Regularity-based Partitioning of Uniform Resources in Real-time Systems *

Yu Li, Albert M. K. Cheng , and Aloysius K. Mok†

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-12-02

April 22, 2012

Keywords: Resource Partitioning, Multiresource, Multiprocessor, Schedulability Bound, Regularity Bound, Real-time Scheduling

Abstract

Hierarchical scheduling is a hot topic in real-time systems. In a hierarchical real-time system, the resource partition is the intermediate level between physical resources and real-time tasks. A resource partition operates on the shared physical resources at a fraction of the rate, and serves as a scheduling interface to the lower-level real-time tasks or resource partitions. Thus a key problem is how to define this scheduling interface on resource partitions. Regularity-bounded methodology is one important type of resource partitioning algorithms. This paper extends Mok and Feng's Regularity-based Resource Partition Model from a single-resource platform to a uniform multiresource platform. We present a resource partitioning algorithm called AAF-Multi Scheduling for solving the time slice overlap problem on a multiresource platform without violating the schedulability bound given by Feng on a single-resource platform. AAF-Multi is a global scheduling algorithm with $O(\Omega \cdot \log \Omega)$ time complexity (Ω = resource amount $\times$ hyper period).

*Supported in part by the National Science Foundation under Award No.0720856.
†Aloysius K. Mok is with University of Texas at Austin.

Regularity-based Partitioning of Uniform Resources in Real-time Systems *

Yu Li, Albert M. K. Cheng , and Aloysius K. Mok†

Abstract

Hierarchical scheduling is a hot topic in real-time systems. In a hierarchical real-time system, the resource partition is the intermediate level between physical resources and real-time tasks. A resource partition operates on the shared physical resources at a fraction of the rate, and serves as a scheduling interface to the lower-level real-time tasks or resource partitions. Thus a key problem is how to define this scheduling interface on resource partitions. Regularity-bounded methodology is one important type of resource partitioning algorithms. This paper extends Mok and Feng's Regularity-based Resource Partition Model from a single-resource platform to a uniform multiresource platform. We present a resource partitioning algorithm called AAF-Multi Scheduling for solving the time slice overlap problem on a multiresource platform without violating the schedulability bound given by Feng on a single-resource platform. AAF-Multi is a global scheduling algorithm with $O(\Omega \cdot \log \Omega)$ time complexity ($\Omega =$ resource amount $\times$ hyper period).

Keywords

Resource Partitioning, Multiresource, Multiprocessor, Schedulability Bound, Regularity Bound, Real-time Scheduling

I. Introduction

In a resource partition model [2], the resource-level scheduler divides the shared resources into temporal partitions with predefined constraints and requirements. Each task runs on the partition it belongs to, and each partition schedules the tasks on it with a task-level scheduler. To avoid interactions with tasks in other applications, tasks pertaining to a specific application are assigned to the same partition.

Furthermore, resource partitioning is a very important aspect of virtualization. There are two different ways of assigning hardware resources to virtual machines. One is fixing virtual machines on dedicated resources, and the other is scheduling virtual machines on shared resources. The former is easier and sometimes more efficient since there is no intervention from the virtual machine monitor. Nonetheless, the latter can provide higher resource utilization though it needs to implement a more complicated mechanism in the virtual machine monitor.

The resource partitioning problem is much more difficult in real-time systems than in general-purpose systems because not only the fairness of the resource allocation, but also the real-time characteristic of the resource partitions needs to be considered. The regularity-based resource partition model [1] characterizes the rate variation of resource partitions from both the temporal and supply dimensions, and can maintain the utilization bounds of both fixed and dynamic priority scheduling algorithms on regular partitions as if they are running on dedicated resources. Feng [3] presented the utilization bounds of both task-level scheduling and resource-level scheduling for the more general case - irregular partitions, as well as the corresponding scheduling algorithms. However, he only investigated resource-level scheduling on a single-resource platform. The overlap problem appears when the same resource-level scheduling algorithm is applied to a multiresource platform (details in Sect. II). In this paper, we attempt to solve the problem of resource partitioning of a multiresource-based platform with the following premises:

A. All resources are uniform.

B. There is no additional overhead when a resource partition is re-assigned from one resource to another.

Actually we shall discuss methodologies for reducing the overall overhead of partition migration between resources in Sect. V if the migration overhead in premise *B.* is considered.

*Supported in part by the National Science Foundation under Award No.0720856.

†Aloysius K. Mok is with University of Texas at Austin.

The rest of this paper is organized as follows. We overview the related work on hierarchical scheduling in Sect. II. Then in Sect. III we review definitions and results of the Regularity-Based Partition Model in [1,3] relevant to our work. Sect. IV defines Binary Partition and two kinds of time slice assignment specifications on it. Our resource partitioning algorithm for multi-resource is introduced in Sect. V. At last, we draw the conclusion in Sect. VI.

II. Related work

Shigero et al. [8] originated the concept of regularity, and showed that subsets of time slots called regular (same as regular partition) can be fully utilized. However, they only presented algorithms that generate two regular subsets. Mok and Feng [1,2,3] introduced the Regularity-based Resource Partition Model as a more systematic work. They introduced irregular partition as a more general case than regular partition, and they make general observations about the schedulabilty bound results both on regular and irregular partitions. Furthermore, they investigated the resource partitioning problem on a single resource, and gave a resulting algorithm based on the notion of Adjusted Available Factor. Since their algorithm does not work for a multiresource platform, we introduce a new methodology for that in this paper.

There are several other hierarchical resource models besides the Regular-based Model, such as the Bounded-Deday Model [10], the Periodic Model[11] and the EDP Model[12]. A number of these models have been extended to multiprocessor scenarios. Lipari and Bini [9] applied the Bounded-Delay Model to a multiprocessor platform by using a partitioned scheduling strategy, which does not allow resource partitions to migrate between processors. However, to the best of our knowledge, no one has ever extended the Regularity-based Model to multiprocessor scenarios.

Although our work is for resource partition scheduling, it shares a lot of similarities with multiprocessor real-time task scheduling if resource partitions are viewed as real-time tasks. However, multiprocessor real-time task scheduling algorithms such as [6, 7] do not suit us because they are only concerned with the deadline and period (sometimes they are the same) requirements of real-time tasks, while our partitioning algorithm considers the regularity bounds of partitions for temporal protection at the inside level. One exception is the pfair approach by Baruah et al. [4, 5]. The best strength of pfair is that it can provide full utilization for periodic tasks scheduled on a multiprocessor system. Pfair constrains the task latency (equals minus of instant regularity) in (-1, 1), which can only ensure that the supply regularity of each task (partition) is 2. Thus the main reason preventing us from using pfair is that pfair cannot generalize regular partitions.

III. Regularity-Based Partition Model

Since our work is an extension of the Regularity-Based Partition Model for task groups with periodic tasks, we first review some concepts introduced in [1,3] to make our presentation self-contained. We only focus on scheduling algorithms at the resource level, and skip those at the task level because our paper is devoted to exploring resource-level partitioning algorithms on a multi-resource platform.

A real-time virtual resource [1] operates at a fraction of the shared physical resources with a varying but bounded rate. A resource partition is a description of how to assign time intervals periodically on physical resources to a real-time virtual resource.

Definition 2.1 *A resource partition* Π *is a tuple* (Γ, P)*, where* Γ *is an array of* N *time pairs* $\{(S_1, E_1), (S_2, E_2), ..., (S_N, E_N)\}$ *that satisfies* $(0 \leq S_1 < E_1 < S_2 < E2 < ... < S_N < E_N \leq P)$ *for some* $N \geq 1$*, and* P *is the partition period. The physical resource is available to a task group executing on this partition only during time intervals* $(S_i + j \times P,\ E_i + j \times P)$*,* $1 \leq i \leq N,\ j \geq 0$*.*

Definition 2.2 *The availability factor of a resource partition* Π *is* $\alpha(\Pi) = (\sum_{i=1}^{n}(E_i - S_i))/P$*.*

Definition 2.3 *The Supply Function* $S(t)$ *of a partition* Π *is the total amount of time that is available in* Π *from time 0 to t.*

The Regularity-based Resource Partition Model studies resource partitioning in the integer domain because resource partitions in this domain have interesting properties. A major restriction is that the partitions can only switch from one to another at integer time instances. The following definitions describes Regular Partition and Irregular Partition by bounding Supply Regularity, which are the most important concepts in [3].

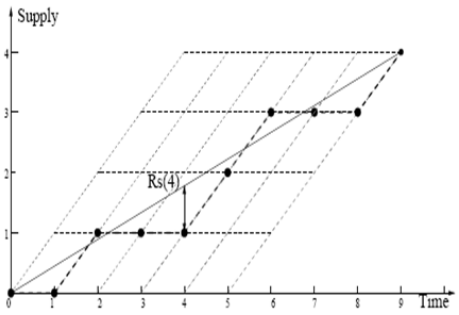

Fig. 1. [1] Instant Regularity at Time 4

Definition 2.4 *The Instant Regularity $Ir(t)$ at time t on partition Π is given by $S(t) - t \cdot \alpha(\Pi)$.*
Definition 2.5 *Let a, b, k be non-negative integers, the Supply Regularity $R_s(\Pi)$ of Partition Π is equal to the minimum value of k such that $\forall a, b$, $|Ir(b) - Ir(a)| < k$.*

Definition 2.6 *A Regular Partition is a partition with supply regularity of 1.*

Definition 2.7 *A k Supply-Irregular Partition is a partition with supply regularity of k, where $k > 1$.*

Regular Partitions and Irregular Partitions, especially the former, can provide wonderful real-time properties. We shall not mention them here since we only care about the resource partitioning problem in this paper. Next, we review the algorithms pertaining to how to schedule Regular Partitions and Irregular Partitions. The main idea is to convert all resource partitions into a group of regular partitions with availability factors of the power of one-half whose schedulability can be easily checked on a single-resource platform.

