IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 23, 2019, accepted February 16, 2020, date of publication February 20, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975254

A Fast Online Task Placement Algorithm for Three-Dimensional Dynamic Partial Reconfigurable Devices

TINGYU ZHOU[1], (Student Member, IEEE), TIEYUAN PAN[2], MICHAEL CONRAD MEYER[1], (Member, IEEE), YIPING DONG[3], AND TAKAHIRO WATANABE[1], (Life Member, IEEE)

[1]Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 8080135, Japan
[2]DENSO Corporation, Kariya 4488661, Japan
[3]China Key System & Integrated Circuit Company, Ltd., Wuxi 214072, China

Corresponding author: Tingyu Zhou (shu-yu@asagi.waseda.jp)

This work was supported in part by the Waseda University Tokutei Kadai under Grant 2018K-301 and Grant 2019C-719.

ABSTRACT Three-dimensional (3D) integration technology provides a great opportunity for reconfigurable devices to increase device performance. Nevertheless, there is no efficient data structure and task placement algorithm to manage 3D dynamic partial reconfigurable (DPR) resources in literature. Inefficient algorithms limit the performance of 3D DPR devices. This study addresses the issue of the 3D task placement problem via a novel data structure named Maximal Empty Cuboid (MEC) list, which is proposed to manage the unoccupied space on the 3D DPR device. No matter if a task is assigned or removed on the device, the MEC list is updated in real-time to record 3D unoccupied resources so that the online task placement can be executed in a shorter time. Experiments are carried out to evaluate the performance of the proposed task placement algorithm, and results demonstrate that the proposed algorithm can make a reduction of at least 39% in terms of the task rejection ratio verifying the algorithm's efficiency.

INDEX TERMS Online task placement algorithm, three-dimensional, dynamic partial reconfigurable devices, maximal empty cuboid.

I. INTRODUCTION

Since Integrated Circuits (ICs) were born in the 1960s, they have been widely used in all aspects of our modern lives. In the field of electronic information technology, General Purpose Processors (GPPs), such as Central Processing Units (CPUs), have become a necessary component of various computing devices.

GPPs with high flexibility can achieve any complicated operations. However, GPPs are not designed for meeting the requirement of fast real-time applications or large-scale data processing, such as large matrix operations, sophisticated image processing, and Artificial Intelligence (AI), etc. To fully accelerate such specialized computation to achieve higher performance, special-purpose ICs, including ASICs, Graphics Processing Units (GPUs), and Digital Signal Processors (DSPs), etc., were designed. Such ICs lose a significant portion of flexibility granted by GPPs in exchange for performance gains. Once a special-purpose processor is manufactured, its functionality can no longer be changed. In a word, GPPs or special-purpose ICs cannot adapt to both the growing flexibility and higher performance requirements.

Reconfigurable devices such as Field-programmable Gate Arrays (FPGAs), provide a good method to balance the computing performance and flexibility. Generally, it is composed of three major components: Programmable Logic Blocks (PLBs), Input-Output Blocks (IOBs), and Interconnections. These programmable hardware resources provide the unique ability of reconfiguration, which means the circuits of the device can be reconfigured to achieve different functions to meet requirements through reconfiguring programmable resources. The reconfigurability enables the device to keep high efficiency of hardware and obtain the programmability of software to fill the gap between GPPs and special-purpose ICs [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

There are two leading reconfigurable technologies: static and dynamic reconfiguration. In static reconfiguration, the reconfigurable device needs to be powered off to load new configuration data to reconfigure the necessary circuits. Then, the device is restarted to execute a new function. In contrast, dynamic reconfiguration allows for loading of new configurations at run-time without taking the device offline. According to the reconfiguration area, global and partial reconfiguration are discussed in dynamic reconfiguration. In global reconfiguration, configuration bitstreams provide all information regarding the complete chip and configure the entire device. However, in partial reconfiguration, only a portion of the device is reconfigured, while the other parts continue to operate without interrupting so that multiple functions can be simultaneously implemented on a single device. In this work, a dynamic partial reconfiguration scheme is explored since its high flexibility and performance provide great potential in the future.

On the other hand, with the growing requirements of device performance and integration, 3D ICs gradually became the future development trend due to its higher integration density, shorter wire-length, and lower energy consumption compared with 2D architecture [2]. In the reconfigurable computing field, the application of a 3D dynamic partial reconfigurable (DPR) device also became a research focus [3], [4].

To achieve higher performance on 3D DPR devices, a lot of problems have to be solved, such as power consumption, heat dissipation, and inter-tasks communication, etc. One of the most important problems is the task placement problem, which means deciding where to place an arriving task on the device for efficient execution. For an application that will be executed on a reconfigurable device, a host processor first divides the application into amounts of small hardware modules (tasks), as execution targets on the reconfigurable device. Each task consists of a series of information, such as the number of required hardware resources, lifetime and execution deadline, etc. During application execution, the task is assigned to proper elements on the device, then executed and finally removed when it finishes. However, due to the limitation of programmable hardware resources, it is usually impossible to have enough resources to assign all arriving tasks to the device simultaneously. Therefore, an appropriate task placement position has to be determined at run-time to obtain maximum utilization ratio of limited hardware resources. This task placement problem is accompanied by the entire application execution phase and directly affects the 3D DPR device performance. Studies for the task placement algorithm to address this problem concentrate on the following two aspects:

- Resource management: record and update programmable resource usage of the device in real-time;
- Placement strategy: decide the task position assignment.

Nevertheless, existing algorithms on this problem for 3D DPR devices are inefficient [5] (time-consuming and low placement quality), and there are no efficient methods for 3D resource management as far as we know.

In this work, we propose a fast online task placement algorithm based on a novel data structure called Maximal Empty Cuboid (MEC) list for 3D DPR devices, which possesses the advantages of fast execution time and high placement quality. The main contributions of this paper are summarized as follows:

- We propose a new data structure called Maximal Empty Cuboid (MEC) list to describe 3D unoccupied reconfigurable resources and prove the upper limit of the number of MECs in theory.
- We develop an MEC enumeration algorithm to quickly update the MEC list to record the resource utilization in real-time once a task is assigned or removed on the 3D DPR device.
- We outline two placement strategies and discuss their impact on placement quality.
- We demonstrate better overall performance in terms of placement quality and execution speed compared to other state of the art approaches [5], [6] through theoretical and simulation studies.

The rest of this paper is organized as follows. Section II gives a brief overview of related studies addressed the task placement problem. In section III, we give the problem formulation, including some basic models and details of our proposed task placement algorithm. In Section IV and V, we evaluate and analyze the proposed algorithm through theories and experiments. Finally, in Section VI, we summarize the paper.

II. RELATED WORKS

The 4D compaction algorithm proposed by Marconi and Mitra [5] focused on the task placement problem on 3D DPR devices. A 3D matrix is used to represent the status and the earliest available time of the 3D programmable resources. The algorithm traverses the 3D matrix and selects the position that makes the arriving task have the earliest start execution time and largest contact area with other ones, which can be considered as four-dimensional compaction, i.e., both in 3D spatial coordinates and time coordinate. The 4D compaction algorithm makes full use of the 3D device resources; however, it has low efficiency due to the significant time consumption for traversing multiple 3D matrices to find the best placement position.

As far as we know, in addition to the 4D compaction algorithm, there are few other effective algorithms, and data structures proposed which target 3D DPR devices in literature. However, the task placement problem for a 2D DPR device has been explored so much, where a lot of strategies and novel data structures were proposed to make more efficient use of limited hardware resources.

Marcroni *et al.* [6] proposed a novel Quad-corner (QC) algorithm, which tries to assign tasks to the four corners of the 2D DPR device according to task size and maintains a free area in the center as much as possible, so as to keep ample and consecutive space for future arriving tasks.

Tabero *et al.* [7] presented a Vertex List Set (VLS) as a geometrical description on the whole reconfigurable device of the available area perimeter. Each vertex list describes the contour of each unoccupied area fragment on the device and some of the vertexes (bottom-left BL and top-right TR) in it are marked as candidates for task placement. During the task placement process, the placement algorithm traverses each vertex list in VLS and tries to perform feasible task insertions at each candidate vertex.

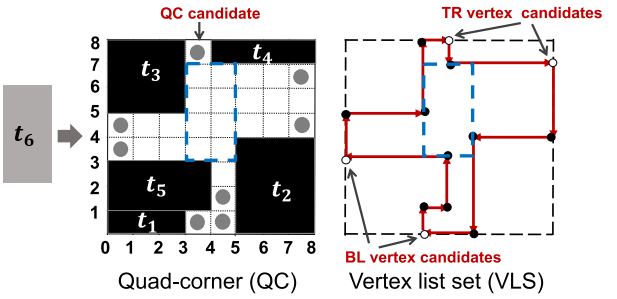


FIGURE 1. An example of QC and VLS representation on a 2D DPR device.

