Sustainability-Oriented Evaluation and Optimization for MPSoC Task Allocation and Scheduling Under Thermal and Energy Variations

Mingsong Chen, *Member, IEEE*, Xinqian Zhang, Haifeng Gu, Tongquan Wei, *Member, IEEE*, and Qi Zhu, *Member, IEEE*

Abstract—Aiming at high performance, more and more Cyber-Physical Systems (CPSs) adopt Multiprocessor System-on-Chips (MPSoCs) as computation units. However, due to increasing integration of transistors on a die, the power densities together with performance variations of MPSoC chips have been increasing dramatically. Consequently, the MPSoC-based CPSs might become unsustainable and unreliable. Although various Task Allocation and Scheduling (TAS) heuristics have been proposed to minimize the hotspot time (i.e., duration of thermal emergency) and energy consumption of MPSoC designs, few of them can guarantee the highest performance yield under process variations without violating energy, thermal and timing constraints. To address these challenges, this paper proposes a novel energy- and thermal-aware TAS evaluation and optimization framework. Based on statistical model checking techniques, our approach enables accurate modeling and reasoning of the performance yield of real-time MPSoC designs under joint energy and thermal constraints. To enable system-level design space exploration, we propose a regression analysis-based method that can drastically reduce the overall exploration efforts. Experimental results show that our fully-automated approach can not only allow accurate sustainability-oriented reasoning of TAS solutions under specified thermal and energy constraints, but also enable the quick search of optimal TAS solutions on different MPSoC architectures with the highest performance yield.

Index Terms—Cyber-Physical Systems, Sustainability, Task Allocation and Scheduling, Statistical Model Checking, Optimization.

1 Introduction

Due to the increasing demand of interactions between cyber world and physical environment, there is a trend towards the development of high-performance *Cyber-Physical Systems* (CPSs), e.g., autonomous automobile systems, unmanned aerial vehicles [1]. To facilitate the simultaneous processing of heterogeneous tasks (e.g., monitoring, control and communication), more and more CPS designers adopt *Multiprocessor System-on-Chip* (MPSoC) which integrates a collection of heterogeneous *Processing Elements* (PEs) such as application-specific instruction-set processors and hardware accelerators on a single die [2]. Since MPSoC allows to fully exploit the capabilities of both hardware and software resources, the stringent CPS design requirements such as real-time response and energy efficiency can be accomplished systematically.

As technology scales, to achieve better overall performance, the number of PEs integrated on an MPSoC chip grows quickly. Due to increasing integration of transistors on a die, the power densities together with process variations have been also increasing dramatically [3], [4]. This brings two big challenges in MPSoC design. The first one is the high energy consumption which affects the sustainability of MPSoC-based CPS designs, especially for mobile autonomous CPSs which are driven by batteries. Typically, higher power densities may lead to higher chip temperature, which not only degrades the performance and depletes system energy storage, but also accelerates both the aging process of MPSoC devices and failure mechanisms (e.g., electro-migration, dielectric breakdown). Consequently, the sustainability of MPSoC-based CPS designs cannot be guaranteed. Therefore, it is very important to restrict the overheating time to reduce the energy consumption and balance the thermal profiles for MPSoC designs. The second challenge is the uncertain PE performance caused by process variations, which has a substantial influence on reliable CPS execution [5]. Due to the difficulty of fabricating small structures consistently across a die, even for the PEs of the same type on the same chip, we cannot assume that they have same performance. In addition to the intrinsic physical variations (e.g., channel length, gate-oxide thickness and threshold voltage) [6], [7], the environmental variations (e.g., temperature, power supply) that depend primarily on architectural designs and PE operations have substantial impacts on the MPSoC performance. If such variations are not considered in MPSoC design, we cannot assure the reliability of the host CPS products [8].

Task Allocation and Scheduling (TAS) plays an important role in the design of energy and thermal efficient MPSoC designs. This is because intelligent TAS strategies can improve the utilization of PEs while satisfying various design constraints (e.g., response time, peak temperature, energy consumption) [9], [10]. However, due to variations across identically designed PEs and chips, the required constraints of MPSoC designs cannot be easily guaranteed [4]. Although traditional MPSoC TAS approaches adopt worst-case timing analysis to obtain feasible TAS solutions, due to significant

- *Mingsong Chen, Xinqian Zhang, Haifeng Gu and Tongquan Wei are with the Shanghai Key Laboratory of Trustworthy of Computing at East China Normal University, Shanghai, 200062, China (email: {mschen, xqzhang, hfgu}@sei.ecnu.edu.cn, tqwei@cs.ecnu.edu.cn). Qi Zhu is with Department of Electrical and Computer Engineering at University of California, Riverside, CA 92521, USA (email: qzhu@ece.ucr.edu). Tongquan Wei is the corresponding author.*
- *This work was supported by the grants from Natural Science Foundation of China (Nos. 61672230, 61672240), Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), and United States National Science Foundation grants CCF-1553757, CCF-1646641, and CCF-1646381.*

performance and power deviations from nominal values in the design and overly pessimistic performance estimation, such approaches may no longer provide viable solutions. To measure TAS strategies under variations, *performance yield* was proposed to define the probability of an assigned TAS instance meeting required MPSoC constraints [11]. Therefore, it can be used as an effective metric to evaluate the sustainability and reliability of MPSoC designs.

Since MPSoC TAS is an NP-Complete problem, various heuristics have been proposed to find a sub-optimal solution to maximize the performance yield [11], [12]. However, due to the complexity of correlated variations (e.g., delay, power and temperature), it is hard for designers to determine which TAS strategy works best for a given MPSoC TAS problem. Therefore, making the quantitative evaluation and comparison among TAS strategies has become an important issue to guarantee the performance yield in MPSoC design. Although existing statistical graph analysis-based methods can deal with simplified execution variations, few of them can accurately model complex parallel task execution scenarios and correlations between different types of variations [4]. Moreover, constraint solving-based approaches can only answer whether a given MpSoC TAS problem satisfies a given constraint. None of these approaches can quantitatively reason why a required performance yield cannot be achieved and how to achieve a better TAS solution with near-optimal performance yield. **Clearly, the bottleneck is the lack of powerful evaluation and optimization methods that can help MPSoC designers to make sustainable and reliable TAS decisions.**

Based on the Statistical Model Checking (SMC) [13] and regression analysis [15] approaches, this paper proposes a novel framework that can effectively conduct performance yield queries and optimization for energy- and thermal-aware MPSoC designs under variations. Our framework adopts the model checker UPPAAL-SMC [16] as the engine of variation-oriented evaluation. Compared with formal model checking approaches, UPPAAL-SMC allows approximate evaluation of complex MPSoC systems, thus it requires far less memory and validation time. By simulating systems using underlying statistical methods (i.e., sequential hypothesis testing and Monte Carlo simulation), UPPAAL-SMC can estimate the satisfaction probability of a specified performance query (i.e., performance yield) under temperature and energy constraints. Generally, the TAS optimization requires evaluation of a large set of feasible solutions. Since the evaluation of a single TAS solution is alreay time-costly, our framework employs regression analysis to predict the best possible TAS solution based on the evaluation of a small subset of labelled TAS solutions. This paper makes two following major contributions:

- We propose a novel UPPAAL-SMC-based approach that can automatically convert TAS solutions under correlated energy and thermal constraints into *Networks of Priced Timed Automata* (NPTA) [16], which enables accurate evaluation of corresponding performance yield.
- We develop a regression analysis-based optimization approach that can quickly find the best possible energy- and thermal-aware TAS solution for a specific MPSoC architecture (i.e., floorplan) under variations.

The remainder of this paper is organized as follows. Section 2 presents related work on variation-aware task allocation and scheduling approaches for MPSoC designs. After a brief introduction to the power and thermal modeling of MPSoCs in Section 3, Section 4 presents the details of our evaluation and optimization framework. Section 5 presents comprehensive experimental results based on a synthetic application. Finally, Section 6 concludes the paper.

2 Related Works

With the advent of the MPSoC architecture in CPS design [2], various application mapping techniques are proposed to optimize the design from the perspectives of performance, temperature and energy [17]. For example, Coskun et al. [18] proposed an ILP-based approach to minimize hotspots and balance temperature distribution on the die for a set of tasks. In [19], Huang and Xu proposed novel task allocation and scheduling algorithms to minimize the expected energy consumption of multi-mode embedded systems under performance and lifetime reliability constraints. In [20], Chantem et al. presented a mixed ILP-based solution to optimize peak temperature under various constraints based on phased steady-state thermal analysis. Although the above approaches are promising in optimizing the temperature and energy, none of them consider the impact of process variations.

Due to the aggressive technology scaling, the effect of process variations in microelectronic circuits is widely investigated [4]. Based on the assumption that the execution time of MPSoC tasks can be approximated with Gaussian distribution [21], various TAS approaches were proposed to minimize the impact of process variation while maximizing the performance. In [11], Wang et al. introduced the concept of performance yield for MPSoC designs. Assuming that the task execution time follows the Gaussian distribution [21], they proposed an efficient TAS algorithm to maximize performance yield based on statistical task graph analysis. In [22], Chon and Kim proposed an efficient method to schedule and bind tasks in an acyclic task graph to MPSoC resources in the presence of resource sharing. Using simulated annealing-based scheduling method and clustering-based performance yield enhancement technique, Huang and Xu [12] took the spatial correlation of within-die variation into account and presented a novel quasi-static scheduling approach to improve the overall performance yield. Although the above approaches can obtain higher performance yield, few of them investigated the performance yield involving the temperature- and energy-based constraints. Moreover, none of the above approaches considers the floorplan information in their variation-aware performance yield analyses.

Machine learning-based algorithms are widely investigated in MPSoC domain for the purpose of design optimization. For example, Coskun et al. [23] proposed an online learning-based low-cost temperature management strategy for multicore systems. Their approach can be used to reduce the adverse effects of hotspots and temperature constraints. In [24], Tan et al. presented a novel online power management technique based on model-free constrained reinforcement learning. Compared with other existing power management method, the proposed approach is capable of exploring the trade-off in the power-performance design space and converging to a better power management policy. To reduce the non-recurring engineering (NRE) costs and time-to-market, Almer et al. [25] proposed a machine learning-based method which can automatically generate near-optimal application specific SoC designs within hours rather than weeks. Although various heuristics were proposed to improve different kinds of performance, none of existing machine learning-based approaches considers the joint energy and thermal constraints under process variations in MPSoC design.

