consequently, it is likely that the solution with the minimum energy consumption results in excessive stress on certain components, leading to unexpected low lifetime reliability of the system. In addition, today's complex embedded systems usually do not stick to a single execution mode throughout their entire service lives. Instead, they work across a set of different interacting applications and operational modes. For instance, modern smart phones not only provide communication service, but also work as MP3 player, game console, and digital camera. For such *multi-mode embedded systems* [18], the above problem can be further exacerbated due to inter-mode resource sharing and the associated possible imbalanced usage of processor cores. For example, an energy-efficient processor in a heterogeneous MPSoC platform might have higher utilization rate in most operational modes, thus aging much faster than other embedded processor due to the excessive stress on it.

From the above, *it is essential to explicitly consider lifetime reliability issue in energy-efficient embedded system designs*. To tackle this problem, in this paper, we first show how to conduct energy-efficient task allocation and scheduling on MPSoC platforms for a single execution mode, taking the lifetime reliability as a constraint. For multi-mode embedded systems, since the overall system's lifetime reliability is also related to the mode execution probabilities, it is not necessary to apply the same constraint to every mode. That is, we can afford to have task allocation and schedules for certain modes with lower reliability if such solutions reduce energy consumption dramatically, and compensate the reliability loss from other execution modes. Based on this observation, we propose to identify a set of "good" task allocation and schedules in terms of lifetime reliability and/or energy consumption for each execution mode. Then, we introduce novel methodologies to obtain an optimal combination of task schedules that minimizes the energy consumption of the entire multi-mode system, while satisfying given systemwide lifetime reliability constraint. Experimental results on various hypothetic multi-mode MPSoC platforms show that the proposed solution can significantly reduce the system energy consumption under reliability constraint.

The reminder of this paper is organized as follows. Section 2 presents preliminaries of this work and formulates the problem studied in this paper. In Section 3, we present our task allocation and scheduling algorithm for multi-mode embedded systems. Section 4 presents our experimental results for hypothetical MPSoC platforms. Finally, Section 5 concludes this work.

2. PRELIMINARIES

2.1 Related Work

Recently, a major trend in embedded system design is towards energy-efficient computing based on the concept of performance on demand, by dynamically adjusting the operational voltage and frequency of processors based on instantaneous processing requirement. There is a rich literature on energy-efficient design methodologies (e.g., [11]), which mainly resort to DVS and slack reclaiming to cut down the energy consumption of the embedded processors. In particular, Schmitz *et al.* [20] proposed an energy-efficient co-synthesis framework for multi-mode embedded systems under the consideration of mode execution probabilities, in which a single execution mode occupies the entire MPSoC at a time.

At the same time, with aggressive technology scaling, the lifetime reliability of today's high-performance integrated circuits has also become a major concern for the industry [3]. The *wearout failure mechanisms* that lead to permanent errors of IC products include electromigration on the interconnects, TDDB in the gate oxides, negative bias temperature instability (NBTI) on PMOS transistors, and thermal cycling (TC), and they were shown to be highly related to the temperature and voltage applied to the circuit. Thus, existing thermal-aware task scheduling techniques may improve the MPSoC's lifetime reliability implicitly, by balancing different processors' temperatures or keeping them under a safe threshold. However, as pointed out in [7], since circuit wearout failures are also dependent on many other factors (e.g., internal structure, operational frequency and voltage), without *explicitly* taking the lifetime reliability into account during task allocation and scheduling, various processor cores may still age differently and thus result in shorter lifetime for the MPSoC-based embedded system.

Recently, the authors of [7] proposed a novel analytical model for the lifetime reliability of multiprocessor platforms, which, unlike prior work (e.g., [4, 21]), is able to capture the processors' accumulated aging effects. They have also introduced a simulated annealing (SA) technique to maximize the service life of MPSoC-based embedded systems under performance constraint. However, energy consumption issues are not considered in [7] and it focused on single execution mode only. In practice, it is not necessary to prolong the service life of embedded systems as much as possible. Rather, energy minimization should be the primary optimization objective with given lifetime reliability being used as a constraint.

2.2 Problem Formulation

Based on the above, the problem investigated in this paper is formulated as follows:

Problem: Given

- the floorplan of the platform-based MPSoC embedded system that consists of ℓ processor cores;
- n execution modes. Each mode i is represented by a directed acyclic task graph $\mathbf{G}_i = (\mathbf{V}_i, \mathbf{E}_i)$, wherein each node in $\mathbf{V}_i$ indicates a task in $\mathbf{G}_i$, and $\mathbf{E}_i$ is the set of directed arcs that represent precedence constraints;
- the joint probability density function[1] that the system is in various modes $f_{Y_1,Y_2,\cdots,Y_n}(y_1,y_2,\cdots,y_n)$, where y_i represents the probability that the system is in execution mode i;
- the execution time $w_{i,j,k}$ of task j of mode i on processor k under maximum supply voltage V_{dd};
- the power consumption $P_{i,j,k}$ of task j of mode i on processor k under maximum supply voltage V_{dd};
- deadline $d_{i,j}$ of task j of mode i, meaning that task j in $\mathbf{G}_i$ should be finished before $d_{i,j}$;
- target service life L and the corresponding reliability requirement $\eta\%$;
- failure mechanism parameters (e.g., activation energy E_a of electromigration) and the corresponding failure distributions;

Determine a periodical task allocation and schedule on the given MPSoC platform for each execution mode such that the expected energy consumption is minimized, under the performance constraints that real-time tasks are finished before deadlines and the lifetime reliability constraint that the system reliability at the target usage life L is no less than $\eta\%$.