Nevertheless, one of the major drawbacks to using QC and VLS is the fact that the limited searching candidates may cause an executable position to not be found for a target task in some cases. For example, in Fig.1, task t_1-t_5 (black regions) are running on a 2D DPR device and the system searches for an available position for a new arriving task t_6. In Fig.1, the candidate locations generated by the QC are marked as grey nodes, and the VLS of available regions is marked by red lines. No matter whether searching for candidates in QC or VLS, t_6 cannot be assigned on the device due to the limitation of searching candidates, although there is enough space (blue frame) to accommodate it. Consequently, the placement quality of these algorithms cannot be guaranteed although the speed of them is fast. In [8]–[11], a Maximal Empty Rectangle (MER) list is proposed to manage 2D unoccupied hardware resources, which is defined as the list of empty rectangles that are not fully covered by other ones. The MER list records all maximal empty rectangles on the 2D DPR device, which guarantees the task can find the available position for it once the position exists to avoid the placement quality problem that appeared in QC and VLS. However, the frequent assignment and removal of tasks lead to the MER list being updated all the time, thus a fast method to enumerate all MERs is required.

Although several efficient placement algorithms for the 2D DPR device have been proposed, there are few 3D task placement algorithms, and they are inefficient, that is, they do not efficiently manage the unoccupied 3D DPR resources. In this paper, an efficient data structure inspired by the concept of MER on the 2D DPR device and the algorithm is presented to solve the 3D task placement problem so that the performance of the 3D DPR device is improved.

III. PROPOSED APPROACH

A. PROBLEM FORMULATION

This section describes the 3D DPR device model, 3D task model, system model, and the proposed data structure. These models are in accordance with related works in [5]–[7].

1) 3D DPR DEVICE AND TASK MODELS

Since the emergence of the concept of the 3D DPR device, various 3D architectures are proposed [12]–[14]. Leeser *et al.* [15] proposed a representative multi-layer architecture, which can be considered as a stack of FPGA circuits with connections between different layers. In particular, the first heterogeneous 3D FPGA was produced by Xilinx in 2012, which is comprised of four FPGA dies, each of which is attached to one silicon interposer and connected using Through-Silicon Via (TSV) technology [16]. Therefore, a reconfigurable device (RD) of the proposed multi-layer 3D DPR device is modeled as RD (W, H, TH), where $W \times H \times TH$ reconfigurable units (RUs) are arranged in a 3D array and interconnections. The position of an RU is represented by the left-most front corner coordinate RU (ux, uy, uz), where $0 \leq ux < W$, $0 \leq uy < H$ and $0 \leq uz < TH$. As shown in Fig.2, the bottom left-most front corner of the 3D DPR device is defined as (0, 0, 0).

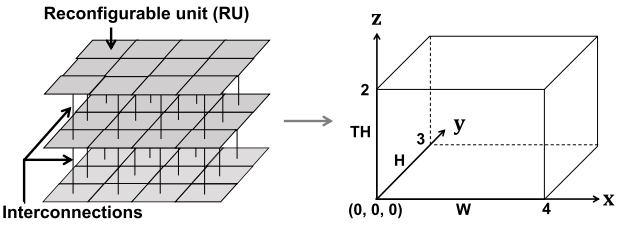


FIGURE 2. A 3D dynamic partial reconfigurable device model.

Tasks are generated by partitioning a given application and considered as basic execution units. A 3D task t_i is defined as $t_i(w_i, h_i, th_i, lf_i, dl_i)$, where $w_i \times h_i \times th_i$ resources are necessary for execution, w_i, h_i and th_i are task width, height and thickness respectively, and lf_i means the lifetime of the task t_i, which is the sum of configuration time and execution time and dl_i means the deadline, which is maximal waiting time of the task t_i before placement. If the waiting time is over dl_i, the task t_i is discarded. A task can be assigned at an arbitrary position on the 3D DPR device if there are enough *empty areas* to be assigned to the task. Empty area means the RUs on the device are not being occupied by any tasks currently. Similar to the previous models in [5]–[7], we assume that tasks are independent, with no priority constraints with each other since these factors are related to the scheduling order of the execution of the tasks.

2) MAXIMAL EMPTY CUBOID LIST

To manage 3D available programmable resources, a Maximal Empty Cuboid (MEC) list is proposed for a description of

empty areas on the 3D DPR device. The MEC list is defined as follows:

Definition 1: An MEC $c_i(cx_i, cy_i, cz_i, cw_i, ch_i, cth_i)$ is an empty cuboid that can not be fully covered by other ones and in which a 3D task may be assigned, where (cx_i, cy_i, cz_i) is the bottom left-most front corner coordinate, cw_i, ch_i and cth_i are the width, height and thickness of the MEC c_i, respectively.

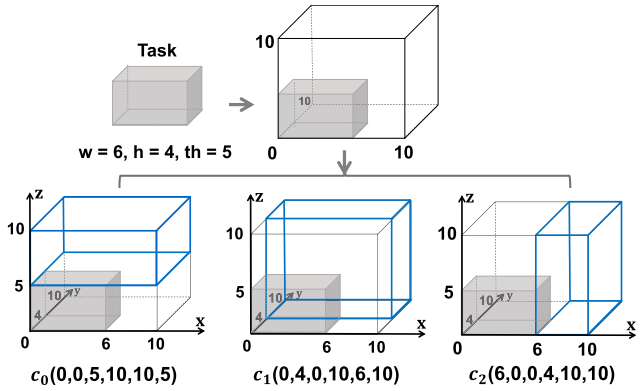


FIGURE 3. An example of an MEC list.

Fig.3 shows an example of an RD (10, 10, 10), where gray-colored region represents a currently running task. In accordance with the device status, three MECs c_0(0, 0, 5, 10, 10, 5), c_1(0, 4, 0, 10, 6, 10) and c_2(6, 0, 0, 4, 10, 10) are generated, which are marked with blue frames.

An important property of an MEC is given in the following theorem.

Theorem 1: Each surface of an MEC must touch another task or the surface of the 3D DPR device.

Proof 1: Assume that there exists an MEC, one surface of which is not in contact with any task or the surface of the 3D DPR device. Thus, the MEC can be extended in the corresponding direction until it touches one task or device surface. However, any new extended cuboid can fully cover the MEC, which contradicts the definition of an MEC and *Theorem* 1 is proved.

A critical problem comes with the application of the MEC list since tasks are constantly assigned and removed on the device. Once the status of the hardware resources on the device changes, it will result in additional time to update the MEC list. Therefore, it is necessary to explore fast and accurate algorithms to enumerate all MECs in the current resource status, which is the main work that we do.

3) ONLINE TASK PLACEMENT MODEL

Fig.4 shows the online task processing model for a 3D DPR device, which is composed of a scheduling stage and a placement stage with three main components: a scheduler, a placer, and an area manager. The scheduler determines which tasks are executed at the current time, the placer is responsible for searching a suitable execution position for the scheduled task and the area manager is used to record and update unoccupied 3D resources [17].

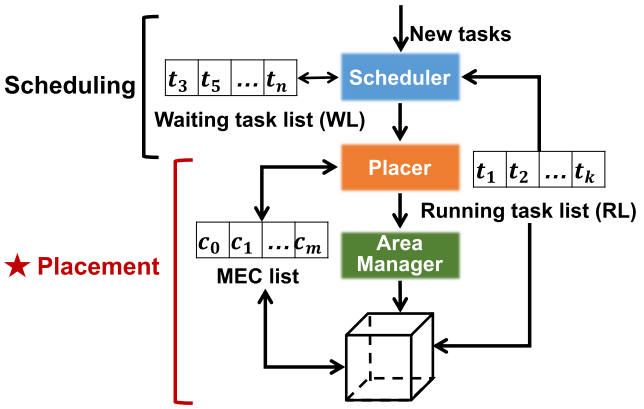


FIGURE 4. Online task processing model.

When a new task arrives at the system based on a given arriving time interval, a task with a closer deadline between the new arriving task and waiting tasks is selected by the scheduler. Then, the placer traverses an MEC list, which describes the empty areas on the 3D DPR device. If there are some MECs that can accommodate the arriving task, choose one of them according to the assignment strategy (described in section III-C) and add the task to a *running task list* (RL). Tasks in RL are sorted by descending order of rest time to execute, where rest time is defined as the remaining execution time of the task. After these operations, the MEC list is updated by the area manager (described in section III-B). If there are no MECs that can accommodate the scheduled task, add the task to a *waiting task list* (WL) where tasks are sorted by descending order of tasks' deadlines dl. Once the scheduler checks that a task execution is finished, remove it from RL and update the MEC list by the area manager. If the current time exceeds the deadline of a task in WL, the task is rejected and removed from WL by the scheduler.

In the whole task processing, our research centers on the placement stage and includes two main aspects:

- Propose a fast MEC enumeration algorithm to update the MEC list at run-time for purpose of managing 3D hardware resources;
- Explore the placement quality based on different assignment strategies.

B. MEC ENUMERATION ALGORITHM

In the placement stage, the proposed MEC list is used to record the unoccupied 3D DPR resources. Once a task is assigned or removed on the device, the MEC list has to be updated. Therefore, the updating speed directly affects the performance of the device. In this section, an efficient MEC enumeration algorithm with four steps: 1) connected MECs selection, 2) stratification, 3) stratified MECs update, and 4) extending, is described to enumerate all new MECs to update the MEC list. Parameters used to describe the proposed MEC enumeration algorithm are summarized in Table 1.

