Towards Overhead-Free Interface Theory for Compositional Hierarchical Real-Time Systems

Jin Hyun Kim, Kyong Hoon Kim, Arvind Easwaran\textsuperscript{\textcopyright}, Member, IEEE, and Insup Lee\textsuperscript{\textcopyright}, Fellow, IEEE

Abstract—A significant amount of research has been conducted in the past on compositional real-time scheduling as it has become a useful foundational theory for real-time operating systems and hypervisors. However, compositional frameworks suffer from abstraction overhead in composing components. In this paper, we decompose the abstraction overhead into: 1) supply abstraction overhead associated with the supply from a resource provider and 2) demand abstraction overhead associated with the component workload. Then, we provide sufficient conditions for each abstraction overhead to be eliminated. In addition, this paper provides a heuristic technique that transforms a component to satisfy the sufficient conditions so that the abstraction overhead can be minimized. In experiments, we show that our technique outperforms two prior overhead-reducing techniques. The reduction in overhead is about 10% on average when compared to a technique that uses a single global period and about 8% on average when compared to a technique based on harmonicity.

Index Terms—Compositional framework (CF), harmonic tasks, hierarchical real-time systems, overhead reduction.

I. INTRODUCTION

A HIERARCHICAL scheduling system (HSS) is a component-based scheduling system wherein a component distributes the resource it receives from the parent component to its subcomponents. Various applications of such a system can be found in avionics, automotive, virtual machines (e.g., Xen hypervisor and VMWare), etc. [1]–[4].

In the last decade, various analysis frameworks have emerged for verifying and validating such HSS (e.g., [5]–[10]).

Model-checking-based schedulability analysis techniques were presented in [5] and [7], but they are prone to the well-known state-exploration problem. A simulation-based schedulability validation tool, called SymTA/S, was presented in [8] and [9] presents model-based Hypervisor analysis. A schedulability analysis paradigm based on network calculus, called real-time calculus, was proposed in [6]. A similar approach based on arrival rate functions was proposed in [10]. However, none of the above frameworks abstract component demand into representative real-time tasks (i.e., interface tasks), and therefore incur high analysis complexity. For the same reason, these frameworks are not suitable for compositional analysis of multilevel HSS developed under the open systems paradigm, because higher-level components cannot use scheduling models based on real-time tasks to allocate resource to subcomponents.

Compositional framework (CF) overcomes the above challenges, and presents an efficient component-based analysis paradigm. This framework is well suited for open systems because components can be independently developed and analyzed under it. Studies in CF have been developed along several directions: new resource models for characterizing resource supply on single and multicore [11]–[16], optimization for improving system efficiency [3], [17]–[20], shared-resource communication [21], cache-aware scheduling and implementation overheads, etc. A resource model of CF is an abstraction of supply from a component or hardware platform and an interface of CF is an abstraction of demand of a component. They are prone to abstraction overhead, i.e., the resource is under-utilized due to these abstractions. In the following motivating problem, we give an example of this under-utilization.

A. Motivating Problem

Let us consider an OS-Hypervisor operating two virtual machines using the scheduling algorithm rate monotonic (RM), as shown in Fig. 1(a). The virtual machines schedule tasks using RM and earliest deadline first (EDF), respectively. Here, the system can be modeled by CF using a periodic resource model (PRM) [18], which provides resource periodically to subcomponents. Since the framework considers the worst-case for resource supply, PRM computes sufficient resource budgets of $C_1$ and $C_2$ as 3 units for 4 time units ($l_1 = (4, 3)$) and 5 units for 8 time units ($l_2 = (8, 5)$), respectively. In this case, the resulting CF does not provide a feasible solution. However, it is easy to see that the system...
is schedulable due to the harmonic property of resource and task periods, as shown in Fig. 1(c).

Note that the periods of \( C_1 \) and \( C_2 \)'s workloads and those of their interfaces are harmonically related to each other. Based on this observation, one could develop a more efficient composition framework. For example, Kramer et al. [22] observed that around 1000–1500 runnables of automotive applications can be divided into ten groups, each of which uses a common period and these group periods are mostly harmonic with each other. It turns out that tasks in a group using the same period benefit from the harmonicity. The above observation motivates this paper such that the harmonicity property is explicitly introduced in CF, even when a component workload is not inherently harmonic, for improving resource utilization of HSS.

To improve resource utilization of HSS, Easwaran et al. [3], [13] introduced a new resource model, called explicit deadline period (EDP) resource model, by extending the PRM of Shin and Lee [12] and Shin et al. [15] with a deadline. Although the EDP model reduces supply abstraction overhead (SAO), it does not eliminate it completely, and further the proposed framework does not consider demand abstraction overhead (DAO). Chen et al. [20] proposed a technique to use the same global period for every interface and thus eliminated abstraction overhead from intermediate (nonleaf) components. However, they did not take into account the increase in runtime overhead such as context switches, which arises due to the use of a common period that must be smaller than or equal to the smallest period of component tasks [23]. Recently, Guo et al. [24] presented how to apply harmonicity in independent scheduling components to remove interface abstraction overhead, wherein component tasks adjust their periods to be harmonic with the component interface. However, they have not shown how to apply harmonicity hierarchically in dependent components of HSS, i.e., components in a parent-child relationship.

We have observed that the regularity in execution pattern of harmonic interface tasks makes the resulting resource supply more efficient so that its worst-case starvation interval (WCSI) is halved when compared to an equivalent generic PRM (based on nonharmonic interface tasks) [12]. Based on this key observation, this paper presents a novel CF, where a demand abstract is developed to represent a component's demand completely independent from any resource model. From an analysis of the root cause of abstraction overheads in CF, we present sufficient conditions to remove these overheads. Using these sufficient conditions, we then present a CF, called HSS-Sync (HSSS), wherein no abstraction overhead is incurred. In addition, we present how to utilize this CF for HSS with nonharmonic tasks in components. To our best knowledge, this paper is the first to utilize harmonicity property for the reduction of abstraction overhead in CF. The contributions of this paper can be summarized as follows.

1) We analyze the root causes of abstraction overheads in CF and identify sufficient conditions to eliminate both demand and SAOs.
2) We develop an overhead-free CF based on the harmonicity property.
3) We develop algorithms to approximate nonharmonic tasks using harmonic tasks and to generate harmonic interfaces for components. These algorithms enable the application of above CF for components with nonharmonic tasks in their workload.

B. Organization

Section II presents the underlying model and problems that we address in this paper. In Section III, we present sufficient conditions to remove demand and SAOs. Section IV provides a new CF, called HSSS. We also show how to apply HSSS to components comprising nonharmonic tasks such that the abstraction overheads are minimized. In Section VI, we compare our framework with various existing abstraction overhead reduction techniques in CF. Section VII discusses the lessons from this paper and concludes this paper.

II. BACKGROUND AND PROBLEM FORMULATION

This section presents the setting and definitions of the underlying system, and the problem that we address in this paper.