Lemma 2.1 *When k regular partitions are combined together, they form a partition with supply regularity of k.*

Definition 2.8 *Given a Partition Π with availability factor α and supply regularity Rs, the Adjusted Availability Factor $AAF(\alpha,\ Rs)$ is the total of the availability factors of partitions that are used to compose Π.*

The algorithm for calculating AAF can be found in [3]. Here we just give some examples.
$AAF(0.67,\ 2) = 0.5 + 0.25 = 0.75$,
$AAF(0.67,\ 3) = 0.5 + 0.125 + 0.0625 = 0.6875$,
$AAF(0.75,\ 2) = AAF(0.75,\ 3) = 0.5 + 0.25 = 0.75$.

Theorem 2.1 *Given $\{(\alpha_i, k_i) :\ i{=}1, 2, ..., n\}$ containing the availability factor and supply regularity of irregular partitions P_i $(i{=}1, 2, ..., n)$, they are schedulable on a single resource if $\sum_{i=1}^{n} AAF(\alpha_i, k_i) \leq 1$.*

Algorithm – AAF-Single

```
1.    for i := 1 to n do
2.        A_i := AAF(α_i, k_i);
3.    end for;
4.    finished := 0;
5.    l := 1;
6.    while (finished < n) do
7.        w := 1 / 2^l;
8.        for i := 1 to n do
9.            if (A_i ≥ w) then
10.               assign time-slices to P_i at level-l;
11.               A_i := A_i − w;
12.               if (A_i = 0) finished++;
13.           end if;
14.       end for;
15.       l++;
```

16. *end while*;

Figure 2 shows an example of the outcome from Algorithm AAF-Single for $\{(0.375,\ 2),\ (0.3125,\ 2),\ (0.3125,\ 2)\}$.

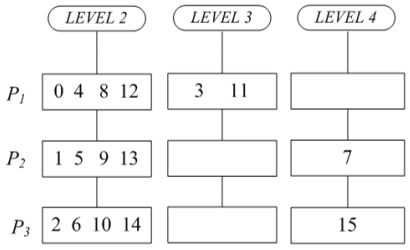

Fig. 2. Scheduling Irregular Partitions on Single Resource

Next we present a new theorem which extends the schedulability condition given in Theorem 2.1 for a single-resource platform to one for a multiresource platform.

Theorem 2.2 *Given* $\{(\alpha_i,\ k_i):\ i{=}1, 2, ..., n\}$ *containing the availability factor and supply regularity of irregular partitions* P_i $(i{=}1, 2, ..., n)$*, they are schedulable on* m *resources if* $\sum_{i=1}^{n} AAF(\alpha_i, k_i) \leq m$.

Our proof of Theorem 2.2 will be presented in Section IV. Here we briefly explain why the schedulability condition in Theorem 2.2 cannot be checked by Algorithm AAF-Single. Consider the example in Figure 3, where $\{(0.75,\ 2),\ (0.625,\ 2),\ (0.625,\ 2)\}$ are AAF and supply regularity of P_1, P_2, and P_3. They cannot be scheduled by Algorithm AAF-Single because an overlap occurs.

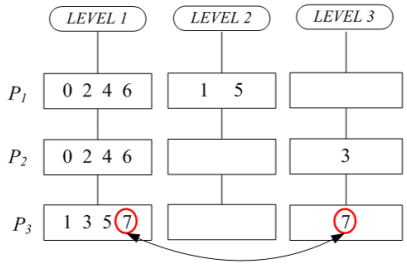

Fig. 3. Scheduling Irregular Partitions on Multiresource

IV. Binary Partition and Binary Partition Instance

In this section, we shall extend some definitions in Section II for convenience of the coming discussions. We introduce the concept of binary partition and binary partition instance to describe in more detail the structure of regular and irregular partitions.

A. Binary Partition

In Sect. II, Feng schedules the irregular partitions after their available factors are transformed to AAF. We introduce the concept of Binary Partition to describe partitions after such transformations.

Definition 3.1 *A Binary Division (BD)* D *is defined as* (l)*, where* l *is a positive integer and denotes the level of* D.

The utilization of D is defined as $U(D) = 1/2^l$.

Definition 3.2 *A Binary Partition (BP)* P *is defined as* (Π, k)*. Non-negative integer* k *denotes the regularity of* P*, and* Π *is a positive integer set* $\{l_i : 0 < i \leq k, 0 < l_1 < l_2 < ... < l_k\}$*.* l_k *denotes the depth of* P.

We say P has level-l division if $l \in \Pi$. The utilization of P is defined as $U(P) = \sum_{i=1}^{k}(1/2^{l_i}) < 1$.

We say P is single while $k = 1$, is combo while $k > 1$, and is empty while $k = 0$.

Definition 3.3 *A Characteristic String of binary partition P is a $'(0|1)*'$ string, whose i-th bit is $'1'$ if and only if P has level-i division. $Cs(P)$ represents the characteristic string of P.*

A binary partition can be represented by its characteristic string for convenience. For example: $'101'$ represents the binary partition $(\{1, 3\}, 2)$.

Like binary numbers, we can define $+$, $-$, $>>$ and $<<$ operations on binary partitions.

Definition 3.4 *For binary partitions $P,\ P_1,\ P_2$, $P = P_1 + P_2$ if and if only $U(P) = U(P_1) + U(P_2)$, and $P = P_1 - P_2$ if and if only $U(P) = U(P_1) - U(P_2)$.*

A Characteristic String can help calculate the sum and difference of binary partitions more easily. For example, $Cs(P_1)$ = $'101'$, $Cs(P_1)$ = $'0011'$, then $Cs(P_1 + P_2)$ = $'1101'$ (like $0.101 + 0.0011 = 0.1101$).

Definition 3.5 *Given a binary partition P=$(\Pi,\ k)$, P>>n represents binary partition $(\{l + n :\ l \in \Pi\},\ k)$, and P<<n represents binary partition $(\{l - n :\ l \in \Pi\},\ k)$ if $\forall l \in \Pi, l > n$.*

For example, if $Cs(P)$ = $'0101'$, $Cs(P<<1)$ = $'101'$, and $Cs(P>>1)$ = $'00101'$.

Definition 3.6 *Given a binary partition $P = (\Sigma,\ k)$ and a positive integer X,*
$Head(P, X)$ represents binary partition $(\Sigma',\ k')$ with $\Sigma' = \{l :\ l \in \Sigma\ and\ l \leq X\}$,
$Tail(P, X)$ represents binary partition $(\Sigma'',\ k'')$ with $\Sigma'' = \{l :\ l \in \Sigma\ and\ l > X\}$.

B. Binary Partition Instance

A Binary Partition Instance(BPI) describes a periodic assignment of time slices to a binary partition. More specifically, a binary partition instance assigns time slices to each level division of a binary partition with a period of 2^L. We say a BPI is valid if there is no overlap among those assigned time slices; otherwise, it is invalid.

Definition 3.7 *A Binary Division Instance (BDI) γ is defined as (D, l, δ). D is a level-l_0 binary division. l is an integer not less than l_0. δ is an integer set containing 2^{l-l_0} time integers in $[0, 2^l)$.*

We say γ is on D. l is the period factor of γ. δ is the (time slice) assigned set of γ.

Definition 3.8 *For a binary partition $P = (\Sigma, k)$, Binary Partition Instance (BPI) Γ is defined as (P, L, Δ). L is an integer not less than the depth of P. Δ is a set of integer sets $\{\delta_{l_i} :\ l_i \in \Sigma\}$ such that $((l_i),\ L,\ \delta_{l_i})$ is a BDI for each $l_i \in \Sigma$.*

We say Γ is on P. L is the period factor of Γ. Integer multiset $\cup\Delta$ is the (time slice) assigned set of Γ. $\gamma_i = ((l_i),\ L,\ \delta_{l_i})$ is the level-l_i BDI of Γ, and Γ is composed of γ_i $(i = 1, 2, ..., k)$.

Γ is valid if and only if $\cap\Delta = \phi$.

Next we define some operations on BDI and BPI, such that we can describe the later proofs more concisely.

Definition 3.9 *Given an integer set δ and a non-negative integers n, l, define*

$Sadd(\delta, n, l) = \{(i + n) \mod 2^l :\ i \in \delta\}$.

Definition 3.10 *Given a BDI $\gamma = (D,\ l,\ \delta)$ and a non-negative integer d, the Scale-out Function is defined as*

$Scale(\gamma, d) = (D,\ l + d,\ \bigcup_{i=0}^{2^d-1} Sadd(\delta,\ i \times 2^l,\ l + d))$.

Definition 3.11 *Given a BPI $\Gamma = (P,\ L,\ \Delta)$ and a non-negative integer d, suppose Γ_i is composed of γ_i $(i = 1, 2, ..., k)$, and $Scale(\gamma_i, d) = (D_i,\ L + d,\ \delta_i)$. The Scale-out Function is defined as*

$Scale(\Gamma, d) = (P,\ L + d,\ \{\delta_i : i = 1, 2, ...k\})$.