Note that, for the ease of discussion, in this work, we assume each execution mode in a multi-mode system corresponds to one directed acyclic task graph only. Our proposed approach, however, could be easily extended to handle multiple task graphs for a single mode by constructing a hyper task graph and performing task scheduling on a hyper-period [13], if necessary. In addition, while there are other hardware resources in the MPSoC platform consuming energy and suffering from wearout failures, we mainly consider processor cores in this work due to their heavy stress and varying operational behaviors. Our work can be easily extended to take other hardware components (with simpler activities) into account, if needed.

2.3 Analytical Models

The energy consumption model used in this work is mainly based on [10, 16], by which we are able to compute the power consumption and execution time of task j (in the task graph for execution mode i) running on processor k given its voltage scaling parameter ρ, denoted by $P_{i,j,k}(\rho)$ and $w_{i,j,k}(\rho)$ respectively. Both are normalized to given values $P_{i,j,k}$ and $w_{i,j,k}$ when $\rho = 1$ (i.e., no voltage scaling). The energy consumption of task schedule for execution mode i depends on the task assignments and the deadline of entire task graph $\max_j\{d_{i,j}\}$. Denoting by $1_{\{i,j\to k\}}$ an indicator function that equals 1 if task j is assigned to processor k while 0 otherwise, the energy consumption of a certain task schedule in unit time is expressed as

$$E_i = \frac{1}{\max_j\{d_{i,j}\}} \sum_j \sum_k P_{i,j,k}(\rho_{i,j,k}) \cdot w_{i,j,k}(\rho_{i,j,k}) \cdot 1_{\{i,j\to k\}} \qquad (1)$$

A lifetime reliability model that captures the accumulated aging effects of IC hard errors has been proposed in [7]. We resort to that model to estimate the reliability stress of task schedules in this work. Let $\alpha_k(T)$ be the aging effect of processor k in unit time provided operational temperature T. The aging effect of processor k in a certain task schedule is therefore

$$A_{i,k} = \sum_j \left[\frac{w_{i,j,k}(\rho)\cdot 1_{\{i,j\to k\}}}{\alpha_k(T_{i,j,k})} - \frac{w_{i,j,k}(\rho)\cdot 1_{\{i,j\to k\}}}{\alpha_k(T_{amb})}\right] + \frac{d_i}{\alpha_k(T_{amb})}$$

where, T_{amb} is the ambient temperature. Then, suppose the system remains in mode i through its service life, we can express system reliability at the target service life L with the aging effect additivity proved in [7] as

$$R_i = \exp\left[-\sum_k \left(\frac{A_{i,k}}{\max_j\{d_{i,j}\}}\cdot L\right)^{\beta_k}\right] \qquad (2)$$

3. PROPOSED ALGORITHM FOR MULTI-MODE EMBEDDED SYSTEMS

Since the lifetime reliability constraint is a systemwide constraint (unlike the performance constraint for real-time tasks), it is not necessary to apply the same reliability constraint to every execution mode for multi-mode MPSoC embedded systems. In this section, we show how to take advantage of this flexibility to minimize energy consumption for multi-mode embedded systems. To be specific, we first generate "good" solutions in terms of reliability and/or energy for each execution mode (Section 3.1-3.3) and then we search for an optimal combination of them to obtain minimized energy while satisfying the systemwide lifetime reliability constraint (Section 3.4).

3.1 Feasible Solution Set

When we loosen the lifetime reliability constraint for a single execution mode, there are many possible task allocation and schedule solutions. However, if one solution is associated with higher energy consumption and at the same time has lower lifetime reliability when compared to another solution, it definitely should not be considered for the combination of solutions of all execution modes.

Before introducing the searching procedure, we first introduce a key concept in our proposed methodology. Among the task allocation and schedule solutions for a certain task graph (denoted as set X), there exists a subset Y that satisfies the following two conditions.

Internal stability *Given two solutions $u, v \in Y$, if u consumes more energy than v, it must have higher lifetime reliability at the target service life, and vice versa.*

External stability *For any solution $w \in X \setminus Y$, there exists at least one solution $u \in Y$ such that u consumes less energy and have higher lifetime reliability than w.*

[1]The mode execution probabilities can be estimated as in [20].

We refer to Y as the *feasible solution set* (i.e., pareto optimal solution set), denoted as $\mathcal{F}$ in the rest of this paper. *Now, our problem of task allocation and scheduling for a single execution mode comes down to identifying $\mathcal{F}$.* Fig. 1 shows an example, wherein several solutions are plotted on a two-dimensional plane as points according to their lifetime reliability and energy consumption. The feasible solution set in this case is $\mathcal{F} = \{\mathbf{O}, \mathbf{D}, \mathbf{E}\}$. Consider a solution outside this set, say **G**. It implies higher energy consumption and lower lifetime reliability than solution **D**. Note that solution **O** should be kept although it violates the global reliability constraint, because it is a possible candidate in the final combined solutions in our multi-mode systems.