A. Background

An HSS consists of components in a tree structure. A component \( C = (W, A) \) is composed of a workload \( W \), a set of tasks, and an algorithm \( A \). A parent component encloses child components, i.e., subcomponents. The HSS defines supply demand (parent–child) relations between a task \( \tau_i \) of \( W \) and a subcomponent \( C \). A parent component delegates a task to run a subcomponent, meaning that the subcomponent’s execution depends on the task. We use the implicit periodic task model \( \tau(p, e) \) for workloads, where \( p \) is the period and \( e \) is the worst-case execution time [25]. A task is called elementary task if it is mapped to no subcomponent. Similarly, a component enclosing only elementary tasks is called an elementary component or leaf component. \( U_W \) and \( U_{\tau_i} \) represent utilization of workload \( W \) and utilization of a task \( \tau_i \), respectively.

A resource model captures a resource supply pattern. For instance, the PRM, \( \Gamma(\Pi, \Theta) \), supplies \( \Theta \) units of resource every \( \Pi \) time units [12]. The resource model is used as an underlying model for the resource supply task as well as the
component interface. The resource supply task of a parent component provides resource to a child component. The component interface represents a collective resource requirement of the component’s workload. In this paper, we use a periodic interface \( I(\mathbb{P}, \mathbb{B}) \), which denotes that the interface \( I \) requires \( \mathbb{B} \) resource units every \( \mathbb{P} \) time units.

We use the harmonicity property of real-time tasks \([26]\) to develop the new CF. For this purpose, we define the following two functions.

**Definition 1 (Harmonic Tasks):** For tasks \( \tau_i \) and \( \tau_j \)

\[
\mathcal{H}(\tau_i, \tau_j) = \begin{cases} 
1 & \text{if } p_i \mod p_j = 0 \lor p_j \mod p_i = 0 \\
0 & \text{Otherwise}.
\end{cases}
\]

Two tasks \( \tau_i \) and \( \tau_j \) are said to be harmonic if and only if \( \mathcal{H}(\tau_i, \tau_j) = 1 \). For notational convenience, we overload \( \mathcal{H} \) so that it can also be used with resource model \( \Gamma(\Pi, \Theta) \). That is, \( \mathcal{H}(\Gamma, \tau) = 1 \) if \( \Pi \) and \( p \) are harmonic.

**Definition 2 (Harmonic Workload):** A workload \( W \) is harmonic if every pair of tasks in \( W \) is harmonic, i.e.,

\[
\forall \tau_i, \tau_j \in W, \quad \mathcal{H}(\tau_i, \tau_j) = 1.
\]

**B. Problem Formulation**

The principle of compositional schedulability analysis is to abstract the details of a component with its interface and then use interfaces to reason about the composition of components. In general, to ensure schedulability analysis, an interface over-approximates the demand of a component. The amount of such over approximation is called abstraction overhead.

The problem addressed in this paper is how to minimize the abstraction overhead.

This paper presents a new periodic abstract demand, based on the periodic task model \([25]\), which is independent from any resource model. We then analyze the overhead of this periodic demand abstraction (PDA) and present techniques for its minimization. In contrast, component interface computation takes into account a specific resource model in existing CFs, so the interface includes overhead depending on resource models \([13], [15]\). As part of the goal of minimizing abstraction overhead, this paper aims to separate demand and supply abstractions and their overheads, so that we can explicitly divide the causes of overheads in CF, address them individually, and make our CF faithful to the open system principle, e.g., independent component analysis.

This paper is based on the following assumptions.

1) The HSS is an open system, where the analysis of a component must be independent of other components.
2) The demand and supply are not synchronized, thus the supply could occur when there is no demand, i.e., a demand task is not scheduled to execute.
3) This paper uses the periodic task model \([25]\) for both demand and supply abstractions.

This paper identifies the root causes of the abstraction overheads and proposes ways to eliminate them. More specifically, we answer the following questions.

1) How can we achieve an ideal supply for a periodic supply model (Section III-A)?
2) What are sufficient conditions to remove the overhead of a PDA (Section III-B)?
3) How can we utilize the ideal periodic resource supply model and the sufficient conditions to remove the DAO for any given component (Section V)?

**III. OVERHEAD OF COMPOSITIONAL FRAMEWORK**

This section discusses two types of abstraction overheads as the demand of a component is abstracted by an interface and the supply is abstracted by a resource model. Each abstraction results in overhead for HSS since it is necessary to do extra resource provisioning to ensure schedulability. We note that there are other causes of overhead, e.g., context switching by interleaved execution of components, cache misses and loading, etc. This paper focuses on the overheads due to demand and supply abstractions, and hence the other overheads are out of scope.

**A. Supply Abstraction Overhead**

Since we assume that the demand and supply are not synchronized, Fig. 2 shows when the worst-case supply delay of PRM \([12]\) happens. In Fig. 2, the demand for time duration \( r \) arrives just after the supply of the previous period of \( \Gamma(\Pi, \Theta) \) has finished. Furthermore, the supply of \( \Gamma(\Pi, \Theta) \) is delayed in the current period for \( \Pi - \Theta \), because it is preempted by other components. This worst-case starvation of PRM is known to be \( 2(\Pi - \Theta) \) \([15]\), which happens because the demand and supply are not synchronized. For the EDP resource model \([13]\), WCSI is \( \Pi + \Delta - 2\Theta \), where \( \Delta \) is the deadline of the periodic supply, because EDP guarantees that the delay in supply is at most \( \Delta \).

Since the WCSI for \( \Gamma(\Pi, \Theta) \) decreases with its period \( \Pi \), one way to reduce WCSI is to use a smaller period, which becomes proportional sharing in the limit. We denote this processor sharing by \( \Omega(\Pi, \Theta) \). Since processor sharing is not realizable in practice, we present an alternative characterization, which can be used to identify a set of tasks where we can achieve minimal WCSI. We next define an ideal PRM (iPRM) as follows.

**Definition 3:** For a given period \( \Pi \) and execution time \( \Theta \), an iPRM, denoted by \( H(\Pi, \Theta) \), is a PRM that guarantees \( \Theta \) resource units in any time interval of duration \( \Pi \).

Note that WCSI of iPRM \( H(\Pi, \Theta) \) is \( \Pi - \Theta \). For given \( (\Pi, \Theta) \), if the demand and supply are not synchronized, then iPRM has a minimal WCSI among all PRMs as the delay of \( \Pi - \Theta \) cannot be avoided in the worst case. The supply bound
function (sbf) of tPRM is defined as follows (from [13]):

$$
sbf_{tPRM}(H, t) = \left\lfloor \frac{t}{\Pi} \right\rfloor \cdot \Theta + \max\left(0, t - (\Pi - \Theta) - \left\lfloor \frac{t}{\Pi} \right\rfloor \cdot \Pi\right).$$

We now define the SAO as follows.

**Definition 4 (SAO):** For a PRM $\Gamma(\Pi, \Theta)$ and a time interval $t$, the SAO of $\Gamma$ is the difference between worst-case supplies of $\Omega(\Pi, \Theta)$ and $\Gamma(\Pi, \Theta)$.