Statistical model checking has been widely used in evaluating system designs under variations [26]. In [27], Chen et al. presented a variation-aware MPSoC design framework that supports the TAS evaluation under the power and timing constraints. However, they did not consider the correlation between the power and temperature of PEs with different floorplan layouts. Moreover, the work in [27] only focuses on TAS evaluation rather than optimization. Although David et al. optimized energy consumption for smart buildings based on UPPAAL-SMC [28], their approach strives to optimize the cost values on a parameterized smart building model, while our approach targets to find a solution with the best possible performance yield from a large set of MPSoC design models.

To the best of our knowledge, our work is the first SMC-based approach that not only can evaluate different performance aspects of TAS strategies considering temperature and energy constraints under process variation, but also can efficiently optimize the floorplan-aware task mapping to achieve the highest performance yield.

3 Background

3.1 UPPAAL-SMC

Relying on the formal models of Network of Priced Timed Automata (NPTA) [29], UPPAAL-SMC [16] provides a user-friendly interface to enable the quantitative performance analysis of complex stochastic systems. NPTA comprises a set of correlated Priced Timed Automata (PTA) that are synchronized via broadcast channels and shared variables. Figure 1 shows an example of an NPTA consisting of two PTAs, i.e., A (id=ida) and B (id=idb), where each PTA has four locations, two variables (e.g., t_1 indicating the delay time and p_1 indicating the power in location A_2 for PTA A) and two local clocks (e.g., c_1 indicating the execution time and e_1 indicating the consumed energy for PTA A), respectively. As a variant of timed automata, PTA allows clocks with different rates in different locations. The value of a primed clock denotes the rate of the clock. For example, $e_1' == p_1$ in PTA A is used to record the energy consumed by PTA A with a rate of p_1. When dealing with the composite state transition of an NPTA, if a PTA process is in a *commit* or *urgent* location (i.e., a location marked with the symbol "C" or "U"), the process will have a zero delay in this location and the next transition should involve an outgoing edge from one of the *commit* or *urgent* locations (*commit* locations have higher priority). Otherwise, after each decision, the PTA process with the shortest delay will attempt to take a transition and all the continuous variables will be updated accordingly. In Figure 1, the synchronization between two PTAs is based on an array of two broadcast channels, i.e., *msg[]*. To filter useless messages, we use the non-deterministic *selections e:msg_t* and the guard condition *e==idb* to filter messages which are not sent to PTA *B*.

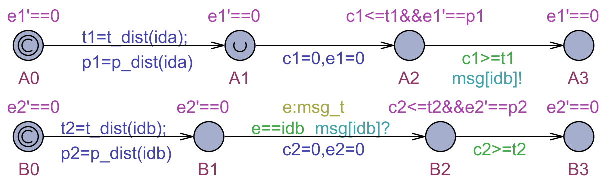


Fig. 1. An NPTA, (A | B)

We use the design pattern shown in Figure 1 to enable the stochastic behavior modeling and evaluation of MPsoC designs. For simplicity, Figure 1 only presents an example considering the variations of power and energy. Currently, UPPAAL-SMC only explicitly supports the uniform and exponential distributions. However, due to the built-in function $random()$ and C-like programming constructs, UPPAAL-SMC can generate constant values following a large set of commonly used distributions (e.g., normal distribution, Poisson distribution). For example, by using the *Box-Muller method*, we can derived values following normal distributions. In this example, functions $t\_dist()$ and $p\_dist()$ are used to derive the random delays (i.e., t_1 and t_2) and power configurations (i.e., p_1 and p_2) for the PTAs following some specified distributions. In the pattern, the location A_2 sets the upper bound for clock c_1 (i.e., $c_1 <= t_1$) and its outgoing transition has the guard condition $c_1 >= t_1$. Therefore, PTA A must stay in location A_2 with a delay of t_1. Meanwhile, the energy consumption rate of A in location A_2 is denoted by the location invariant $e_1' == p_1$. Based on the above template using message-based synchronization among PTAs, arbitrarily complex stochastic behaviors of MPSoC designs can be modeled.

To enable the quantitative evaluation of NPTA-based designs, UPPAAL-SMC employs the property-based performance queries in cost-constrained temporal logic [29] format. The query is in the form of $Pr[cost <= bound](<> expr)$, where $[cost <= bound]$ indicates the bound of cost (e.g., time, power), and the expression $<> expr$ asserts that the predicate $expr$ should be hold eventually. If the $cost$ is not specified explicitly, $[<= bound]$ indicates the bound of system time. Based on the specified *probability of false negatives* (i.e., α) and *probability uncertainty* (i.e., ε), UPPAAL-SMC will generate and execute a fixed number of random runs. By monitoring these runs bounded by either time or design constraints, the probability range of each query (i.e., $[p-\varepsilon, p+\varepsilon]$) with a specified confidence degree (i.e., $1-\alpha$) will be reported, where p indicates the success ratio of the given query. The performance query details will be described in Section 4.4.

3.2 Power and Thermal Modeling

It is important to note that our TAS evaluation and optimization framework itself is independent of the power and thermal models used. In other words, users may define their own power and thermal models by modifying the functions which calculate new power and new temperature, and these models can be easily converted to corresponding NPTA templates in a similar way as presented in Section 4.3. To illustrate the usage of our framework, this subsection presents a general power and thermal model, which can model the correlation between power consumption and temperature of PEs.

Assume that there are m PEs in an MPSoC design. For each PE, the power consumption consists of two parts: dynamic power P_{dyn} and leakage power P_{leak}. Since dynamic power consumption is weakly coupled with temperature variation, we consider the value of P_{dyn} as a constant. Note that leakage power consumption is a strong function of temperature. Thus the overall power of the i^{th} PE can be formulated as:

$$P_i(t) = P_{leak,i}(t) + P_{dyn,i}. \quad (1)$$

Since our framework takes the temperature impacts among PEs into consideration, we adopt the well-known RC thermal model [32], [35] to investigate the temperature correlation among PEs. In the RC model, C_i denotes the thermal capacitance of the i^{th} PE, and $R_{i,j}$ indicates the thermal resistance between the i^{th} PE and the j^{th} PE. The temperature of the i^{th} PE can be formulated as:

$$C_i \cdot \frac{\mathrm{d}T_i(t)}{\mathrm{d}t} + \frac{T_i(t) - T_{amb}}{R_{i,i}} + \sum_{j \neq i} \frac{T_i(t) - T_j(t)}{R_{i,j}} = P_i(t) \quad (2)$$

where T_{amb} indicates ambient temperature. Note that, although Equation (2) is a first order Ordinary Differential Equation (ODE), it does not need to be solved within our framework. This is because UPPAAL-SMC allows location invariants in the form of ODEs with primed clocks [16].

4 Our SMC-Based Evaluation and Optimization Approach

This section presents our TAS evaluation and optimization framework in details. Our approach focuses on how to improve the performance yield under energy and thermal constraints considering the variations of time and power.

4.1 Notations and Problem Definition

Besides the two types of variations (task execution time and power) addressed in this paper, our approach can be easily extended to model other types of variations, based on the template shown in Figure 1. To accurately describe the stochastic behaviors of MPSoC designs, we adopt the distribution-based methods for modeling variations. The MPSoC TAS evaluation and optimization under variations studied in this work is formulated as follows.

- Let $G = (V, E)$ be a task graph in the form of directed acyclic graph (DAG), where $V = \{v_1, \ldots, v_n\}$ denotes the task set and E indicates precedence constraints between tasks.
- Let F be the floorplan of an MPSoC design which consists of a set of PEs $PE = \{p_1, \ldots, p_m\}$ with specific placement. Let $PT = \{pt_1, \ldots, pt_k\}$ be the set of PE types of F. We use the PE type function $T_{PE} : PE \to PT$ to specify the type of an PE p_i ($p_i \in PE$), and use the task type function $T_{task} : V \to PT$ to denote the type of PEs to which task v_j ($v_j \in V$) can be allocated. A task v_j can be assigned to p_i only if $T_{task}(v_j) = T_{PE}(p_i)$.
- Let the function $ET : V \times PT \to \mathbb{R}^+$ be the nominal execution time function for tasks, where $ET(v_i, pt_j)$ represents the nominal execution time of task v_i running on a PE of type pt_j. Let $DIST$ be the set of probability distributions. We use $ETD : V \times PT \to DIST$ to specify the execution time variation of tasks, where $ETD(v_i, pt_j) = dist$ denotes the execution time of task v_i running on a PE of type pt_j following the distribution $dist$.
- Let the function $PD : PT \to DIST$ specify the power variations of PEs, where $PD(pt_i) = dist$ indicates that the power consumption of PEs of type pt_i follows the distribution $dist$. Considering the PE power variations and temperature influence among PEs, we use the function $Power : PE \times \mathbb{R}^+ \to \mathbb{R}^+$ to denote real time power consumption of PEs, i.e., $Power(p_i, t)$ denotes the power consumption of p_i at time t.
- Let the function $Energy(t) = \int_0^t \sum_{i=1}^m Power(p_i, t)\mathrm{d}t$ denote the overall energy consumed by the MPSoC design till time t.
- Let the function $Hotspot : \mathbb{R}^+ \to \{0, 1\}$, denote whether there exists a PE on F with a temperature higher than the specified hotspot temperature at time t. The function $T_{hotspot}(t) = \int_0^t Hotspot(t)\mathrm{d}t$ is a metric that denotes the overall MPSoC overheating time till time t.
- Let mp, rt, $t_{hotspot}$, me, and fu be design constraints, which denote the power limit, response time, overheating time limit, energy limit, and available function units, respectively.
- A **TAS solution** $tass_x$ is a 2-tuple in the form of $(<v_{x,1}, \ldots, v_{x,n}>, <vp_{x,1}, \ldots, vp_{x,n}>)$ indicating that the task $v_{x,i}$ is assigned to a PE virtually indexed by $vp_{x,i} \in \mathbb{N}^+_{\leq m}$ without violating the constraint fu. The schedule sequence $<v_{x,1}, \ldots, v_{x,n}>$ is a permutation of tasks of V in an order such that $v_{x,j}$ cannot be dispatched earlier than $v_{x,i}$ if $i<j$.
- Let $FP : \mathbb{N}^+_{\leq m} \to PE$ be a PE mapping function, where $FP(vp_i)$ indicates the real PE virtually indexed by vp_i. Considering the floorplan F, a **mapped TAS solution** $mtass_x$ is mapping from tasks to real PEs. It is a 2-tuple $(<v_{x,1}, \ldots, v_{x,n}>, <FP(vp_{x,1}), \ldots, FP(vp_{x,n})>)$ where $FP(vp_{x,i})$ indicates the assigned PE running the task $v_{x,i}$.
- Let $mtass$ be a mapped TAS solution, and $END(v_i, mtass)$[1] be the finish time of task v_i when executing $mtass$. Under the variations (i.e., execution time variation ETD and power variation PD) and design constraints (i.e., mp, rt, $t_{hotspot}$, and me), the performance yield $PY(mp, rt, t_{hotspot}, me, mtass)$ of the MPSoC with $mtass$ denotes the probability that the design meets the constraints such that $\sum_{i=1}^m Power(p_i, t) \leq mp$ for any t, $Max_{i=1}^n END(v_i, mtass) \leq rt$, $T_{hotspot}(rt) \leq t_{hotspot}$, and

