IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 8, 2021, accepted October 13, 2021, date of publication October 15, 2021, date of current version October 25, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3120743

Q-CSF: Quantum-Aware Compositional Scheduling Framework for Hierarchical Real-Time Systems

JAEWOO LEE[1], (Member, IEEE), AND HYEONGBOO BAEK[2]
[1]Department of Industrial Security, Chung-Ang University, Seoul 06974, South Korea
[2]Department of Computer Science and Engineering, Incheon National University, Incheon 22012, South Korea

Corresponding author: Hyeongboo Baek (hbbaek@inu.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) Support Program through supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP) under Grant IITP-2021-2018-0-01799; in part by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of Korea under Grant 20199710100060; and in part by the National Research Foundation of Korea (NRF) funded by the Korea Government (MSIT) under Grant 2021R1F1A1059277.

ABSTRACT Component-based design has received considerable attention owing to its advantages in terms of security and safety when developing modern embedded systems. To effectively allocate computing resources to components in these systems, real-time component-based scheduling theory has been studied from various perspectives. The main advantage of component-based scheduling theory is that it guarantees the schedulability of an independent component and composability of multiple components. However, the existing component scheduling theory cannot be directly applied to real hardware platform due to an impractical assumption that resource allocation must be conducted across the continuum of real numbers, whereas actual operating systems (or virtualization systems) allocate resources in units of scheduling time quantum. In this study, we proposed a new efficient resource allocation and supply mechanism for quantumized hardware platforms while using real-number-based component interface. In simulation results with randomly-generated workloads, our approach reduced the overhead of existing approaches by up to 97.1% in an individual component. In composition of multiple components, our approach has up to 0.41 better acceptance ratio than existing approaches.

INDEX TERMS Component-based systems, schedulability analysis, compositional scheduling framework, supply bound function, time quantum.

I. INTRODUCTION

Traditional embedded systems that perform dedicated functions in restricted environments are evolving toward cyber physical systems (CPS), which operate more complicated functions involving both physical and computing aspects [1], [2]. A compelling example of CPS is the autonomous driving system (ADS), whose functions include 1) sensing and controlling physical devices, and 2) perception and planning with algorithms. The most important requirement for such functions is the timing guarantee, such that each periodic task must be completed in a predefined time (i.e., a deadline) [1]. For instance, the adaptive cruise-control system [1] of an ADS correctly operates only when its embedded camera and LiDAR tasks deliver sensing data in a timely manner to the motor-control task through associated computing interfaces.

There have been a few approaches that attempted to realize real-time CPS design [3]–[7]. Among them, the component-based method has received considerable attention, owing to its advantages in terms of security and safety [3], [7]. Component-based design aims to simplify CPS design by splitting it into sub-procedures and multiple components. For example, an ADS includes various components such as a driving system that includes engine and braking tasks, and a control system that encompasses steering tasks. Furthermore, merging developed components into a combined system necessitates component abstraction. The high-level idea of component abstraction is providing interfaces that communicate between components and externals while hiding complicated architectural details. Because each component

The associate editor coordinating the review of this manuscript and approving it for publication was Laxmisha Rai.

is individually developed and operated, the component-based systems are normally attack- and fault-resilient, compared to traditional systems.

Component-based design has been incorporated into real-time systems by exploiting component scheduling theory. Component scheduling framework (CSF) [3], [7] aims to effectively allocate computing resources (e.g., processors and networks) to multiple components. Then, virtualization technique on operating systems promotes to schedule task (or component) and allocate computing resources under cloud platforms [8]–[13]. Among them, RT-Xen [12], [13] is one representative example on a real-time cloud platform, which supports the timely execution for components in a Xen virtualization environment.

In the CSF, each component has its own workload, which can be a set of tasks or a set of components. Component can schedule its workload by its own scheduler. If a parent component has multiple child components as its workload (refer to Figure 1), the parent component need to schedule its child components. Each component has its component interface which specifies resource demands in terms of time duration. From the perspective of the parent component, a child component can be transformed into a task by its component interface. Then, the schedulability of each component is guaranteed by comparing the worst-case resource-supply (e.g., derived by the supply bound function) and resource demand (e.g., derived by the request bound function). However, the original component scheduling theory [3] makes an impractical assumption that the resource allocation can be conducted in units of real numbers, whereas actual operating systems or virtualization systems allocate resources in units of quantum time. To address this problem, an existing study [13] utilized integer-based resource-supply and resource-demand policies, but it found that it did not derive effective scheduling algorithms and relevant implementations. In this study, we proposed a new efficient supply bound function for quantumized hardware platforms while using real-number-based component interface.

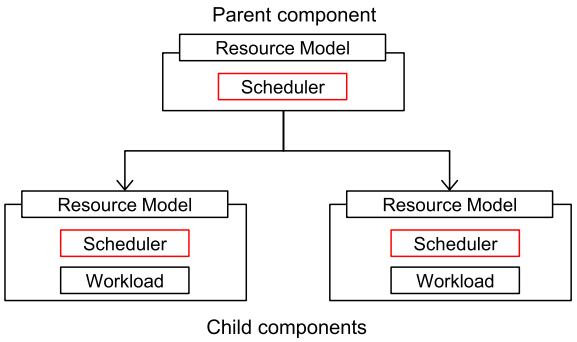


FIGURE 1. System model for a component-based hierarchical system (the top-level parent component is the root of the system, i.e., root component).

This paper makes the following contributions:

- We develop a new quantum-aware compositional scheduling framework (Q-CSF) of which every scheduling decision is quantum-based for hierarchical real-time systems.
- With Q-CSF, we propose an integer-based supply bound function (iSBF) to the address resource-allocation problems using quantumized hardware platforms.
- We present a new schedulability analysis for Q-CSF and algorithm to compute a resource efficient component interface for Q-CSF.
- From component-level simulation results, we show that Q-CSF reduced the interface overhead of the existing approach for quantumized platforms by up to 97.1%. From system-level simulation results, we show that Q-CSF has up to 0.41 better acceptance ratio than the existing approach.

The rest of the paper is structured as follows: Section II presents the system model and background. Section III explains the challenges and our framework. Section IV evaluates the proposed framework, and Section V discusses related work. Finally, Section VI concludes the paper.

II. SYSTEM MODEL AND BACKGROUND

In this section, we present our system model and assumptions (Section II-A), problem forumation (Section II-B), and a theoretical background (Section II-C) for real-time hierarchical systems. We summarize our notations in Table 1.