For a PRM $\Gamma$, the SAO is computed by

$$O_{sup}^{sa}(\Gamma, t) = \int_{0}^{t} (sbf_{tPRM}(\Omega, x) - sbf_{tPRM}(\Gamma, x)) \, dx. \quad (1)$$

PRMs may result in different SBFs for the same period and budget parameters, depending on their properties. For instance, the PRM of [12] and tPRM result in different sbfs.

A resource model with less SAO makes more tasks feasible, and hence is desirable. For instance, Fig. 3 shows the supply $\Gamma(\Pi, \Theta)$ and $\Gamma(\Pi, \Theta)$ regularity of the periodic supply. We define a regular PRM as follows.

**Lemma 1:** If a PRM $\Gamma(\Pi, \Theta)$ is regular, it is an tPRM.

**Proof:** Let supply$(\Gamma, t_{0}, t_{1})$ compute the amount of resource supplied by $\Gamma$ during $[t_{0}, t_{1} + t]$, i.e., an interval $t$ starting at $t_{0}$. For $\Gamma(\Pi, \Theta)$ and $0 < t_{1} \leq \Pi$ and $n \in \mathbb{N}_{>0}$, supply$(\Gamma, (n-1) \cdot \Pi, t_{1})$ + supply$(\Gamma, (n-1) \cdot \Pi + t_{1}, \Pi - t_{1}) = \Theta$ by definition. For any two consecutive periods of a regular $\Gamma$, the model supplies resource such that supply$(\Gamma, (n-1) \cdot \Pi, t_{1}) + \Theta$ is supply$(\Gamma, n \cdot \Pi, t_{1} + t_{i})$ and supply$(\Gamma, (n-1) \cdot \Pi + t_{i}, \Pi - t_{i})$ is supply$(\Gamma, n \cdot \Pi + t_{i}, \Pi - t_{i})$, where $n + t_{i} = t_{j}$. Then, supply$(\Gamma, (n-1) \cdot \Pi + t_{i}, \Pi - t_{i})$ + supply$(\Gamma, n \cdot \Pi, t_{j}) = \Theta$, since supply$(\Gamma, (n-1) \cdot \Pi, t_{1}) + \Theta$ is supply$(\Gamma, (n-1) \cdot \Pi + t_{i}, \Pi - t_{i})$.

Note that $\Gamma(\Pi, \Theta)$ is a supply model that is not synchronized with the demand.

$\Pi - t_{i} = \Theta$. Hence, a regular $\Gamma$ always provides $\Theta$ resource units for any time interval $\Pi$. Thus, $\Gamma$ is an tPRM.

Now, we present a sufficient condition to enforce a PRM to be regular. Under RM, tasks with harmonic periods are scheduled at the same points in each period. If harmonic tasks are scheduled under EDF and their ties are identically broken, then each task also executes at the same points in each period. Thus, a PRM implemented by a task under the above harmonic conditions is regular if the PRM is serving the top (root) component fully utilizing resources.

For instance, the resource supply tasks $\tau_{1}$ and $\tau_{2}$ in Fig. 4 are harmonic, and hence the corresponding resource models are regular. Then, components $C_{1}$ and $C_{2}$ supplied by tasks $\tau_{1}$ and $\tau_{2}$ can, respectively, utilize as much resource as the individual resource utilizations of $\tau_{1}$ and $\tau_{2}$.

In the following lemma, we provide conditions for a resource supply task $\tau_{i}$ to become regular when it is supplied by a nonroot parent component using regular PRM $\Gamma$. For instance, $\tau_{3}$ in Fig. 4 is regular since $\tau_{1}$ supplying resources for $C_{1}$ enclosing $\tau_{3}$ is also regular.

**Lemma 2:** Consider a regular resource model $\Gamma(\Pi, \Theta)$ supplying resources to workload $W$. For each $\tau_{i} \in W$, $\tau_{i}$ is regular if

$$(\forall \tau_{i} \neq \tau_{j}, \tau_{j} \in W, \mathcal{H}(\tau_{i}, \tau_{j}) = 1 \land \mathcal{H}(\tau_{i}, \Gamma) = 1) \land \cup U_{W} \leq U_{\Gamma}.$$  

**Proof:** Since $W$ is harmonic, the best-case and worst-case response times of each task $\tau_{i}$ is $e_{i} + \sum_{k \in H(\tau_{i})} e_{k} / p_{k}$, where $H(\tau_{i})$ is the set of tasks whose priority is higher than $\tau_{i}$. The resource supply of regular $\Gamma$ is proven to be an tPRM from Lemma 1. So the delay in its resource supply is always the same in each supply period. Suppose that the supply delay of $\Gamma$ is $\epsilon$. The best-case and worst-case response times of each task $\tau_{i}$ supplied by $\Gamma$ must then be equal to $e_{i} + \sum_{k \in H(\tau_{i})} e_{k} / p_{k} + \epsilon$.

Note that Lemma 2 is applicable to resource supply tasks. Fig. 4 shows that all resource supply tasks are harmonic in each component, and hence their supplies satisfy the conditions of tPRM. $C_{1}$ enclosing $\tau_{3}$ and $\tau_{4}$ is supplied by $\tau_{1}$. Then, $\tau_{3}$ and $\tau_{4}$ supply resources to $C_{3}$ and $C_{4}$. Since $\tau_{1}$, $\tau_{3}$, and $\tau_{4}$ are harmonic, the supply tasks $\tau_{3}$ and $\tau_{4}$ satisfy the conditions of tPRM.
B. Demand Abstraction Overhead

We define a PDA \( D(P, E) \), based on the periodic task model [25]. Our goal is to determine when it does not incur supply related overheads like WCSI. It is defined as follows.

**Definition 6 (PDA):** For a workload \( W \) under scheduling algorithm A, \( (P, E) \) is said to be a PDA, denoted by \( D(P, E) \), if, whenever a resource supply can schedule \( D \), it can also schedule \( W \).

PDA is a supply independent interface, while the existing interface models rely on a resource supply model. Hence, it is free from any resource-model-dependent overhead.

Now, we define bandwidth-DAO (bDAO) of a PDA in terms of its resource utilization. For a workload \( W \) and a PDA \( D(P, E) \), the bDAO is defined by \( O^{bDAO}(W, D) = U_W - E/P \). A PDA with no bDAO is said to be zero-bDAO.

Now, we propose how to remove bDAO. In [17], it is shown that a workload under EDF scheduled by a resource supply is schedulable if the component interface period divides the workload periods and the total utilization of the workload is less than or equal to that of the resource supply. We apply this condition to remove bDAO from the PDA abstraction of a workload scheduled under EDF.

**Lemma 3 (From [17]):** For a workload \( W \) scheduled under EDF, a PDA \( D(P, E) \) is zero-bDAO if

\[
\left( U_W = \frac{E}{P} \right) \land (\forall \tau_i \in W, p_i \mod P = 0).
\]

Now, we present a condition to eliminate bDAO from the PDA abstraction of a workload scheduled under RM. It has been shown that a harmonic task set scheduled under RM can make full use of resources [26]. The following lemma uses this to show that if all the tasks in a workload are harmonic with each other and the PDA, and the PDA has the same resource utilization as the workload, then the PDA has no abstraction overhead.