3.2 Searching Procedure for a Single Mode

The simulated annealing technique could be used to obtain a single solution with minimum energy consumption under lifetime reliability and performance constraint for the task scheduling problem (e.g., [7]). To find a set of feasible solutions as candidates for later multi-mode combination, we modify the classic procedure and highlight the difference with Fig. 2.

In a typical SA-based algorithm, it is only necessary to keep the current solution during the searching procedure and the best solution explored so far, but in our method, every newfound solution is first checked with its cost, which will be introduced in Section 3.2.2, to determine whether it should be accepted (Line 7). If accepted and meeting performance constraint, depending on which feasible solution set identification strategy is chosen (detailed in Section 3.3), it is either added into the possible solution set $\mathcal{P}$ (Line 11) or identified together with original feasible solution set (Line 13). Note that, if the static strategy is chosen, after this searching procedure all found solutions are kept in the set $\mathcal{P}$. The static identification is conducted and yields the feasible solution set $\mathcal{F}$ (Line 16). In contrast, if we perform the dynamic identification during the searching procedure, the resulting set $\mathcal{P}$ is essentially the feasible solution set $\mathcal{F}$ (Line 18).

3.2.1 Solution Representation

The most straightforward representation to specify schedule **X** is (*schedule order sequence*; *resource binding sequence*), which can be used to construct task schedules directly [7] and corresponds to unique reliability, energy consumption and schedule length (also known as *makespan*). This simple representation, however, leads to huge design exploration space. Let v_i be the number of nodes in the task graph for execution mode i. With ℓ processors in the system, the solution exploration space is as high as $(v_i! \cdot \ell^{v_i})$.

Fortunately, we notice that, without considering DVS, both energy consumption and lifetime reliability depend mainly on task allocation and are almost independent of their schedule orders (see Eq. (1) and Eq. (2)). In addition, with known task allocation, there is a rich literature (e.g., [14, 15]) on how to conduct task scheduling to meet deadlines and reduce the total schedule length. Based on the above, we propose to represent the solutions with task allocation only, shrinking the solution space to ℓ^{v_i}. The random move strategy is also quite simple with this representation, i.e., we randomly pick up a task and assign it to another processor core, different from the original one.

3.2.2 Cost Function

As the simulated annealing searching procedure is guided to the solution with minimum cost, we should define a cost function such that

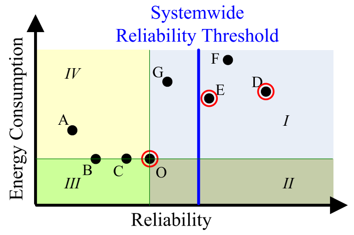

Figure 1: Feasible Solution Set Identification.

```
1   Build an initial solution $\mathbf{X}$ and initialize $\mathcal{P}$ as $\{\mathbf{X}\}$
2   While $T > T_{end}$
3       For the iteration $i$ from 1 to $N_{iter}$
4           $\mathbf{X}' \leftarrow$ Random_Move($\mathbf{X}$)
5           $C_{old} \leftarrow$ Cal_Cost($\mathbf{X}$)
6           $C_{new} \leftarrow$ Cal_Cost($\mathbf{X}'$)
7           If $C_{new} < C_{old}$ or $\exp\left(\frac{C_{old}-C_{new}}{T}\right) > rand()$
8               $\mathbf{X} \leftarrow \mathbf{X}'$ // accept $\mathbf{X}'$
9               If $\mathbf{X}' \notin \mathcal{P}$ and $\mathbf{X}'$ meet performance constraint
10                  If static identification
11                      $\mathcal{P} \leftarrow$ Include($\mathcal{P}$, $\mathbf{X}'$)
12                  Else If dynamic identification
13                      $\mathcal{P} \leftarrow$ Identify($\mathcal{P}$, $\mathbf{X}'$) // see Sec. 3.3.2
14      $T \leftarrow T \times R_{cooling}$

15  If static identification
16      $\mathcal{F} \leftarrow$ Identify($\mathcal{P}$) // see Sec. 3.3.1
17  Else If dynamic identification
18      $\mathcal{F} \leftarrow \mathcal{P}$
```

Figure 2: Main Flow of Searching for Feasible Solution Set.

the task schedule with higher reliability and lower energy consumption has less cost. Also, to meet the performance constraint, a heavy penalty should be given to the task assignment that cannot meet this constraint. Denoting by $m_{i,j}$ the earliest finish time of task j given resource binding of mode i, we therefore define the cost function as

$$C_i = \gamma \cdot 1_{\{\exists j:\, m_{i,j} > d_{i,j}\}} - R_i \times E_i$$

Here, γ represents a significant large number. The first term, which equals γ if there exists at least one task (say, task j) exceeds its deadline (i.e., $m_{i,j} > d_{i,j}$) while 0 otherwise, is the penalty of violating performance constraint. To determine this value, we need to estimate the schedule length, which depends on not only assignment but also the schedule. While we can use exhaustive search to obtain the minimum value and the corresponding task schedule, given the task allocation solution, we can resort to well-studied heuristics (e.g., [15]) to acquire an optimized task schedule.