Definition 3.12 *Given BPIs $(\Gamma_1 = (\Sigma_1, k_1),\ L_1,\ \Delta_1)$, $(\Gamma_2 = (\Sigma_2, k_2),\ L_2,\ \Delta_2)$ where $\Sigma_1 \bigcap \Sigma_2 = \phi$, suppose*

$L_{max} = \max\{L_1,\ L_2\}$,
$Scale(\Gamma_1, L_{max} - L_1) = ((\Sigma_1, k_1),\ L_{max},\ \Delta_1')$,
$Scale(\Gamma_2, L_{max} - L_2) = ((\Sigma_2, k_2),\ L_{max},\ \Delta_2')$.

Define $\oplus$ operation of two BPIs as

$\Gamma_1 \oplus \Gamma_2 = ((\Sigma_1 \cup \Sigma_2,\ k_1 + k_2),\ L_{max},\ \Delta'_1 \cup \Delta'_2)$.

NOTE: BDI $((l_0),\ l,\ \delta)$ *is automatically transformed into BPI* $((\{l_0\},\ 1),\ l, \{\delta\})$ *prior to* $\oplus$ *operations.*

At the end of this section, we introduce the instant regularity for BDI and valid BPI. For a given BDI $\gamma = (D = (l_0), l, \delta)$, the availability factor $\alpha = U(D) = 1/2^{l_0}$, and the supply function $S(t)$ can be defined as:

$$S(t+1) = \begin{cases} S(t) + 1 & if\ (t\ mod\ 2^l) \in \delta; \\ S(t) & otherwise. \end{cases}$$

Similarly, for a given valid BPI $\Gamma = (P, L, \Delta)$, the availability factor $\alpha = U(P)$, and the supply function $S(t)$ can be defined as:

$$S(t+1) = \begin{cases} S(t) + 1 & if\ (t\ mod\ 2^L) \in \cup\Delta; \\ S(t) & otherwise. \end{cases}$$

For both BDI and BPI, the instant regularity $Ir(t)$ at time t is given by $S(t) - t \cdot \alpha$. The definition of instant regularity is also inherited from the regularity-based partition model (Definition 2.4). The next two subsections will introduce two special types of BPI whose instant regularity can be easily confined.

C. Regular Binary Partition Instance

Regular Binary Partition Instance(Regular BPI) is extended from regular partition defined in Section II. Before presenting the formal definition, we first introduce Regular Binary Division Instance(Regular BDI). If all the time slices assigned to a BDI consist of an arithmetic progression, then it is called a regular BDI.

Definition 3.13 *A BDI* $r{=}((l_0), l, \delta)$ *is regular if* $\exists \rho \in [0, 2^{l_0}), \delta = \{\rho + i * 2^{l_0} : i{=}0, 1, 2, ..., 2^{l-l_0}-1\}$. ρ *is called the offset of regular BDI* r.

For example, given a regular BDI $r{=}((1), 3, \delta)$, if the offset of r is 1, δ is $\{1, 3, 5, 7\}$.

Lemma 3.1 *The Supply regularity of any regular BDI is* 1.

Proof: The instant regularity of r has minimal values at time instants just before r is assigned time slices, and has maximal values at time instants just after r is assigned time slices. Suppose r is on (l_0), and ρ is the offset of r, then $\forall a, \forall b, |Ir(r,b) - Ir(r,a)| \leq Ir(r, \rho+1) - Ir(r,\rho) = 1 - (1/2^{l_0})$. ■

Lemma 3.2 *If a BDI* r *is regular,* $Scale(r,\ d)$ *is regular for any positive integer* d.

Lemma 3.3 *Given a regular BDI* $r{=}((l_0),\ l,\ \delta)$, *and* $l' = \max\{l,\ l_0 + 1\}$, *there exist regular BDIs* $r_1{=}((l_0 + 1),\ l',\ \delta_1)$, $r_2{=}((l_0+1),\ l',\ \delta_2)$, *such that* $\delta_1 \cup \delta_2 = \delta'$ (δ' *is the assigned set of* $Scale(r,\ l'-l)$). *Specifically, we say* $(r_1,\ r_2)$ *is a split of* r.

Proof: Suppose the offset of r is ρ. We only need to let the offset of r_1 be ρ, and let the offset of r_2 be $\rho + 2^{l_0}$. ■

Definition 3.14 *A BPI is regular if its each level BDI is regular.*

For example, BPI $((\{1, 3\}, 2), 3, \{\{1, 3, 5, 7\}, \{2\}\})$ is regular.

Lemma 3.4 *if a regular BPI* $R{=}((\Sigma, k),\ L,\ \Delta)$ *is valid, the supply regularity of* R *is* k.

Proof: (Referring Lemma 3.1) Suppose regular BPI R is composed of $r_i{=}(D_i, l_i, \delta_i)$, $i = 1, 2, ..., k$. Then $\forall a, \forall b$, $|Ir(R,b) - Ir(R,a)| = |\sum_{i=1}^{k}(Ir(r_i,b) - Ir(r_i,a))| \leq \sum_{i=1}^{k} |Ir(r_i,b) - Ir(r_i,a)| < k$. ■

Lemma 3.5 *If a BPI* R *is regular,* $Scale(R,\ d)$ *is regular for any positive integer* d.

Similar to that a binary partition can be easily split, a regular BPI can be also split into two or more regular BPIs. Lemma 3.6 proves the case of splitting one BPI into two, and Theorem 3.1 presents the more general cases.

Lemma 3.6 *Given a regular BPI* $R{=}(P,\ L,\ \Delta)$, *and binary partitions* P_1, P_2 *where* $P_1 + P_2 = P$, *there exist regular BPIs* $R_1{=}(P_1,\ L',\ \Delta_1)$ *and* $R_2{=}(P_2,\ L',\ \Delta_2)$, *such that* $L' = \max\{L,\ depth\ of\ P_1,\ depth\ of\ P_2\}$ *and* $(\cup\Delta_1) \cup (\cup\Delta_2) = \cup\Delta'$ (Δ' *is the assigned set of* $Scale(R,\ L'-L)$). *Specifically, We say* $(R_1,\ R_2)$ *is a split of* R *on* $(P_1,\ P_2)$.

Proof: i) When k=0, P is an empty BP, so P_1 and P_2 are both empty BPs. Thus $((P_1,\ L,\ \phi),(P_2,\ L,\ \phi))$ is a split of R on $(P_1,\ P_2)$.

ii) Suppose when $k < K$ (K is a positive integer), regular BPI can always be split. When $k = K$, suppose the first $'1'$ in Cs(P) occurs at the x-th bit from left . Since $Cs(P_1) + Cs(P_2) = Cs(P)$, there are two cases on the x-th bits of $Cs(P_1)$ and $Cs(P_2)$.

Case 1: one x-th bit is $'1'$, the other is $'0'$. Suppose the x-th bit of $Cs(P_1)$ is $'1'$. Change the x-th bit of $Cs(P)$ from $'1'$ to $'0'$, and get a new BP P'. And get another new BP P_1' by the same change to P_1. Suppose r=$((x),\ L,\ \delta_x)$ is the level x regular BDI of R, let R'=$(P',\ L,\ \Delta - \{\delta_x\})$. Since P'=P_1'+P_2, and the regularity of P' is $K-1$, by the induction assumption, there exists $((P_1',\ L',\ \Delta_1'),(P_2,\ L',\ \Delta_2))$ which is a split of P' on $(P_1',\ P_2)$. Then $((P_1',\ L',\ \Delta_1') \oplus r,\ (P_2,\ L',\ \Delta_2))$ is a split of R on (P_1, P_2).

Case 2: Both the x-th bits are $'0'$. Since $Cs(P_1) + Cs(P_2) = Cs(P)$, at least one of the x+1-th bits is $'1'$. There are two cases on the $(x+1)$-th bits.

Case 2.1: Both $(x+1)$-th bits are $'1'$. Change the x-th bit of $CS(P)$ from $'1'$ to $'0'$, and get a new BP P'. Change the $(x+1)$-th bits of $CS(P_1)$ and $CS(P_2)$ from $'1'$ to $'0'$, and get two new BP P_1' and P_2'. Suppose r=$((x),\ L,\ \delta_x)$ is the level x regular BDI of R, let R'=$(P',\ L,\ \Delta - \{\delta_x\})$. Since P'=P_1'+P_2', and the regularity of P' is $K-1$, by the induction assumption, there exists $((P_1',\ L',\ \Delta_1'),(P_2',\ L',\ \Delta_2'))$ which is a split of R' on $(P_1',\ P_2')$. By Lemma 3.3, $\exists(r_1,\ r_2)$ which is a split of r. Then $((P_1',\ L',\ \Delta_1') \oplus r_1,\ (P_2',\ L',\ \Delta_2') \oplus r_2)$ is a split of R on $(P_1,\ P_2)$.