**Lemma 4:** For a harmonic \( W \) scheduled under RM, a PDA \( D(P, E) \) is zero-bDAO if

\[
\left( U_W = \frac{E}{P} \right) \land (\forall \tau_i \in W, \mathcal{H}(\tau_i, P) = 1 \land p_i \geq P).
\]

**Proof:** Since \( W \) is scheduled under RM and harmonic, it requires resources at the rate of \( U_W \). Since \( \forall \tau_i \in W, \mathcal{H}(\tau_i, P) = 1 \land p_i \in \mathbb{N} \), \( \tau_i \) requires the amount \((e_i/p_i)\cdot P\) of resource in every period \( P \). Hence, \( W \) requires the total amount \( \sum_{\tau_i \in W} (e_i/p_i) \cdot P \) of resource in every period \( P \). Let \( E = \sum_{\tau_i \in W} (e_i/p_i) \cdot P \). Then, \( W \) requires \( (E/P) \) amount of resource in every period \( P \). Thus, \( D(P, E) \) sufficiently represents the total resource requirement of \( W \) if \( W \) is harmonic and \( U_W = (E/P) \).

In Fig. 4, since the periods of \( \tau_7 \) and \( \tau_8 \) scheduled under RM are harmonic with values 5, 15, and 45, \( D(5, 1), D(15, 3) \), and \( D(45, 9) \) can be demand abstraction tasks with zero-bDAO.

Similarly, the periods of \( \tau_9 \) and \( \tau_{10} \) scheduled under EDF are divisible by 5, 15, and 30. Thus, \( D(5, 2), D(15, 6) \), and \( D(30, 12) \) are demand abstraction tasks with zero-bDAO for this case.

C. Composition Overhead-Free Components

In this section, we present sufficient conditions that HSS has no abstraction overheads. For given PDA and IPMM, we present conditions that an interface \( I \) is generated such that it satisfies PDA and IPMM satisfies \( I \) without loss of resources. Interface and PDA are both workload demand abstracts and subject to bDAO. However, the interface differs from the PDA in that it depends on a resource model of its parent component. Thus, a (better) component interface can be computed after its parent component’s resource model is known. We present conditions for a workload schedulable by a PDA to be also schedulable by an interface when a IPMM is given. For a given IPMM, \( H(\Pi, \Theta) \), a PDA schedulable by this IPMM satisfies the following conditions.

**Lemma 5:** A PDA \( D(P, E) \) is schedulable by an IPMM \( H(\Pi, \Theta) \) if

\[
\left( E = P \cdot \frac{\Theta}{\Pi} \right) \land (P \mod \Pi = 0).
\]

**Proof:** For \( H(\Pi, \Theta) \), \( sbf_{IPMM}(H, t) \) is such that

\[
sbf_{IPMM}(H, k\Pi) = k\Theta
\]

where \( k \in \mathbb{N}_{>0} \) for the following reason. Since \( t = k\Pi \) and \( k \in \mathbb{N}_{>0} \),

\[
sbf_{IPMM}(H, (t\mod \Pi)) = k\Theta + \max(0, k\Pi - (\Pi - \Theta) - k\Pi)
\]

\[
= k \cdot \Theta + \max(0, -(\Pi - \Theta))
\]

\[
= k \cdot \Theta.
\]

Now, we prove the lemma as follows. \( k\Pi = P \) since \( P \mod \Pi = 0 \). \( (E/P) = (k\Theta/k\Pi) \) since \( (E/P) = (\Theta/\Pi) \) and \( k\Pi = P \). \( sbf_{IPMM}(H, k\Pi) = k\Theta \) since \( \forall t, O^{bDAO}(\Gamma, t) = 0 \). Since \( (E/P) = (k\Theta/k\Pi) \) holds as shown in Fig. 5. D is schedulable by H.

To generate an interface \( I \) with zero-bDAO, we have the following theorem and corollary.

**Theorem 1:** For an IPMM \( H(\Pi, \Theta) \) and zero-bDAO \( D(P, E) \), if

\[
\left( E = P \cdot \frac{\Theta}{\Pi} \right) \land (P \mod \Pi = 0) \land (P \mod \Pi = 0)
\]

then the interface \( I(P, P) \) is schedulable and zero-bDAO.

**Proof:** In Lemma 5, we showed that D is schedulable by \( I \) such that \( ([E/P] = [\Theta/\Pi]) \land (P \mod \Pi = 0) \). Similarly,
Lemma presents overhead-free composition of a component. If the parameters of the composition have no abstraction overheads. First, we define a harmonic component as follows.

Definition 7: A component with a PDA $D(P, E)$ is harmonic if the task periods in its workload and the PDA's period are harmonic.

As indicated in Theorem 1, a PDA or an interface $I$ is zero-bDAO if $E = P \cdot U_W$ or $\Theta = P \cdot U_W$. Let us assume that an IPRM $H(\Pi, \Theta)$ provides resources to a harmonic component. If the parameters of $H$ are identical to $D$ of the component, we can say the component abstraction incurs no abstraction overhead or is overhead-free. Thus, the following lemma presents overhead-free composition of a component under RM scheduling algorithm.

Lemma 6: For a given component $C$ with RM scheduling algorithm, if the resource model $H(P, E)$ is an IPRM and $D(P, E)$ is harmonic with the workload, then the interface $I(P, E)$ has no abstraction overhead.

Proof: Since a PDA $D(P, E)$ is harmonic with the workload, the PDA is zero-bDAO. If the same parameter $(P, E)$ is used for $H$, $D$, and $I$, then they satisfy the three conditions of Theorem 1.

The interface $I$ derived from the zero-bDAO PDA is called harmonic interface.

B. Divisible Composition of Component Under EDF

A harmonic component scheduled under EDF can achieve overhead-free composition using Lemma 6. However, in the case of EDF, we can use the divisible constraint rather than the stricter harmonic constraint for abstraction overhead elimination.

IV. HIERARCHICAL SCHEDULING SYSTEMS WITH SYNC

In this section, we show how an HSS free from abstraction overheads and component scheduling overheads is built, depending on scheduling algorithms RM and EDF.

A. Harmonic Composition of Component Under RM

We now discuss the abstraction of a component scheduled under RM such that the composition has no abstraction overhead. First, we define a harmonic component as follows.

Definition 8: A component using PDA $D(P, E)$ is divisible if all task periods are individually divisible by $P$.

As indicated in Theorem 1, a divisible interface produces no DAO if $E = P \cdot U_W$ under EDF. Thus, the following lemma presents overhead-free composition of a component under EDF.

Lemma 7: For a given component $C$ scheduled under EDF, if the resource model $H(P, E)$ is an IPRM and $D(P, E)$ is divisible with the workload, then the interface $I(P, E)$ has no abstraction overhead.

Proof: Since the PDA $D(P, E)$ is divisible with the workload, the PDA is zero-bDAO. If the same parameter $(P, E)$ is used for $H$, $D$, and $I$, then they satisfy the three conditions of Theorem 1.