TABLE 1. Variable definitions.

Parameter	Definition
$RD(W, H, TH)$	3D DPR device with size $W \times H \times TH$.
$t(w, h, th, lf, dl)$	Target task t with size $w \times h \times th$, life time lf and deadline dl that will be assigned or removed.
$p(x, y, z)$	Position of the task t.
$C = \{c_0, c_1, ..., c_i\}$	Maximal Empty Cuboid (MEC) list.
$C_c = \{c_0, c_1, ..., c_j\}$	Connected MEC list and $C_c \subseteq C$.
$c(cx, cy, cz, cw, ch, cth)$	An MEC $c \in C$ with the bottom left-most front corner coordinate (cx, cy, cz) and size $cw \times ch \times cth$.
$LA = \{l_0, l_1, ..., l_{th+1}\}$	Layer list.
$l(lx, ly, lz, lw, lh, lth)$	A layer $l \in LA$ with the bottom left-most front corner coordinate (lx, ly, lz) and size $lw \times lh \times lth$.
$s(sx, sy, sz, sw, sh, sth)$	A stratified MEC with the bottom left-most front corner coordinate (sx, sy, sz) and size $sw \times sh \times sth$.

Algorithm 1 MEC Enumeration Algorithm

Require:
The MEC list: C.
The Connected MEC list: C_c.
The assigned or removed target task: t (w, h, th, lf, dl).
The 3D DPR device: RD(W, H, TH)
The position of target task t: p (x, y, z).
The layer list: $LA = \{l_0, l_1, \ldots, l_{th+1}\}$.

Ensure:
Update the MEC list C.

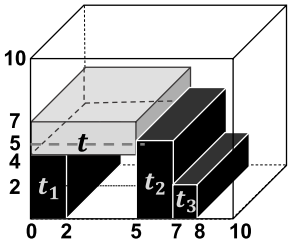
1: $C \leftarrow \varnothing$;
2: **for** each $l_i \in LA$ **do**
3: $l_i \leftarrow \varnothing$;
4: **end for**
5: SelectConMEC(C,C_c,x,y,z,w,h,th); /*Section III-B.1*/
6: StratifyMEC(C_c, LA, x, y, z, th, TH); /*Section III-B.2*/
7: UpdateStraMEC(LA, x, y, z, w, h, th); /*Section III-B.3*/
8: ExtendLayer(LA, z, th); /*Section III-B.4*/
9: **for** each s_i in LA **do**
10: add s_i to C;
11: **end for**

The overview of the MEC enumeration algorithm is shown in Algorithm 1. It is applied to obtain the updated MEC list after the target task $t(w, h, th, lf, dl)$ being assigned or removed at position $p(x, y, z)$. Firstly, connected MECs of the target task t are selected from MEC list C and moved in connected MEC list C_c (line 5). After that, these connected MECs in C_c are stratified and stored in a layer list LA (line 6). Stratified MECs in candidate layers are updated according to the task assignment or removal position p (line 7). Then, by extending the updated stratified MECs bottom-up, all updated MECs can be obtained (line 8). Finally, the MEC enumeration algorithm updates the MEC list C by adding the updated MECs into C (line 9-11). Each step is explained in detail below.

1) CONNECTED MECS SELECTION (SELECTCONMEC)

When a task is assigned or removed, the MEC that is *connected* to the task needs to be updated in the MEC list. The definition of 'connected' is given as follows:

Definition 2: If there exists an overlap or contact between an MEC and a task, the MEC is called a connected MEC of the task, otherwise it is unconnected.

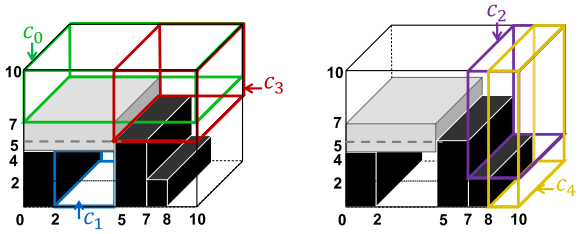


Task	Position	w	h	th
t	(0, 0, 4)	5	10	3
t_1	(0, 0, 0)	2	10	4
t_2	(5, 0, 0)	2	10	5
t_3	(7, 0, 0)	1	10	2

FIGURE 5. Tasks on a 3D DPR device RD (10, 10, 10).

TABLE 2. MEC list in Fig.5.

Maximal Empty Cuboid c_i	$c_i(cx_i, cy_i, cz_i, cw_i, ch_i, cth_i)$
c_0	(0, 0, 7, 10, 10, 3)
c_1	(2, 0, 0, 3, 10, 4)
c_2	(7, 0, 2, 3, 10, 8)
c_3	(5, 0, 5, 5, 10, 5)
c_4	(8, 0, 0, 2, 10, 10)



(a) Connected MECs (c_0, c_1 and c_3) (b) Unconnected MECs (c_2 and c_4)

FIGURE 6. Connected and unconnected MECs in Fig.5.

Fig. 5 shows an example of an RD(10, 10, 10), where black-colored regions are the area occupied by running tasks (t_1, t_2, and t_3) and gray-colored cuboid t is a target task that has finished executing to be removed. According to the task information shown in Fig. 5, there are five MECs and detailed information of them are listed in Table 2, where the c_0, c_1 and c_3 are marked by colored frames in Fig.6(a), which represent the MECs connected to the gray-colored task t since c_0, c_1, and c_3 are directly touched by the target task t. Fig.6(b) shows the unconnected ones, c_2 and c_4.

Since the task assignment and removal merely affect the MECs connected to them (proved in *Theorem* 2 below), the others in the MEC list have no need to be operated in the subsequent steps of the MEC enumeration algorithm. Thus, the subsequent operations can be significantly reduced by specifically selecting the connected MECs from the MEC list.

Theorem 2: When a target task is assigned or removed on a 3D DPR device, unconnected MECs of the task are not affected by this operation.

Proof 2: For a task t, assume that there exists an unconnected MEC c', which is affected when the task t is removed from the 3D DPR device. The task removal operation releases resources occupied by the task t, so that the affected MEC c' can be extended to a larger one. Since each surface of an MEC must touch at least one task or surface of the device (proved in Theorem 1), the extension of the c' means that at least one task that c' in contact with is removed. Since only task t is removed from the device at this time, there is only one possible fact that the task t is the task that contact with the MEC c'. This contradicts the definition of an unconnected MEC in Definition 2. Similarly, the theorem is proved.

2) STRATIFICATION (STRATIFYMEC)

To reduce the complexity of updating connected MECs at the three-dimensional level, we layer the 3D DPR device for subsequent processing in this stratification step.

The stratification step includes two primary operations: 1) build a layer list LA and select candidate layers, 2) cut connected MECs and store stratified MECs information into corresponding candidate layers.

Firstly, a layer list LA based on the placement position and thickness of the target task is generated. The 'layer' is defined as follows:

Definition 3: A layer l_i $(lx_i, ly_i, lz_i, lw_i, lh_i, lth_i) \in LA$ is a space that is generated by horizontally slicing the 3D DPR device according to the top and bottom surface of the assigned or removed task, where (lx_i, ly_i, lz_i) is the bottom left-most front corner coordinate, lw_i, lh_i and lth_i are the width, height, and thickness of the layer l_i, respectively. Due to the horizontal stratification, the basic information of the layer l_i can be represented as l_i $(0, 0, lz_i, W, H, lth_i)$, where W and H are the width and height of the device, respectively. Therefore, the above definition of a layer l_i can be simplified to l_i (lz_i, lth_i).

In Fig.7(a), given a target task t (w, h, th, lf, dl), it will be assigned or removed at position $p(x, y, z)$. Fig.7(b) shows the front view of the task t, where the space of the 3D DPR device $\mathrm{RD}(W, H, TH)$ is cut into $LA = \{l_0, \ldots, l_i, \ldots, l_{th+1}\}$ in the z-axis, where $0 \le i \le th+1$. The z-axis value and information of each layer is described in Fig.7(b).

As the layer list LA is constructed, connected MECs for the target task are horizontally stratified and stored in corresponding layers according to their z-axis value, where 'stratified MEC' is defined as follows:

Definition 4: A stratified MEC $s_i(sx_i, sy_i, sz_i, sw_i, sh_i, sth_i)$ is an empty cuboid that is generated from horizontally slicing connected MECs based on layer information, where (sx_i, sy_i, sz_i) is the bottom left-most front corner coordinate, sw_i, sh_i and sth_i are the width, height, and thickness of the stratified MEC s_i, respectively.

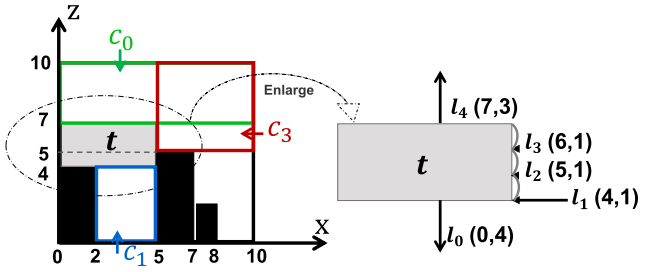
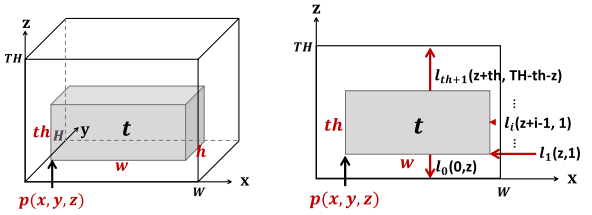


FIGURE 7. Principle to generate a layer list *LA*.