1. Note that $END(v_i, mtass)$ is affected by ETD, and $Power(p_i, t)$ is affected by PD and temperature influence among PEs.

$Energy(Max_{i=1}^{n} END(v_i, mtass)) \leq me.$

Note that TAS solutions do not take floorplans into account. The exploration of a TAS solution is to assign tasks to corresponding indexed virtual PEs only considering the task precedence relations (denoted by E) and constraint of available function units (denoted by fu). Although there may exist multiple feasible TAS solutions for the given MPSoC design constraints, for each TAS strategy our approach only adopts the one that is found first. The difference between distinct mapped TAS solutions using the same strategy is that they are based on different floorplan-based PE mapping functions. Given a set of mapped TAS solutions generated by different TAS strategies with different PE mapping functions, the TAS evaluation process is to figure out which mapped TAS solution has the best performance under variations. Unlike evaluation, the optimization process will try to evaluate all possible TAS solutions for the same strategy with the same design constraints but different PE mapping functions. The goal of optimization is to find one mapped TAS solution with the highest performance yield under variations.

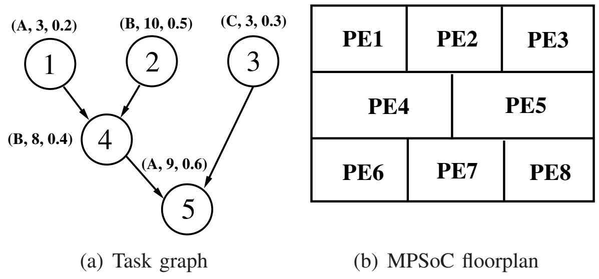


Fig. 2. An example of floorplan-aware MPSoC TAS

As an example shown in Figure 2, we are trying to schedule the tasks shown in Figure 2(a) onto an MPSoC design with the floorplan specified in Figure 2(b). In Figure 2(a), each task is labelled with the information of allocated PE type and execution time variation. For example, task v_1 is labelled with a setting $(A,3,0.2)$ indicating that this task will be assigned to PEs of type A and its execution time follows the Gaussian distribution $N(3,0.2^2)$. Assume that $T_{PE}(PE_1) = T_{PE}(PE_2) = T_{PE}(PE_3) = A$, $T_{PE}(PE_4) = T_{PE}(PE_5) = B$, and $T_{PE}(PE_6) = T_{PE}(PE_7) = T_{PE}(PE_8) = C$. Let $tass$= $(<v_1,v_2,v_3,v_4,v_5>,<1,5,6,4,1>)$ be a TAS solution of the problem. Assume that we have a PE mapping function indicates the binary relation of virtual indexed PEs to real PEs, i.e., $\{(1,v_1),(2,v_2),(3,v_3),(4,v_5),(5,v_4),(6,v_6),(7,v_7),(8,v_8)\}$. Based on such PE mapping function, we can get a mapped TAS solution $mtass_1 = (<v_1,v_2,v_3,v_4,v_5>,<PE_1,PE_4,PE_6,PE_5,PE_1>)$ for $tass$. Note that $mtass_2 = (<v_1,v_3,v_2,v_4,v_5>,<PE_1,PE_5,PE_6,PE_4,PE_3>)$ is not a mapped TAS solution generated from $tass$, since v_1 and v_5 are not mapped to the same PE. Under the execution time and power variations coupled with the energy and thermal constraints, it is difficult for existing approaches to determine which mapped TAS solution ($mtass_1$ or $mtass_2$) has a better performance yield. Furthermore, it is a major challenge to answer whether $mtass_1$ or $mtass_2$ has the highest performance yield among all the feasible mapped TAS solutions for the given task graph and MPSoC platform.

4.2 Our Framework

Figure 3 presents our proposed UPPAAL-SMC-based TAS evaluation and optimization framework. Based on the MPSoC design information (i.e., a task graph with task execution information and the specification of the adopted MPSoC platform) and the TAS strategy, our framework can generate one TAS solution and figure out all the corresponding mapped TAS solutions. By combining the power model, thermal model and platform variation information of the MPSoC design, our framework automatically transforms the mapped TAS instances into corresponding executable NPTA models. Meanwhile, the specified design constraints can be converted to properties to enable the quantitative evaluation of the mapped TAS instances. By checking the generated NPTA models and properties using the UPPAAL-SMC model checker, we can compare among the mapped TAS solutions. For the purpose of optimization, it is required to evaluate all the possible mapped TAS solutions, which is time-consuming. To reduce the optimization time, our framework employs the regression analysis that only needs to check a small subset of the sampled mapped TAS solutions. By using the Back Propagation Neural Network (BPNN)-based approach [33] on the evaluated NPTA models in the regression set, we can predict and rank the performance yield of the remaining NPTA models in the prediction set. Finally, the selected top-ranking NPTA models will be evaluated using UPPAAL-SMC to validate the prediction results. Since only a small part of the mapped TAS solutions are evaluated, the overall TAS optimization time can be significantly reduced. The following sub-sections will describe the major components of our framework in detail.

4.3 NPTA Model Generation

Aiming at evaluating mapped TAS instances using SMC, we need to first covert the mapped TAS instances into executable NPTA models. In our approach, we adopt six kinds of PTAs: task, PE, power monitor, temperature monitor, hotspot monitor, and hotspot timer. The task PTA models the execution of a single task. The PE PTA presents the behavior of a single PE dealing with multiple assigned tasks. The power monitor, hotspot monitor and hotspot timer PTAs monitor the power usage and overheating time of the whole chip, while the temperature monitor models the temperature change for a single PE. To facilitate the model construction, we decouple an NPTA model for MPSoC designs into two parts: i) front-end models which describe the common behaviors of MPSoC designs, and ii) back-end configuration which consists of necessary data structures (e.g., thermal models, task graph DAG, variation information, synchronization, etc.) to guide the stochastic simulation for the specified TAS solution. Note that in our approach, all TAS solutions share the same front-end models. To simplify the model and property generation, we introduce a dummy task (i.e., a task whose execution time is 0) with $tid = 0$ to merge all the tasks without any successors. In our approach, we assume that there are $T+1$ tasks (with $tid \in [0,T]$) and P PEs (with $pid \in [0,P-1]$).

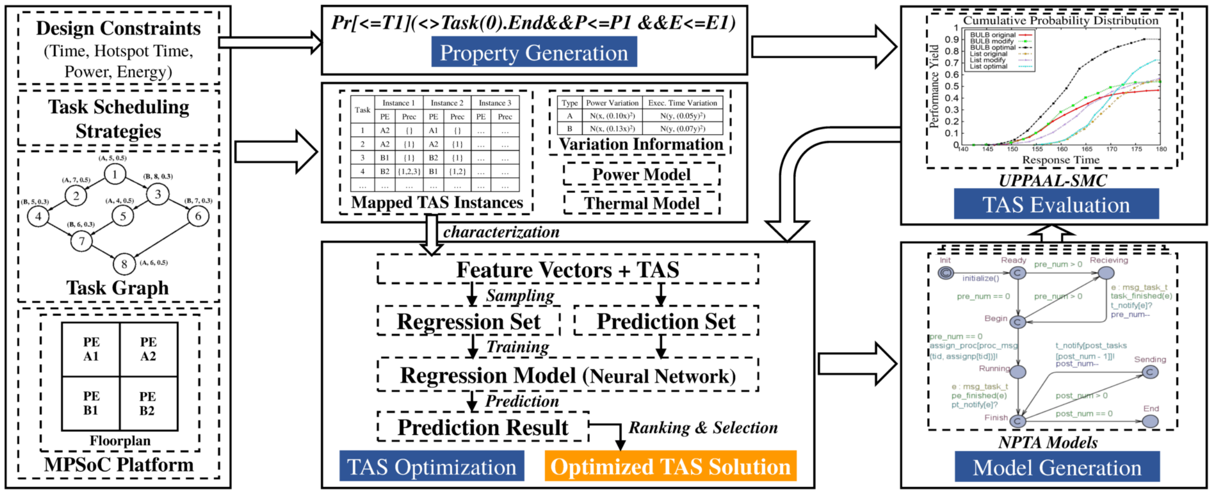


Fig. 3. Our TAS strategy evaluation and optimization framework

4.3.1 Back-end Configuration Generation

According to the semantics of the task graph, a task can be executed only when its precedent tasks are all completed. As an example shown in Figure 2, for $mtass_1$, task v_4 can start execution only when both of its predecessors v_1 and v_2 are finished. To model the concurrent execution of tasks, we adopt a precedence matrix *PM[T+1][T+1]* to indicate the task precedence relations (indicated by both the elements of the 2-tuple $tass_x$ as defined in Section 4.1), where *PM[i][j]=1* indicates that task v_j can start its execution immediately when task v_i is finished. Note that besides the task precedence relations posed by the edges of a task graph, *PM* also contains the task precedence relations derived during the task allocation and scheduling. For example, assume that $<\ldots,v_i,\ldots,v_j,\ldots>$ is a schedule sequence, where v_i and v_j are parallel tasks, i.e., there is no path from v_i to v_j in the task graph. If v_i and v_j are mapped to the same PE, we need to set $PM[i][j]$ to 1. Based on this precedence matrix, each task should be aware of the status of its predecessors and successors. In our back-end configuration, we define two arrays *pre_count[T+1]* and *post_count[T+1]* to denote the number of predecessors and successors for each task. Both arrays are initialized based on the precedence matrix.