We use *minus* for the second term because solutions with lower R_i are with less lifetime reliability and hence should have higher cost during the simulated annealing process. Without considering DVS, E_i and R_i can be computed by Eq. (1) and Eq. (2). Yet it is very important to note that DVS is helpful to both energy savings and reliability improvement in most cases, as shown in Fig. 3. We therefore make use of the timing slacks to determine appropriate voltage scales for DVS-enabled processors to minimize energy under performance and lifetime reliability constraint.

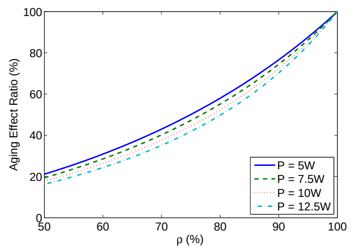

Figure 3: Influence of Voltage Scaling on Aging Effect.

To be specific, we first compute the range of possible task execution time and the associated voltage scaling parameter with the following procedure. Given the task schedule reconstructed with solution representation, suppose the entire task schedule is fulfilled ahead of deadline $\max_j\{d_{i,j}\}$ without voltage scaling, all tasks in the schedule are uniformly "stretched" and the ratio is denoted by global scale parameter θ_i^g. In addition, we notice that some tasks can be further elongated without affecting the scheduling of any other tasks. That is to say, a task can be further extended if it finishes before the starting time of all of its successors on task graph and the starting time of the next task on the same processor core. We define the local scale parameter for such a task j as $\theta_{i,j,k}^l$, given it is assigned to processor k. In particular, these values are set to 1 if no extension is appropriate. Thus, the execution time of task j is bounded by $w_{i,j,k} \cdot \theta_i^g \cdot \theta_{i,j,k}^l$ and thus the upper bound of voltage scaling parameter could be computed accordingly.

Then, as in most cases a processor core can only work under a few voltage levels, we determine voltage assignment for each task by choosing the voltage state that results in the maximum cost reduction. And finally we reevaluate the energy consumption of the resource binding sequence by using Eq. (1).

3.3 Feasible Solution Set Identification

With a number of recorded solutions obtained using the above procedure, this section is concerned with identifying the feasible solution set out of them.

3.3.1 Static Strategy

Let us start from a simple case in which all solutions have been found out and stored before our feasible solution set identification process (this assumption will be lifted later).

Before presenting the proposed approach, we first introduce some definitions. All the found task allocation and schedule solutions are examined according to the lifetime reliability model and energy model in Section 2.3, and marked on a two-dimensional plane whose x-axis and y-axis represent the lifetime reliability at the end of target service life (provided the system remains in this mode) and the expected energy consumption, respectively. With respect to a point, we divide the plane into four domains, referred as domain *I*, *II*, *III*, and *IV*. Fig. 1 illustrates a domain division respect to point **O**. The positive x-axis and negative y-axis belong to domain *II*; while the negative x-axis and positive y-axis belong to domain *IV*. We define *original point* as the point with the lowest energy consumption and, in case of ties, with the highest lifetime reliability on this plane, denoted as point **O**.

Theorem 1 *Point* **O** *corresponds to a feasible solution, and all other feasible solutions only exist in its domain I.*

Proof Since solution **O** consumes the least energy, there is no points falling in its domain *II* or *III*. For the solutions in its domain *IV*, since they are associated with higher energy consumption and lower reliability than **O**, they cannot be feasible. Also, solutions with the same energy consumption as that of **O** but lower lifetime reliability (e.g., point **B** and **C** in Fig. 1) are apparently not feasible. Therefore, all feasible solutions are in domain *I* of point **O**. □

It is also worth noting that the converse of this theorem is not true, that is, not all points in domain *I* of point **O** are feasible solutions.

We propose to sweep domain *I* with respect to point **O** in a counterclockwise manner, and check the reached points (i.e., solutions) according to the following theorem. A feasible solution set $\mathcal{F}$ is initialized as $\{\mathbf{O}\}$ before sweeping. If a reached solution is a feasible one, it is included into $\mathcal{F}$ before examining the next one.

Theorem 2 *A new solution* **N** *is a feasible one if and only if it is in domain I or III of all elements in set* $\mathcal{F}$.

Proof Suppose point **N** is feasible and it is in the domain *II* of another feasible solution $\mathbf{X}\in\mathcal{F}$, by counterclockwise sweeping we must have come across point **N** before **X** and put it into $\mathcal{F}$ already. Therefore, the supposition does not hold. Next, we assume point **N** is in the domain *IV* of any solution **X** in set $\mathcal{F}$. In this case, solution **N** costs more energy and results in lower reliability when compared to solution **X**, and hence is not a feasible solution. □

Consider the example shown in Fig. 1. By sweeping counterclockwisely, we first come across point **D**. It is in the domain *I* of point **O**, which is the only element in $\mathcal{F}$ at this moment, and therefore included into the feasible solution set. Next one is **E**. It is in domain *I* of point **O** and domain *III* of point **D**, and hence also included. Then, we come across point **F**, which is not a feasible solution as it falls into domain *IV* of point **D**. The final one is **G**. As it locates in domain *IV* of **D** and **E**, it is also not feasible. Therefore, the end result is $\mathcal{F}$={**O, D, E**}.