Case 2.2: Only one of $(x+1)$-th bits is $'1'$; suppose it is P_1's. Since $Cs(P_1) + Cs(P_2) = Cs(P)$, the $(x+1)$-th bit of $CS(P)$ must be $'0'$. Change the x-th bit of $CS(P)$ from $'1'$ to $'0'$, the $(x+1)$-th bit from $'0'$ to $'1'$, and get a new BP P'. Change the x-th bit of $CS(P_1)$ from $'1'$ to $'0'$, and get P_1'. Suppose r=$((x),\ L,\ \delta_x)$ is the level x regular BDI of R, and $(r_1,\ r_2)$ is a split of r. Let $R' = (P',\ L,\ \Delta - \{\delta_x\}) \oplus r_1$. Since P'=P_1'+P_2, if $\exists((P_1',\ L',\ \Delta_1'),(P_2,\ L',\ \Delta_2))$ is a split of R' on $(P_1',\ P_2)$, we can get that $((P_1',\ L',\ \Delta_1') \oplus r_2,\ (P_2,\ L',\ \Delta_2))$ is a split of R on $(P_1,\ P_2)$. So it only need R' be split. The regularity of R' is K, and the first $'1'$ in $CS(P')$ occurs at the $(x+1)$-th bit from left, so we can return step ii). The recursive procedure must end because of the finiteness of P, P_1 and P_2. ■

Theorem 3.1 *Given a regular BPI $R = (P,\ L,\ \Delta)$, and binary partitions P_i $(i$=$1, 2, ..., n)$ where $P = \sum_{i=1}^{n} P_i$, there exist regular BPIs R_i=$(P_i,\ L',\ \Delta_i)$ $(i$=$1, 2, ..., n)$, such that $L' = \max\{L,\ \max_{i=1}^{n}(depth\ of\ P_i)\}$ and $\bigcup_{i=1}^{n}(\cup\Delta_i) = \cup\Delta'$ (Δ' is the assigned set of $Scale(R,\ L'-L)$). Specifically, We say $(R_1, R_2, ..., R_n)$ is a split of R on $(P_1, P_2, ..., P_n)$.*

Proof: can be easily concluded from Lemma 3.6. ■

For example:

$R = (P,\ 4,\ \{\{0, 2, 4, 6, 8, 10, 12, 14\},\ \{1, 9\},\ \{3\}\})$, $Cs(P)$ = $'1011'$. Suppose $Cs(P_1)$ = $'011'$, $Cs(P_2)$ = $'0101'$, then

$R_1 = (P_1,\ 4,\ \{\{0, 4, 8, 12\},\ \{1, 9\}\})$,

$R_2 = (P_2,\ 4,\ \{\{2, 6, 10, 14\},\ \{3\}\})$.

Definition 3.15 *Right Shift Operation of regular BDI and regular BPI.*

Given a regular BDI r=$((l_0), l, \delta)$ with offset ρ, r>>n is a regular BDI $((l_0 + n), l + n, \delta')$ with offset $\rho \times 2^n$.

Given a regular BPI R=$(P,\ L,\ \Delta)$ composed of r_i $(i$=$1, 2, ..., k)$, R>>n is a regular BPI $(P$>>$n,\ L$+$n,\ \Delta')$ composed of r_i>>n $(i$=$1, 2, ..., k)$.

D. Sub-Regular Binary Partition Instance

The initial idea of Sub-Regular Binary Partition Instance (Sub-Regular BPI) came from the effort to avoid overlaps while scheduling irregular partitions on multiresource platforms (mentioned in Section II). Take a look at Figure 4. How good is the idea of swapping those overlapped time slices with other partitions?

Definition 3.16 *BDI s=$((l_0), l, \delta)$ is sub-regular if there exist regular BDI r=$((l_0), l, \delta_0)$ and integer set $\lambda \subset \delta_0$ where $\delta = \delta_0 - \lambda \cup Sadd(\lambda,\ 1,\ l)$.*

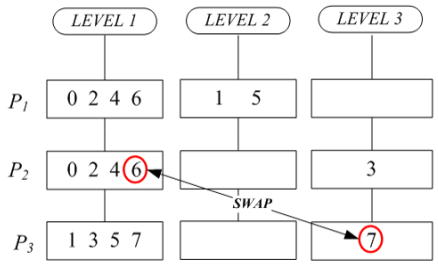

Fig. 4. Swapping Overlapped Time slices with Other Partitions

We say r is the origin of s, and s is shifted from r. λ is the (time slice) shifted set of s. Time slice t is shifted on s if $(t \ mod \ 2^l) \ \in \lambda$.

Regular BDI is a special case of sub-regular BDI when the shifted set is empty.

For example: s is a sub-regular BDI shifted from $((1),\ 3,\ \{1,3,5,7\})$ with shifted set $\{1,5,7\}$, then $s = ((1),\ 3,\ \{2,3,6,0\})$.

Lemma 3.7 *Given a sub-regular BDI s,* $\forall a$, $\forall b$, $|Ir(s,b) - Ir(s,a)| \leq 1$.

Proof: Suppose s is shifted from $r = ((l_0),\ l,\ \delta)$. Then

$$Ir(s,\ t) = \begin{cases} Ir(r,\ t{-}1) - (1/2^{l_0}) & if\ t{-}1\ is\ shifted, \\ Ir(r,\ t) & otherwise. \end{cases}$$

So according to Lemma 3.1 and its proof,
$|Ir(s,b) - Ir(s,a)| \leq |Ir(r,b') - Ir(r,a')| + (1/2^{l_0}) \leq 1$
$(b' = b \ \ or \ \ b-1; \ \ a' = a \ \ or \ \ a-1)$. ■

Lemma 3.8 *If a BDI s is sub-regular,* $Scale(s,\ d)$ *is sub-regular for any positive integer d.*

Lemma 3.9 *Given a sub-regular BDI* $s = ((l_0),\ l,\ \delta)$, *and* $l' = \max\{l, l_0+1\}$, *there exist sub-regular BDIs* $s_1 = ((l_0+1),\ l',\ \delta_1)$, $s_2 = ((l_0+1),\ l',\ \delta_2)$, *such that* $\delta_1 \cup \delta_2 = \delta'$ (δ' *is the assigned set of* $Scale(s,\ l'-l)$). *Specifically, we say* (s_1, s_2) *is a split of s.*

Proof: Similar to Lemma 3.3. The only difference is that splitting sub-regular BDIs requires splitting their time slice shifted sets. ■

Definition 3.17 *A BPI is sub-regular if its each level BDI is sub-regular.*

Additionally, the origin of a sub-regular BPI is the regular BPI composed of the origin of its each level BDI.

For example, the origin of sub-regular BPI
$((\{1,2\},\ 2),\ 3,\ \{\{0,3,4,7\},\ \{1,6\}\})$ is
$((\{1,2\},\ 2),\ 3,\ \{\{0,2,4,6\},\ \{1,5\}\})$.

Lemma 3.10 *if a sub-regular BPI* $S{=}((\Sigma,k),\ L,\ \Delta)$ *is valid, and* $k > 1$, *the supply regularity of* S *is* k.

Proof: Suppose S is composed of sub-regular BDIs s_i $(i{=}1,2,...,k)$, then according to Lemma 3.7, $\forall a$, $\forall b$,

$|Ir(S,b) - Ir(S,a)| = |\sum_{i=1}^{k} Ir(s_i,b) - Ir(s_i,a)| \leq k$.

If $|Ir(S,b) - Ir(S,a)| = k$, each $Ir(s_i,b) - Ir(s_i,a)$ must be 1 (or each be -1). That is impossible because the instant regularity of s_i $(i{=}1,2,...,k)$ can not reach maximal (or minimal) values at the same time instant when S is valid. ■

Lemma 3.11 *If a BPI* S *is sub-regular,* $Scale(S,\ d)$ *is sub-regular for any positive integer d.*

Theorem 3.2 *Given a sub-regular BPI* $S{=}(P,\ L,\ \Delta)$, *and binary partitions* $P_i(i{=}1,2,...,n)$ *where* $P{=}\sum_{i=1}^{n} P_i$, *there exist sub-regular BPIs* $S_i{=}(P_i,\ L',\ \Delta_i)$ $(i{=}1,2,...,n)$, *such that* $L'{=}\max\{L,\ \max_{i=1}^{n}(depth\ of\ P_i)\}$ *and* $\bigcup_{i=1}^{n}(\cup\Delta_i) = \cup\Delta'$ (Δ' *is the assigned set of* $Scale(S,\ L'-L)$). *Specifically, we say* $(S_1, S_2, ..., S_n)$ *is a split of* R *on* $(P_1, P_2, ..., P_n)$.

Proof: Similar to the proof of Theorem 3.1. ■

Definition 3.18 *Right Shift Operation of sub-regular BDI and sub-regular BPI.*

Given a sub-regular BDI s shifted from r with shifted set λ, $s>>n$ is a sub-regular BDI shifted from $r>>n$ with shifted set $\{t \times 2^n : \ t \in \lambda\}$.

Given a sub-regular BPI $S=(P,\ L,\ \Delta)$ composed of s_i $(i=1, 2, ..., k)$, $S>>n$ is a sub-regular BPI $(P>>n,\ L+n,\ \Delta')$ composed of $s_i>>n$ $(i=1, 2, ..., k)$.

V. AAF-Multi Resource Partitioning Algorithm

As discussed in the previous section, a sub-regular BPI can guarantee that its supply regularity does not exceed the regularity of the BP it is on, except in the case where it is on a single BP (Lemma 3.10, 3.7). We shall determine how to schedule sub-regular BPIs on a multiresource platform while avoiding the problem on single BPs.

Definition 4.1 *$(S_1, S_2, ..., S_n)$ is called a Sub-Regular Binary Partition Scheduling (AAF-Multi Scheduling) of $(P_1, P_2, ..., P_n)$ on m resources if S_i $(i=1, 2, ..., n)$ is a valid sub-regular BPI on binary partition P_i, and each time slice is assigned at most m times to $S_1, S_2, ..., S_n$.*

A. Inductive Algorithm of AAF-Multi Scheduling

An inductive algorithm of AAF-Multi Scheduling is presented in this subsection, which can conclude Theorem 2.2 given in Sect. II.