(a) A 3D task t at position p. (b) Front view of the 3D DPR device and layer list information.

FIGURE 8. An example of stratification.

TABLE 3. Layers and stratified MECs in Fig.7.

Layer	Stratified MEC s_i	c_j(source of s_i)
$l_0(0, 4)$	$s_0(2, 0, 0, 3, 10, 4)$	c_1
$l_1(4, 1)$	None	
$l_2(5, 1)$	$s_0(5, 0, 5, 5, 10, 1)$	c_3
$l_3(6, 1)$	$s_0(5, 0, 6, 5, 10, 1)$	c_3
$l_4(7, 3)$	$s_0(0, 0, 7, 10, 10, 3)$,$s_1(5, 0, 7, 5, 10, 3)$*	c_0, c_3

* Redundant stratified MEC.

Fig.8 shows the front view of the 3D DPR device in Fig.6(a) and the device is stratified into layer $l_0 - l_4$ according to the principle shown in Fig. 7. In Table 3, the first column shows the layer list information, the second column shows the stratified MECs information in each layer and c_j in the third column means the stratified MEC s_i comes from the stratification of the connected MEC c_j.

For example, the stratified MECs s_0 and s_1 in the layer l_4 are generated from the stratification of connected MECs c_0 and c_3, respectively. Besides, a redundant MEC may exist, which can be completely covered by other stratified MECs in the same layer. The redundant MEC is marked by * in Table 3 and needs to be removed.

Note that, the layers l_0, l_1, l_2, and l_4 in Table 3, are marked as *candidate layers* for the next processing step, to the opposite l_3 is a non-candidate layer.

The principle for selecting a candidate layer is given as follows:

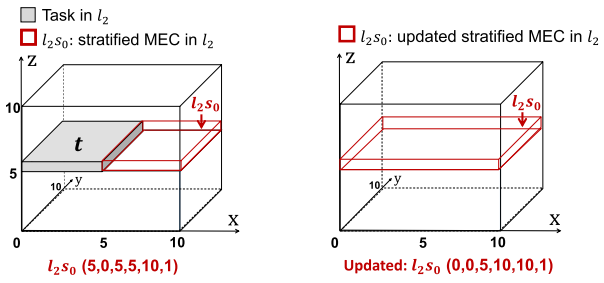
Definition 5: For two adjacent layers l_i and l_{i+1}, l_{i+1} is a non-candidate layer if the information of stratified MECs in the two layers are completely the same except for their z-axis value and thickness. Conversely, if not completely overlapping, l_{i+1} is a candidate layer.

For the bottom layer l_0 and the top layer l_{th+1}, as long as there exists stratified MECs stored in which, it is marked as a candidate layer. Furthermore, l_1 is always considered as a candidate layer since the bottom of the target task t is at l_1.

Algorithm 2 shows the pseudo-code of the stratification step. We traverse the connected MEC list and compare the task thickness with the bottom and top of its connected MECs since the layer at the bottom or top of a connected MEC must be a candidate layer (line 1-20). In each candidate layer, the information of stratified MECs is stored. Furthermore, to avoid the lack of the stratified MECs information stored in the candidate layer, the connected MEC list and candidate layer information are compared again (line 22-30). It should be noted that the redundancy check is necessary (line 31-36) before each stratified MEC is stored in its corresponding layer.

3) STRATIFIED MECS UPDATE (UPDATESTRAMEC)

Stratified MECs in candidate layers except for l_0 and l_{th+1}, need to be updated when assigning or removing the target task. From layer l_1 to l_{th}, the stratified MECs in the same layer have the same thickness; thus, the ones in the same layer can be considered as MERs in the x-y plane. A method proposed in [11] is used to enumerate new MERs in a 2D plane after the device status changes.



(a) Before removing (b) After removing

FIGURE 9. An example of stratified MEC update in l_2.

Fig. 9 shows an example of a stratified MECs update process in layer l_2 mentioned in Fig.8. When the gray-colored task t with size $5 \times 10 \times 3$ located at position $(0, 0, 4)$ is removed, the stratified MEC in l_2 is updated from Fig.9(a) to Fig.9(b). Similarly, the process of task assignment is reverse to the removal process. Table 4 shows the stratified MECs update results from the example in Fig.8, where the stratified MECs marked with red are the updated ones.

4) EXTENDING (EXTENDLAYER)

In the previous steps, connected MECs on the 3D DPR device have been updated in each layer for management. Contrary to the stratification, in the final step, we need to extend updated stratified MECs bottom-up to generate ultimate MECs and obtain the final updated MEC list. The overview of the extending step is described in Fig. 10, where the th is the

Algorithm 2 StratifyMEC(C_c, LA, x, y, z, th, TH)

Require:
Task's position and thickness: x, y, z, th.
3D DPR device's thickness: TH.
Connected MEC list: C_c.
The layer list: $LA = \{l_0, l_1, \ldots, l_{th+1}\}$.
Integers: i, j.

Ensure:
Stratify connected MECs into candidate layers. **Function** Stratify (C_c, LA, x, y, z, th)

1: **for** each $c(cx, cy, cz, cw, ch, cth) \in C_c$ **do**
2: **if** $cz < z$ **then**
3: generate $s(cx, cy, cz, cw, ch, z - cz)$;
4: RedundancyCheck(l_0, s);
5: mark $l_0(0, z)$ as a candidate layer;
6: **end if**
7: **if** $cz + cth > z + th$ **then**
8: generate $s(cx, cy, z+th, cw, ch, cz + cth - (z+th))$;
9: RedundancyCheck(l_{th+1}, s);
10: mark $l_{th+1}(z+th, TH - z - th)$ as a candidate layer;
11: **end if**
12: **if** $(cz \geq z)$ and $(cz < z + th)$ **then**
13: generate $s(cx, cy, cz, cw, ch, 1)$;
14: RedundancyCheck(l_{cz-z+1}, s);
15: mark $l_{cz-z+1}(cz, 1)$ as a candidate layer;
16: **end if**
17: **if** $(cz + cth > z)$ and $(cz + cth < z + th)$ **then**
18: mark $l_{cz+cth-z+1}(cz + cth, 1)$ as a candidate layer;
19: **end if**
20: **end for**
21: mark $l_1(z, 1)$ as a candidate layer;
22: **for** each candidate layer $l_i(lz_i, lth_i)$ **do**
23: $lz_i \leftarrow i + z - 1$
24: **for** each $c(cx, cy, cz, cw, ch, cth) \in C_c$ **do**
25: **if** $(cz \leq lz_i)$ and $cz + cth > lz_i$ **then**
26: generate $s(cx, cy, lz_i, cw, ch, 1)$;
27: RedundancyCheck(l_i, s);
28: **end if**
29: **end for**
30: **end for**
End Function
Function RedundancyCheck (l, s)
31: **if** there is a s' in l that is included in s **then**
32: replace s' with s;
33: **end if**
34: **if** there is no s' in l includes s **then**
35: add s into l;
36: **end if**
End Function

TABLE 4. Candidate layers and updated stratified MECs in Fig.8.

Candidate layer	Updated stratified MECs $s_i(sx_i, sy_i, sz_i, sw_i, sh_i, sth_i)$
$l_0(0,4)$	$s_0(2,0,0,3,10,4)$
$l_1(4,1)$	$s_0(0,0,4,5,10,1)$
$l_2(5,1)$	$s_0(0,0,5,10,10,1)$
$l_4(7,3)$	$s_0(0,0,7,10,10,3)$

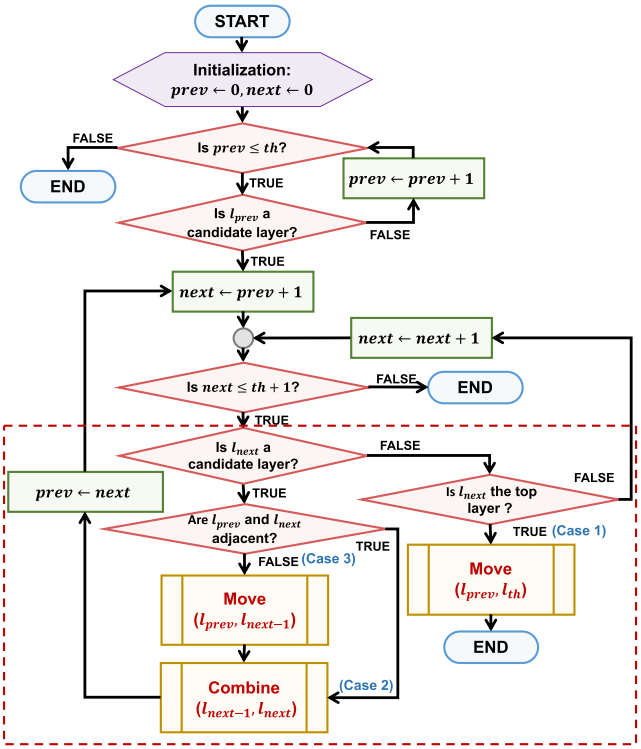


FIGURE 10. Flowchart of extending step.

thickness of the target task. From the bottom to the top layer, the candidate layers are searched and extended one by one.