To model the variation information of MPSoC platforms, we use two multi-dimensional arrays *tvar* and *pvar* in the back-end configuration to specify the distributions of execution time and power consumption for each PE. Note that our framework supports a wide range of widely used distribution models which facilitate the modeling of time and power variations. For example, if designers select the Gaussian distribution to model the power consumption variation, a two dimensional array *pvar[T+1][2]* will be used to specify the mean value and standard deviation of power for each PE. Based on such distribution information, the arrays *real_time[T+1]* and *real_power[P]* in the back-end configuration will be initialized to denote the real execution time for each task and the real power for each PE at the beginning of simulation.

Due to the correlation between the power and temperature of PEs as introduced in Section 3, to accurately model the TAS execution, our approach allows the RC modeling as presented in Section 3. In the back-end configurations, we use two matrices $R[P][P]$ and $C[P][P]$ to denote the thermal resistance matrix and thermal capacitance matrix with constant coefficients, respectively. Both matrices can be either obtained from MPSoC producers or generated by floorplanning tools.

To enable the PTA synchronization, our framework adopts the messages of different types to conduct the synchronization for specific purposes. For example, in our approach, task PTAs, PE PTAs, and the power PTA can only accept messages of type *msg_task_t*, *msg_proc_t* and *msg_power_t*, respectively. For the communication between task PTAs, the back-end configuration contains a broadcast channel array *t_notify[T+1]*, where *t_notify[i]* indicates a private channel for the i^{th} task to receive notifications from predecessor tasks. With respect to task allocation, the array *assignp[T+1]* in the back-end configuration is used to indicate the floorplan-aware task-to-PE mapping relation (i.e., the PE mapping function defined in Section 4.1), and the broadcast channel array *assign_proc*$[(T+1)\times(P)]$ is used to enable the dispatching of tasks to PEs, where the channel *assign_proc[e]* is used to assign a task with $tid = e/P$ to a PE with $pid = e\%P$. When a PE finds that the current task v_j finishes its execution, the PE will notify v_j via its private channel $pt\_notify[j]$ about its completion. To model the real-time power of MPSoC designs, the back-end configuration consists of two channel arrays *requestP[P]* and *freeP[P]* which can dynamically update the power consumption status of currently running PEs. Since each PE has an associated temperature monitor, to enable the switch of monitoring states, PE p_i needs to send notification messages via the channels *m_task_start[i]* and *m_task_finish[i]*. The back-end configuration also defines four urgent channels (*temp_high*, *temp_lower*, *hotspot_timing_start*, *hotspot_timing_stop*), which can trigger transitions immediately when their associated guards satisfy.

4.3.2 Front-end Configuration Generation

Based on the global data structures defined in the back-end configuration, the front-end models can be instantiated to enable the stochastic simulation. According to the semantics of task graphs, all the tasks have the same behavior pattern. Therefore, we only need to construct one task PTA model for all the tasks. Figure 4 presents the details of the task PTA. The *Init* state initializes the data structures of the whole NPTA. For example, based on the precedence matrix *PM*, we can figure out the successor tasks (saved in the local array *post_tasks[T+1]*) and the count of predecessor and successor tasks of the current task (denoted by local variables *pre_num* and *post_num*). *Init* state also sets the real execution time for each task and the real power consumption for each PE with the randomly generated values based on the specified distribution information. The *Receiving* state tries to receive the notifications from all its predecessors. As soon as all the predecessors are finished (i.e., *pre_num =0*), the *Running* state will dispatch the task to the PE with $pid = assignp[tid]$. After receiving the execution completion notification from the PE via channel *pt_notify[tid]* as shown in the *Finish* state, the *Sending* state tries to notify all its successors saved in *post_tasks* about the completion of the current task.

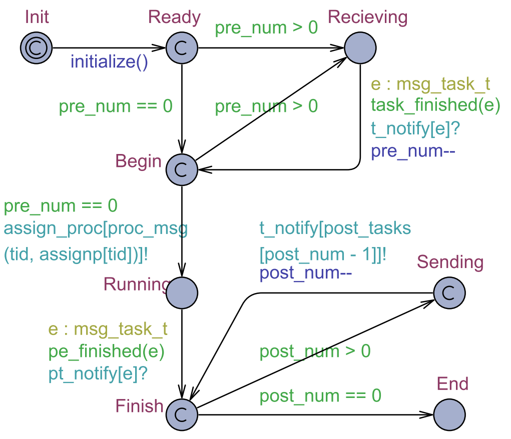


Fig. 4. Front-end model of a task

PEs have the same behavior pattern as shown in Figure 5. In our approach, each PE has a task queue to keep the ready tasks in the dispatching order. The PE PTA model consists of three major states. The *Waiting* state tries to receive new tasks when its task queue is empty. Before entering the *Running* state, the PE will notify the power monitor to update the overall power via channel *request_p[pid]* and notify its temperature monitor to switch its temperature changing mode via the channel *m_task_start[pid]*. In the *Running* state, the PE will choose task v_x at the head of the queue for execution and the execution time is saved in *real_time[x]* which follows the specified distribution. When task v_x completes its execution (indicated by the *Finish* state), the PE will notify v_x about its completion, and notify its power and temperature monitors to conduct the corresponding state switch.

The power monitor is created to monitor the overall real-time power and accumulated energy consumption for TAS solutions. As shown in Figure 6, the model has two states. The *Handling* state deals with the requests form PEs for increasing or decreasing the power. The *Waiting* state waits for new power requests and records the accumulated consumed energy. It is

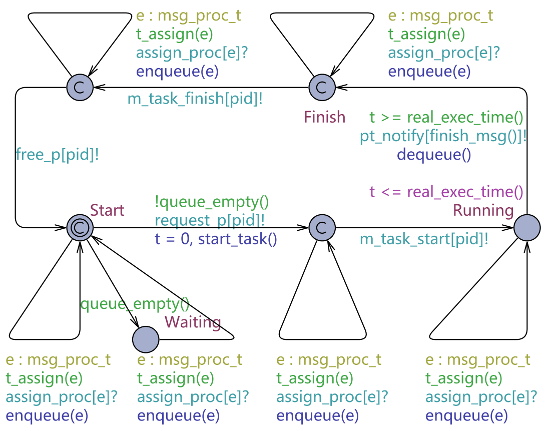


Fig. 5. Front-end model of a PE

important to note that, due to the correlation between the PE temperature and power, we use the function *power_update()* to periodically update the overall power of all the PEs by using the formula presented in Equation (1). The value of the latest overall power is saved in *current_power*.

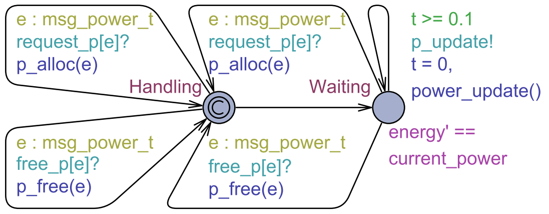


Fig. 6. Front-end model of the power and energy monitor

During the simulation of TAS solutions, the temperature of different PEs is different. In order to reflect the real-time temperature for each PE, we adopt the PTA shown in Figure 7 for each PE. The model has two states (*P_idle* and *P_running*) which indicates whether there is a task running on the PE. When calculating the temperature, the *P_idle* state only considers the leakage power, while the *P_running* state takes both dynamic and leakage power into account. The temperature rate functions *temp_idle()* and *temp_running()* are created based on the formulas presented in Equation (2).

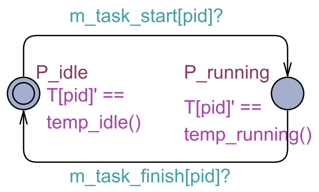
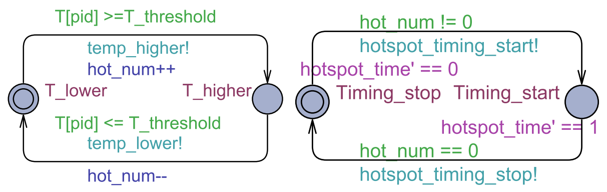


Fig. 7. Front-end model of the temperature monitor

When evaluating MPSoC temperature, *hotspot* is an important factor which contributes to the system reliability and cooling cost. Based on the PTAs shown in Figure 8, our approach supports the quantitative analysis of hotspot features. For each PE, we adopt a PTA shown in Figure 8(a) to indicate whether the temperature of current PE p_{pid} exceeds the threshold temperature, i.e., *T_threshold*. For the overall MPSoC design, we use one hotspot monitor to record the up-to-date overheating time $T_{hotspot}(t)$ as defined in Section 4.1.