A. SYSTEM MODEL AND ASSUMPTIONS

We consider hard real-time systems where missing deadlines cause the system failure, whereas soft real-time systems tolerates some deadline misses. Therefore, our performance metrics is resource efficiency or acceptance ratio.[1]

Our target system is a hierarchical (compositional) real-time system comprising a set of real-time components, where a component may include a set of child components, in a tree-like manner as shown in Figure 1. A component is noted by $\mathbb{C} = (W, \Gamma, A)$ where W is a workload (described below), Γ is the resource model as the interface of the component (described below) and A is the scheduling algorithm (e.g., deadline monotonic (DM) or earliest deadline first (EDF)) that is used to schedule W. In this study, we only consider deadline monotonic (DM) scheduling algorithm in which a task having a smaller relative deadline has a higher priority. For simplicity of representation, we denote $\mathbb{C} = (W, \Gamma, A)$ by $\mathbb{C} = (W, \Gamma)$ because A is fixed to DM.[2]

1) COMPONENT WORKLOAD

A workload W in a component $\mathbb{C}$ consists of n real-time tasks: $W = \{\tau_1, \tau_2, \cdots, \tau_n\}$. We consider a constrained-deadline sporadic task, where the deadline of the task (D_i) is equal to or less than the period of the task (T_i), and the next job of the

[1] We do not consider soft real-time performance metric such as turnaround time, response time, and deadline miss ratio.

[2] We will consider EDF and other scheduling policies in future work.

TABLE 1. Descriptions for symbols and acronyms.

Term	Description	Term	Description
$\mathbb{C}$	A component	W	A component workload
Γ	The resource model	A	A scheduling algorithm
τ	A task set	τ_i	The i-th task
n	The number of tasks in W	T_i	The minimum inter-job arrival time
C_i	A minimum inter-arrival time or period of τ_i	D_i	The deadline of the task
J_i^j	The j-th job of a task τ_i	Π	The resource period in PRM Γ
Θ	The amount of resource supply within a resource period Π	$HP(\tau_i)$	The task set having a higher priority than τ_i
u_i	The task utilization of τ_i	U_b	Utilization bound
ADS	Autonomous driving system	CBD	Component-based design
CPS	Cyber-physical systems	CSF	Component scheduling framework
DBF	Request bound function	DM	Deadline monotonic (scheduling algorithm)
EDF	earliest deadline first (scheduling algorithm)	SBF	Supply bound function
PRM	Periodic resource model	iSBF	Integer-based SBF
RBF	Request-bound function	Q-CSF	Quantum-aware compositional scheduling framework
WCET	worst-case execution time		

task releases at any random time instant after the release time of the current job plus the period of the task. Each task τ_i is characterized by (T_i, C_i, D_i), where

- $T_i \in \mathbb{N}$ (a natural number, i.e., positive integer) is the minimum inter-job arrival time (or *period*),
- $C_i \in \mathbb{R}^+$ (positive real number) is the worst-case execution time (WCET),[3] and
- $D_i \in \mathbb{N}$ is the relative deadline (the relative time of the deadline of a real-time job from the release time of the job while absolute deadline indicates the deadline of a real-time job from time 0).

A task τ_i generates an infinite sequence of jobs $\{J_i^1, J_i^2, \cdots\}$: if a job J_i^j is released (or arrived) at t, then the absolute deadline of J_i^j is $t+D_i$, and the next job J_i^{j+1} may be released at any time after $t + T_i$. The term "deadline" indicates the relative deadline unless specified otherwise. A task τ_i is schedulable if every job J_i^j of τ_i completes its execution before its absolute deadline. Under the DM scheduling policy, workload W is schedulable if every task τ_i in W is schedulable.

2) RESOURCE MODEL

In component-based system design, component interfaces eliminate implementation details and facilitate their easy use of components. In CSF [7], the *resource model* abstracts (computing) resource demand for a real-time component as the component interface. The representative resource model is periodic resource model (PRM) [3], $\Gamma = (\Pi, \Theta)$, which consists of the resource period, $\Pi \in \mathbb{N}$, and the (computing) resource supply within the resource period, $\Theta \in \mathbb{R}^+$. PRM $\Gamma = (\Pi, \Theta)$ means that the resource model Γ can supply Θ units of resources (such as 2ms CPU execution) in every Π time units (such as 10ms).

[3]The worst-case execution time may be computed by static analysis with mathematical computation, which may lead a positive real number, not a natural number.

B. PROBLEM FORMULATION

The problem of CSF has not much considered under quantumized platforms (resource cannot be supplied in fraction of time quantum). In this paper, we consider the following problems:

- For a given resource model (e.g., PRM), how can we compute the possible worst-case resource supply of in terms of time interval length considering quantumized platforms?
- For a given component where resource model and component workload are fixed, how can we determine the schedulability of the component under quantumized platforms?
- For a given component workload, how can we compute the resource-efficient resource model under quantumized platforms?

C. BACKGROUND

In this subsection, we introduce the background knowledge of the schedulability analysis of real-time systems and compositional (hierarchical) systems. To analyze the schedulability of real-time systems, we must investigate the worst-case resource demand of each task to complete its execution miss and resource supply provided by the platform for a given time interval. Hence, by identifying the upper-bounded resource demand of each task and the lower-bounded resource supply for a given time interval, we can judge whether each task is schedulable. The request-bound function (RBF) gives the worst-case resource demand[4] of task τ_i among workload W for a given time duration t under the DM scheduling algorithm. We can then compute RBF as follows:

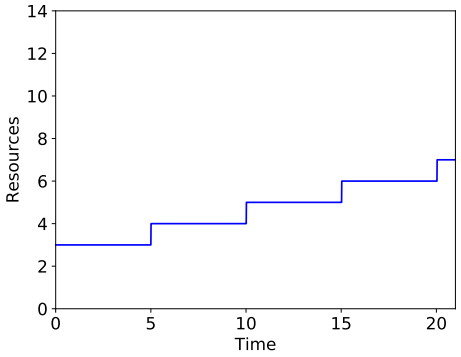
$$\mathsf{RBF}(W, i, t) = \sum_{k \in HP(\tau_i)} \left\lceil \frac{t}{T_k} \right\rceil C_k + C_i \tag{1}$$

where $HP(\tau_i)$ is the task set having a higher priority than τ_i.

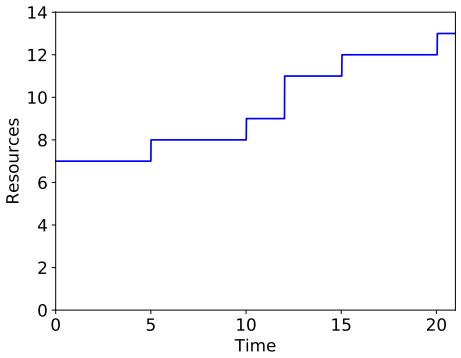
[4]The worst-case release pattern of sporadic task is periodic.

The following example shows how to calculate the resource demand of a task among a given workload.

Example 1: Consider a workload, $W = \{\tau_1, \tau_2, \tau_3\}$, *where* $\tau_1 = (T_i = 5, C_i = 1, D_i = 5)$, $\tau_2 = (12, 2, 12)$, *and* $\tau_3 = (18, 4, 17)$. *Figure 2 draws the resource demand of* $\mathsf{RBF}(W, 2, t)$ *and* $\mathsf{RBF}(W, 3, t)$ *according to Equation* (1). *Note that* $\mathsf{RBF}(W, 1, t)$ *is a constant function.*



(a) $\mathsf{RBF}(W, 2, t)$



(b) $\mathsf{RBF}(W, 3, t)$

FIGURE 2. **The illustration of** $\mathsf{RBF}(W, 2, t)$ **and** $\mathsf{RBF}(W, 3, t)$ **in Example 1.**

We then consider the worst-case resource supply: for non-hierarchical systems, the worst-case resource supply for a given time interval t is t. Thereafter, we can check whether the workload is schedulable. The following lemma presents the schedulability condition under the DM scheduling algorithm.