Based on the position of the two candidate layers, there are two main operations: *Move* (Algorithm 3) and *Combine* (Algorithm 4) in the extending step. There are three cases marked in Fig. 10 as follows:

- **case 1** : Layer l_{prev} is the last candidate layer and $prev < th$: $move(l_{prev},\ l_{th})$, as shown in Fig.11(a).
- **case 2** : Layer l_{prev} and l_{next} are candidate layers and adjacent: $combine(l_{prev},\ l_{next})$, as shown in Fig.11(b).
- **case 3** : Layer l_{prev} and l_{next} are candidate layers, while non-adjacent: $move(l_{prev},\ l_{next-1})$, then, $combine(l_{next-1},\ l_{next})$, as shown in Fig.11(c).

The pseudo-code of the move operation is shown in Algorithm 3, where the thickness of the moved stratified MECs should be updated (line 2 in Algorithm 3).

The pseudo-code of the combine operation is shown in Algorithm 4, where stratified MECs from two candidate layers are compared. When two compared stratified MECs $[s_i(sx_i,sy_i,sz_i,\ sw_i,sh_i,sth_i)$ in the layer l_{next-1} and $s_j(sx_j,sy_j,sz_j,sw_j,\ sh_j,\ sth_j$ in the layer $l_{next}]$ are mapped in

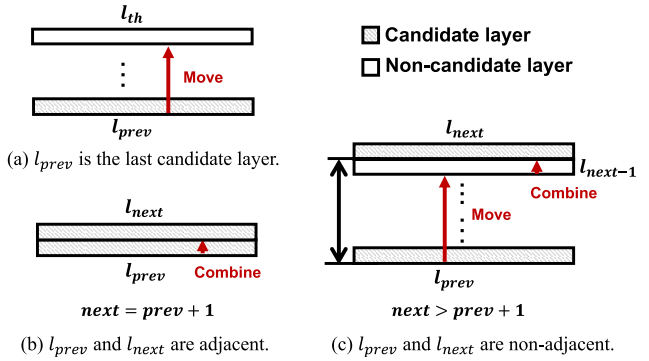


FIGURE 11. Extending step based on the position of candidate layers.

Algorithm 3 Move (l_a, l_b)

Require:
Two layers: l_a, l_b.
Ensure:
Move all stratified MECs in l_a to l_b.
1: **for** each stratified MEC $s_i(sx_i, sy_i, sz_i, sw_i, sh_i, sth_i)$ in l_a **do**
2: $sth_i \leftarrow sth_i + (b - a)$;
3: move s_i to l_b;
4: **end for**

the same x-y plane, there are five kinds of relative positions between them, which are given as follows:

- $s_i \supset s_j$ means that s_i includes s_j.
- $s_i \subset s_j$ means that s_i is included in s_j.
- $s_i = s_j$ means that s_i is equal to s_j, where $sx_i = sx_j$, $sy_i = sy_j$, $sw_i = sw_j$ and $sh_i = sh_j$.
- $s_i \cap s_j$ means that a part of s_i overlaps with s_j.
- $s_i \times s_j$ means that there is no overlap between s_i and s_j.

In the last case, the combine operation cannot be done since no overlap means there are no MECs that exist on which their bottom belongs to the s_i and top belongs to the s_j.

Fig.12 shows four scenarios where the combine operation is possible. In case (a), the thickness of s_j is updated (line 3-7 in Algorithm 4). In case (b), the thickness of s_i is extended (line 8-12 in Algorithm 4) and s_i is marked to move to l_{next} (line 27-29 in Algorithm 4). In case (c), s_i and s_j are merged to s_j (line 13-17 in Algorithm 4). In case (d), a new MEC s_{new} is generated and stored in l_{next} (line 18-24), where s_{new} becomes $s_{new}(max(sx_i, sx_j),\ max(sy_i, sy_j),\ min(sz_i, sz_j),\ min(sx_i + sw_i, sx_j + sw_j) - max(sx_i, sx_j),\ min(sy_i + sh_i, sy_j + sh_j) - max(sy_i, sy_j),\ max(sz_i + sth_j, sz_j + sth_j) - min(sz_i, sz_j))$.

Fig. 13 shows an example of the extending step in Table 4, where l_0, l_1, l_2 and l_4 are candidate layers. From l_0 to l_1, the combine operation is done directly. The s_0 in l_0 is included by the s_0 in l_1, therefore, the thickness of s_0 in l_0 is updated and moved to l_1, which is represented as s_1 in l_1. From l_1 to l_2, s_0 and s_1 in l_1 need to be compared with the s_0 in l_2, respectively. The combine operation is the same as before and extending results are shown in Fig. 13(b). From l_2 to l_4, they

Algorithm 4 Combine (l_{next-1}, l_{next})

Require:
Two layers: l_{next-1}, l_{next}.

Ensure:
Combine stratified MECs in l_{next-1} and l_{next}.

1: **for** each s_i in l_{next-1} **do**
2: **for** each s_j in l_{next} **do**
3: **if** $s_i \supset s_j$ **then**
4: **if** $sz_i \leq sz_j$ **then**
5: $sth_j \leftarrow sz_j + sth_j - sz_i$;
6: **end if**
7: **end if**
8: **if** $s_i \subset s_j$ **then**
9: **if** $sz_i + sth_i \leq sz_j + sth_j$ and $sz_i < sz_j$ **then**
10: $sth_i \leftarrow sz_j + sth_j - sz_i$ and mark s_i;
11: **end if**
12: **end if**
13: **if** $s_i = s_j$ **then**
14: **if** $sz_i \leq sz_j$ **then**
15: merge s_i and s_j in l_{next};
16: **end if**
17: **end if**
18: **if** $s_i \cap s_j$ **then**
19: **if** no s' in l_{next-1} that $s_{new} \subseteq s'$ and $sz' < sz_i$ **then**
20: **if** no s'' in l_{next} that $s_{new} \subseteq s''$ and $sz'' + sth'' > sz_j + sth_j$ **then**
21: add s_{new} to l_{next};
22: **end if**
23: **end if**
24: **end if**
25: **end for**
26: **end for**
27: **for** each marked s_i **do**
28: move s_i to l_{next};
29: **end for**

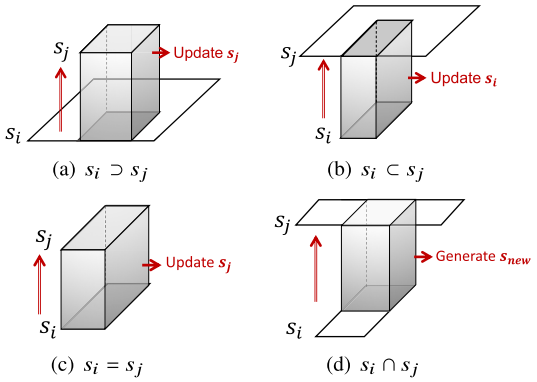
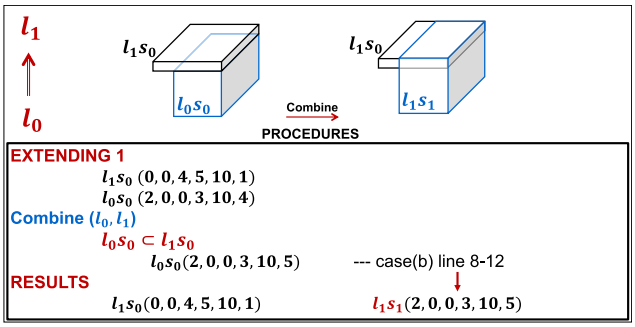
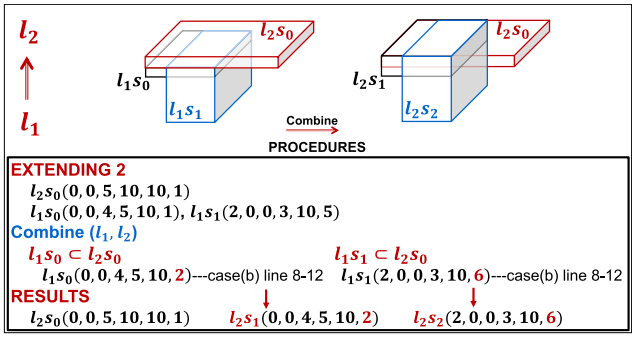


FIGURE 12. Relationships between s_i in l_{next-1} and s_j in l_{next}.