(a) Hotspot Monitor (b) Hotspot Timer

Fig. 8. Front-end models for hotspot

4.4 Property Generation

To compare the performance yield of generated NPTA models considering the temperature and power constraints, we assume that MPSoC designers would like to figure out the problems like *"what is the probability such that the task graph can be completed under the hotspot time constraint x and energy constraint y within time z?"* or *"given a certain amount of energy/hotspot time x, what is the probability a specific task graph can be completed under the time constraint y and hotspot time/energy constraint z?"*. In our framework, the above queries can be automatically converted using the following property templates:

- Pr[<=z] (<> T(0).End && hotspot_time<x && energy<=y)
- Pr[energy<=x] (<> T(0).End && hotspot_time<=z && time<=y)
- Pr[hotspot_time<=x] (<> T(0).End && energy<=z && time<=y)

Here, $T(0).End$ denotes the completion of the task graph, *time* means the overall elapsed time, *hotspot_time* indicates the overheating time, and *energy* represents the overall energy consumption. Based on the monitoring of a large number of stochastic simulation runs, UPPAAL-SMC will report the probability distribution of successful simulations after the check finishes. Such information can be used to evaluate the performance of TAS solutions.

4.5 TAS Optimization

Since our approach focuses on the evaluation and optimization of task allocation and scheduling under thermal and energy constraints, we need to consider the underlying MPSoC floorplans and corresponding task-to-PE mappings. After figuring out a feasible TAS solution for a given task graph and resource constraints using existing TAS heuristics (e.g., BULB [36], list scheduling [37]), our TAS optimization tries to figure out a task-to-PE mapping (i.e., PE mapping function) that can lead to the highest performance yield for the TAS solution. To quickly explore such an optimal result, our approach adopts BPNN [33] as the TAS optimization engine as shown in Figure 3. The reason why we choose BPNN to perform regression analysis and prediction of performance yield is because BPNN outperforms other popular regression models when reasoning the performance metrics of variation-aware models [14]. It needs less regression model generation time but still can achieve the expected prediction accuracy. It is important to note that BPNN is not the only option to perform the regression. In fact, other regression methods (e.g., *Support Vector Regression* [34], *M5 Model Tree*) can be easily integrated into our framework for the TAS optimization.

```
Algorithm 1: Our TAS optimization approach
Input: i) A TAS solution $tas$;
       ii) A floorplan of the MPSoC platform $fp$;
       iii) The performance query $prop$;
Output: An optimized mapped TAS solution with best possible
        performance
TASOptimization($tas$, $fp$) begin
    1: $\{FP_1,\ldots,FP_n\} = FloorPlanFuncGen(fp)$;
    2: $TAS = \{tas_1,\ldots,tas_n\} = \{FP_1(tas),\ldots,FP_n(tas)\}$;
    3: $FV = \{fv_1,\ldots,fv_n\} = Characterization(TAS)$;
    4: $(Reg, Pre) = Sampling(TAS, FV)$;
    5: $max\_py = 0$, $best\_tas = \emptyset$;
    6: $reg\_result = \{\}$, $pre\_result = \{\}$;
    for each (tas, fv) pair in Reg do
        7: $model = NPTAGen(tas)$;
        8: $py = SMC(model, prop)$;
        9: $best\_tas = max\_py \geq py \ ?\ best\_tas \ :\ tas$;
        8: $max\_py = MAX(max\_py, py)$;
        10: $reg\_result = reg\_result \bigcup (tas, fv, py)$;
    end
    11: $predictor = Regression(reg\_result)$;
    for each (tas, fv) pair in Pre do
        12:
        $pre\_result = pre\_result \bigcup (tas, fv, predictor((tas, fv)))$;
    end
    13: $ranked\_tas = Rank(pre\_result)$;
    14: $sel\_tas = Select(ranked\_tas)$;
    while $sel\_tas.empty() == FALSE$ do
        15: $tas' = sel\_tas.deque()$;
        16: $temp\_py = SMC(NPTAGen(tas'), prop)$;
        if $temp\_py \geq max\_py$ then
            17: $best\_tas = tas'$;
            18: $max\_py = temp\_py$;
        end
    end
    19: Return $(best\_tas, max\_py)$;
end
```

Algorithm 1 outlines the major steps in our TAS optimization approach. For a given floorplan, step 1 enumerates all possible PE mapping functions. Based on these mapping functions, we can obtain all the mapped TAS solutions for a given unmapped TAS solution in step 2. By extracting the feature vectors of the mapped TAS solutions in step 3, step 4 conducts the sampling to split the mapped TAS solutions into two separate sets, i.e., the regression set (dented by *Reg*) and prediction set (denoted by *Pre*). The regression set is used to generate the regression model, which can be used to predict the performance yield for the mapped TAS solutions in the prediction set. Steps 5 initializes the best mapped TAS solution and its performance yield, while steps 6 initializes the regression and prediction results. Steps 7-10 evaluate the mapped TAS solutions in the regression set one by one and save the evaluation results in *reg_result*. Meanwhile, the best mapped TAS solution as well as its performance yield are saved in *best_tas* and *max_py*, respectively. Based on the evaluation results of the TAS solutions in regression set, step 11 generates the regression model, which is used to predict the performance yield for the mapped TAS solutions in the prediction set as shown in step 12. Step 13 sorts the mapped TAS solutions with regard to their performance yield in a descending order. Step 14 selects three mapped TAS solutions with the highest predicted performance yield from the prediction set. Steps 15 and 16 re-evaluate these selected solutions using the UPPAAL-SMC. If a solution is found to be better than the best solution searched so far, steps 17 and 18 will record this solution as the new best solution. Finally, step 19 reports the optimized mapped TAS solution with the best possible performance yield under the power and temperature constraints. Note that feature selection (step 3) and sampling (step 4) are two key steps in our approach, since they determine the accuracy of the prediction. The following subsections will describe these two steps in detail.

4.5.1 Feature Selection for TAS Solutions

To guarantee the accuracy of prediction, it is required to figure out the features which can significantly reflect the performance of the mapped TAS solutions. Since our approach focuses on finding PE mapping function for a TAS solution to achieve the highest performance yield, based on the criteria of significance, independence and diversity, our approach investigates the following two kinds of features of each PE for the given floorplan and TAS solution: i) the physical attributes of PE p_i (i.e., $R_{i,i}$ and C_i in the RC model [35] as defined in Section 3.2); and ii) the task to PE mapping information. Due to the same task dispatching order and same mapped PE types for all the mapped TAS solutions, we do not consider the task dependence relation and the PE variation information in the feature selection. For a mapped TAS solution, there may be multiple tasks allocated to the same PE. To simplify the feature representation, we use the following encoding to denote the tasks assigned to the PE p_i:

$$M_i = \sum_{j=1}^{n} (X_{i,j} \times 2^j) \tag{3}$$

where $X_{i,j} = 1$ indicates that task v_j is allocated to the PE p_i., and 0 otherwise. As an example shown in Figure 2, the TAS solution $mtass_1$ has two tasks (i.e., v_1 and v_5) assigned to PE_1. Therefore, we can get $M_1 = 1\times 2^1 + 1\times 2^5 = 34$ for PE_1. To calculate the overall running time of a PE, we use the following formula:

$$T_i = \sum_{j=1}^{n} (X_{i,j} \times ET(v_j, T_{PE}(p_i))) \tag{4}$$

which indicates the accumulated running time of tasks on PE p_i. For the example of $mtass_1$, we can get $T_1 = 1\times 3 + 1\times 9 = 12$, since the overall nominal running time of tasks v_1 and v_5 in Figure 2(a) is 12.

In the feature vector, each PE p_i is represented by a 4-tuple in the form of $(R_{i,i}, C_i, M_i, T_i)$. If there are p PEs in the MPSoC design, the feature vector will contain p such 4-tuples. As an example shown in Figure 2, assuming that $mtass_1$ needs to be characterized, the following shows its feature vector.

$$<(19.5,0.2,34,12),\ (2.15,0.25,0,0),\ (19.5,0.2,0,0),\ (19,0.22,4,10),\ (19,0.22,16,8),\ (20.2,0.24,8,3),\ (20.7,0.27,0,0),\ (20.2,0.24,0,0)>$$

4.5.2 TAS Solution Sampling

The goal of sampling is to collect a set of feasible mapped TAS solutions for the regression model generation. To avoid biased prediction, such a set should be as representative as possible. If all the sampled TAS solutions have the same performance yield value, the prediction will be inaccurate. Therefore, it is required that the sampled TAS solutions should be evenly distributed in the whole solution space. Since we focus on TAS optimization, some observable TAS solutions with high performance yield should be included in the regression set. To achieve these two goals, our approach employs a hybrid sampling method with two strategies. In the first strategy, we uniformly sample the TAS solutions that are sorted based on their generation order. Assuming that there are m TAS solutions in total sorted based on its generation order, and n TAS solutions to be sampled, we may conduct the sampling by choosing every other $\lceil m/n \rceil$ TAS solutions. The second strategy tries to sample as many TAS solutions with high performance yield as possible. Since allocating tasks to PEs that are close to the die center can easily result in hotspots, to avoid such scenarios, our approach assigns each PE with a *bonus coefficient* which encourages allocating tasks to border PEs. The PEs on the die border have higher bonus, while the PEs far away from the die border have lower bonus. Assuming that $(< v_{x,1}, \ldots, v_{x,n} >, < PE_{x,1}, \ldots, PE_{x,n} >)$ is a mapped TAS solution, its overall bonus can be calculated using the formula $\sum_{i=1}^{n} ET(v_{x,i}, T_{PE}(PE_{x,i})) * Bonus(PE_{x,i})$. As an example shown in Figure 2, assuming that $Bonus$ is a binary relation $\{(PE_1,0.8), (PE_2,0.5), (PE_3,0.8), (PE_4,0.6), (PE_5,0.6), (PE_6,0.7), (PE_7,0.4), (PE_8, 0.7)\}$, $mtass_1$ has a bonus of $3\times 0.8 + 10\times 0.6 + 3\times 0.7 + 8\times 0.6 + 9\times 0.8 = 22.5$. By using this strategy, if there are n TAS solutions to be sampled, the n ones with highest bonus values will be sampled.

Note that both sampling strategies have their own strengths. For the first strategy, it can be used to obtain TAS solutions with different performance yields. For the second strategy that selects TAS solutions with highest bonus, the performance yields of the samples are quite similar. However, from their feature vectors, the regression analysis process can effectively learn why the performance yield is high. To guarantee the prediction accuracy, our approach employs a hybrid approach that combines the advantages of above two sampling strategies. By default, the hybrid sampling approach generates the same number of TAS solutions for the two different sampling approaches. It also allows designers to specify their own proportion for samples with different strategies.