Lemma 4.1 *Given an array of binary partitions $P_1, P_2, ..., P_n$ where $\sum_{i=1}^{n} U(P_i) = m$. L is the maximal depth of P_i $(i=1, 2, ..., n)$, and τ is an arbitrary integer in $[1, n]$. There exists AAF-Multi Scheduling of $(P_1, P_2, ..., P_n)$ on m resources - $(S_1, S_2, ..., S_n)$, such that*

1) S_τ is regular.

2) L is the period factor of S_i $(i=1, 2, ..., n)$.

Proof: i) When $m=1$, such AAF-Multi Scheduling exists(see Theorem 2.1 and Algorithm AAF-Single).

ii) Suppose When $m<M$ $(M>1)$, such AAF-Multi Scheduling always exists. Consider $m=M$.

First merge the given BPs like this: each time select two BPs with the lowest utilization, and if the sum of their utilization is less than 1, replace these two BPs by their sum in the array. The merging process stops when the utilization sum of any two BPs in the array is ≥ 1.

After merging, we get a new BPs array $P'_1, P'_2, ..., P'_{n'}$. suppose P_τ is merged into $P'_{\tau'}$, and L' $(L' \leq L)$ is the maximal depth of P'_i $(i=1, 2, ..., n')$. Notice that if we can find a AAF-Multi Scheduling of $(P'_1, P'_2, ..., P'_{n'})$ on M resources and $P'_{\tau'}$ gets a regular assignment, the resulting AAF-Multi Scheduling of $(P_1, P_2, ..., P_n)$ can be easily obtained through splitting and scale-out (see Theorem 3.1, 3.2, and Lemma 3.5, 3.11).

Suppose the two BPs having the lowest and second lowest unitization in $(P'_1, P'_2, ..., P'_{n'})$ are P'_α and P'_β.

Case 1: If $U(P'_\alpha) + U(P'_\beta) = 1$, schedule P'_α and P'_β on one of the given resources, and schedule the remaining BPs on the other $M-1$ resources. According to the induction assumption, such SRP scheduling exists.

Case 2: If $U(P'_\alpha) + U(P'_\beta) > 1$. Suppose there are N time BPs in $P'_1, P'_2, ..., P'_{n'}$ having utilization $> 1/2$. Because $\sum_{i=1}^{n'} U(P'_i) = M$ and the sum of any two BPs in $P'_1, P'_2, ..., P'_{n'}$ is great than 1, N must be in $[M, 2M)$. On the other hand, there is at most one BP having utilization $\leq 1/2$, and suppose it is P'_{min} (P'_{min} could be empty).

Case 2.1: if $N \in (M, 2M)$, separate $P'_1, P'_2, ..., P'_{n'}$ into three groups. Group A contains $M-1$ time BPs $(A_1, A_2, ..., A_{M-1})$ having utilization $> 1/2$ and excluding $P'_{\tau'}$, and Group B contains $N-M$ time BPs $(B_1, B_2, ..., B_{N-M})$ having utilization $> 1/2$ and excluding $P'_{\tau'}$. Group C includes the remaining two BPs (C_1, $C_2 = P'_{min}$). Notice that $P'_{\tau'}$ is in group C.

As shown in Figure 5, assign time slices $1, 3, 5, ...$ to the level-1 BDs of C_1 (getting BDI s_1) and $A_1, A_2, ..., A_{M-1}$ (getting BDIs $s_2, s_3, ..., s_M$ correspondingly), and assign time slices $0, 2, 4, ...$ to the level-1 BDs of $B_1, B_2, ..., B_{N-M}$ (getting BDIs $s_{M+1}, s_{M+2}, ..., s_N$ correspondingly).

$Q_1 = Tail(C_1,\ 1)$,

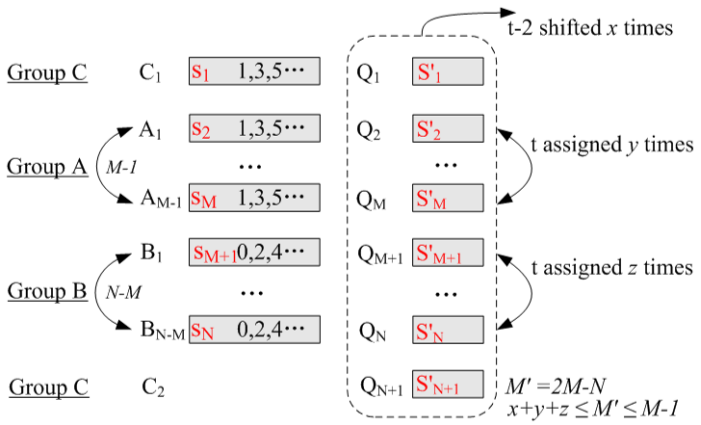

Fig. 5. Assigning after Merging (Lemma4.1 Case2.1)

$Q_{1+i} = Tail(A_i,\ 1), \quad i = 1, 2, ..., M-1$
$Q_{M+i} = Tail(B_i,\ 1), \quad i = 1, 2, ..., N-M$
$Q_{N+1} = C_2.$

Let $Q'_i = Q_i{<<}1$ $(i{=}1, 2, ...N{+}1)$. Then

$\sum_{i=1}^{N+1} Q'_i = 2\sum_{i=1}^{N+1} Q_i = 2(M - (N/2)) = 2M{-}N.$

Let $M'{=}2M{-}N$, $M'{\in}(0,\ M)$. According to the induction assumption, ∃AAF-Multi Scheduling of $(Q'_1, Q'_2, ..., Q'_{N+1})$ on M' resources - $(S''_1, S''_2, ..., S''_{N+1})$, such that

1) S''_1 is regular if $P'_{\tau'} = C_1$, otherwise S''_{N+1} is regular.
2) All period factors are $L'{-}1$. (L' is max depth of $Q_1, Q_2, ..., Q_{N+1}$).

This AAF-Multi Scheduling can help construct the target AAF-Multi Scheduling on $(C_1, A'_1, ..., A'_{M-1}, B'_1, ..., B'_{N-M}, C_2)$ through the following two steps. (Figure 6 can help understand the next two paragraphs)

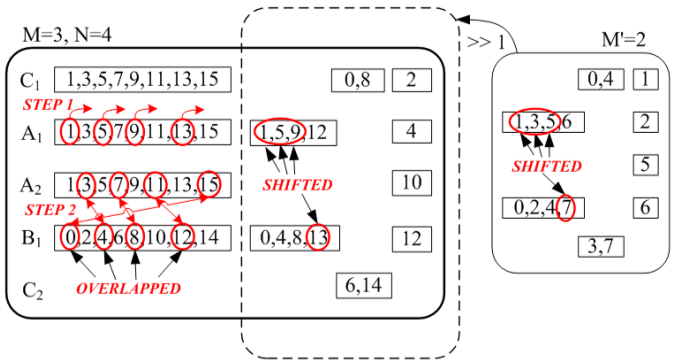

Fig. 6. Scheduling Example (Lemma4.1 Case2.1)

Step 1 - Balance Shifted Time slices

Let $S'_i = S''_i{>>}1$ $(i{=}\ 1, 2, ..., N{+}1)$. Because each time slice in $\{0, 1, 2, 3...\}$ is assigned to $S''_1, S''_2, ..., S''_{N+1}$ M' times, each times slice $\{0, 2, 4, 6...\}$ should be assigned to $S'_1, S'_2, ..., S'_{N+1}$ M' times if there was no shifted time slice. Consider those shifted time slices. ∀ an even integer $t \in [0, 2^{L'})$, suppose time slice $T(t{-}2)$ $(T(t) = t\ mod\ 2^{L'})$ is shifted x times in $S'_1, S'_2, ..., S'_n$. Until now in all BPIs $S'_1 \oplus s_1, S'_2 \oplus s_2, ..., S'_N \oplus s_N, S'_{N+1}$, t is assigned $M{+}x$ times and $T(t{-}1)$ is assigned $M{-}x$ times, which is unbalanced. We can solve this problem by replacing time slice $T(t{-}1)$ by time slice t x times in $s_2, s_3, ..., s_M$. Suppose t is assigned to $S'_2, S'_3, ..., S'_M$ y times, and

assigned to $S'_{M+1}, S'_{M+2}, ..., S'_N$ z times, then $x+y+z \leq M' \leq M-1$ because $t/2$ is assigned to $S''_1, S''_2, ..., S''_{N+1}$ M' times. So there are at least $x+z$ time slice $T(t-1)$ in $s_2, s_3, ..., s_M$ which could be replaced by t. Choose x of them, and perform the replacements (if $T(t-2)$ is shifted in $S'_j (j \in [2, M])$, choose s_j first. Thus the overlaps between s_j and S'_j can be removed). After balancing the shifted time slices for all even integers $t \in [0, 2^{L'})$, each time slice in $[0, 2^{L'})$ should be assigned to $S'_1 \oplus s_1, S'_2 \oplus s_2, ..., S'_N \oplus s_N, S'_{N+1}$ M times. Furthermore, we only changed $s_2, s_3, ..., s_M$ from regular to sub-regular, and did not touch the other parts.