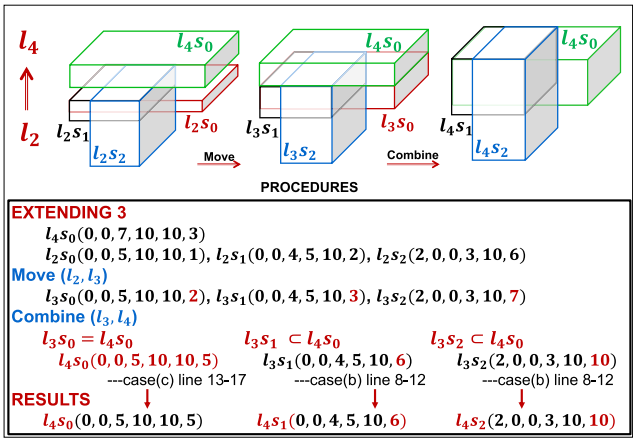
are non-adjacent layers, thus a move operation is performed to update the stratified MECs in l_2 and move them to l_3. Then in the combine operation, s_0, s_1, and s_2 in l_3 are compared with s_0 in l_4, respectively. It should be noted that, s_0 in l_3 is



(a) Extending 1



(b) Extending 2



(c) Extending 3

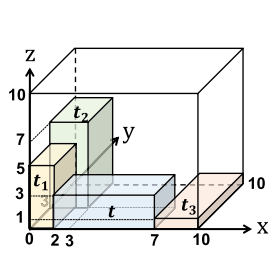
FIGURE 13. Extending step of stratified MECs in Table 4.

equal to s_0 in l_4, thus we merge them into an entire MEC and store it in l_4, which is marked as s_0 in l_4 in Fig.13(c). After processing all the candidate layers, all the stratified MECs have been extended from the bottom to the top layer to get the final extended MECs.

Finally, the layer list *LA* needs to be traversed and all the extended MECs are moved to the MEC list (line 9-11 in Algorithm 1). As a result, the connected MECs c_0, c_1 and c_3 in Fig. 6 are updated into: $c_0'(0, 0, 5, 10, 5)$, $c_1'(0, 0, 4, 5, 10, 6)$ and $c_3'(2, 0, 0, 3, 10, 10)$.

C. ASSIGNMENT STRATEGIES

The MEC list stores all of the MECs of the 3D DPR device in the current status, thus a scheduled task can find the



Task	Position	w	h	th	lf
t	(2, 0, 0)	5	3	3	4
t_1	(0, 0, 0)	2	3	5	8
t_2	(0, 3, 0)	3	7	7	7
t_3	(7, 0, 0)	3	10	1	9

FIGURE 14. Tasks and timeline of a 3D DPR device RD(10, 10, 10).

executable position by searching the MEC list. If several MECs can accommodate the target task, it is necessary to use a proper approach as an assignment strategy to choose the best one. In this section, several assignment strategies are introduced and combined with the proposed MEC enumeration algorithm.

1) BEST-FIT STRATEGY (BF)

If several MECs can accommodate the scheduled task, the Best-Fit strategy [18] selects one MEC which has the smallest volume difference dif, which is calculated as follows: given a task $t(w, h, th, lf, dl)$ and an MEC $c_i(cx_i, cy_i, cz_i, cw_i, ch_i, cth_i)$,

$$dif(c_i, t) = cw_i \times ch_i \times cth_i - w \times h \times th \tag{1}$$

If there are several MECs with the same dif, we calculate the body diagonal $dia(c_i)$, defined as follows:

$$dia(c_i) = \sqrt{cw_i^2 + ch_i^2 + cth_i^2} \tag{2}$$

then the MEC with the smallest body diagonal is assigned to the task.

2) ADJACENCY HEURISTIC STRATEGY (ADJ)

The adjacency heuristic was proposed in [19]. When a placed task touches other tasks or the surface of the 3D DPR device, they are defined to be adjacent. Adjacent value is the sum of surface area where a task and its adjacent ones are touching, taking into account the time of contact. The main idea of this strategy is selecting the position which has the largest adjacent value with other tasks and device surfaces, so as to make the remaining space on the device more complete and continuous.

Given a task $t(w, h, th, lf, dl)$ and the task set T which is adjacent to task t assigned at position (x, y, z), the adjacent value A can be calculated as follows:

$$A(x, y, z, t) = \sum_{t_i \in T} AT(t_i, t) \times min(lf, e_i) + AR(t) \times lf \tag{3}$$

where AT means the adjacent area between t and t_i, e_i is the rest of the execution time of task t_i and AR represents the adjacent area between the task and the 3D DPR device.

As shown in Fig. 14, at time 6, three tasks t_1, t_2, t_3 are executed on a $10 \times 10 \times 10$ device, whose information is shown in the table in Fig.14. At this time, if a new task t with dimensions $5 \times 3 \times 3$ arrives at the system and begins to execute at the position $(2, 0, 0)$, the adjacent area $AR(t)$ between task t and the 3D DPR device can be calculated as $5 \times 3 + 5 \times 3 = 30$. According to the relative position of the task t with t_1, t_2 and t_3, the adjacent area $AT(t_1, t)$, $AT(t_2, t)$, and $AT(t_3, t)$ are 9, 3 and 3, respectively. Based on the execution timeline of tasks, the remaining execution time of t_1, t_2 and t_3 can be calculated easily. Therefore, the adjacent value $A(2, 0, 0, t)$ is equal to $AT(t_1, t) \times min(4, 2) + AT(t_2, t) \times min(4, 3) + AT(t_3, t) \times min(4, 6) + AR(t) \times 4 = 159$.

Finally, according to Eq.3, the system can get all adjacent values when a new task is placed in different possible positions and select the position with the largest $A(x, y, z, t)$.

IV. THEORETICAL ANALYSIS

In this section, theoretical analysis is discussed to evaluate the performance of the proposed task placement algorithm. Definitions of additional variables are shown in Table 5, where a task $t(w, h, th, lf, dl)$ will be assigned or removed at position (x, y, z), n is the number of tasks that are currently running on the 3D DPR device RD(W, H, TH), and M is the number of MECs which contains m MECs connected to the task t.

TABLE 5. Variable definitions.

Parameter	Definition
n	The number of tasks currently running on the device.
M	The number of MECs.
m	The number of MECs that connected to target task t.

TABLE 6. Time complexity of MEC enumeration algorithm.

Step	Time complexity
Connected MECs Selection	$O(M)$
Stratification	$O(m \times th)$
Stratified MECs Update	$O((n + (min(w, h) + m)^3) \times th)$
Extending	$O(m^2 \times th)$
Total	$O(n + (n + (min(w, h) + m)^3) \times th)$

A. TIME COMPLEXITY

The time complexity of each step in the proposed MEC enumeration algorithm is shown in Table 6. Firstly, the MEC list is traversed to select connected MECs. Thus the time complexity of this step is $O(M)$. Then, the size and position information of each connected MEC is compared with the target task t to select candidate layers and stratify connected MECs into corresponding candidate layers.

Thus the time complexity of stratification is $O(m \times th)$. For each layer, the time complexity of stratified connected MEC update is $O(n + (w + m)^3)$, as proved in [11]. The maximum number of candidate layers that needs to be updated is th, which means all the layers from l_0 to t_{th} are candidate layers. Therefore, the time complexity for stratified MECs update is $O((n + (min(w, h) + m)^3) \times th)$. In the final extending

step, from the bottom layer to the top layer, each candidate layer is compared one by one. Thus, the time complexity is $O(m^2 \times th)$.

To ensure the efficiency of processing the MEC list, the total number of MECs M when n tasks are executing on the 3D DPR device should be explored. In Appendix, the upper limit of the number of MECs is proved to not be larger than $12n + 9$. Furthermore, Fig.15 shows the experimental results about the relationship between the number of tasks, MECs, and connected MECs, where the x-axis represents the number of tasks that are currently running on the 3D DPR device RD(100, 100, 100). The blue line provides confirmatory evidence that with the running task number increases, the number of MEC increases in a linear manner. Besides, from the red line, we can find that the number of connected MECs is far less than that of MECs, which proves that the connected MECs selection step effectively reduces the scope of exploration for subsequent steps.

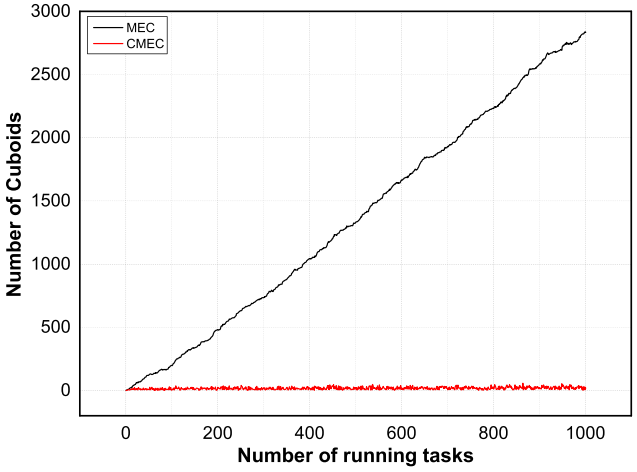


FIGURE 15. Relationship between the number of tasks, MECs, and connected MECs.

Thus the total time complexity of the above MEC enumeration algorithm is $O(n + (n + (min(w, h) + m)^3) \times th)$.