3.3.2 Dynamic Strategy

It is important to highlight that the above static identification strategy requires to find all solutions and store them before the identification process, thus involving heavy memory overhead, especially when there are a large number of possible solutions. This problem can be avoided if we can identify feasible solutions in a random order. By doing so, feasible solution set $\mathcal{F}$ is updated whenever a newfound solution is generated, and hence we only need to maintain the current feasible solution set. In the following, a dynamic identification strategy to achieve the above objective is presented.

The feasible solution set $\mathcal{F}$ is initialized as empty. Every newfound solution is processed according to the following three rules.

Rule 1 *If the new solution is in domain I or III of ALL elements in set* $\mathcal{F}$*, it should be included into the feasible set* $\mathcal{F}$.

Rule 2 *If the new solution is in domain II of ANY solution* **X** *(where* $\mathbf{X}\in\mathcal{F}$*), we include the new solution into* $\mathcal{F}$ *and at the same time eliminate* **X** *from* $\mathcal{F}$.

Rule 3 *If the new solution is in domain IV of ANY solution* **X** *(where* $\mathbf{X}\in\mathcal{F}$*), we ignore the new solution.*

Rule 1 is consistent with Theorem 2. Rule 2 holds because the new solution must consume less energy and have higher reliability than solution **X**, if it is in domain *II* of **X**. Hence, **X** should be replaced by the new solution. As for Rule 3, given the new solution is in domain *IV* of solution **X**, it is worse than **X** in terms of both reliability and energy saving, and hence is not feasible. Note that, it is impossible that a newfound solution is marked in domain *II* of some feasible set elements and domain *IV* of some other elements simultaneously.

Consider the example shown in Fig. 1 again. Let us examine the dynamic identification with a random order: **C, O, E, D, F, B, A, G**. The procedure is shown in Fig. 4, leading to the same feasible solution set as that obtained using static strategy (see Section 3.3.1).

original $\mathcal{F}$	new solution	updated $\mathcal{F}$
∅	**C**	{**C**}
{**C**}	**O**	{**O**}
{**O**}	**E**	{**O, E**}
{**O, E**}	**D**	{**O, E, D**}
{**O, E, D**}	**F**	{**O, E, D**}
{**O, E, D**}	**B**	{**O, E, D**}
{**O, E, D**}	**A**	{**O, E, D**}
{**O, E, D**}	**G**	{**O, E, D**}

Figure 4: An Example Identification Procedure.

3.4 Multi-Mode Combination

Given the feasible solution set for every execution mode, we show how to combine them to achieve minimum energy consumption under systemwide lifetime reliability constraint in this subsection.

Suppose the feasible solution set $\mathcal{F}_i$ of mode i is composed of q_i elements. We denote by E_i^ℓ the energy consumption given the ℓ^{th} feasible solution ($\ell = 1,\cdots,q_i$) for execution mode i. In addition, we introduce a function $1_{\{\ell\Rightarrow i\}}$, which equals to 1 if we choose the ℓ^{th} solution from the feasible solution set $\mathcal{F}_i$ for mode i; otherwise 0. With these notations, the energy consumption for a fixed execution probability combination can be expressed as

$$E(y_1,y_2,\cdots,y_n) = \sum_i y_i \cdot \sum_\ell E_i^\ell \cdot 1_{\{\ell\Rightarrow i\}}$$

Subject to the constraint that

$$\forall i: \quad \sum_\ell 1_{\{\ell\Rightarrow i\}} = 1 \tag{3}$$

More generally, when we consider the usage information obtained from a large user group, we can use a joint probability density function $f_{Y_1,Y_2,\cdots,Y_n}(y_1,y_2,\cdots,y_n)$ to represent the probabilities that the system is in various execution modes. Thus, the optimization objective becomes to minimize the expected energy consumption over service life, that is, to minimize $\mathbf{E}[E(Y_1,Y_2,\cdots,Y_n)]$, where $\mathbf{E}[X]$ indicates the expectation of random variable X.

As the performance constraints of all execution modes have been guaranteed in the proposed searching procedure, besides Eq. (3) we need to consider the lifetime reliability constraint only. Similar to the energy consumption issue, we compute the expected value given the fixed probabilities y_i for execution mode i as

$$R(y_1, y_2, \cdots, y_n) = \exp\big[-\sum_k \big(\sum_i \frac{y_i \cdot \sum_\ell A^\ell_{i,k} \cdot 1_{\{\ell \Rightarrow i\}}}{\max_j\{d_{i,j}\}} \cdot L\big)^{\beta_k}\big]$$

where, $A^\ell_{i,k}$ represents the aging effect of task schedule ℓ on processor k in execution mode i. The reliability constraint comes down to keep the expected lifetime reliability over the execution probability distribution exceeding a threshold, i.e.,

$$\mathbf{E}[R(Y_1, Y_2, \cdots, Y_n)] \geq \eta\%$$

We therefore formulate the multi-mode solution combination problem as follows:

$$\boxed{\begin{array}{ll} \text{min} & \mathbf{E}[E(Y_1, Y_2, \cdots, Y_n)] \\ \text{st.} & \mathbf{E}[R(Y_1, Y_2, \cdots, Y_n)] \geq \eta\% \\ & \forall i : \sum_\ell 1_{\{\ell \Rightarrow i\}} = 1 \\ & \forall i, \ell : 1_{\{\ell \Rightarrow i\}}\text{=0 or 1} \end{array}}$$

This optimization problem can be solved quite efficiently since the number of execution modes for an embedded system is typically small (tens of modes for very complex systems).