The interface $I$ derived from the zero-bDAO and divisible PDA is called divisible interface.

C. Overhead-Free Hierarchical Composition

This section presents an overhead-free CF with harmonic or divisible components. We first show the following conditions.

1) If the interface of a component under RM is harmonic or the interface of a component under EDF is divisible, then the interface is zero-bDAO.

2) If a resource model is a resource supply task of a harmonic component, then the resource model becomes an IPRM with no SAO.

Then, if a CF satisfies the above two conditions, it has no abstraction overhead.

For example, the system in Fig. 4 meets the harmonicity condition for RM components and the divisibility condition for EDF components. Note that the resource model of an EDF component with a divisible interface is not an IPRM. In Fig. 6, the component $C_2$ is a divisible component. The tasks $r_5$ and $r_6$ of $C_2$ may not be IPRMs because they may not be regular under EDF.

Lemma 8: If each component in a hierarchical system is harmonic and the resource supply is the same as the interface, the composition has no abstraction overhead.

Proof: The proof is direct from Theorem 1, since $\Pi = P$ and $\Theta = B$ for resource supply $H(\Pi, \Theta)$ and interface $I(P, B)$.
V. COMPOSITIONAL FRAMEWORK-SYNC FOR NONHARMONIC TASKS

This section discusses how arbitrary periodic tasks can be approximated to an HSSS so as to remove abstraction overheads of intermediate components. In particular, we present: 1) an adaptation (proximation) of a nonharmonic component to a harmonic component and 2) an algorithm that determines an interface period such that it is harmonic with the underlying component’s workload.

A. Proximation of Nonharmonic Workload Tasks

For a nonharmonic task set, we dedicate a periodic resource server [27] to each task. We call the dedicated periodic server proxy task and denote it by \( v(\Pi, \Theta) \). A given task can run only when the associated proxy task runs. In our framework, we force the proxy task in a component to be harmonic with each other and the component interface as well, so that the interface incurs no abstraction overhead and becomes an tIPRM. Lemma 9 presents a computation that creates a proxy task \( v(\Pi, \Theta) \) for a given task.

**Lemma 9 (Budget for Proxy Task):** For a given task \( \tau(p, e) \) and period \( \Pi \), the resource demand \( \Theta \) of the proxy task \( v(\Pi, \Theta) \) is computed by

\[
\Theta = \frac{e + \Pi - \alpha}{k + 1} \quad \text{if} \quad 0 \leq \alpha \leq \Pi - \frac{\alpha}{k} \\
\Theta = \frac{\Pi - \frac{\alpha}{k} < \alpha}{k + 1} \quad \text{if} \quad \Pi - \frac{\alpha}{k} < \alpha \leq \Pi
\]

where \( p = k\Pi + \alpha \).

**Proof:** Note that a proxy task serves a single task. The demand of this single task is represented by \( dbf_{\text{EDF}}(W, t) = \sum_{v_i \in W}\{t/p_i\}e_i \) [18]. Given a task \( \tau(p, e) \) and an tIPRM \( H = (\Pi, \Theta) \) such that \( \exists r \leq p, dbf_{\text{EDF}}(\tau), t = sbf_{\text{IPRM}}(H, t), \) let \( p = k\Pi + \alpha \). Then, there are two cases depending on the value of \( \alpha \): 0 \leq \alpha \leq \Pi - \Theta [case of Fig. 7(a)] and \( \Pi - \Theta < \alpha \leq \Pi [case of Fig. 7(b)]. For each case, \( \Theta \) is computed as follows.

- **Case 1 (\( 0 \leq \alpha \leq \Pi - \Theta \)):** Let \( k\Pi \leq p \leq k\Pi + \Theta \) such that \( 0 \leq \alpha \leq \Pi - \Theta \) and \( p = k\Pi + \alpha \). Then, the minimum value of \( k\Theta \) is equal to \( e \) such that \( dbf_{\text{EDF}}(\tau), t = sbf_H(t) \) where \( k\Pi \leq t \leq k\Pi + \Theta \). Thus, \( \Theta = (e/k) \).

- **Case 2 (\( \Pi - \Theta < \alpha \leq \Pi \)):** Let \( (k + 1)\Pi - \Theta < p \leq (k + 1)\Pi \) such that \( \Pi - \Theta < \alpha \leq \Pi \). Then \( k\Theta \leq e \leq (k + 1)\Theta \) such that \( dbf_{\text{EDF}}(\tau), t = sbf_H(t) \) where \( k\Pi - \Theta < t < k\Pi \). Thus, for \( \alpha, e + \Pi - \alpha = (k + 1)\Theta \), then \( \Theta = [(e + \Pi - \alpha)/(k + 1)] \).

**Example 1:** Suppose that a task \( \tau(16, 5) \) is given. If the period \( \Pi \) of \( v(\Pi, \Theta) \) is 5, then the task’s period \( p = 16 = 3 \times 5 + 1 \) where \( \alpha = 1 \). \( \Theta = 5/3 = 1.667 \) such that \( 0 \leq \alpha \leq 1 \leq 5 - \Theta \). With a different period \( \Pi = 6 \), the period \( p = 16 = 2 \times 6 + 4 \) where \( \alpha = 4 \). Thus, \( \Theta = [(5 + 6 - 4)/(2 + 1)] = 7/3 = 2.667 \) such that \( 6 - \Theta < \alpha \leq 6 \).

Fig. 8 shows the mappings from tasks \( (\tau_1, \ldots, \tau_12) \) to the corresponding proxy tasks \( (\nu_7, \ldots, \nu_12) \). We call such a creation of proxy tasks proximation in this paper. Creating a proxy task for a given task may cost an overhead, called proximation overhead, which is defined as \( U_{\nu} - U_{\tau} \). While the abstraction overhead of a PRM-based CF is incurred by every component, the proximation overhead is incurred only by tasks in leaf components. Using numerical analysis and simulation experiments in Section VI, we show that the proximation overheads in our approach are much smaller than the component abstraction overheads of existing approaches.

B. Harmonic Interface Generation

This section presents an algorithm that determines interfaces of components and proxy tasks for a given system. The challenging issues in deriving interfaces are: 1) to meet harmonicity requirements of interface periods and 2) to reduce...
proximation overheads. The component interface’s period should be harmonic with all the workload tasks’ periods. In addition, proxy tasks in leaf components must also be harmonic with each other. This harmonicity condition leads to no abstraction overheads in the resulting hierarchical system.

The proximation overhead depends on the proxy task period which can be any number less than or equal to the original task’s period. We prefer a larger period for the proxy task that is close to the original task’s period to avoid implementation-related overheads (e.g., context switching and interrupt handling [20]).

The first step of the algorithm is to decide the base period for harmonic composition of the system. When we derive the interface of a component, the period of the interface should be harmonic with both its workload tasks and the interfaces of other sibling components.