Lemma 1 (From [14]): A given workload W *is schedulable under the DM scheduling algorithm if*

$$\forall \tau_i (\exists t \leq D_i, \mathsf{RBF}(W, i, t) \leq t). \tag{2}$$

Next, we consider the schedulability of a component-based (hierarchical) system (e.g., Figure 1). Checking the schedulability of a real time component is different from that of a traditional (non-hierarchical) system (Lemma 1). To analyze schedulability in CSF, we introduce a *Supply Bound Function (SBF)* [7] that gives the worst-case resource supply of PRM Γ in a given time interval t.

Figure 3 illustrates the worst-case pattern of resource-supply of PRM Γ for a time interval t. Let's consider arbitrary resource supply from PRM Γ. Resource may be supplied in the beginning of the period, the middle of the period, or the end of the period. The worst-case resource-supply happens when resource is not supplied until the deadline of the first job of some task. Then, we consider the worst-case resource starvation scenario[5] among all possible resource-supply pattern: the resource of the previous resource period (before time zero) is supplied at the beginning of the resource period, and the first resource is supplied at the end of the resource period. Then, the length of the worst-case resource starvation is $2(\Pi - \Theta)$, as shown in Figure 3. Next, the worst-case of the following resource supply happens when resource is supplied at the end of the following resource period.

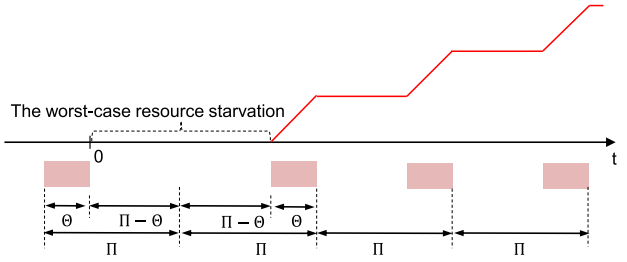


FIGURE 3. **The illustration of the worst-case pattern of resource supply of** $\mathsf{SBF}(\Gamma, t)$.

Then, we compute SBF as follows:

$$\mathsf{SBF}(\Gamma, t) = k\Theta + \max(0, t - 2l - k\Pi) \tag{3}$$

where $l = \Pi - \Theta$ and $k = \max(0, \lfloor \frac{t-l}{\Pi} \rfloor)$. Note that l and k indicate the unserviced time across the period (i.e., the difference between Π and Θ) and the number of whole resource supplies within the time interval,[6] respectively.

Extending Lemma 1 with the SBF, the next lemma shows the schedulability condition of a real-time component.

Lemma 2 (From [7]): Consider a real-time component $\mathbb{C} = (W, \Gamma)$. *The component is schedulable under the DM scheduling algorithm if*

$$\forall \tau_i (\exists t \leq D_i, \mathsf{RBF}(W, i, t) \leq \mathsf{SBF}(\Gamma, t)). \tag{4}$$

Next, we present a way to compute the component interface if the component workload is given. The abstraction problem indicates how to find a resource-efficient component interface (i.e., PRM model $\Gamma = (\Pi, \Theta)$) for a given component workload. The existing work [3], [7] considers the abstraction problem to find a proper resource supply Θ for a given component workload W and a given resource period Π. Then, the problem is computing the minimum Θ that can schedule W with component period Π. By solving the equation in Lemma 2 using W and Π, we find the proper Θ. Refer to the details of the computations in [3] and [15].

However, the solution of the abstraction problem is a real-number interface even if all the workload parameters are integers. Hence, the solution is not directly applicable to

[5] In the perspective of a component (or a task), resource starvation duration indicates the largest time duration that resource is not supplied from the higher-level component (or the system).

[6] Alternatively, k means the number of intervals of Πs after time point l in Figure 3.

quantum-based real platforms. The next example shows why the solution is a real-number.

Example 2: Consider a component where its workload is $W = \{\tau_1, \tau_2\}$ *where* $\tau_1 = (T_i = 5, C_i = 1, D_i = 5)$ *and* $\tau_2 = (7, 1, 7)$. *Assume that component period is given by* $\Pi = 3$. *If* $\Theta = 1$, *then Equation* (4) *of Lemma 2 does not hold for* τ_2 *because* $\mathsf{RBF}(W, 2, t) > \mathsf{SBF}(\Gamma, t)$ *for all* $t \leq 7$. *It means that workload* W *is not schedulable when* $\Gamma = (3, 1)$. *If* $\Theta = 2$, *then Equation* (4) *holds for both* τ_1 *and* τ_2. *If* $\Theta = 1.7$, *then Equation* (4) *still holds for both* τ_1 *and* τ_2. *For* τ_2, *we observed that* $\mathsf{RBF}(W, 2, t) \leq \mathsf{SBF}(\Gamma, t)$ *at time* 7: $\mathsf{RBF}(W, 2, 7) = 3$ *and* $\mathsf{SBF}(\Gamma, 7) = 1 * 1.7 + \max(0, 1.4) = 3.1$. *The above calculation indicates that the minimum* Θ *exists between* 1 *and* 1.7, *meaning that the solution of the abstraction problem is a real-number.*

III. THE QUANTUM-AWARE COMPOSITIONAL SCHEDULING FRAMEWORK

We introduce the challenge of the quantum-aware resource supply in Section III-A and present our approaches in Section III-B. Then, we analyze the schedulability of our approaches and present the algorithm to compute the resource-efficient resource model in Section III-C.

A. CHALLENGES FOR QUANTUM-AWARE RESOURCE SUPPLY

In real digitized embedded systems, we cannot supply a fractional unit of resources (e.g., allocate 0.7 processor time quantum to task A). Possible resource allocation is to assign a task into the processor while executing the task for least one time quantum. However, the solution of abstraction problem is a real-number interface, even if all workload parameters are integers (refer to Example 2). To utilize the solution of the abstraction problem (i.e., resource-efficient real-number PRM interface), we require a quantum-aware resource supply mechanism.

One naive way to provide a quantum-aware resource supply is to use *nPRM*, which is the PRM $\Gamma = (\Pi, \Theta)$ where $\Pi \in \mathbb{N}$ and $\Theta \in \mathbb{N}$. To supply sufficient resources for the component workload, we transform the real-number PRM $\Gamma = (\Pi, \Theta)$ into the nPRM Γ' as follows:

$$\Gamma' = (\Pi, \lceil \Theta \rceil).$$

The next example shows the difference between the PRM and the nPRM when abstracting the component workload into the component interface.

Example 3: Suppose that the workload of a component is abstracted to the PRM $(3, 1.6)$. *For the same workload, we can abstract it into the nPRM* $(3, 2)$. *Although the nPRM can support quantumized resource supply, it has 0.133 larger overhead than the PRM in terms of utilization:* $2/3 - 1.6/3 = 0.1333$.