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

We conduct two sets of experiments with different task graphs and hypothetical MPSoC platforms to evaluate the proposed approach. All task graphs are generated by TGFF [5]. The power consumption of tasks on processor cores are randomly generated, while the range is set according to state-of-the-art technology [22]. Although the proposed approach is applicable for the combination of multiple failure mechanisms, since there is no public data on the influence weight of various hard errors, we use the well-studied electromigration failure model presented in [6] for our experiments. The parameters are set to cross-sectional area of conductor $A_c = 6.4 \times 10^{-8} cm^2$, the current density $J = 1.5 \times 10^6 A/cm^2$ and the activation energy $E_a = 0.48eV$. The simulated annealing parameters are set to initial temperature 100, terminal temperature $T_{end} = 10^{-5}$, cooling rate $R_{cooling} = 0.99$, and iteration count $I = 20$.

To demonstrate the effectiveness of the proposed algorithms, we compare the proposed multi-mode approach, the proposed single-mode approach wherein all execution mode needs to satisfy the given reliability constraint, and a greedy heuristic constructed by ourselves due to the lack of prior work on the same topic. In this heuristic, we first build a task schedule to shorten the schedule length and reduce energy consumption with a list scheduling heuristic [2]. Note that, to take both energy consumption and performance into account, the *critical path length* of task j on processor k is redefined as $\text{cpl}'(\tau_j, p_k) = \text{cpl}(\tau_j, p_k) - P_{j,k}$ according to the heuristics in [8], where $\text{cpl}(\tau_j, p_k)$ is the critical path length of a task τ_j scheduled on processor p_k [2]. We then attempt to meet the reliability constraint in a greedy manner, that is, if the system reliability constraint is violated, the processor core with the minimum lifetime reliability is selected in each iteration and the task causing the highest stress is assigned onto another core without violating performance constraints. The moved task is then locked. The procedure terminates if no more moves are available or reliability constraint can be met.

4.2 Case Study

We consider three task graphs with task quantities 6, 7, and 3 (denoted as task graph (a), (b), and (c) respectively hereafter) in the first experiment, each corresponding to a particular execution mode, and schedule them on two processor cores with failure distribution slope parameter β = 1.5, 2, respectively. The probabilities that the system is in execution modes (a)-(c) are set to be 0.3, 0.3, and 0.4, respectively. By conducting the proposed searching procedure we obtain a set of feasible solutions for each mode, sorted in the order of R_i and listed in Table 1. Some schedules (e.g., solution 2-7 for task graph (a)) cannot meet the lifetime reliability constraint, but they are kept because some other task graphs (e.g., task graph (b)) may provide reliability margin.

Task Graph Label	Solution No.	R_i (%)	E_i	$\max_j\{m_{i,j}\}$	Resource Binding Sequence
(a)	1	39.16	23.179	52.835	(0, 0, 1, 0, 1, 1)
	2	**36.70**	22.992	46.742	(1, 0, 1, 0, 1, 0)
	3	**34.91**	22.370	50.028	(0, 0, 1, 1, 1, 1)
	4	**34.18**	21.312	37.021	(1, 0, 1, 0, 1, 1)
	5	**27.92**	20.503	34.214	(1, 0, 1, 1, 1, 1)
	6	**17.05**	20.061	33.807	(1, 1, 1, 0, 1, 1)
	7	**11.80**	19.253	31.001	(1, 1, 1, 1, 1, 1)
(b)	1	65.09	15.437	71.702	(1, 0, 1, 1, 1, 0, 0)
	2	59.19	15.358	78.068	(1, 1, 1, 1, 0, 0, 0)
	3	56.94	14.921	62.918	(1, 0, 1, 1, 0, 0, 0)
	4	47.54	14.910	57.684	(1, 0, 0, 1, 0, 0, 0)
	5	45.49	14.488	52.101	(1, 0, 1, 0, 0, 0, 0)
	6	**36.51**	14.477	46.867	(1, 0, 0, 0, 0, 0, 0)
(c)	1	44.05	23.036	33.275	(1, 1, 0)
	2	41.98	22.559	25.813	(0, 1, 0)
	3	39.77	19.889	27.235	(1, 0, 1)
	4	**35.33**	17.034	23.355	(1, 0, 0)
	5	**27.10**	16.556	15.892	(0, 0, 0)

R_i: lifetime reliability at the target service life given mode i;
E_i: energy consumption for execution mode i;

Table 1: Feasible Solution Set (End Result).

Then, by optimization, we obtain the combination with the minimum expected energy consumption 16.875 meeting the lifetime reliability requirement $R_i \geq 36.8\%$ (or e^{-1}) at the target service life $L = 10$ years [12]. We choose solution 7, 4, 5 for task graph (a), (b), and (c), respectively.