Since the component interface is derived from the workload, the generation procedure is accomplished from leaves to the root component in the component hierarchy graph. Fig. 9 shows the pseudo-code of deriving a component interface. We use the depth-first-search mechanism to compute the interface, so that the function findCompInterface calls itself recursively if the workload task is a component (Line 3-4). For example, the function call of findCompInterface with the root component and the base harmonic period set \( H \) (line 7 of Fig. 9). The execution time of the interface is determined by the utilization of the workload (line 10 of Fig. 9).

We derive all the interfaces by calling the function findCompInterface for task \( \tau_i \) in a leaf component. Then, we extend it to the UB of a single task. Once the proxy task of a task or the interface of a component is determined, the next proxy task or component interface period should be harmonic with the period of the recently computed proxy task or interface. For example, when the proxy task period of \( v_0 \) in Fig. 8 is determined as 27, the proxy task period of \( \tau_{10} \) should be harmonic with 9 and 27. Thus, we add the new proxy task or interface period in the set of harmonic periods \( H \).

When all the tasks or subcomponents in a component workload have determined their proxy tasks or interfaces, the component interface can be determined easily. The interface period becomes the minimum among the periods of proxy tasks or interfaces in the workload. Since we need the interface period to be harmonic with the workload, the minimum period satisfies this condition (line 9 of Fig. 9). The execution time of the interface is determined by the utilization of the workload (line 10 of Fig. 9).

Once the proxy task of a task or the interface of a component is determined, the next proxy task or component interface period should be harmonic with the recently computed proxy task or interface. For example, when the proxy task period of \( v_0 \) in Fig. 8 is determined as 27, the proxy task period of \( \tau_{10} \) should be harmonic with 9 and 27. Thus, we add the new proxy task or interface period in the set of harmonic periods \( H \).

When all the tasks or subcomponents in a component workload have determined their proxy tasks or interfaces, the component interface can be determined easily. The interface period becomes the minimum among the periods of proxy tasks or interfaces in the workload. Since we need the interface period to be harmonic with the workload, the minimum period satisfies this condition (line 9 of Fig. 9). The execution time of the interface is determined by the utilization of the workload (line 10 of Fig. 9).

We derive all the interfaces by calling the function findCompInterface with the root component and the base harmonic period. Since the algorithm searches all positive integer-valued periods less than half of a task or interface period, the proposed algorithm runs in pseudo-polynomial time.

---

**Mathematical Formulation**

Let \( v_i \) be a task and \( H_i \) be the set of harmonic periods for \( v_i \) in the workload. The utilization bound (UB) of the task \( v_i \) is given by the following equation:

\[
\text{UB}_{v_i} = \frac{\Theta_i}{\Pi_i} \leq \frac{(k+1)U_i}{k+U_i}
\]

where \( p_i = k\Pi_i + \alpha \) and \( k \geq 1, 0 \leq \alpha < \Pi_i \).

**Proof:** Let \( \text{ldbfb}_{IPRM}(v_i, t) \) be the utilization bound of task \( v_i \) in the workload, which denotes the linear function passing through the point \((k\Pi_i + (\Pi_i - \Theta_i), k\Theta_i)\) as shown in Fig. 10. Then, for \( k\Pi_i \leq t \leq (k+1)\Pi_i \), \( \text{ldbfb}_{IPRM}(v_i, t) \) is less than or equal to \( \text{sbf}_{IPRM}(v_i, t) \).
Suppose that \( p_i = k\Pi_i + \alpha_i \) and \( \{(k + 1)U_i/[k + U_i]\} \leq U_{ij} \).

Then

\[
(k + 1)U_i \leq (k + U_i)U_{ij}.
\]

\[
U_i \leq \frac{kU_{ij}}{k + 1 - U_{ij}} = \frac{k\Theta_i}{\Pi_i(k + 1 - \frac{\Theta_i}{\Pi_i})} = k\Pi_i - \Theta_i.
\]

\[
e_i \leq \left( \frac{k\Pi_i + \Pi_i - \Theta_i}{k}\right)p_i\text{dbf}_A(\tau_i, p_i) = e_i \leq \text{ldbf}_{\text{IPRM}}(\tau_i, p_i) \leq \text{sbf}_{\text{IPRM}}(v_i, p_i).
\]

Since \( \text{dbf}_A(\tau_i, p_i) = e_i \) for any scheduling algorithm \( A \), the demand of the task at time \( p_i \) is less than or equal to the resource supply by the proxy task, which guarantees the schedulability of the task.

Now we extend the proxy task UB in Lemma 10 to a workload as shown in the following theorem.

**Theorem 2 (UB of a Workload: UBW):** A workload \( W \) is schedulable by \( \text{IPRM} \) if

\[
U_T = \frac{\Theta_i}{\Pi_i} \geq \text{UBW} = \frac{n}{k_i + U_i} (k_i + 1)U_i
\]

where \( p_i = k_i\Pi_i + \alpha_i \) (\( k_i \geq 1, 0 \leq \alpha_i < \Pi \)).

**Proof:** In the proposed framework, we assign a proxy task \( v_i(\Pi_i, \Theta_i) \) for each task \( \tau_i \in W \). Since all proxy tasks have the same period as the interface period, the component becomes harmonic so that there is no composition overhead among \( \Gamma \) and \( v_i \)'s. Now, let \( \Theta = \Theta_i + \Theta_2 + \ldots + \Theta_n \), where each \( \Theta_i \geq \Pi(\{(k_i + 1)U_i/[k_i + U_i]\}) \). Then, such a proxy resource distribution satisfies (6). Suppose that the component workload with such a proxy resource allocation is not schedulable. However, each task is schedulable by the proxy task \( (\Pi_i, \Theta_i) \) according to Lemma 10 because \( U_{ij} \geq \{(k_i + 1)U_i/[k_i + U_i]\} \). Therefore, this contradicts the assumption that the workload is not schedulable.

**VI. PERFORMANCE EVALUATION**

In this section, we compare the proposed CF with other overhead-reducing frameworks for hierarchical real-time systems. Specifically, we consider the following frameworks.

1) PRM [18]: CF based on PRM.

2) PRM with a single global period (PRM-single) [20]: Every interface is given a single period, which is less than or equal to \( p_{\text{min}} \), the smallest task period in the system.

3) A combination technique—harmonic component by using period-shrinking (PRM-sync-Trans): Components are harmonic and task periods are shrunken to be harmonic with interface periods.

4) Our technique (PRM-sync-Proxy).

The third scheme extends [24] to compose independent components into a system with harmonic interface periods. We adopt a simple harmonic transformation of tasks based on period-shrinking for the third scheme.

**A. Composition Overhead of Independent Components**

In this comparison, we want to show the efficiency of the proposed proximation-based composition approach. The proposed framework of this paper does not produce any abstraction overheads in intermediate components, but the proximation overhead incurred by leaf components remain.

Since the resource bound in (6) depends on individual task utilizations, we analyze it in two ways. First, we analyze it under the assumption of individually identical task utilization (i.i.u.). Second, we measure the average resource bound for randomly generated task sets.