Step 2 - Remove Overlaps

Another problem we need to solve is the overlaps between $s_{M+1}, s_{M+2}, ..., s_N$ and $S'_{M+1}, S'_{M+2}, ..., S'_N$. As mentioned in Step 1, time slice t is assigned to $S'_{M+1}, S'_{M+2}, ..., S'_N$ z times. Unfortunately, all of them are overlapped with $s_{M+1}, s_{M+2}, ..., s_N$. Our strategy is to exchange those overlapped time slice t in $s_{M+1}, s_{M+2}, ..., s_N$ with time slice $T(t-1)$ in $s_2, s_3, ..., s_M$ (according to the analysis in step 1, there are still at least z time slice $T(t-1)$ in $s_2, s_3, ..., s_M$ which could be replaced by t). This solution can keep $s_{M+1}, s_{M+2}, ..., s_N$ sub-regular and valid because time slice t and $T(t-1)$ cannot be assigned to the same S'_j $(j \in [M+1, N])$ simultaneously (because $t/2$ can only be assigned to S''_j at most once). So after performing all of the exchanges, $S'_1 \oplus s_1, S'_2 \oplus s_2, ..., S'_N \oplus s_N, S'_{N+1}$ become valid.

$(S'_1 \oplus s_1, S'_2 \oplus s_2, ..., S'_N \oplus s_N, S'_{N+1})$ is the target AAF-Multi Scheduling of $(C_1, A'_1, ..., A'_{M-1}, B'_1, ..., B'_{N-M}, C_2)$. Let's check the two conditions it must satisfy.

1) It can guarantee $P'_{\tau'}$ gets a regular BPI (We never changed the assigned sets of s_1, S'_1 and S'_{N+1}).

2) The common period factor is L'.

Ignoring the order, $C_1, A'_1, ..., A'_{M-1}, B'_1, ..., B'_{N-M}, C_2$ is same as $P'_1, P'_2, ..., P'_{n'}$. So we have constructed the AAF-Multi Scheduling of $(P'_1, P'_2, ..., P'_{n'})$ on M resources. As mentioned before, the resulting AAF-Multi Scheduling of $(P_1, P_2, ..., P_n)$ can be constructed through splitting and scale-out.

Case 2.2: if $N = M$. There must be $M+1$ time BPs in $P'_1, P'_2, ..., P'_{n'}$, so $n' = M+1$. In this case, only one BP has utilization $< 1/2$, and each of others has utilization $> 1/2$.

Case 2.2.1: if $U(P'_{\tau'}) > 1/2$. Suppose X is the minimum level at which $P'_1, P'_2, ..., P'_{M+1}$ have $M+1$ time BDs . Without loss of generality, suppose $P'_{\tau'} = P'_1$, and $U(P'_{M+1}) < 1/2$.

Case 2.2.1.1: if $U(P'_M) < 2^{-(X-1)}$, P'_M starts from level X. As shown in Figure 7, assign time- slices $(1, 3, 5...) \times 2^{(i-1)}$ to each BD of $P'_1, P'_2, ..., P'_M$ at level i for $i = 1, 2, ..., X-1$ (getting BDI $s_{i,j}$ at level i of P'_j). Assign time-slices $(1, 3, 5...) \times 2^{(X-1)}$ to $P'_2, P'_3, ..., P'_{M+1}$ at level X (getting BDI $s_{X,(j \bmod M)}$ in P'_j), and assign time-slices $(0, 2, 4...) \times 2^{(X-1)}$ to P'_1 at level X (getting BDI s_0).

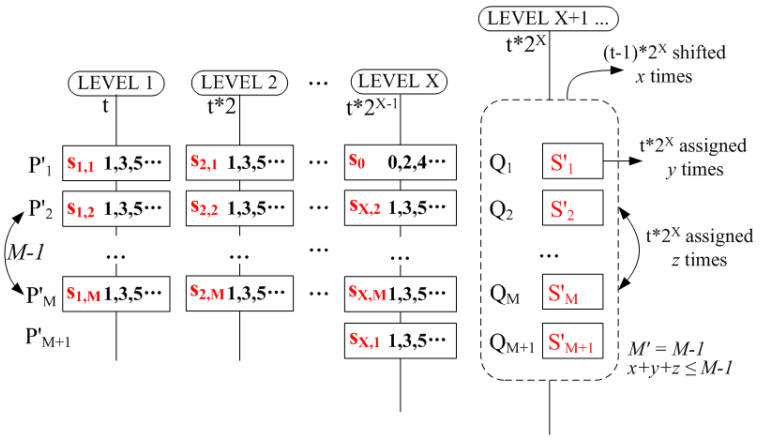

Fig. 7. Assigning after Merging (Lemma4.1 Case2.2.1.1)

Let Q_i=$Tail(P_i,\ X)$, Q'_i=Q_i<<X, i=$1, 2, ..., M+1$, then $\sum_{i=1}^{M+1} U(Q'_i) = M-1$. According to the induction assumption, $\exists$SRBP scheduling of $(Q'_1, Q'_2, ..., Q'_{M+1})$ on $M-1$ resources - $(S''_1, S''_2, ..., S''_{N+1})$, such that

1) S_1'' is regular.
2) All period factors are $L'-X$. (L' is max depth of $Q_1, Q_2, ..., Q_{M+1}$).

Let $S_i'=S_i''>>X$, $i=1, 2, ..., M+1$. The following processing is very similar to that in Case 2.1. We need to balance the shifted time-slices in $S_1', S_2', ..., S_{M+1}'$, and remove overlaps between S_1' and s_0. For conciseness, we may skip some details later in this case.

Step 1 - Balance Shifted Time-slices

$\forall\ t \cdot 2^X \in [0,\ 2^{L'})$. Suppose time-slice $T((t-1) \cdot 2^X)$ ($T(t) = t\ mod\ 2^{L'}$) is shifted in $S_1', S_2', ..., S_{M+1}'$ x times. Time-slice $t \cdot 2^X$ is assigned to S_1' y times (y is 0 or 1), and assigned to $S_2', S_3', ..., S_M'$ z times. Then $x+y+z \leq M-1$. Similar to Case 2.1, our balancing strategy is to replace time-slice $T((t-1) \cdot 2^X+ 1)$ by $t \cdot 2^X$ in $s_{i,j}$ ($i=1, 2, ..., X$, $j=2, 3, ..., M$) x times. Meanwhile, we need to keep $s_{i,j}$ sub-regular. We can achieve such one replacement in $s_{1,\psi}, s_{2,\psi}, ..., s_{X,\psi}$ by Algorithm 4.1.

Algorithm 4.1

```
1.    l := 1;
2.    d := 1;
3.    Y := X;
4.    t' := (t · 2^X − 1) mod 2^L';
5.    for i := 1 to (2^X − 1) do
6.        shift time-slice t' in s_{l,ψ};
7.        t'−−;
8.        if (l = 1) then
9.            l = 1 + d++;
10.           if (d = Y) then
11.               d := 1;
12.               Y−−;
13.           fi;
14.       else
15.           l := 1;
16.       fi;
17.   od for;
```

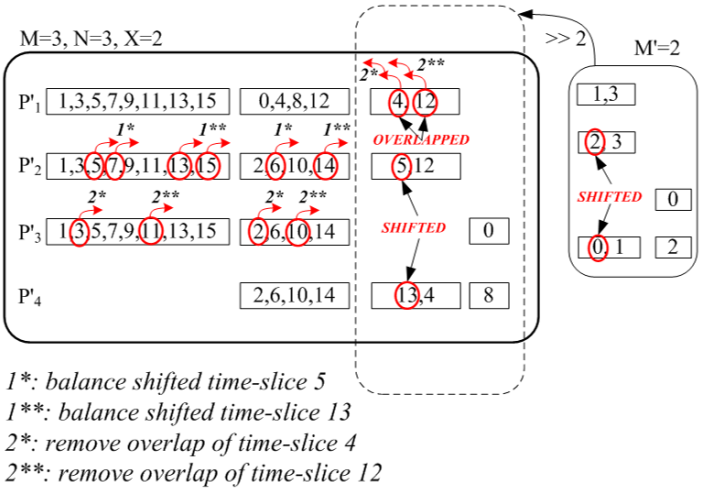

Fig. 8. Scheduling Example (Lemma4.1 Case2.2.1.1)

Step 2 - Remove Overlaps

There exist overlaps only between S_1' and s_0. We can remove the overlaps by exchanging time-slice $t \cdot 2^X$ in S_1' with $T(t \cdot 2^X - 2^{X-1})$ in $s_{1,\psi}, s_{2,\psi}, ..., s_{X,\psi}$ by Algorithm 4.1.

Case 2.2.1.2: if $U(P'_M) \geq 2^{-(X-1)}$, P'_M starts before level X. For each BD of P'_M before level X, borrow it to the partition without BD at the same level. Then we can perform scheduling as in Case 2.2.1.1, and after that return all borrowed BDs to P'_M (notice that the returning will not bring out overlaps because the distance between any two BDIs on P'_M is greater than 1 if there is no shifted time-slice). A scheduling example is presented in Figure 9.