About the assignment strategies, the time complexity of the Best-Fit strategy is $O(n)$ since the MEC list is traversed to select the best one to assign the scheduled task. Additionally, the adjacency heuristic strategy needs to traverse the running task list, MEC list, and surfaces of the 3D DPR device to calculate the adjacent value $A(x, y, z, t)$, thus the time complexity is $O(n^2)$.

B. SPACE COMPLEXITY

Space complexity of the proposed algorithm consists of the following parts: 1) an MEC list C, 2) a layer list LA. Thus, the total space complexity is $O(M + m \times th)$.

V. EXPERIMENTAL RESULTS AND EVALUATION

A. EXPERIMENTAL SETUP

To evaluate the performance of the proposed task placement algorithm, we compare it with existing algorithms that focus on the 3D task placement problem. Based on the literature survey, we implement two algorithms for comparison. The first one is the 4D compaction algorithm [5], which has higher placement quality compared to other existing 3D task placement algorithms. Another one named 3D QC is an extension of the Quad-corner algorithm proposed in [6]. The Quad-corner algorithm has high efficiency when targeting on a 2D DPR device. Thus we extend it from 2D to 3D.

We construct an experimental framework in C language to implement our proposed and baseline algorithms. The experimental environment is macOS 10.15.1, GCC 4.8 on 1.4 GHz Quad-Core Intel Core i5 Processor with 8 GB Memory.

Experiments are performed on a 3D DPR device with $50 \times 50 \times 50$ reconfigurable units. There are five task sets, each containing 500 tasks. Moreover, each task is randomly generated according to the information given in Table 7, where w_r, h_r, th_r and lf_r are ranges of task's width, height, thickness and lifetime in each task set, respectively.

TABLE 7. Task sets for simulation.

Task set	w_r, h_r, th_r	$lf_r\,[ms]$	Number of tasks
TS1	[1-5]	[100-500]	500
TS2	[5-10]		
TS3	[5-15]		
TS4	[5-20]		
TS5	[10-20]		

Note that the size of tasks in TS1[1-5] are so small that the 3D DPR device RD(50, 50, 50) can accommodate all 500 tasks of TS1 simultaneously.

B. EVALUATION INDICATORS

Runtime overhead and placement quality of a task placement algorithm have a significant influence on the performance of the 3D DPR device. Thus, the average runtime and rejection ratio introduced below, are chosen as two evaluation indicators.

Average runtime is the sum of searching time and updating time, which is an indication of the speed of the task placement algorithm. Searching time is the average time to find an available position when a task is assigned. Updating time is the average time to manage free space after removing or assigning a target task.

Rejection ratio is used to evaluate the placement quality of the task placement algorithm. When a task can not find an available position until its given deadline, it will be rejected.The equation to calculate the rejection ratio is defined as follows:

$$Rejection\ ratio = \frac{\sum\limits_{\forall t_i \in REJECT} w_i \times h_i \times th_i \times lf_i}{\sum\limits_{\forall t_j \in TOTAL} w_j \times h_j \times th_j \times lf_j} \tag{4}$$

where $REJECT$ means the set of tasks that are rejected from the total task set $TOTAL$, where w, h, th, and lf represent the width, height, thickness, and lifetime of the corresponding task, respectively.

TABLE 8. Evaluation of average runtime.

Algorithm	Process	Average runtime (μs)									
		TS1[1-5]	Ratio	TS2[5-10]	Ratio	TS3[5-15]	Ratio	TS4[5-20]	Ratio	TS5[10-20]	Ratio
4D compaction [5]	**Total**	**11458**	249.09	**7840**	82.53	**4399**	29.73	**2717**	21.23	**2081**	37.16
	(Searching/updating)	(11456/2)		(7838/2)		(4397/2)		(2716/1)		(2081/<1)	
3D QC [6]		**24**	0.52	**169**	1.78	**161**	1.09	**186**	1.45	**187**	3.34
		(24/<1)		(168/1)		(158/3)		(181/5)		(178/9)	
Proposed (MEC/BF*)		**46**	1.00	**95**	1.00	**148**	1.00	**128**	1.00	**56**	1.00
		(1/45)		(5/90)		(3/145)		(2/126)		(< 1/56)	

* Our proposed task placement algorithm is the combination of the MEC enumeration algorithm and the Best-Fit strategy.

C. EVALUATION OF TASK PLACEMENT ALGORITHM

1) EVALUATION OF RUNTIME OVERHEAD

For each task set shown in Table 7, we generate 500 tasks and repeat 100 times to do experiments to get the average runtime. To evaluate the runtime overhead fairly, the deadline of each task is set to a value large enough to guarantee there is no task rejected on the 3D DPR device. Experimental results of total average runtime, which is the sum of searching time and updating time for each algorithm, are shown in Table 8, where the proposed task placement algorithm is the combination of the MEC enumeration algorithm and the Best-Fit strategy (MEC/BF).

As shown in Table 8, the total average runtime of proposed MEC/BF algorithm is 46 μs, 95 μs, 148 μs, 128 μs and 56 μs for the task sets TS1 to TS5, respectively. For all task sets, the speed of the MEC/BF algorithm is much faster than the 4D compaction algorithm, especially when the task size is small enough, the searching time of 4D compaction algorithm is greatly increased. The reason is that the 4D compaction algorithm has to traverse all scheduled tasks, running tasks and each RU on the 3D DPR device, to find the best position to assign arriving tasks. The time complexity of 4D compaction algorithm is $O(W \times H \times TH \times max(n_s \times n))$, where W, H, TH, n_s and n are the width, height, thickness of the 3D DPR device, the number of scheduled tasks, and running tasks, respectively. In addition, the 3D QC needs to traverse the running tasks on the 3D DPR device with time complexity $O(w \times h \times th \times n)$, where w, h, th is the width, height, and thickness of the assigned task, respectively. Therefore, when the task size is small enough, such as TS1 in Table 8, the 3D QC has the shortest total average runtime compared with 4D compaction and the proposed MEC/BF. Nevertheless, the speed of 3D QC is significantly affected by the task size so that when the task size becomes larger from TS2, the proposed MEC/BF algorithm performs better.

In order to better analyze the runtime overhead of the algorithms, Table 8 also shows the time spent by each algorithm on searching and updating.

The searching time of the MEC/BF algorithm is shorter than 4D compaction and 3D QC because the Best-Fit strategy finds the available position by searching an MEC list with a time complexity of $O(n)$.

From TS1 to TS5, with the task size becoming larger, the number of running tasks or scheduled tasks on the 3D DPR device is decreased. Thus, the searching time of the 4D compaction algorithm is significantly reduced. For the 3D QC algorithm, the searching time is related to not only the number of running tasks but also the task size. Therefore, from TS4[5,20] to TS5[10,20], the searching time of 3D QC is decreased significantly since the number of running tasks on the 3D DPR device is reduced even though the task size becomes larger. For the proposed MEC/BF algorithm, Fig. 16 shows the number of MECs corresponding to the number of tasks currently running on the 3D DPR device for the task sets TS1 to TS5, where the number of MECs for TS2 is considerably more than TS1, so that the searching time is increased from 1 μs to 5 μs. By contrast, from TS2 to TS5, the searching time is decreased with the decreasing number of MECs on the device.

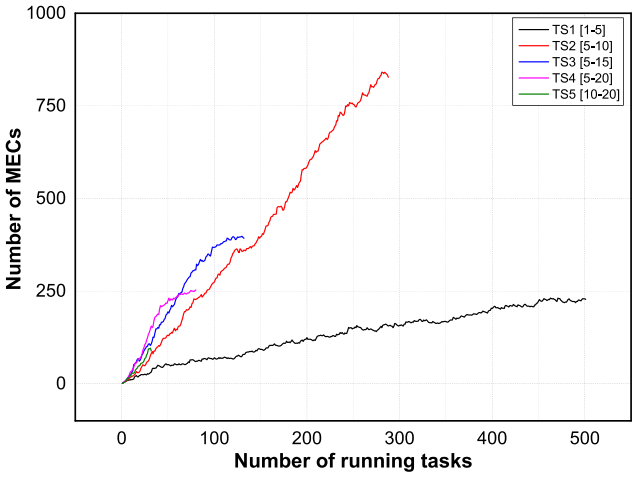


FIGURE 16. Number of MECs when task set TS1 to TS5 executed on the RD(50, 50, 50).

In terms of updating time, the proposed MEC/BF algorithm has the longest updating time. The reason is that the MEC enumeration process needs to search all connected MECs and update the information of them, while the 3D QC only needs to update the 3D matrix occupied by the assigned or removed task with time complexity $O(w \times h \times th)$.