We compare this result with that obtained by the baseline greedy heuristic and the proposed single-mode algorithms. The selected solutions are illustrated in Table 2. The baseline approach cannot provide a solution meeting this requirement because of task graph (a). On the other hand, because the single-mode method does not allow the reliability constraint violation of any modes, it results in significant reliability margin. The multi-mode method, by contrast, makes full use of such margin, and therefore provides 14.1% energy saving when compared with the single-mode approach.

4.3 Sensitivity Analysis

It is interesting to analyze the impact of lifetime reliability threshold on the effectiveness of the proposed algorithm. As shown in Fig. 5, no solutions can be found using the baseline method with a tight reliability threshold (i.e., higher than 32%), while the single-mode approach results in good solutions until the reliability threshold increases to 39%. The proposed multi-mode approach is able to give solutions when the reliability threshold is as high as 49%.

If the reliability threshold can be relaxed, both the energy consumptions obtained by the single-mode method and multi-mode one decrease, and finally they converge when the threshold decreases to 11.8%. In the range 11.8–39%, the proposed approach always leads to better result than the single-mode one. The energy consumption of the baseline approach also decreases with the relaxation of reliability requirement, but it always results in the highest energy consumption among these methods. In particular, when the reliability threshold is 11.8%, the energy consumption obtained with the baseline method is 17.27. With the same energy consumption, the proposed single-mode and multi-mode approach are able to achieve much higher lifetime reliability 27% and 42% (see the black stars) after 10-year service. Generally speaking, the warrantee period for electronic products are shorter than their designed service life, and we are interested in the failure rates at this time point. Suppose our system's warrantee period is 3-year, the failure rates for these three methods are 17.9%, 15.2% and 11.7%, respectively. In other words, there are roughly 6.2% less failures with our proposed method when compared to the greedy heuristic at the end of the warrantee period.

Approach	Task Graph Label	$R_i(\%)$	E_i	$\max_j\{m_{i,j}\}$	Resource Binding Sequence	$\mathbf{E}[E]$
Baseline	(a)	–	–	–	–	–
	(b)	42.71	15.008	43.283	(0, 0, 0, 1, 0, 0, 0)	
	(c)	41.98	22.559	25.813	(0, 1, 0)	
Single Mode	(a)	39.16	23.179	52.835	(0, 0, 1, 0, 1, 1)	19.255
	(b)	45.49	14.488	52.101	(1, 0, 1, 0, 0, 0, 0)	
	(c)	39.77	19.889	27.235	(1, 0, 1)	
Multi Mode	(a)	11.80	19.253	31.001	(1, 1, 1, 1, 1, 1)	16.875
	(b)	47.54	14.910	57.684	(1, 0, 0, 1, 0, 0, 0)	
	(c)	27.10	16.556	15.892	(0, 0, 0)	

Table 2: Energy Consumption Comparison between the Single-Mode Method and the Multi-Mode Combination Approach.

4.4 Extensive Results

In this experiment, we consider relatively large MPSoC platform and the associated task graphs, wherein we have 5 execution modes and their task quantities are 69, 7, 14, 3, and 29 (denoted as task graph (d)-(h) respectively), mapping onto a 8-core heterogeneous MPSoC. The maximum in- and out-degree of these task graphs are 3 and 2 respectively. The task graphs are also generated by TGFF [5]. The failure distribution slope parameters β for processor cores are 2, 4, and six 1.5, respectively.

As shown in Fig. 6, only when the reliability threshold is lower than 29%, we can obtain solutions for all execution modes using the baseline greedy heuristic. In this range, both single-mode and multi-mode approaches provide up to 26.3% and 27.8% energy reduction when compared with the greedy solutions, respectively.

The single-mode approach is able to meet tighter reliability constraints and save more energy when compared to the greedy heuristic. But still, when the reliability threshold is higher than 39.7%, there is no solution for some execution modes and hence the entire system. For instance, when threshold is 45%, no task schedule meeting both constraints can be found for modes (d) and (e). With the proposed multi-mode combination approach, however, we can obtain a solution for the entire multi-mode system. This is because some other modes, such as (g), provides sufficient reliability margin. Besides, the multi-mode approach always achieves lower energy consumption when compared with the single-mode approach, because it explores a larger solution space that includes the solution obtained by the single-mode method.

Finally, we allocate these task graphs onto a 8-core homogeneous MPSoC platform, where all processor cores have the same architecture and hence the same failure distribution ($\beta = 1.5$), and consume the same execution time and power consumption on each task. In this case, the three methods obtain solutions with the same energy consumption when the reliability constraint is set as $\eta\% = 27.7\%$, but the proposed multi-mode approach is able to achieve higher lifetime reliability, $\eta\% = 37.3\%$. This is because, for each single execution mode, all task schedules in the feasible solution set obtained with the proposed algorithm have the same lifetime reliability and energy consumption, but different resource allocation. The proposed multi-mode combination step tends to choose the task schedules for multiple execution modes such that all processor cores have similar wearout stress. This feature, however, cannot be exploited by the other two methods.