As shown in Example 3, the drawback of nPRM is inefficient resource utilization, which is severe for smaller Π. Our challenge for quantum-aware resource supply is to reduce inefficiency even for small Π.

B. A NEW RESOURCE SUPPLY MODEL FOR Q-CSF

In this subsection, we propose a new resource supply model for quantumized platforms, called *integer Supply Bound Function (iSBF)*, which calculate the minimum resource supply considering scheduling quantum in real hardware platforms. For given PRM Γ and time interval t, we propose an iSBF as follows:

$$\mathsf{iSBF}(\Gamma, t) = \lfloor k\Theta \rfloor + \max(0, t - l - m - k\Pi) \qquad (5)$$

where $l = \Pi - \lfloor \Theta \rfloor$, $k = \max(0, \lfloor \frac{t-l}{\Pi} \rfloor)$, $m = \Pi - QS(k, \Theta)$, and $QS(k, \Theta) = \lfloor k\Theta \rfloor - \lfloor (k-1)\Theta \rfloor$. We illustrate the worst-case resource-supply pattern considering quantumization in Figure 4. To express the quantumized resource supply, we define the function $QS(k, \Theta)$. It computes the maximum integer resource-supply during $k + 1$ periods, which is the floor value of the cumulative resource-supply from time 0 to $k + 1$ period. We note that the worst-case resource starvation considering the maximum integer resource-supply is $2\Pi - \lfloor \Theta \rfloor - QS(1, \Theta))$, which is also expressed in the design of Equation (5).

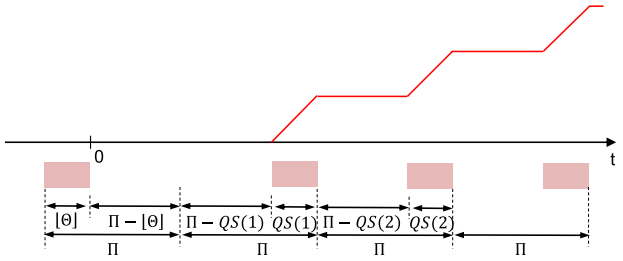


FIGURE 4. The illustration of the worst-case pattern of resource supply of *iSBF*(Γ, t).

Next, we examine characteristics of iSBF. First, iSBF can compute the worst-case integer units of resource supply for a given time interval from a specific resource model. Second, the remaining resource supply after the quantumization of the resource supply (fractional part of resource supply) is accumulated to the next resource supply. Third, the worst-case resource starvation of iSBF $(2\Pi - \lfloor \Theta \rfloor - QS(1, \Theta))$ is larger than that of SBF $(2\Pi - 2\Theta)$, owing to quantumized resource supply.

The next example shows the difference of SBF and iSBF, illustrating in Figure 5.

Example 4: Consider the PRM $\Gamma = (3, 1.6)$. *Figure 5 illustrates the* $\mathsf{SBF}(\Gamma, t)$ *and the* $\mathsf{iSBF}(\Gamma, t)$ *in terms of time interval* t. *The worst-case resource starvation of* SBF *is* 2.8 *while that of* iSBF *is* 4. *The resource supply of* Γ *until time* 5 *is* $QS(1) = \lfloor 1.6 \rfloor = 1$. *The resource supply of* Γ *during time* $[5, 8)$ *is* $QS(2) = \lfloor 2 * 1.6 \rfloor - \lfloor 1.6 \rfloor = 2$.

C. SCHEDULABILITY ANALYSIS OF Q-CSF

In this section, we present the schedulability analysis of Q-CSF and the computation method of the minimum-bandwidth PRM under Q-CSF. Extending Lemma 2 with iSBF, Theorem 1 shows the schedulability condition of

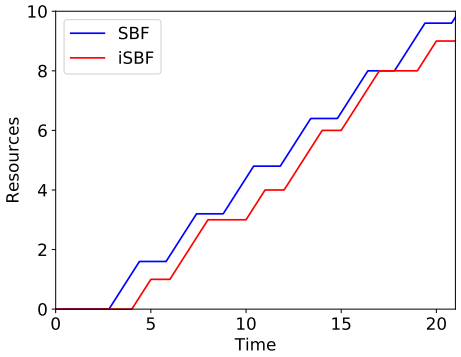


FIGURE 5. **The comparison of SBF and iSBF in Example 4.**

a real-time component under quantum-based systems. The proof of Theorem 1 is constructed based on the proof of Theorem 4.2 in [7].

Theorem 1 (Scheduliability Condition Under Q-CSF): Consider a real time component $\mathbb{C} = (W, \Gamma)$ *where* Γ *is PRM. The component is schedulable under the DM scheduling algorithm if*

$$\forall \tau_i (\exists t \leq D_i, \mathsf{RBF}(W, i, t) \leq \mathsf{iSBF}(\Gamma, t)). \tag{6}$$

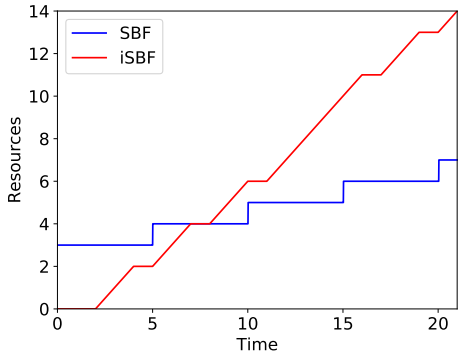
Proof: The task set is schedulable under DM scheduling algorithm if each task is schedulable. Consider a task τ_i. Because the scheduling algorithm is DM, task τ_i completes its execution before the deadline of the task D_i if the execution requirement from tasks whose priorities are higher than τ_i and the execution requirement of τ_i is satisfied with resources supplied by iSBF with Γ at time $t \in [0, D_i]$, i.e., $\exists t$, $\mathsf{RBF}(W, i, t) \leq \mathsf{iSBF}(\Gamma, t)$. ■

We present an example to check the schedulability of a real-time component under Q-CSF.

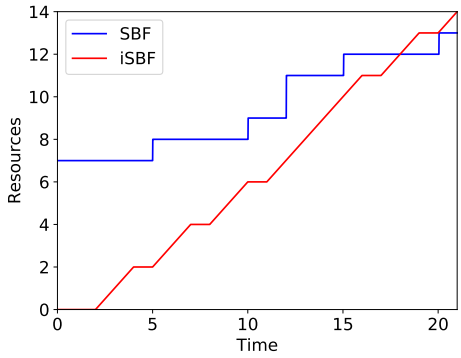
Example 5: Consider the workload W *in Example 1 and PRM* $\Gamma = (3, 2.25)$. *For* τ_1, *we have* $\mathsf{RBF}(W, 1, t) \leq \mathsf{iSBF}(\Gamma, t)$ *at* $t = 3 \leq D_1$, *which is 5. For* τ_2, *we have* $\mathsf{RBF}(W, 2, t) \leq \mathsf{iSBF}(\Gamma, t)$ *at* $t = 7 \leq D_2$, *which is 12. For* τ_3, *we have* $\mathsf{RBF}(W, 3, t) \leq \mathsf{iSBF}(\Gamma, t)$ *at* $t = 17 \leq D_3$, *which is 17. Then, Equation* (6) *of Theorem 1 holds. Figure 6 shows the schedulability of* τ_2 *and* τ_3.