5 CASE STUDY

5.1 Experiment Setup

This section presents the experimental results on a synthetic example using our proposed framework. In this example, we model the dynamic power consumption of PEs using the method presented in [35], and the static power consumption of PEs using the approach provided by [31]. We adopted the UPPAAL-SMC (version 4.1.19 with parameters $\varepsilon = 0.05$, $\alpha = 0.05$) as the engine of variation-aware TAS solution evaluation. We used the built-in BPNN tool from MATLAB to generate the regression model (with a setting of 50 neurons in hidden layer, $MSE = 0.001$, and a maximum of 50 epoch iterations)[2]. For the BPNN regression model generation, we employed the training function *scaled conjugate gradient backpropogation* to update weight and bias values. We used the *tan-sigmoid* transfer function in the hidden layer which generates outputs between -1 and 1. In the output layer, we used the *log-sigmoid* transfer function which generates outputs between 0 and 1 indicating the value of performance yields. We implemented the remaining components of our framework including the TAS solution generation heuristics, TAS to NPTA converter, property generator and TAS optimization procedure using the C programming language. For TAS solution generation, our framework supports three heuristics (i.e., BULB [36], list scheduling [37], and PVTS [11]) that can efficiently achieve feasible TAS solutions. In this experiment, we set the supply voltage of PEs to 1.1V, the ambient temperature to 35°C, and the temperature of thermal emergency (i.e., hotspot) to 85°C. All the experiments were conducted on Linux desktops with 4.0GHz CPU and 8 GB RAM.

TABLE 1
Power and Time Variations for MPSoC PEs

Type	Power Variation	Exec. Time Variation
A	$N(x, (0.10x)^2)$	$N(y, (0.05y)^2)$
B	$N(x, (0.13x)^2)$	$N(y, (0.07y)^2)$
C	$N(x, (0.15x)^2)$	$N(y, (0.10y)^2)$

By using the open source tool TGFF [38] which is designed to generate pseudo-random task-graphs, we obtained a 22-node task graph with a maximum input degree of 3 and a maximum output degree of 2. Beside ID information, each task node is assigned with its corresponding PE type and execution time information. Note that the execution time variations of all the tasks follow the distributions for PEs as specified in Table 1. For example, if the expected execution time of a task v running on PEs of type A is 10 time units, its execution time will follow the Gaussian distribution $N(10, 0.5^2)$, since the execution time of PEs of type A follows the Gaussian distribution $N(y, (0.05y)^2)$.

In the experiment, we assumed that the target MPSoC platform consists of 12 PEs, including 5 PEs of type A, 2 PEs of type B and 5 PEs of type C. To demonstrate the evaluation capability of our approach, we investigated the performance yield of the same TAS solution under different architecture. We conducted the experiment based on two architectures with the different floorplans (i.e., *MPSoC(5,5,2)* and *MPSoC(5,2,5)*) as shown in Figure 9. Due to the different floorplan layouts, the RC models of these two MPSoC platforms are quite different, which will inevitably affect the evaluation results. For each of above floorplan, there exists $5! \times 2! \times 5! = 28800$ different PE mapping functions. In other words, for a given TAS solution, there are 28800 different mapped TAS solutions. Note that both the floorplans shown in Figure 9 have a symmetric structure. If we want to figure out the best solution for a TAS problem under variations, we only need to evaluate $28800/2 = 14400$ mapped TAS solutions.

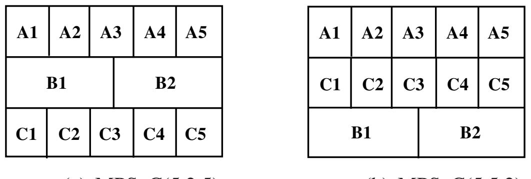


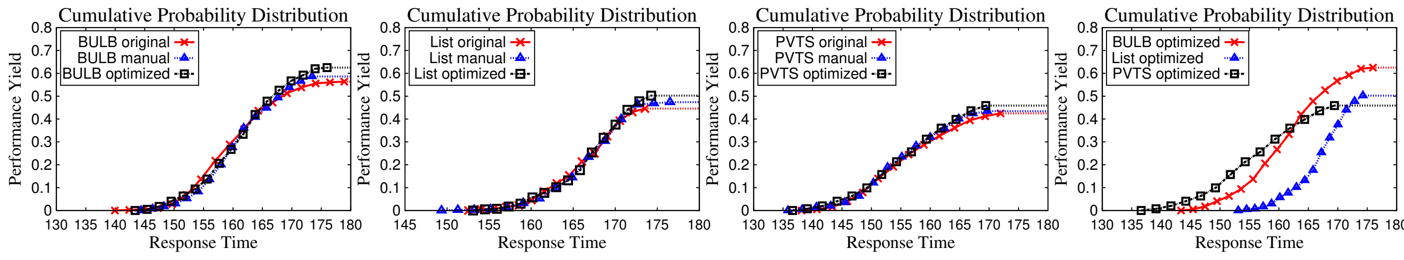
Fig. 9. Two MPSoC floorplans

Similar to the work presented in [11], [27], we assume that the task execution time and power consumption follow the Gaussian distribution as presented in Table 1. For example, the power of type A PE follows the Gaussian distribution $N(x, (0.10x)^2)$, where x indicates the real time power (leakage power + dynamic power) of the PEs of type A. For example, if the current power of the PE is 10, its real power value at the same time will follow the Gaussian distribution $N(10, (1)^2)$. The execution time of PEs of type A follows the Gaussian distribution $N(y, (0.05y)^2)$, where y denotes the expected execution time of a task that can be executed on the PEs of type A. For instance, if the mean execution time of a task allocated to some PE of type A is 15, its execution time will follow the Gaussian distribution $N(15, 0.75^2)$. It is important to note that, since our framework offers a large set of distribution models and comprehensive programming constructs, it allows accurate modeling of variations and correlations of PEs.

5.2 Time-Oriented Evaluation and Optimization

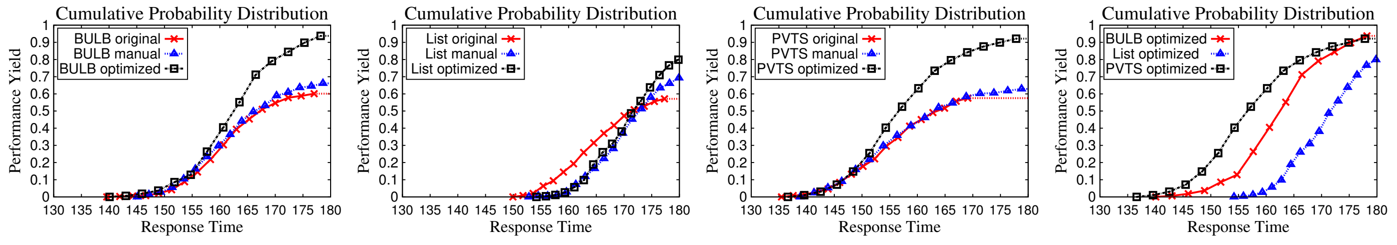
Assume that we want to achieve a TAS solution that can meet the design constraints such that "*with an energy quota of 12000 Joules, the task graph can be finished within 180 seconds while the max power cannot exceed 82 Watts and the overheating time cannot be longer than 60 seconds*". Based on the given power and functional unit constraints, we generated three TAS solutions based on the following three strategies, respectively: i) the BULB [36] which is a power-constrained time minimization method for task allocation and scheduling, ii) list scheduling method (denoted by *List*) [37] which is a promising heuristic to quickly find a near-optimal TAS solution, and iii) process variation-aware task scheduling approach (denoted by *PVTS*) proposed in [11] which can effectively find near-optimal TAS solutions with highest performance yield. Note that neither *BULB* nor *List* takes the performance

2. In this example, we used the MATLAB BPNN tool which has only one hidden layer by default. Based on the Kolmogorov's theorem [30], we found that 50 neurons can effectively generate an accurate regression model with under the constraint of Mean Square Error (MSE). We set the maximum number of epoch iterations to 50 in this example, and our approach can find the regression model within 50 epoch iterations. Note that if within 50 epoch iterations we cannot obtain a regression model satisfying the given constraints, we need to increase the maximum number of iterations or tune the other parameters (e.g., increasing the number of hidden neurons) until a satisfying regression model is found.



(a) BULB (b) List Scheduling (c) PVTS [11] (d) Optimized

Fig. 10. Time oriented evaluation results for (5,2,5)



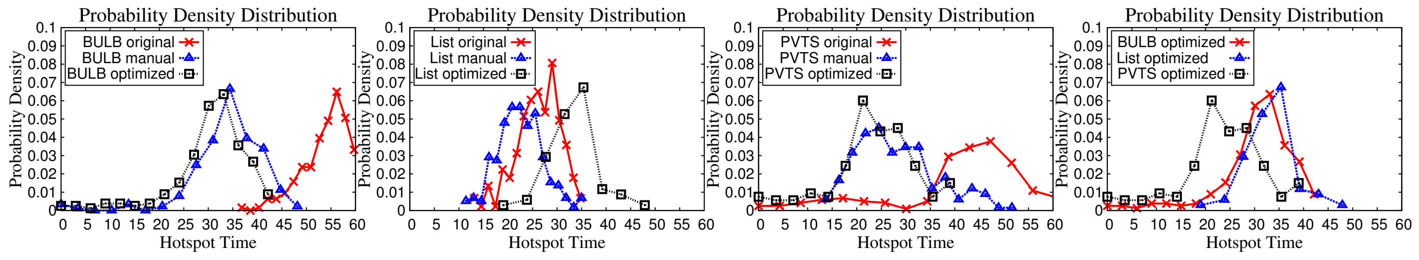
(a) BULB (b) List Scheduling (c) PVTS [11] (d) Optimized

Fig. 11. Time oriented evaluation results for (5,5,2)

variations into account during the search of optimal TAS solutions. Unlike *BULB* and *List*, *PVTS* adopts a list scheduling like approach based on dynamic priorities calculated from unscheduled tasks with uncertain delays. For each obtained TAS solution with a given floorplan, we generated 14400 mapped TAS solutions. By using our developed tools, all these mapped TAS solutions can be automatically converted to NPTA models, and the design constraints can be automatically transformed to the properties. To enable the TAS optimization, we sampled 100 mapped TAS solutions in total (50 by using the first sampling strategy and 50 by using the second sampling strategy) to construct the regression set. Since the whole process including the mapped TAS solution generation, NPTA model/property generation and TAS solution sampling costs less than 3 seconds, compared with the TAS evaluation time, such time can be neglected.