If we assume that each task has the same utilization, (6) becomes \( (k + 1)U_W/(k + U_W/n) \), where \( k = \text{min}_{i \in W} k_i \) and \( n \) is the number of tasks in \( W \). We can then compare the resource bound of the proximation-based CF against that of PRM under RM and EDF based on the in [18]. Fig. 11 shows the resource bounds of three schemes when \( n = 4 \). The results for other \( n \) show a similar pattern.

The proposed scheme generally shows lower resource bound in lower utilization. As \( k \) increases, the resource bounds of the proposed scheme performs better than PRM-RM and becomes close to PRM-EDF. In the case of \( k = 1 \), the proposed scheme under i.i.u. is worse than PRM-RM (PRM under RM). It is because the component overhead of the proposed technique is the sum of proximation overheads of individual tasks and a large interface period \( (k = 1) \) increases each proximation overhead of tasks.

Next, we analyze the resource bounds for randomly generated tasks. We generate \( n \) tasks for each workload utilization range, where \( n \) is randomly selected from 2 to 6. Individual task utilization is generated using a normal distribution with mean \( U_W/n \) and standard deviation 50% of the utilization range (i.e., 0.05). We also randomly select \( k \) from 1 to 10. We measure the average resource bounds of 500 random task sets.
for each utilization range and show the results in Fig. 12(a). The resource bound overhead of Fig. 12(b) is obtained using \((UB^W - U^W)/U^W\) for a given resource bound UB^W.

The resource bound of the proposed scheme performs better than PRM when the workload utilization is less than about 70%. However, for higher workload utilizations, the proposed scheme has higher resource bounds than PRM-EDF. This is because the resource bound of the proposed scheme depends on individual task approximation overheads, as shown in Fig. 12(b).

**B. Composition Overhead**

In simulation experiments, we use a system structure with the depth of three as in Fig. 13. Each leaf component has three tasks, resulting in a system that contains eighteen tasks in all. The task period is randomly generated between \(p_{min}\) and \(p_{max}\) for each component as shown in Table I.

For a given system utilization range of \([U^W_{min}, U^W_{max}]\), we generate each task period and execution time such that the total utilization resides in the range. As shown in Fig. 14, we varied the utilization range from 0.2 to 0.6 in an interval of 0.05. We define the utilization overhead as the difference between the top-layer workload utilization by composition and the actual utilization of eighteen tasks. For each utilization range, we generate 100 task sets and measure the average overhead.

Fig. 14(a) shows the average utilization overhead for each utilization range of task sets, while Fig. 14(b) shows the overheads in percentile compared to the task set utilization. As shown in Fig. 14, the proposed scheme shows the lowest overhead of about 5%. Although both PRM-sync-Trans and PRM-single have no intermediate overhead, the difference comes from the additional reduction due to the harmonicity requirement between leaf components and their proxy tasks. Since PRM-sync-Trans incurs more overhead than PRM-sync-Proxy, task proximation seems to be more efficient than just reducing task periods for harmonicity.

Fig. 15 shows the utilization overhead in each component of PRM and PRM-sync-Proxy. Since PRM does not consider intermediate component overheads, it generates additional composition overhead in \(C_1\) and \(C_2\) as shown in Fig. 15. The proposed framework removes all intermediate composition overheads, so \(C_1\) and \(C_2\) in Fig. 15 has solid bars. Note that PRM-single is also a special case of the proposed framework because all components use a common period, resulting in harmonicity. Another improvement comes from the reduction of overhead in leaf components with RM scheduling algorithm. For example, \(C_4, C_6,\) and \(C_8\) in Fig. 15 show high overheads because of RM algorithm.

**C. Runtime Overhead of HSS**

The smaller the interface period, the more is the runtime overhead such as context switch and scheduling [23]. For
instance, the common period of PRM-single [20] needs to be smaller than or equal to ρ_{min} of the system, implying that it may induce frequent context switches. The proxy task of our framework also imposes such a runtime overhead. So this section compares the runtime overhead of the proposed technique against prior techniques. We only counted the number of context switches of components per time unit in simulations, since context switches are one of the major runtime overheads.

Fig. 16 shows the average number of context switches of four schemes. PRM-single has many context switches due to the small interface period. On the contrary, PRM-sync-Proxy has an additional layer due to task proximation, it shows more context switch overhead than PRM-sync-Trans, but still significantly lower than PRM-single.

VII. CONCLUSION
OS-Hypervisor engineers can better utilize resources using our technique as follows.
1) OS-Hypervisor can better utilize the resources by using harmonic interfaces.
2) If the host system manages the virtual system to be harmonic, regardless of host’s scheduling algorithm, it can guarantee Θ/Π bandwidth of resource supply for the virtual system’s resource request of (Π, Θ).
3) If the host system running EDF manages the virtual system to be divisible, it can guarantee Θ/Π bandwidth of resource supply for the virtual system’s resource request of (Π, Θ).

This paper discussed two abstraction overheads of prior CF and presented sufficient conditions to remove these overheads. In addition, we provided novel techniques and algorithms that allow tasks to be composed in an HSS with low overheads. We showed that our technique reduces overhead by about 10% on average when compared to a technique that uses a single global period and by about 8% on average when compared to a technique based on harmonicity.

Our proximation technique is not optimal since the base period of harmonic periods is randomly selected such that it is less than the minimal period among all workload tasks. In the future work, we plan to devise an algorithm to compute optimal harmonic periods of components and investigate how to reduce the overheads when applying the proposed technique in practical systems.