Although we can check schedulability of a real-time component with a given resource model and a given component workload, we do not know how to find the resource-efficient resource model yet. Our next step is to find the resource-efficient PRM for a given component workload under Q-CSF, which is called *abstraction* process. Theorem 1 checks the schedulability of a component when workload W and PRM Γ are given. Utilizing Equation (6) of Theorem 1, we present Algorithm 1 (an abstraction algorithm for Q-CSF), which computes the minimum-bandwidth schedulable PRM when workload W is given.

At a high-level view, Algorithm 1 computes the minimum Θ_i to schedule the workload W for each task τ_i, and it assigns the maximum of Θ_i to Θ to schedule all tasks. In Lines 1–9, we consider each task τ_i in workload W. In Lines 2–7,



(a) The schedulability of τ_2



(b) The schedulability of τ_3

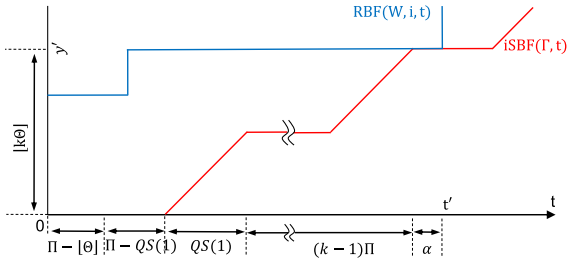
FIGURE 6. **The schedulability of τ_2 and τ_3 in Example 5.**

Algorithm 1 Abstraction Algorithm for Q-CSF

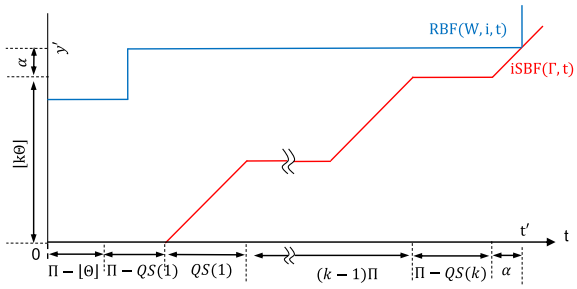
Require: W(component workload), Π (interface period)
Ensure: the minimum-bandwidth schedulable PRM Γ.

1: **for** $i \leftarrow 1$ to $|W|$ **do**
2: **for** $t' \leftarrow 1$ to D_i **do**
3: $y' = \mathsf{RBF}(W, i, t')$
4: Compute Θ'_1 satisfying Equation (7)
5: Compute Θ'_2 satisfying Equation (8)
6: $\Theta_i^{t'} = \min(\Theta'_1, \Theta'_2)$
7: **end for**
8: $\Theta_i = \min_{t'}(\Theta_i^{t'})$
9: **end for**
10: $\Theta = \max_i(\Theta_i)$
11: $\Gamma = (\Pi, \Theta)$

we consider each time interval t' that is less than or equal to the deadline of the task D_i. In Line 3, we compute the maximum requested resource demand y' for task τ_i and time t' by the RBF function (Equation (1)). In Lines 4–6, we compute the minimum budget for the given τ_i and t', which is denoted by $\Theta_i^{t'}$. To compute $\Theta_i^{t'}$, we consider two cases depending on the relation of RBF and iSBF. Because RBF is a step function and iSBF is a piecewise linear function, there exist two different cases for the minimum budget $\Theta_i^{t'}$ that satisfy $iSBF(\Gamma, t') \geq RBF(W, i, t')$, as shown in Equation (6) of Theorem 1. In the first case, the requested resource demand



(a) Case 1: y' is divisible by $\Theta_i^{t'}$



(b) Case 2: y' is not divisible by $\Theta_i^{t'}$

FIGURE 7. Two cases for computing budgets in the abstraction algorithm (Algorithm 1).

y' is divisible by $\Theta_i^{t'}$. In Line 4, we compute the budget for case 1, denoted by Θ'_1, as follows:

$$t' = \Pi - \lfloor\Theta\rfloor + k\Pi + \alpha,$$
$$y' = \lfloor k\Theta\rfloor \tag{7}$$

which is illustrated in Figure 7(a). In the second case, the requested resource demand y' is not divisible by $\Theta_i^{t'}$.

In Line 5, we compute the budget for case 2, denoted by Θ'_2, as follows:

$$t' = \Pi - \lfloor\Theta\rfloor + k\Pi + \Pi - QS(k) + \alpha,$$
$$y' = \lfloor k\Theta\rfloor + \alpha \tag{8}$$

which is illustrated in Figure 7(b). In Line 6, we set $\Theta_i^{t'}$ to the minimum of Θ'_1 and Θ'_2 (computed in Lines 4–5) because either Θ'_1 or Θ'_2 can satisfy $\mathsf{iSBF}(\Gamma, t') \geq \mathsf{RBF}(W, i, t')$. In Line 8, we set Θ_i to the minimum value among $\Theta_i^{t'}$ for all t' because we only need any $\Theta_i^{t'} \leq T_i$ that satisfies $\mathsf{iSBF}(\Gamma, t') \geq \mathsf{RBF}(W, i, t')$, as shown in Equation (6). In Line 10, we set Θ to the maximum value among Θ_i for all τ_i because Θ must satisfy $\exists t \leq D_i, \mathsf{RBF}(W, i, t) \leq \mathsf{iSBF}(\Gamma, t)$ for all τ_i, as shown in Equation (6). In Line 11, we return the minimum-bandwidth schedulable $\Gamma = (\Pi, \Theta)$ with the Θ that is calculated through the algorithm.

We present the complexity of Algorithm 1. In Algorithm 1, we need to keep the value of Θ_i for each task. Therefore, the space complexity is $O(n)$ where n is the number of tasks. Next, we consider the time complexity of Algorithm 1: the number of for-loop in Line 1 is n and the number of for-loop in Line 2 is up to $\max(D_i)$. Therefore, the time complexity is pseudo-polynomial, $O(nd)$ where $d = \max(D_i)$.

IV. EVALUATION

We evaluate our framework inside a single component (Section IV-B). We also evaluate our framework for an entire hierarchical systems having multiple components (Section IV-C). We also provide the detailed simulation results with numerical data, which is available online.[7]

A. SIMULATION SETUP

We compare Q-CSF with two versions of the original CSF [3], [7]:

- Q-CSF: our proposed approach under quantumized platforms.
- CSF [3], [7]: an original CSF approach with the real-number resource model, which cannot be used in quantumized platforms.
- CSF(ceil): CSF approach with the ceiling value of the real number resource model, which is a naive extension of the original CSF for quantumized platforms.

For evaluation metrics, we consider the utilization of resource model and the interface overhead in the component-level simulation. The utilization of resource model can be calculated by ... The interface overhead can be calculated by the difference between the utilization of resource model and the utilization of component workload: $U_w - U_\Gamma$. In system-level simulation, we consider the acceptance ratio for evaluation metrics.