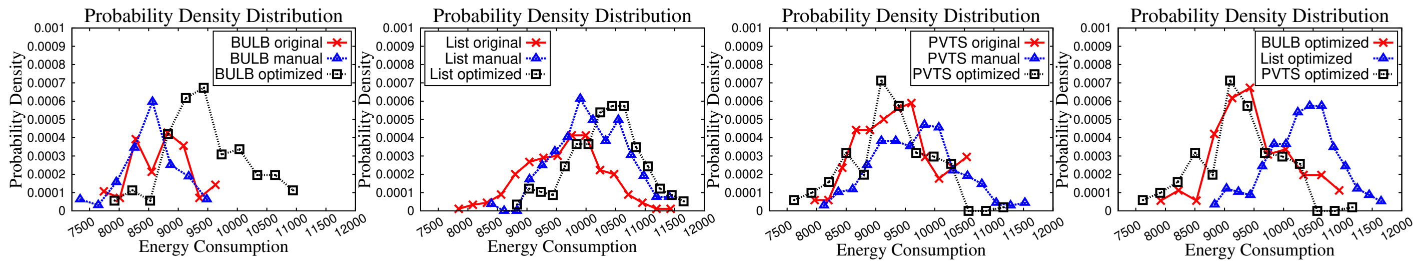
Assuming that we want to check the performance yield from the perspective of completion time, we can use the property *Pr [<=180](<> T(0).End && energy<=12000 && max_power<=82 && hotspot_time<=60)*. Based on the two given floorplans, Figure 10 and Figure 11 show the evaluation results of the different mapped TAS solutions generated by the approaches BULB, *List* and *PVTS*. Here, each figure has four sub-figures. While the first three sub-figures compare the performance of different mapped TAS solutions for the same strategy, the last sub-figure compares the best performance of the mapped TAS solutions for different strategies. In these figures, *original* denotes the TAS solution with the first generated PE mapping function, while *manual* indicates that the PE mapping is provided manually by some experienced MPSoC designer. The *optimized* PE mapping is produced by using our TAS optimization approach. Note that an optimized PE mapping does not mean the real "global optimal" mapping which can achieve the highest performance yield. This is because our approach relies on the performance yield prediction using regression analysis rather than evaluating all the TAS solutions. **It can be observed that our approach can visually reflect the relations between performance yield and response time for the given query.**

From Figure 10 we can observe that the performance yield of all the BULB-based TAS solutions are better than the performance yield of all the list scheduling-based TAS solutions. Furthermore, from both figures we can find that in the first three sub-figures the optimized mapped TAS solutions generated by our approach achieve the best performance yield compared with its counterparts produced by the same TAS strategy. Especially, in Figure 11(a) , by using our optimization approach we can obtain a significant improvement of the solution *BULB optimized* (with a performance yield of 0.91) over the solution *BULB original* (with a performance yield of 0.60). The similar trend can also be observed in Figure 11(c). From Figure 10(d) and Figure 11(d) we can find that the mapped TAS solutions generated by both the BULB strategy and our optimization approach can achieve the highest performance yield. This is because that the TAS solutions generated by BULB have the smallest response time under specified design constraints. Unlike *List* and *PVTS*, the SMC simulation failures against the query in this case is mainly due to the violation of the expression part of the query (i.e., *<> T(0).End && energy<=12000 && max_power<=82 && hotspot_time<=60*) rather than the time bound (i.e., *[<=180]*). **By using our optimization approach, the tasks are more likely to be allocated to colder PEs with few neighbour tasks, which drastically reduces the chance of performance query violations. Consequently, the best performance yield can be achieved.** Interestingly, in Figure 11(d) the mapped TAS instances *BULB optimized* and *PVTS optimized* achieve similar performance yield at time 1. However, designers may prefer *PVTS optimized*, since its average response time is smaller.



(a) BULB (b) List Scheduling (c) PVTS [11] (d) Optimized

Fig. 12. Hotspot time oriented evaluation results for (5,5,2)



(a) BULB (b) List Scheduling (c) PVTS [11] (d) Optimized

Fig. 13. Energy oriented evaluation results for (5,5,2)

To justify the efficacy of our proposed approach, we conducted the full search for the "real" optimal solutions. Due to the limited time and computer resources, we only investigated the case of BULB strategy with the floorplan *(5,5,2)* (i.e., the case shown in Figure 11(a)) by checking all the 14400 mapped TAS solutions. It is important to note that the average evaluation time of a single mapped TAS solution against the above property needs around one hour. To accelerate the optimization process, we used four machines with 4 cores each for the parallel evaluation. We spent around one month to achieve the "real" optimal mapped TAS solution. In this case, we found that the "real" optimal solution is the same as the solution obtained using our approach, which only required 7 hours with the same setting.

TABLE 2
Comparison of Different Regression Methods

	SMC Time	Regression Method	Reg. Time	Pred. Time	Pred. Value	Real Value
BULB	8859	ε-SVR	< 1	< 1	0.980	0.928
		BPNN	< 1	< 1	0.933	0.937
List	15433	ε-SVR	< 1	< 1	0.766	0.770
		BPNN	< 1	< 1	0.802	0.799
PVTS	7417	ε-SVR	< 1	< 1	0.992	0.901
		BPNN	< 1	< 1	0.947	0.921

To show the effectiveness of our regression analysis based approach using BPNN, we performed the optimization for the three TAS strategies with floorplan *(5,5,2)* using the well established regression tool ε-SVR from the SVM library LIBSVM [34]. For the regression model generation in ε-SVR, we adopted the Radial Basis Function (RBF) kernel to enable the non-linear regression for the training set, and employed a 10-fold cross validation. To enable the prediction of performance yield, we scaled the output of the regression which can generate an output value between 0 and 1. Table 2 shows the comparison results of the two regression methods. In this table, column 1 denotes the name of TAS strategies. Column 2 gives the SMC time (in minutes) for checking the 100 mapped TAS solutions in the regression set. While column 3 indicates the two regression methods, columns 4 and 5 present the time (in seconds) for regression model generation and performance yield prediction, respectively. The last two columns shows the predicted value and real value for the best mapped TAS solution obtained using Algorithm 1. From this table, we can find that both regression methods can quickly ($<$ 1 second) find an optimized mapped TAS solution. Especially, in this experiment we can find that the regression models generated by BPNN outperforms the ones generated by ε-SVR, since they can find the mapped TAS solutions with the highest real performance yield in all the three cases. Furthermore, we can observe that BPNN can achieve a smaller difference between predicted value and real value of performance variations. In other words, the prediction based on BPNN regression is more accurate.

5.3 Hotspot time and Energy Oriented Evaluation

Since the MPSoC platform with layout *(5,5,2)* can achieve the best performance yield under the specified design constraints, based on this architecture we conducted the further hotspot time and energy-oriented analysis. Hotspot time indicates the overheating time of a given MPSoC platform, which strongly affects the reliability and cooling cost of the designed chips. Therefore, it is required that the hotspot time should be as short as possible. To further analyze the overheating time of the achieved mapped TAS solutions as shown in Figure 11, we re-evaluated the same mapped TAS solutions against the property *Pr [hotspot_time<=60](<> T(0).End && energy<=12000 && max_power<=85)*. Figure 12 shows the evaluation results. It is important to note that Figure 11 shows the cumulative probability distribution, while Figure 12 shows the distribution of the probability density. From Figure 12, we can find that the *optimized* mapped TAS solutions may not achieve the least overheating time. For example, in Figure 12(b), the *List optimized* has the largest hotspot time. This is because that the optimized mapped TAS solutions derived in Figure 11 focus on the optimization of the response time. It only requires the hotspot time constraint to be satisfied for the mapped TAS solutions. From the four sub-figures in Figure 12, we can observe that the solution *PVTS optimized* has the lowest chance to be overheated, though its performance yield is slightly lower than *BULB optimized* (as shown in Figure 11(d)). Compared with the solution *BULB optimized* which has the highest chance to have a hotspot time of 35 seconds, the solution *PVTS optimized* has a lower chance (i.e., 22 seconds at its peak) to be overheated under variations. In this case, designers may prefer the mapped TAS solution *PVTS optimized*, since it has a higher reliability and needs less cooling cost.

Similarly, we performed the energy oriented evaluation for the design *MPSoC(5,5,2)* using the property *Pr [energy<=12000] (<> T(0).End && hotspot_time<=60 && max_power<=85)*. Figure 13 shows the analysis results. Note that Figure 13(d) does not reflect the optimized solutions for the purpose of saving energy. Interestingly, in Figure 13(a) we can find that although the mapped TAS solution *BULB optimized* can achieve the highest performance yield, its MPSoC chips which meet the design constraint have a higher chance to consume more energy than the mapped TAS solutions *BULB List* and *BULB manual*. If designers care for the energy consumption much more than the performance yield, they would like to choose the mapped TAS solution *BULB manual*.

6 Conclusion and Future Work

To fully exploit the capabilities of heterogeneous hardware and software resources, MPSoC chips have been increasingly embedded in cyber-physical systems. While more and more PEs are integrated on a die to achieve better performance, drastically increasing energy consumption and performance variations are becoming two key challenges in MPSoC design, which make cyber-physical systems unsustainable and unreliable. To address these challenges, this paper proposes an efficient energy- and thermal-aware evaluation and optimization approach for MPSoC TAS. Based on statistical model checking, our framework enables automated evaluation of various complex queries to quantitatively reason the performance of TAS solutions. To reduce the overall optimization efforts, our framework employs the regression analysis to quickly explore TAS solutions with near optimal performance yield considering both temperature and energy constraints. Comprehensive experimental results demonstrate the effectiveness of our approach.