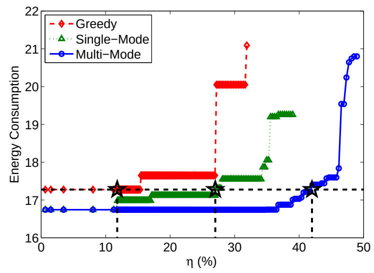

Figure 5: Variation in Energy Consumption with Reliability Threshold.

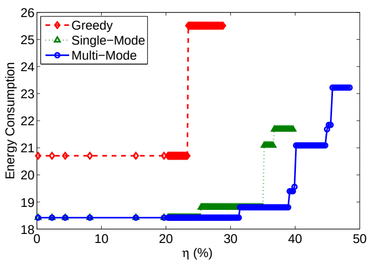

Figure 6: Comparison between Energy Consumption of Multi-Mode System under Constraints.

5. CONCLUSION

In this work, we propose novel task allocation and scheduling algorithms to minimize the expected energy consumption of multi-mode embedded systems under performance and lifetime reliability constraints. As shown in our experimental results, the proposed methodology is able to meet tight reliability constraints and results in significant energy savings, especially for heterogeneous MPSoCs.

6. ACKNOWLEDGEMENTS

This work was supported in part by the General Research Fund CUHK417406, CUHK417807, and CUHK418708 from Hong Kong SAR Research Grants Council (RGC), in part by National Science Foundation of China (NSFC) under grant No. 60876029, and in part by a grant N_CUHK417/08 from the NSFC/RGC Joint Research Scheme.

7. REFERENCES

[1] ARM. ARM11 PrimeXsys Platform. http://www.jp.arm.com/event/images/forum2002/02-print_arm11_primexsys_platform_ian.pdf.
[2] P. Bjorn-Jorgensen and J. Madsen. Critical Path Driven Cosynthesis for Heterogeneous Target Architectures. In *Proc. CODES*, pp. 15–19, 1997.
[3] S. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. In *IEEE D&T*, 25(6):10–16, Nov.-Dec. 2005.
[4] A. Coskun, et al. Analysis and optimization of mpsoc reliability. *Journal of Low Power Electronics*, 15(2):159–172, February 2006.
[5] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: Task Graphs for Free. In *Proc. CODES*, pp. 97–101, 1998.
[6] A. K. Goel. *High-Speed VLSI Interconnections*. IEEE Press, 2nd edition, 2007.
[7] L. Huang, F. Yuan, and Q. Xu. Lifetime Reliability-Aware Task Allocation and Scheduling for MPSoC Platforms. In *Proc. DATE*, pp. 51–56, 2009.
[8] W.-L. Hung, et al. Thermal-Aware Task Allocation and Scheduling for Embedded Systems. In *Proc. DATE*, pp. 898–899, 2005.
[9] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proc. ISLPED*, pp. 197–202, 1998.
[10] R. Jejurikar, C. Pereira, and R. Gupta. Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems. In *Proc. DAC*, pp. 275–280, 2004.
[11] N. K. Jha. Low Power System Scheduling and Synthesis. In *Proc. ICCAD*, pages 259–263, 2001.
[12] E. Karl, et al. Multi-Mechanism Reliability Modeling and Management in Dynamic Systems. *IEEE Trans. on VLSI Systems*, 16(4):476–487, April 2008.
[13] V. Kianzad and S. S. Bhattacharyya. CHARMED: A Multi-objective Co-synthesis Framework for Multi-mode Embedded Systems. In *Proc. ASAP*, pp. 28–40, 2004.
[14] Y.-K. Kwok and I. Ahmad. Static task scheduling and allocation algorithms for scalable parallel and distributed systems: Classification and performance comparison. In Y. C. Kwong, editor, *Annual Review of Scalable Computing*, pp. 107–227, 2000.
[15] G. Liao, et al. A Comparative Study of Multiprocessor List Scheduling Heuristics. In *Proc. Hawaii International Conference on System Sciences*, pp. 68–77, 1994.
[16] S. Martin, et al. Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Lower Power Microprocessors under Dynamic Workloads. In *Proc. ICCAD*, pp. 721–725, 2002.
[17] NVIDIA Provides Second Quarter Fiscal 2009 Business Update. http://www.nvidia.com/object/io_1215037160521.html.
[18] H. Oh and S. Ha. Hardware-Software Cosynthesis of Multi-Mode Multi-Task Embedded Systems with Real-Time Constraints. In *Proc. CODES*, pp. 133–138, 2002.
[19] A. Sangiovanni-Vincentelli, et al. Benefits and Challenges for Platform-based Design. In *Proc. DAC*, pp. 409–414, 2004.
[20] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Cosynthesis of Energy-Efficient Multimode Embedded Systems With Consideration of Mode-Execution Probabilities. *IEEE Trans. on CAD*, 24(2):153–169, February 2005.
[21] J. Srinivasan, et al. The Case for Lifetime Reliability-Aware Microprocessors. In *Proc. ISCA*, pp. 276–287, 2004.
[22] S. Zhang and K. S. Chatha. Approximation Algorithm for the Temperature-Aware Scheduling Problem. In *Proc. ICCAD*, pp. 281–288, 2007.
[23] C. Zhu, et al. Reliable Multiprocessor System-on-Chip Synthesis. In *Proc. CODES*, pp. 239–244, 2007.