We describe simulation environment. We use a desktop computer with Intel i9 CPU with 32GB memory. Simulation code is written in Java language (simulation data generation and shedulability analysis) and Python language (drawing simulation result graph with python Matplotlib library[8]). Our simulation workloads (task sets) is generated according to widely-used workload generation techniques [13], [16]. For a workload, we vary its utilization bound (U_b) from 0.3 to 0.7 in steps of 0.05, which results in 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, and 0.7. For a task,

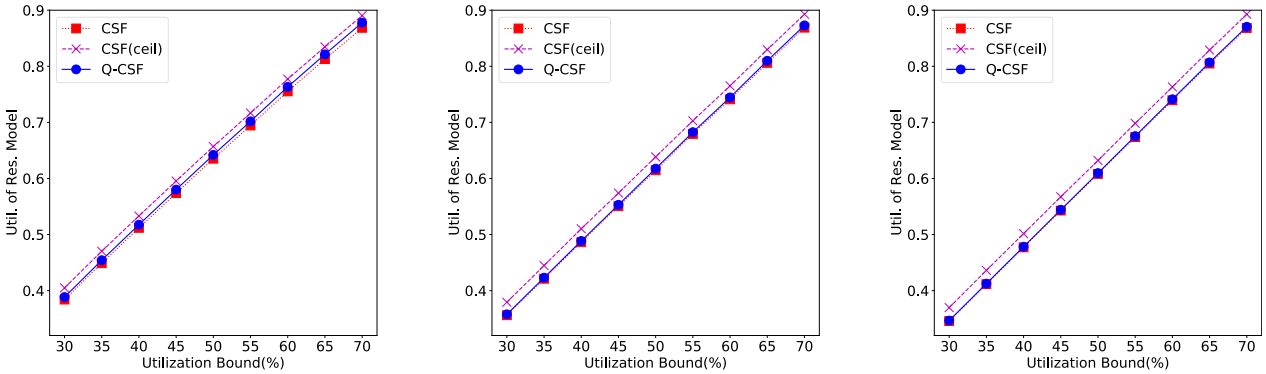
- Task period (T_i) is an integer that is uniformly drawn from [50, 300] unless specified.
- Task deadline (D_i) is an integer that is uniformly drawn from $[0.8 * T_i, T_i]$.
- Task utilization (u_i) is a real number that is uniformly drawn from [0.002, 0.1] unless specified.
- Task execution time (C_i) is calculated based on task period and task utilization: $C_i = T_i * u_i$.

We repeat the task generation until the utilization of the workload is larger than U_b. Then, we discard the task added last.

B. COMPONENT-LEVEL SIMULATION

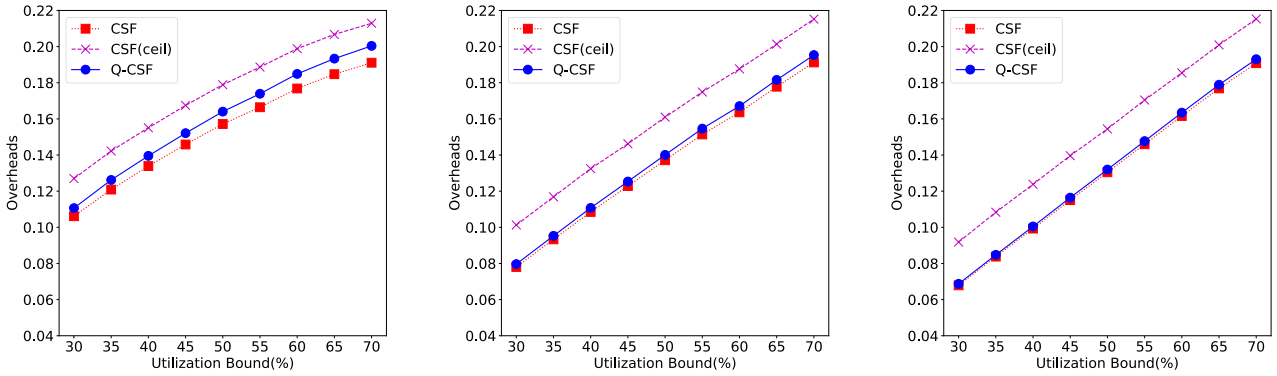
We evaluate the resource efficiency of Q-CSF in a component with a random workload (its generation is described in Section IV-A).

[7]URL: https://icpslab.github.io/qcsf_exp.xlsx
[8]URL: https://matplotlib.org



(a) Small task periods (b) Medium task periods (c) Large task periods

FIGURE 8. Utilization of resource model for different ranges of task period.



(a) Small task periods (b) Medium task periods (c) Large task periods

FIGURE 9. Interface overheads for different ranges of task period.

1) IMPACT OF TASK PERIOD

We evaluate Q-CSF for different task period ranges, which are periods uniformly drawn from [50, 200] (called *small task periods*), periods uniformly drawn from [200, 500] (called *medium task periods*), and periods uniformly drawn from [500, 1000] (called *large task periods*). The number of workloads for each data point is 3, 000 workloads and the number of total workloads is 81, 000, which is computed as $3000 \times 9 \times 3$.

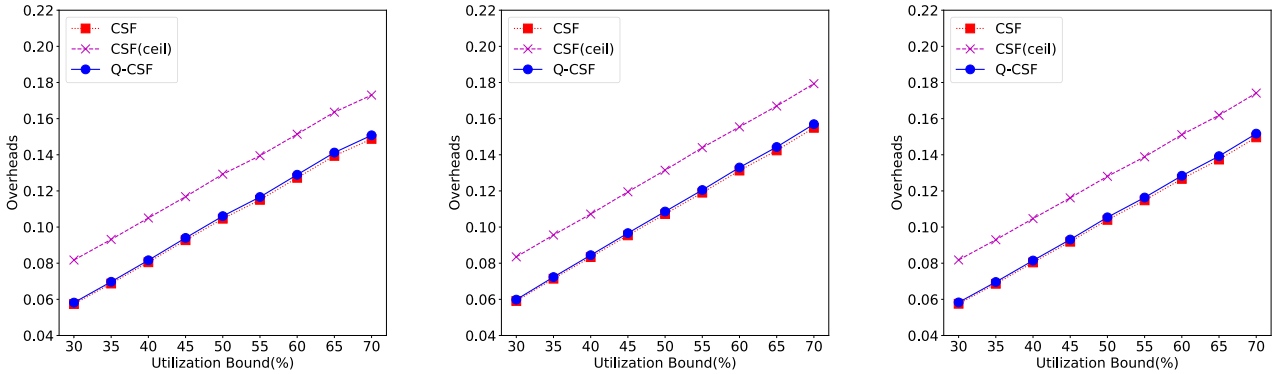
Figure 8 shows the average utilization of the resource model for the random component workloads varying the utilization bound U^b for different ranges of the task period. The required utilization of the CSF(ceil) is 6.92% larger than the CSF in the worst case. However, the required utilization of Q-CSF is only 1.18%, 0.48%, and 0.26% larger than the CSF in small, medium, and large task periods, respectively. For the current quantum-based hardware platform, the CSF that supplies a fractional unit of resources is not directly applicable. Hence, the Q-CSF supplying an integer unit of resources is more resource-efficient than the CSF(ceil).