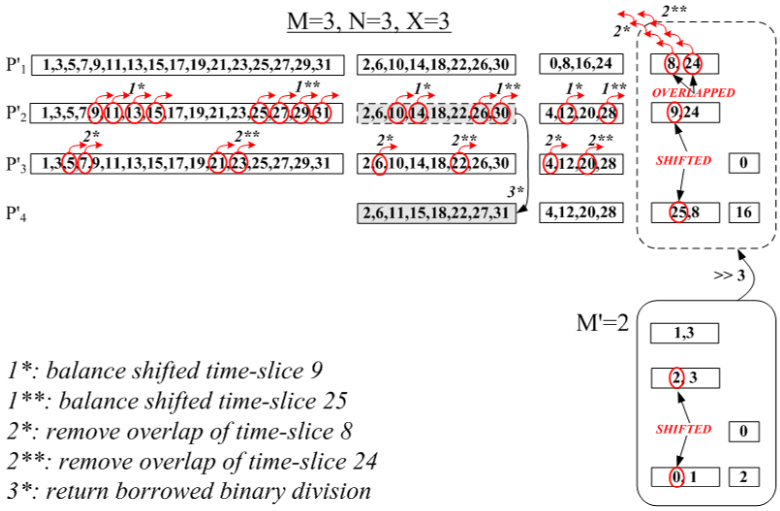

Fig. 9. Scheduling Example (Lemma4.1 Case2.2.1.2)

Case 2.2.2: if $U(P'_{\tau'}) < 1/2$. Suppose $X+1$ is the minimum level at which $P'_{\tau'}$ has BD. Without loss of generality, suppose $P'_{\tau'} = P'_1$. As shown in Figure 10, we assign time-slices $(1, 3, 5...) \times 2^{(i-1)}$ to each BD of $P'_2, P'_2, ..., P'_{M+1}$ at level-i for i=$1, 2, ..., X$.

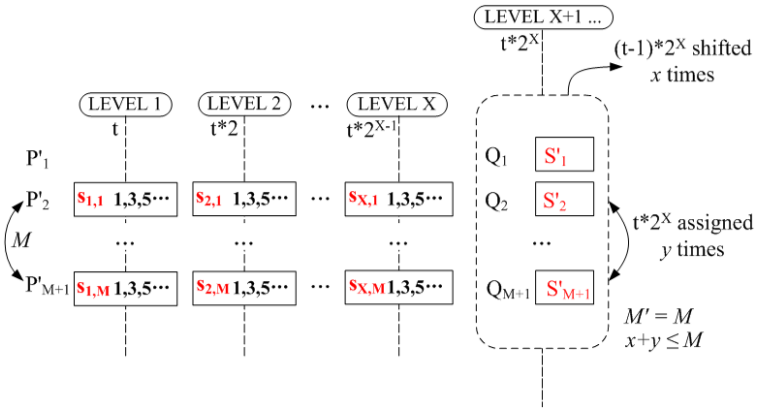

Fig. 10. Assigning after Merging (Lemma4.1 Case2.2.2)

Let Q_i=$Tail(P'_i,\ X)$, Q'_i=Q_i<<X $(i = 1, 2, ..., M+1)$. Then $\sum_{i=1}^{M+1} Q'_i = M$, and $U(Q'_1) \geq 1/2$. According to case 2.2.1 (if there are M time BDs on level $X+1$) or case 2.1 (if there are $M+1$ time BDs on level $X+1$), $\exists$ SRBP scheduling of $(Q'_2, Q'_2, ..., Q'_{M+1})$ on M resources - $(S''_2, S''_2, ..., S''_{M+1})$, such that

1) S''_1 is regular.

2) The common period factor is $L'-X$ (L' is the maximum depth of $P'_2, P'_2, ..., P'_{M+1}$).

Let S'_i=S''_i>>X $(i$=$1, 2, ..., M+1)$. We only need to balance the shifted time-slices in $S'_2, S'_3, ..., S'_{M+1}$. $\forall$ $t \cdot 2^X \in [0, 2^{L'})$, Suppose time-slice $T((t-1) \cdot 2^X)$ $(T(t) = t\ mod\ 2^{L'})$ is shifted x times in $S'_2, S'_3, ..., S'_{M+1}$, and $t \cdot 2^X$ is assigned to $S'_2, S'_3, ..., S'_{M+1}$ y times, then $x + y \leq M$. We can achieve balancing through Algorithm 4.1. ■

Lemma 4.2 *Given an array of single BPs $P_1, P_2, ..., P_n$ in order of non-increasing utilizations. If $\sum_{i=1}^{n} U(P_i) \geq 1$, there exists a positive integer $j \in (1, n]$, such that $\sum_{i=1}^{j} U(P_i) = 1$.*

Proof: If not so, there must exist a positive integer $k < n$ where $\sum_{i=1}^{k} U(P_i) < 1$, and $\sum_{i=1}^{k+1} U(P_i) > 1$. Let $S = \sum_{i=1}^{k} U(P_i)$, then S is a multiple of $U(P_k)$, because $U(P_1), U(P_2), ..., U(P_k)$ are all multiples of $U(P_k)$. And since 1 is a multiple of $U(P_k)$, $1-S$ (great than 0) is a multiple of $U(P_k)$, so $1-S$ must be a multiple of $U(P_{k+1})$. This means $1-S \geq U(P_{k+1})$, which contradicts $\sum_{i=1}^{k+1} U(P_i) = S + U(P_{k+1}) > 1$. ■

Theorem 4.1 *Given an array of binary partitions $P_1, P_2, ..., P_n$ where $\sum_{i=1}^{n} U(P_i) = m$. Suppose the regularity of P_i is k_i ($i = 0, 1, ..., n$). There exists AAF-Multi Scheduling of ($P_1, P_2, ..., P_n$) on m resources - ($S_1, S_2, ..., S_n$), such that the supply regularity of S_i is k_i ($i = 0, 1, ..., n$).*

Proof: The resulting AAF-Multi Scheduling must satisfy that each single BP gets a regular BPI instead of a sub-regular BPI because the supply regularity of a single sub-regular BPI could be 2 (see Lemma 3.7).

i) When m=1, such AAF-Multi Scheduling exists (see Theorem 2.1). Furthermore, all assigned BPIs are regular BPIs.

ii) Suppose such AAF-Multi Scheduling always exists when $m<M$, consider $m=M$. Suppose the BP array $Q_1, Q_2, ..., Q_h$ contains all single BPs in $P_1, P_2, ..., P_n$ in order of non-increasing untilizations. The BP array $C_1, C_2, ..., C_{n-h}$ contains the remaining BPs. Let $U = \sum_{i=1}^{h} Q_i$.

Case 1: if $U \geq 1$, there exists integer $j \in [1, h]$ where $\sum_{i=1}^{j} U(Q_i) = 1$ (according to Lemma 4.2). Single BPs $Q_1, Q_2, ..., Q_j$ can be scheduled on one of the given resources, each getting a regular BPI. And according to the induction assumption, the remaining BPs $Q_{j+1}, Q_{j+2}, ..., Q_h$ and $C_1, C_2, ..., C_{n-h}$ can be scheduled on the other $M-1$ resources.

Case 2: if $U<1$, let binary partition $Q = \sum_{i=1}^{h} Q_i$. According to Lemma 4.1, $\exists(S', S''_1, S''_2, ..., S''_{n-h})$ which is a AAF-Multi Scheduling of $(Q, C_1, C_2, ..., C_{n-h})$ on M resources, where S' is regular. Then according to Theorem 3.1, S' can be split into a regular BPI array $S'_1, S'_2, ..., S'_h$ on $Q_1, Q_2, ..., Q_h$ correspondingly. Therefore, $(S'_1, S'_2, ..., S'_h, S''_1, S''_2, ..., S''_{n-h})$ is the resulting AAF-Multi Scheduling of $(Q_1, Q_2, ..., Q_h, C_1, C_2, ..., C_{n-h})$. ■

Theorem 2.2 *Given $\{(\alpha_i,\ k_i) :\ i=1, 2, ..., n\}$ containing the availability factor and supply regularity of irregular partitions P_i (i=1, 2, ..., n), they are schedulable on m resources if $\sum_{i=1}^{n} AAF(\alpha_i, k_i) \leq m$.*

Proof: Irregular partition P_i corresponds to a binary partition $Q_i = (\Pi_i,\ k'_i)$ where $U(Q_i) = AAF(\alpha_i, k_i)$ and $k'_i \leq k_i$. According to Theorem 4.1, P_i (i=1, 2, ..., n) can be scheduled on m resources. ■

B. Time Complexity of AAF-Multi Scheduling

We now consider the time complexity of the inductive algorithm presented in Lemma 4.1. Suppose n BPs are scheduled on m resources, and the maximum depth of these BPs is l. This inductive algorithm finishes after k iterations. In the i-th iteration (i=1, 2, ..., k), the amount of resources is M_i, the amount of BPs is N_i, and their maximum depth is L_i. Then M_k=m, N_k=n, L_k=l. In each iteration, the time-consuming part focuses on three kinds of operations - merging BPs, splitting sub-regular BPIs, and adapting time slices according to the result of the previous iteration. The time complexity is simplified based on: $k \leq L$, $n \leq m \cdot 2^L$. $k \leq L$ is because L_i is strictly decreasing, and $n \leq m \cdot 2^L$ is because there are only $m \cdot 2^L$ time slices.