REFERENCES
[1] Green Hills Software. (2017). Safety Critical Products: ARINC 653 Partition Scheduler. [Online]. Available: https://www.ghs.com/products/safety_critical/arinc653.html/
[2] M. S. Mollison, J. P. Erickson, J. H. Anderson, S. K. Baruah, and J. A. Sorensen, “Mixed-criticality real-time scheduling for multicore systems,” in Proc. 10th IEEE Int. Conf. Comput. Inf. Technol., Jun. 2010, pp. 1864–1871.
[3] A. Easwaran, I. Lee, O. Sokolsky, and S. Vestal, “A compositional scheduling framework for digital avionics systems,” in Proc. 15th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA), Aug. 2009, pp. 371–380.
[4] I. Lee et al., “Realizing compositional scheduling through virtualization,” in Proc. IEEE 18th Real Time Embedded Technol. Appl. Symp., Apr. 2012, pp. 13–22.
[5] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, “TIMES: A tool for schedulability analysis and code generation of real-time systems,” in Proc. FORMATS, 2003, pp. 60–72.
[6] L. Thiele, E. Wandeler, and N. Stoimenov, “Real-time interfaces for composing real-time systems,” in Proc. 6th ACM Int. Conf. Embedded Softw. (EMSOFT), 2006, pp. 34–43.
[7] G. Weiss, R. Alur, A. Bicchi, and G. Buttazzo, “Automata based interfaces for control and scheduling,” in Proc. Int. Workshop Hybrid Syst. Comput. Control, 2007, pp. 601–613.
[8] R. Henia et al., “System level performance analysis—The SymTA/S approach,” IEE Proc. Comput. Digit. Techn., vol. 152, no. 2, pp. 148–166, Mar. 2005.
[9] A. Kohn et al., “Timing analysis for hypervisor-based I/O virtualization in safety-related automotive systems,” SAE Int. J. Passenger Cars Electron. Electr. Syst., vol. 10, no. 2, pp. 368–379, 2017.
[10] T. A. Henzinger and S. Matic, “An interface algebra for real-time components,” in Proc. 12th IEEE Real Time Embedded Technol. Appl. Symp. (RTAS), San Jose, CA, USA, 2006, pp. 253–266.
[11] A. K. Mok, X. Feng, and D. Chen, “Resource partition for real-time systems,” in Proc. 7th IEEE Real Time Technol. Appl. Symp. (RTAS), 2001, pp. 75–84.
[12] I. Shin and I. Lee, “Periodic resource model for compositional real-time guarantees,” in Proc. 24th IEEE Real Time Syst. Symp. (RTSS), Dec. 2003, pp. 2–13.
[13] A. Easwaran, M. Anand, and I. Lee, “Compositional analysis framework using EDP resource models,” in Proc. 28th IEEE Int. Real Time Syst. Symp. (RTSS), Dec. 2007, pp. 129–138.
[14] W.-J. Chen, P.-C. Huang, Q. Leng, A. K. Mok, and S. Han, “Regular composite resource partition in open systems,” in Proc. IEEE Int. Real Time Syst. Symp. (RTSS), Paris, France, Dec. 2017, pp. 34–44.
[15] I. Shin, A. Easwaran, and I. Lee, “Hierarchical scheduling framework for virtual clustering of multiprocessors,” in Proc. Euromicro Conf. Real Time Syst., Jul. 2008, pp. 181–190.
[16] Y. Chang, R. Davis, and A. Wellings, “Schedulability analysis for a real-time multiprocessor system based on service contracts and resource partitioning,” Dept. Comput. Sci., Univ. York, York, U.K., Rep. YCS 432, 2008.
[17] T.-W. Kuo and C.-H. Li, “A fixed-priority-driven open environment for real-time applications,” in Proc. 20th IEEE Real Time Syst. Symp., Phoenix, AZ, USA, 1999, pp. 256–267.
[18] I. Shin and I. Lee, “Compositional real-time scheduling framework with periodic model,” ACM Trans. Embedded Comput. Syst., vol. 7, no. 3, 2008, Art. no. 30.
[19] A. Easwaran, M. Anand, I. Lee, and O. Sokolsky, “On the complexity of generating optimal interfaces for hierarchical systems,” in Proc. Int. Workshop Compositional Theory Technol. Real Time Embedded Syst., 2008.
[20] S. Chen, L. T. X. Phan, J. Lee, I. Lee, and O. Sokolsky, “Removing abstraction overhead in the composition of hierarchical real-time systems,” in Proc. 17th IEEE Real Time Embedded Technol. Appl. Symp., Chicago, IL, USA, Apr. 2011, pp. 81–90.
[21] A. Biondi, G. C. Buttazzo, and M. Bertogna, “Schedulability analysis of hierarchical real-time systems under shared resources,” IEEE Trans. Comput., vol. 65, no. 5, pp. 1593–1605, May 2016.
[22] S. Kramer, D. Ziegenbein, and A. Hamann, “Real world automotive benchmark for free,” in Proc. WATER, 2015.
[23] L. T. X. Phan, M. Xu, J. Lee, I. Lee, and O. Sokolsky, “Overhead-aware compositional analysis of real-time systems,” in Proc. Real Time Embedded Technol. Appl. Symp., Philadelphia, PA, USA, 2013, pp. 237–246.
[24] C. Guo, X. Hua, H. Wu, D. Lautner, and S. Ren, “Best-harmonically-fit periodic task assignment algorithm on multiple periodic resources,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 5, pp. 1303–1315, May 2016.
[25] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” J. ACM, vol. 20, no. 1, pp. 46–61, 1973.
[26] T.-W. Kuo and A. K. Mok, “Load adjustment in adaptive real-time systems,” in Proc. Real Time Syst. Symp., Dec. 1991, pp. 160–170.
[27] B. Sprunt, L. Sha, and J. Lehozcky, “Aperiodic task scheduling for hard-real-time systems,” Real Time Syst., vol. 1, no. 1, pp. 27–60, 1989.
Jin Hyun Kim received the Ph.D. degree from the Department of Computer Science and Engineering, Korea University, Seoul, South Korea, in 2011.

He was a Post-Doctoral Fellow with the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, and Aalborg University, Aalborg, Denmark. He was a Researcher with INRIA/IRISA, Rennes, France. Since 2015, he has been a Post-Doctoral Fellow with the University of Pennsylvania, Philadelphia, PA, USA. His current research interests include cyber physical system and real-time system model checking, timing analysis, formal methods, and process algebras.

Kyong Hoon Kim received the B.S., M.S., and Ph.D. degrees in computer science and engineering from POSTECH, Pohang, South Korea, in 1998, 2000, and 2005, respectively.

Since 2007, he has been a Professor with the Department of Informatics, Gyeongsang National University, Jinju, South Korea. From 2005 to 2007, he was a Post-Doctoral Research Fellow with the CLOUDS Laboratory, Department of Computer Science and Software Engineering, University of Melbourne, Melbourne, VIC, Australia. His current research interests include real-time systems, cloud computing, avionics software, and security.

Arvind Easwaran (M’13) received the Ph.D. degree in computer and information science from the University of Pennsylvania, Philadelphia, PA, USA, in 2008.

He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, which he joined in 2013. His current research interests include cyber-physical systems, embedded real-time systems, and formal methods.

Insup Lee (F’01) received the B.S. degree from UNC-Chapel Hill, Chapel Hill, NC, USA, and the Ph.D. degree from UW-Madison, Madison, WI, USA.

He is a Cecilia Fitler Moore Professor of Computer and Information Science with the University of Pennsylvania, Philadelphia, PA, USA, where he has been the Co-Director of Penn Health Tech, since 2017, the Director of the PRECISE Center, since 2008, and holds a secondary appointment with the Department of Electrical and Systems Engineering. His current research interests include cyber-physical systems, real-time and embedded systems, runtime assurance and verification, medical cyber-physical systems, connected health, Internet of Medical Things, and security of cyber physical systems. The theme of his research activities has been to assure and improve the correctness, safety, and timeliness of life-critical embedded systems.

Dr. Lee was a recipient of the IEEE TC-RTS Outstanding Technical Achievement and Leadership Award in 2008, and the Best Paper Award in IEEE RTSS 2003, CEAS 2011, IEEE RTSS 2012, ACM/IEEE ICCPS 2014, IEEE CPSNA 2016, ISORC 2018, and IEEE RTAS 2012. He has served on several program committees and chaired several international conferences and workshops. He has also served on various steering and advisory committees of conferences and technical societies, and the editorial boards of several scientific journals. He was the Chair of the IEEE Computer Society Technical Committee on Real-Time Systems from 2003 to 2004, and an IEEE CS Distinguished Visitor Speaker from 2004 to 2006. He has been the Chair of ACM SIGBED since 2015. He is an ACM Fellow.