To analyze the results of the simulation (Figure 8) in detail, we present Figure 9, which shows the average interface overhead. We observed that the interface overhead of CSF(ceil) is 35.25% larger than CSF in the worst case. However, the interface overhead of Q-CSF is only 4.87%, 2.15%, and 1.20% larger than CSF in small, medium, and large task periods, respectively. We observed that the interface overhead increases with the relatively smaller task periods. Q-CSF reduced the interface overhead of CSF(ceil) by up to 86.16%.

2) IMPACT OF TASK UTILIZATION

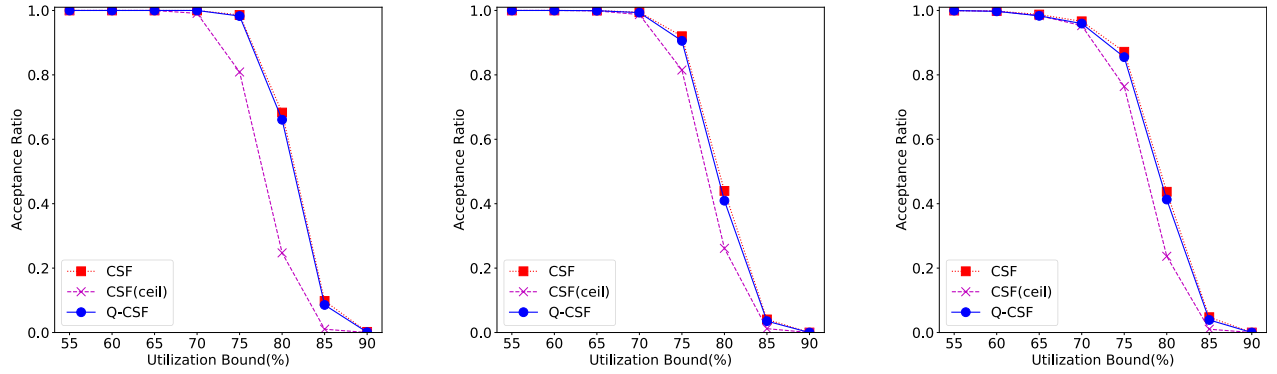
We evaluated the Q-CSF for different distribution of task utilization: one uniform distribution (uniformly over [0.002, 0.1]) and three bimodal distribution (uniformly over either [0.002, 0.05] or [0.05, 0.1]), with respective probabilities of 8/9 and 1/9 (light), 6/9 and 3/9 (medium), and 4/9 and 5/9 (heavy). The number of workloads for each data point was 3,000 and the number of total workloads is 108,000, which is computed as $3000 \times 7 \times 4$.

Figure 10 shows the average interface overheads of resource models for the random component workloads varying the utilization bound U^b for different distribution of task utilization. We observed that the interface overhead of CSF(ceil) is 42.42% larger than CSF in the worst case. However, the interface overhead of Q-CSF is only 1.43%, 1.22%, 1.33% and 1.45% larger than the CSF in uniform, bimodal



(a) Uniform distribution (b) Bimodal distribution (medium) (c) Bimodal distribution (heavy)

FIGURE 10. **Interface overheads for different distribution of task utilization.**



(a) $\Pi = 25$ (b) $\Pi = 50$ (c) $\Pi =$ random number in [25,75]

FIGURE 11. **Acceptance ratio for two-level hierarchical systems.**

(light), bimodal (medium), and bimodal (heavy) distributions, respectively. Regardless of distribution, Q-CSF shows similar overheads. Q-CSF reduced the interface overhead of CSF(ceil) by up to 96.58%.

C. SYSTEM-LEVEL SIMULATION

To evaluate the Q-CSF in hierarchical scheduling systems, we consider a two-level hierarchical scheduling system with multiple child components: the system schedules child components using the DM scheduler and each child component schedules its own workload (task set) using the DM scheduler. Similar to Section IV-B, we randomly generate workloads (two-level hierarchical scheduling systems). For each system, we vary its utilization bound (U_b) from 0.55 to 0.90 in steps of 0.05, which results in 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, and 0.90. For a component C_j,

- Component utilization (λ_j) is a real number uniformly drawn from $\left[0.05, \min(0.3, U_b - \sum \lambda_j)\right)$.
- We repeat the process to generate a task (using the default parameters from Section IV-B) until the utilization of the task set is larger than λ_j.
- Then, we discard the task added last.

We repeat the process to generate a component until the sum of the utilization of child components is greater than $U_b - 0.05$ (i.e., $\sum \lambda_j > U_b - 0.05$).

We evaluate the Q-CSF for different component periods: two identical periods, $\Pi = 25$ and $\Pi = 50$, and one random period, which is uniformly drawn from [25, 75]. The number of workloads for each data point is 3,000 and the number of total workloads is 72,000, which is computed as $3000 \times 8 \times 3$.

Figure 11 shows the average acceptance ratio of the resource model for the random systems varying the utilization bound U^b for different component periods. We observed that the difference of the average acceptance ratio between CSF and CSF(ceil) is 0.436 in the worst case. However, the average acceptance ratio of Q-CSF is close to CSF: the difference of the average acceptance ratio between CSF and Q-CSF is only 0.022, 0.030, and 0.024 in a small component period ($\Pi = 25$), a large component period ($\Pi = 50$), and a random component period, respectively. We observed that Q-CSF has up to 0.44, 0.18, and 0.20 better acceptance ratio than CSF(ceil) in a small component period, a large component period, and a random component period, respectively. For the larger component period, the difference of acceptance ratio among three approaches is small. However, for smaller

TABLE 2. Comparison between related work.

	Theory	Practicality	Description
Shin and Lee [3], [7]	✓		the very first work for component-based scheduling theory
Bini *et al.* [17]	✓		abstraction methods for different service mechanisms for multiprocessor platforms
Shin *et al.* [18]	✓		a method to minimize the utilization of clusters for multiprocessor cluster systems
Gu *et al.* [19]	✓		CSF extension to mixed-criticality systems
Yang *et al.* [20]	✓		a new notion of budget-based mixed-criticality scheduling
Easwaran *et al.* [15]	✓	✓	an explicit deadline periodic resource model
Kim *et al.* [21]	✓	✓	elimination of abstraction overhead
Chen *et al.* [22]	✓	✓	a new interface abstraction for both schedulability and associativity
RT-Xen [12]		✓	real-time Xen virtualization without theory support
Lee *et al.* [13]		✓	RT-Xen extension to support CSF theory
This Q-CSF	✓	✓	a new CSF theory considering quantum-based platforms, e.g., RT-Xen

component period, Q-CSF is much better than CSF(ceil) and close to the ideal CSF. For random component period, the trend is similar to the case of the smaller component period.