In our approach, the execution of UPPAAL-SMC is based on the underlying simulation engine using statistical methods. From our experiments, we can find that for complex MPSoC TAS problems the evaluation of a single mapped TAS instance under variations requires a long simulation time. To address this problem, in future we plan to incorporate the state space reduction techniques (e.g., model slicing and model abstraction) in our approach, which may improve the overall evaluation and optimization time of our approach.

References

[1] E. A. Lee and S. A. Seshia, “Introduction to Embedded Systems - A Cyber-Physical Systems Approach,” LeeSeshia.org, 2011.

[2] L. Jozwiak, “Heterogeneous MPSoC Technology for Modern Cyber-physical Systems,” *Journal of Microelectronics, Electronic Components and Materials*, vol. 44, no. 4, pp. 264–279, 2014.

[3] J. H. Choi, A. Bansal, M. Meterelliyoz, J. Murthy and K. Roy, “Leakage power dependent temperature estimation to predict thermal runaway in FinFET circuits,” in *Proceedings of International Conference On Computer Aided Design (ICCAD)*, 2006, pp. 583–586.

[4] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. Rosing, M. Srivastava, S. Swanson and D. Sylvester, “Underdesigned and opportunistic computing in presence of hardware variability,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 32(1):8-23, 2013.

[5] A. Agarwal, D. Blaauw and V. Zolotov, “Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations,” in *Proceedings of International Conference on Computer Aided Design (ICCAD)*, 2003, pp. 900–907.

[6] S. Y. Borkar, “Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation,” *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[7] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, “VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, no. 1, pp. 3–13, 2008.

[8] E. A. Lee, “Cyber Physical Systems: Design Challenges,” in *Proceedings of International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.

[9] L. Huang, F. Yuan and Q. Xu, “On Task Allocation and Scheduling for Lifetime Extension of Platform-Based MPSoC Designs,” *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 22, no. 12, pp. 2088–2099, 2011.

[10] L. Singhal, S. Oh and E. Bozorgzadeh, “Yield maximization for system-level task assignment and configuration selection of configurable multiprocessors,” in *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2008, pp. 249-254.

[11] F. Wang, Y. Chen, C. Nicopoulos, X. Wu, Y. Xie and N. Vijaykrishnan. “Variation-aware task and communication mapping for MPSoC architecture,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, no. 2, pp. 295–307, 2011.

[12] L. Huang and Q. Xu, “Performance yield-driven task allocation and scheduling for MPSoCs under process variation,” in *Proceedings of Design Automation Conference (DAC)*, 2010, pp. 326–331.

[13] K. G. Larsen, “Priced timed automata and statistical model checking,” in *Proceedings of International Conference on Integrated Formal Methods (IFM)*, 2013, pp. 154-161.

[14] M. Chen, S. Huang, X. Fu, X. Liu and J. He. “Statistical model checking-based evaluation and optimization for cloud workflow resource allocation,” *IEEE Transactions on Cloud Computing*, 2017, accepted.

[15] T. Hastie, R. Tibshirani and J. Friedman, “The elements of statistical learning: data mining, interference, and prediction,” *Springer*, 2009.

[16] A. David, K. Larsen, A. Legay, M. Mikucionis and D. Poulsen, “Uppaal SMC tutorial,” *International Journal on Software Tools for Technology Transfers (STTT)*, vol. 17, no. 4, pp. 397–415, 2015.

[17] P. Marwedel, J. Teich, G. Kouveli, I. Bacivarov, L. Thiele, S. Ha, C. Lee, Q. Xu and L. Huang, “Mapping of applications to MPSoCs,” in *Proceedings of International Conference on Harware/Software Codesign and System Synthesis (CODES+ISSS)*, 2011, pp. 109–118.

[18] A. Coskun, T. Rosing, K. Whisnant and K. Gross, “Temperature-aware MPSoC scheduling for reducing hot spots and gradients,” in *Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC)*, 2008, pp. 49–54.

[19] L. Huang and Q. Xu, “Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint,” in *Proceedings of Design, Automation and Test in Europe (DATE)*, 2010, pp. 1584–1589.

[20] T. Chantem, X. S. Hu and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 19, no. 10, pp. 1884–1897, 2006.

[21] S. Hervert and D. Marculescu, "Characterizing chip-multiprocessor variability-tolerance," in *Proceedings of Design Automation Conference (DAC)*, 2008, pp. 313–318.

[22] H. Chon and T. Kim, "Resource sharing problem of timing variation-aware task scheduling and binding in MPSoC," *The Computer Journal*, vol. 53, no. 7, pp. 883–894, 2010.

[23] A. K. Coskun, T. S. Rosing and K. C. Gross, "Temperature management in multiprocessor SoCs using online learning," in *Proceedings of Design Automation Conference (DAC)*, 2008, pp. 890–893.

[24] Y. Tan, W. Liu and Q. Qiu, "Adaptive power management using reinforcement learning," in *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, 2009, pp. 461–467.

[25] O. Almer, M. Gould, B. Franke and N. Topham, "Selecting the optimal system: automated design of application-specific systems-on-chip," in *Proceedings of International Workshop on Network on Chip Architectures (NoCArc)*, 2011, pp. 43–50.

[26] F. Gu, X. Zhang, M. Chen, D. Große and R. Drechsler, "Quantitative timing analysis of UML activity diagrams using statistical model checking," in *Proceedings of Design, Automation and Test in Europe (DATE)*, 2016, pp. 780-785.

[27] M. Chen, D. Yue, X. Qin, X. Fu and P. Mishra, "Variation-aware evaluation of MPSoC task allocation and scheduling strategies using statistical model checking," in *Proceedings of Design, Automation and Test in Europe (DATE)*, 2015, pp. 199-204.

[28] A. David, D. Du, K. Larsen, A. Legay and M. Mikucionis, "Optimizing control strategy using statistical model checking," in *Proceedings of NASA Formal Methods*, 2013, pp. 352–367.

[29] A. David, K. Larsen, A. Legay, M. Mikucionis, D. Poulsen, J. Vliet and Z. Wang, "Statistical Model Checking for Networks of Priced Timed Automata," in *Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2011, pp. 80–96.

[30] V. Kurkova, "Kolmogorov's theorem and multilayer neural networks," *Neural Networks*, vol. 5, no. 3, pp. 501–506, 1992.

[31] Y. Liu and H. Yang, "Temperature-aware leakage estimation using piecewise linear power models," *IEICE Trans. on Electronics*, vol. 93, no. 12, pp. 1679–1691, 2010.

[32] I. Ukhov, M. Bao, P. Eles and Z. Peng, "Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems," in *Proceedings of Design Automation Conference (DAC)*, 2012, pp. 197-204.

[33] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[34] LIBSVM. *http://www.csie.ntu.edu.tw/∼cjlin/libsvm*.

[35] M. Fan,V. Chaturvedi, S. Sha and G. Quan, "An analytical solution for multi-core energy calculation with consideration of leakage and temperature dependency," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, 2013, pp. 353-358.

[36] M. Narasimhan and J. Ramanujam, "A fast approach to computing exact solutions to the resource-constrained scheduling problem," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 6, no. 4, pp. 490–500, 2001.

[37] D. Gajski, N. Dutt, A. Wu and S. Lin, "High-level synthesis: introduction to chip and system design," *Kluwer Academic*, 1992.

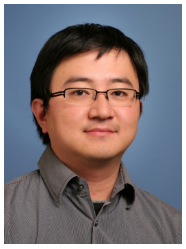
[38] R. P. Dick, D. L. Rhodes and W. Wolf, "TGFF: Task graphs for free," in *Proceedings of International Workshop on Hardware/Software Codesign (CODES)*, pp. 97–101, 1998.

Mingsong Chen (S'08–M'11) received the B.S. and M.E. degrees from Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006 respectively, and the Ph.D. degree in Computer Engineering from the University of Florida, Gainesville, in 2010. He is currently a Professor with the Computer Science and Software Engineering Institute at East China Normal University. His research interests are in the area of design automation of cyber-physical systems, parallel and distributed systems, formal verification techniques and cloud computing. He is an Associate Editor of IET Computers & Digital Techniques, and Journal of Circuits, Systems and Computers.

Xinqian Zhang received the B.E. degree from the Software Engineering Institute of East China Normal University in 2015. He is currently a Ph.D. student in the Department of Embedded Software and System, East China Normal University, Shanghai, China. His research interests are in the area of computer architecture, design automation of embedded systems, and software engineering.

Haifeng Gu Haifeng Gu received the B.E. degree from the Department of Computer Science and Technology, Sichuan Normal University, Chengdu, China, in 2013. He is currently a Ph.D. student in the Department of Embedded Software and System, East China Normal University, Shanghai, China. His research interests are in the area of Hardware/Software co-validation, symbolic execution, statistical model checking, and software testing.

Tongquan Wei (S'06-M'11) received his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University. His research interests are in the areas of green and reliable embedded computing, cyber-physical systems, parallel and distributed systems, and cloud computing. He serves as a Regional Editor for Journal of Circuits, Systems, and Computers since 2012. He also served as Guest Editors for several special sections of IEEE TII and ACM TECS.



Qi Zhu (M'12) is an Assistant Professor of Electrical and Computer Engineering at the University of California, Riverside (UCR). He received his B.E. degree in Computer Science from the Tsinghua University, China in 2003, and his Ph.D. degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 2008. Prior to joining UCR, He was a research scientist at the Strategic CAD Labs in Intel from 2008 to 2011. His research interests include model-based design and software synthesis for cyber-physical systems, CPS security, energy-efficient buildings and infrastructures, and system-on-chip design. He is a recipient of the 2016 CAREER award from the National Science Foundation, and best paper awards of ACM Transactions on Design Automation of Electronic Systems 2016, International Conference on Cyber-Physical Systems 2013, Design Automation Conference 2007 and 2006.