Merging - Obviously, the total amount of BP merging operations in all iterations is smaller than n, and the time complexity of each merging operation is $O(L)$. So the time complexity of merging is

$$T_1 = O(n \cdot L) = O(m \cdot L \cdot 2^L).$$

Splitting - Notice that the time consumption of splitting sub-regular BPIs stems primarily from that of splitting those involved time slice shifted sets. The amount of all time slices is $m \cdot 2^L$, and each time slice is involved in BD splitting at most L times, because it would be assigned to a lower level BD after splitting the BD owning it. So the overall time complexity of splitting is

$$T_2 = O(m \cdot L \cdot 2^L).$$

Adapting - In the i-th iteration, there are $M_{i-1} \cdot 2^{L_{i-1}}$ time slices coming from the previous iteration. For each time slice shifted or overlapped, the time complexity of adapting is $O(2^{L_i - L_{i-1}})$. So the overall time complexity of this part is

$$T_3 = O(\sum_{i=1}^{k}(M_{i-1} \cdot 2^{L_{i-1}} \cdot 2^{L_i - L_{i-1}})) = O(m \cdot 2^L).$$

In summary, the time complexity of AAF-Multi Scheduling is

$$T = T_1 + T_2 + T_3 = O(m \cdot L \cdot 2^L).$$

Let $\Omega = m \cdot 2^L$, Ω is the total amount of time slices in AAF-Multi Scheduling. The time complexity is also presented as

$$T(\Omega) = O(\Omega \cdot \log \Omega).$$

C. Reducing Partition-Migration Overhead

Since AAF-Multi is a global scheduling algorithm, the partition migrations between resources cannot be fully avoided. We mentioned in Sect. I that we could develop some methodologies to reduce the overall migration overhead.

Time Slices on Partitions

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C_1	x	x	x	x		x		x	x	x		x		x		x
A_1		x	x	x	x	x	x	x		x	x	x	x		x	x
A_2	x	x			x	x			x	x	x		x	x		
B_1	x		x	x	x		x	x	x		x	x	x	x	x	x
C_2							x								x	

Partitions on Resources (Migration happens for 18 times)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R_1	C_1	C_1	C_1	C_1	A_1	C_1	A_1	C_1	C_1	C_1	A_1	C_1	A_1	C_1	A_1	C_1
R_2	A_2	A_1	A_1	A_1	A_2	A_1	B_1	A_1	A_2	A_1	A_2	A_1	A_2	A_2	B_1	A_1
R_3	B_1	A_2	B_1	B_1	B_1	A_2	C_2	B_1	B_1	A_2	B_1	B_1	B_1	B_1	C_2	B_1

Fig. 11. Resource Allocation of the Case in Figure 6

Figure 7 shows two tables. The first one is the resulting AAF-Multi scheduling of the case in Figure 6. It can be observed from the table that at each time instant, three and only three partitions get time slices. The second table determines how the three resources R_1, R_2, R_3 choose the partitions on them at each time instant. A simple strategy is applied such that the partitions are picked by the fixed order from top to bottom in the first table. 18 times partition-migration happen in those cells with circles.

These partition-migrations can be grouped into two categories. A partition-migration is in Type-One category if it happens after a time slice that this partition just get executed. Otherwise, it is a Type-Two partition-migration. For example, the first migration of partition A_1 is Type-One, and the first migration of partition A_2 is Type-Two. These two types of partition-migration have different impacts on the performance, especially on an SMP platform, which can provide an identical latency for memory access from different processors. Obviously, Type-One partition-migrations would get much more penalty from TLB and cache misses than Type-Two partition-migrations.

A simple strategy can eliminate all the Type-One migrations. When the resources pick the partitions at a specific time instant, those partitions that are just executed in the previous time slice would remain on the same resources. Figure 8 shows the better result after optimization. We extend the hyper period to 48 to eliminate Type-One migrations crossing consecutive periods.

D. Simulation

We show the performance of AAF-Multi in this subsection by simulation. We generate regular or irregular partitions whose availability factors are randomly chosen in $(0, 1)$, and supply regularities are randomly chosen in

Type-One Migrations are eliminated after optimization

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R_1	C_1	C_1	C_1	C_1	A_2	A_2	C_2	C_1	C_1	C_1	B_1	B_1	B_1	B_1	B_1	B_1
R_2	A_2	A_2	B_1	B_1	B_1	C_1	B_1	B_1	B_1	A_1	A_1	A_1	A_1	C_1	C_2	C_1
R_3	B_1	A_1	A_1	A_1	A_1	A_1	A_1	A_1	A_2	A_2	A_2	C_1	A_2	A_2	A_1	A_1
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R_1	B_1	A_1	A_1	A_1	A_1	A_1	A_1	A_1	A_2	A_2	A_2	C_1	A_2	A_2	A_1	A_1
R_2	C_1	C_1	C_1	C_1	A_2	A_2	C_2	C_1	C_1	C_1	B_1	B_1	B_1	B_1	B_1	B_1
R_3	A_2	A_2	B_1	B_1	B_1	C_1	B_1	B_1	B_1	A_1	A_1	A_1	A_1	C_1	C_2	C_1
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R_1	A_2	A_2	B_1	B_1	B_1	C_1	B_1	B_1	B_1	A_1	A_1	A_1	A_1	C_1	C_2	C_1
R_2	B_1	A_1	A_1	A_1	A_1	A_1	A_1	A_1	A_2	A_2	A_2	C_1	A_2	A_2	A_1	A_1
R_3	C_1	C_1	C_1	C_1	A_2	A_2	C_2	C_1	C_1	C_1	B_1	B_1	B_1	B_1	B_1	B_1

Fig. 12. Resource Allocation after optimization

$[1, MaxReg]$, where $MaxReg$ is a varying parameter. Figure 9 and 10 show the schedulable ratio of randomly generated partition sets on 2 resources and 16 resources. The horizontal axis represents the utilization ratio of randomly generated partition sets, and the vertical axis represents the schedulable ratio of these partitions sets by AAF-Multi. From the charts, we can see that irregular partitions require less utilization overhead than regular partitions when their Availability Factors are converted to AAF for scheduling.

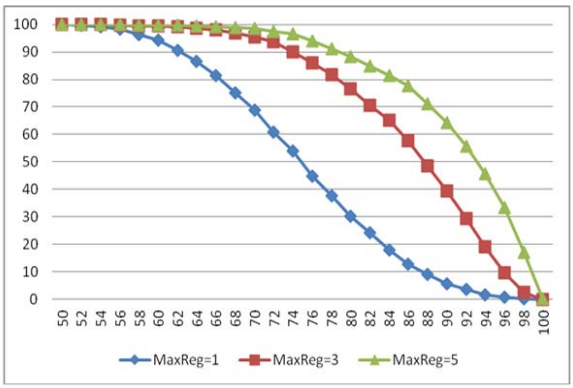

Fig. 13. Schedulable Ratio of AAF-Multi on 2 Resources

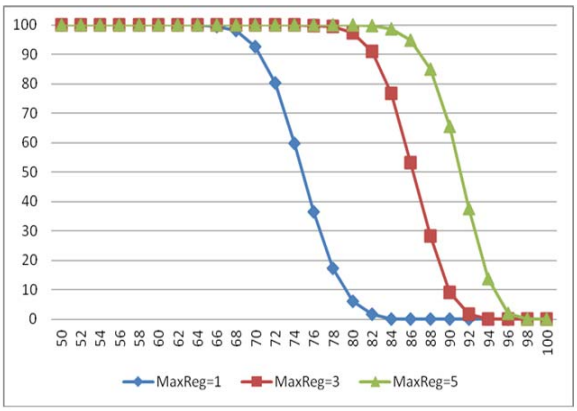

Fig. 14. Schedulable Ratio of AAF-Multi on 16 Resources

VI. Conclusion

In this paper, we investigate the problem of applying the regularity-based temporal resource partition model to a uniform multiresource platform. We introduce the concept of binary partition, and develop an inductive resource partitioning algorithm (AAF-Multi) for it on a multiresource platform. Meanwhile, the schedulability bound remains the same as in the former result on a single resource.

Our partitioning techniques can supply compositionality and isolation for real-time applications on a multiresource platform such that they appear to execute on their dedicated resources. In a virtualization environment, our techniques can also help the virtual machines monitor provide virtual resources to real-time virtual machines with more temporal protection. Meanwhile, our partitioning algorithm can guarantee very good time complexity.

There are still many open issues in this problem. First, the time consumption of AAF-Multi Scheduling could be reduced. Second, the schedulability bound may not be sufficiently tight, and could be improved by algorithms similar to pfair [4, 5]. Last but not least, resource partitioning of a non-uniform multiresource platform still needs further investigation. As a future work, we shall implement our resource partition model in a real-time Linux kernel running on an SMP platform.

References

[1] A. K. Mok and X. Feng. Towards compositionality in real-time resource partitioning based on regularity bounds. RTSS, 2001.

[2] A. K. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. RTAS, 2001.

[3] X. Feng. Design of real-time virtual resource architecture for largescale embedded systems. Ph.D. dissertation, Department of Computer Science, The University of Texas at Austin, 2004.

[4] S. Baruah, N. Cohen, G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 1996.

[5] S. Baruah, J. Gehrke, and G. Plaxton. Fast scheduling of periodic tasks on multiple resources. IPPS, 1995.

[6] D.-I. Oh and T. P. Baker. Utilization bounds for n-processor rate monotonescheduling with static processor assignment. Real-Time Syst, 1998.

[7] S. Baruah and N. Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. IEEE Transactions on Computers, 2006.

[8] S. Shigero, M. Takashi, and H. Kei. On the schedulability conditions on partial time slots. RTCSA, 1999.

[9] G. Lipari and E. Bini. A framework for hierarchical scheduling on multiprocessors: from application requirements to run-time allocation. RTSS, 2010.

[10] A. Mok, X. Feng, and D. Chen. Resource partition for realtime systems. In RTAS, 2001.

[11] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In RTSS, 2003.

[12] A. Easwaran, M. Anand, and I. Lee. Optimal compositional analysis using explicit deadline periodic resource models. In RTSS, 2007.