V. RELATED WORK

Beginning with the seminal work of Shin and Lee [3], [7], scheduling theory for the component-based real-time systems was extensively studied in various domains. Pertaining to multiprocessor platforms, Bini *et al.* [17] proposed abstraction methods for different types of service mechanisms such as periodic servers, static partitions, and the fluid-based time partitions. Additionally, Shin *et al.* [18] targeted cluster-based real-time systems and proposed a compositional scheduling algorithm that minimizes the utilization of individual clusters. Regarding mixed-criticality systems, Gu *et al.* [19] tried to overcome the shortcomings of the common assumption of mixed-criticality systems that all high-criticality tasks require additional computing resources simultaneously at a mode transition. To these ends, they proposed parameters to model the expected number of high-critical tasks simultaneously while demanding more resources and incorporating them into the compositional scheduling model for better resource usage. Yang *et al.* [20] introduced a new notion of budget-based mixed-criticality scheduling for component-based real-time systems.

For practicality, a few studies have focused on minimizing the abstraction overhead of component-based scheduling. Easwaran *et al.* [15] proposed an explicit deadline periodic resource model that applied the notion of a deadline to an existing periodic resource model to reduce the abstraction overhead. Furthermore, Kim *et al.* [21] attempted to eliminate abstraction overhead associated with the resource-supply and resource-demand. Chen *et al.* [22] identified some important properties of component-based real-time systems and proposed a new interface abstraction and composition framework to achieve both schedulability and associativity.

As system virtualization mechanisms have been developed [23], a several component-based scheduling mechanisms have been proposed to support virtualized real-time systems. RT-Xen [12] received a considerable attention, owing to its simplicity and efficiency in terms of system implementation and computing-resource utilization. The first version of RT-Xen did not support the component-based scheduling theory, and the study of Shin and Lee [3] was limited in that it did not consider a scheduling quantum. To overcome these shortcomings, Lee *et al.* [13] improved RT-Xen to consider the notion of time quantum while applying component-based scheduling theory. However, there exists the resource inefficiency in previous work in [13] due to rounding up the real-number of resource as quantum values. In this paper, we aim at relieving such limitations with a novel SBF, scheduling analysis, and component interface calculation methods for real digitalized hardware platforms. In Table 2, we compare our approach with other related work.

VI. CONCLUSION

The compositional scheduling framework is an effective paradigm to supports real time properties in component-based embedded systems. However, it is not directly applicable to real quantumized hardware platforms due to lack of consideration on quantumized resouce allocation. In this study, we proposed quantum-aware compositional scheduling framework, Q-CSF, to address the issue of time quantum in real digitalized hardware platforms. We present integer-based supply bound function (iSBF) to allocate resource for quantumized platforms. We also present schedulability analysis and interface computation algorithm under Q-CSF. In simulation results, we showed that Q-CSF has only a 4.8% overhead in component-level experiments and an up to 0.026 smaller acceptance ratio in system-level experiments, compared with theoretical CSF approaches.

In future work, we would like to investigate time quantum issue on multiprocessor component-based systems and mixed-criticality component-based systems.

REFERENCES

[1] R. I. Davis, T. Feld, V. Pollex, and F. Slomka, "Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling," in *Proc. IEEE 19th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2014, pp. 51–62.

[2] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Syst.*, vol. 26, no. 2, pp. 161–197, 2004.

[3] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proc. Int. Symp. Syst. Chip*, Dec. 2003, pp. 2–13.
[4] V. Nelis, J. Goossens, and B. Andersson, "Two protocols for scheduling multi-mode real-time systems upon identical multiprocessor platforms," in *Proc. 21st Euromicro Conf. Real-Time Syst.*, Jul. 2009, pp. 151–160.
[5] V. Nelis, B. Andersson, J. Marinho, and S. M. Petters, "Global-EDF scheduling of multimode real-time systems considering mode independent tasks," in *Proc. 23rd Euromicro Conf. Real-Time Syst.*, Jul. 2011, pp. 205–214.
[6] P. Rattanatamrong and J. A. B. Fortes, "Mode transition for online scheduling of adaptive real-time systems on multiprocessors," in *Proc. IEEE 17th Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2011, pp. 25–32.
[7] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, pp. 1–39, Apr. 2008, doi: 10.1145/1347375.1347383.
[8] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid Metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, Nov. 2011.
[9] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Comput. Oper. Res.*, vol. 40, pp. 3045–3055, Dec. 2013.
[10] C. Cheng, J. Li, and Y. Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," *Tsinghua Sci. Technol.*, vol. 20, no. 1, pp. 28–39, Feb. 2015.
[11] M. B. Gawali and K. S. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, pp. 1–16, 2018.
[12] S. Xi, J. Wilson, C. Lu, and C. Gill, "RT-Xen: Towards real-time hypervisor scheduling in Xen," in *Proc. ACM Int. Conf. Embedded Softw. (EMSOFT)*, 2011, pp. 39–48.
[13] J. Lee, S. Xi, S. Chen, L. T. X. Phan, C. Gill, I. Lee, C. Lu, and O. Sokolsky, "Realizing compositional scheduling through virtualization," in *Proc. IEEE 18th Real Time Embedded Technol. Appl. Symp.*, Apr. 2012, pp. 13–22.
[14] J. W. S. Liu, *Real-Time Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000. [Online]. Available: http://www.amazon.com/Real-Time-Systems-Jane-W-Liu/dp/0130996513
[15] 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.
[16] B. Andersson, K. Bletsas, and S. Baruah, "Scheduling arbitrary-deadline sporadic task systems on multiprocessors," in *Proc. Real-Time Syst. Symp.*, Nov. 2008, pp. 385–394.
[17] E. Bini, G. Buttazzo, and M. Bertogna, "The multi supply function abstraction for multiprocessors," in *Proc. 15th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2009, pp. 294–302.
[18] 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.
[19] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin, "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *Proc. 27th Euromicro Conf. Real-Time Syst.*, Jul. 2015, pp. 13–24.
[20] K. Yang and Z. Dong, "Mixed-criticality scheduling in compositional real-time systems with multiple budget estimates," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2020, pp. 25–37.
[21] J. H. Kim, K. H. Kim, A. Easwaran, and I. Lee, "Towards overhead-free interface theory for compositional hierarchical real-time systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2869–2880, Nov. 2018.
[22] 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. IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2011, pp. 81–90.
[23] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Dec. 2003, doi: 10.1145/1165389.945462.

JAEWOO LEE (Member, IEEE) received the B.S. and M.S. degrees in computer science and engineering from Seoul National University, Republic of Korea, in 2006 and 2008, respectively, and the Ph.D. degree in computer and information science from the University of Pennsylvania, USA, in 2017. He is currently an Assistant Professor with Chung-Ang University, Republic of Korea, where he joined, in 2018. He has been a Postdoctoral Research Fellow with Seoul National University, from 2017 to 2018. His research interests include cyber-physical systems, real-time embedded systems, and information system security.

HYEONGBOO BAEK received the B.S. degree in computer science and engineering from Konkuk University, South Korea, in 2010, and the M.S. and Ph.D. degrees in computer science from KAIST, South Korea, in 2012 and 2016, respectively. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Incheon National University (INU), South Korea. He was a Senior Researcher at Agency for Defense Development (ADD), from 2018 to 2019. His research interests include cyber-physical systems, real time embedded systems, and system security. He won the Best Paper Award from the 33rd IEEE Real Time Systems Symposium (RTSS), in 2012.