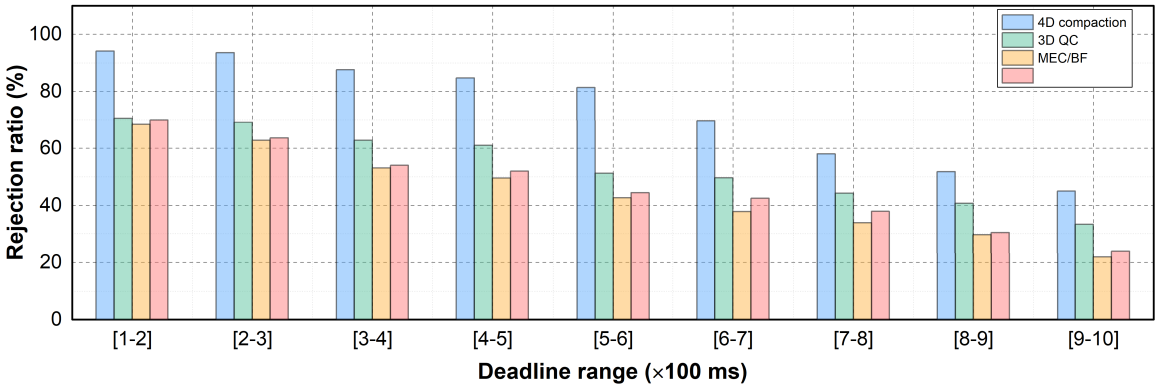


FIGURE 17. Rejection ratio with different deadline ranges.

The updating time of the 4D compaction algorithm is shorter than 3D QC and MEC/BF due to the updating operation completed by removing or adding a task into a scheduled task list or a running task list.

With the task size becoming larger, the updating time of 3D QC is longer than before since a larger task size means that a larger 3D matrix requires updating. From TS1 to TS3, the updating time of MEC/BF becomes longer, although the number of running tasks on the 3D DPR device gradually decreases. This is because the thickness of a task also has a significant influence on the runtime of the MEC enumeration algorithm. From TS4 to TS5, relative to the size of the 3D DPR device $50 \times 50 \times 50$, the size of the task is so large that only a small number of tasks can be executed on the device simultaneously, thus the updating time is gradually decreased compared with TS3.

In general, the runtime efficiency of proposed MEC/BF algorithm is much improved in comparison with 4D compaction and 3D QC algorithms.

2) EVALUATION OF PLACEMENT QUALITY

To explore the relationship between task deadline and rejection ratio, tasks with different deadline ranges [1, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 7], [7, 8], [8, 9] and [9, 10] ($\times$100 ms) are given based on the task's width, height and thickness ranges of [5-15]. Task arrival time interval (AT) is set to 300 µs, which means a new task arrives at the 3D DPR device every 300 µs. As shown in Fig.17, with the increase in task deadline, the rejection ratio of each method decreases gradually. For the reason that, if the deadline for each task is shorter, it is more difficult for the task placement algorithm to meet the deadline.

To explore the relationship between the task arrival time interval (AT) and rejection ratio, tasks with different ATs 100 µs, 200 µs, …, 1900 µs and 2000 µs are given. All the task's width, height, and thickness are randomly generated between [5-15], and the deadline is set between [1-10] ($\times$100) ms. As shown in Fig.18, the rejection ratio for each algorithm decreases with the increase of AT since a longer task arrival time interval allows more possibilities for running tasks to be finished, so the rejection ratio is lower.

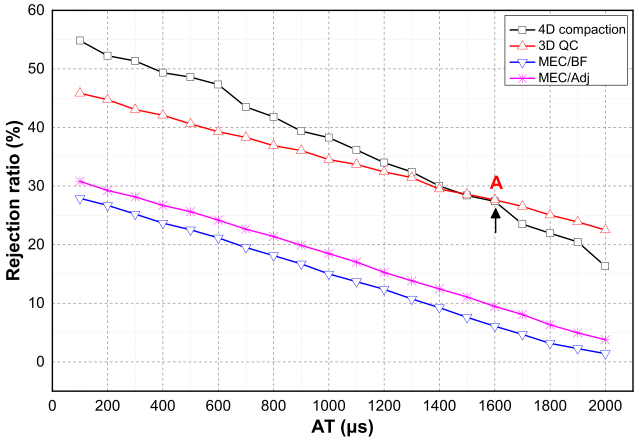


FIGURE 18. Rejection ratio with different task arrival time intervals (AT).

The data gathered in Fig. 17 and Fig. 18 suggests that proposed MEC/BF algorithm has the lowest rejection ratio. Compared with the previous works, the total average rejection ratio is reduced at least 39%. Moreover, the 4D compaction algorithm has the highest average rejection ratio. Although the 4D compaction algorithm places the arrival tasks more compactly on the 3D DPR device by considering the relative position between scheduled tasks and running tasks [5], it takes a much longer time to search for a feasible position for the current task than 3D QC and the proposed MEC/BF and MEC/Adj algorithms. The longer runtime means other new arriving tasks have to take more time to wait for calling so that it is much easier to be rejected. The rejection ratio of 3D QC is higher than MEC/BF and MEC/Adj algorithm because the limitation of searching candidates in 3D QC prevents the arriving task be placed although there is enough space on the 3D DPR device to accommodate it. In contrast, the proposed MEC/BF and MEC/Adj algorithms can assign an arriving task as long as there is enough space. The rejection ratio of the MEC/Adj algorithm is higher than MEC/BF, as the time complexity of adjacency heuristic $O(n^2)$ is higher than

Best-Fit $O(n)$, although the adjacency heuristic places the task more compactly than Best-Fit.

Fig. 18 demonstrates that the rejection ratio of the 4D compaction algorithm decreases faster than 3D QC. Moreover, from point A the rejection ratio of 4D compaction starts to be lower than 3D QC. The reason is that the 4D compaction algorithm has a runtime close to 3D QC due to the reduced number of running tasks on the device, and the algorithm places tasks more compactly than 3D QC. When the runtime advantages of 3D QC are not obvious, the gap in the rejection ratio is gradually decreased.

VI. CONCLUSION AND FUTURE WORK

In this paper, to solve the problem of online task placement on 3D dynamic partial reconfigurable devices, we propose a novel data structure named MEC and an efficient algorithm. When a task is assigned or removed, the connected MECs of the target task are selected, and processed by stratifying, updating, and extending. We ensure the efficiency of searching the MEC list by proving the upper limit of the number of MECs theoretically and experimentally. Furthermore, the experimental results demonstrate that the runtime overhead of the proposed task placement algorithm is greatly decreased, and the rejection ratio is at least reduced by 39% compared to 4D compaction [5] and 3D QC [6]. Therefore, the performance of the 3D DPR device can be greatly improved.

Since the task placement is only one step in the overall process of online task processing for dynamic partial reconfigurable devices, as part of our future work, a scheduling order based on task priority will be taken into consideration. Furthermore, in reality, tasks require frequent data transmission in most applications, therefore the development of a new placement strategy to further minimize task communication overhead is necessary.

APPENDIX

THE UPPER LIMIT OF THE NUMBER OF MECS

For a 2D planer placement problem, a *bar-visibility graph* mentioned in [20], [21] is redefined as a graph that the vertices of the graph correspond to the rectangle or the boundary of the 2D DPR device and two vertices of the graph are visible whenever there exists a vertical sightline between two rectangles corresponding to the vertices [11]. An edge between two vertices represents the visible relationship.

For a layout of tasks on a 3D DPR device, we consider the six boundaries of the device as six surfaces L, R, B, F, U and D (shown in Fig. 19(a)). According to the definition of the bar-visibility graph, the visible relationship can be represented as a left-right bar-visibility graph in the x-axis direction, a front-back bar-visibility graph in the y-axis direction and an up-down bar-visibility graph in the z-axis direction. We consider tasks and the six surfaces as vertices. Thus the 3D layout can be transformed into a *cuboid-visibility graph*, which is defined as follows:

Definition 6: A cuboid-visibility graph $G(V, E)$ is a union of the left-right bar-visibility graph $G_x(V, E_x)$, the front-back bar-visibility graph $G_y(V, E_y)$ and the up-down visibility graph $G_z(V, E_z)$, where its vertex set V is the same as G_x, G_y, or G_z and its edge set $E = E_x \cup E_y \cup E_z$.

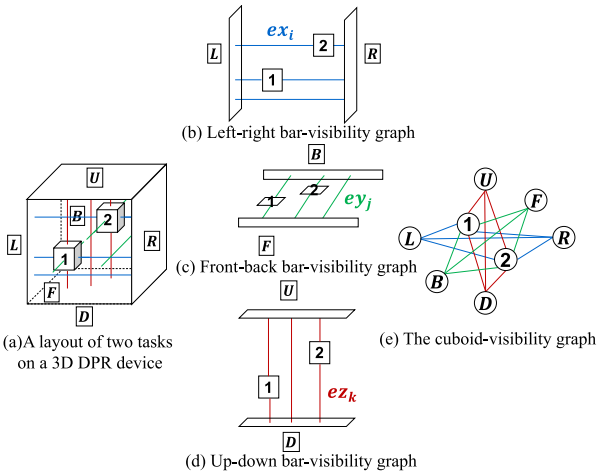


FIGURE 19. An example of transforming a layout into a cuboid-visibility graph.

Theorem 1 proved that each surface of an MEC must touch another task or the surface of the 3D DPR device. Thus, the edge between two vertices in a cuboid-visibility graph represents there exists at least one MEC that touch with the task or device's surface represented by the vertices.

Theorem 3: Given a 3D DPR device and n non-overlapping tasks placed on the device, where $n \geq 1$, the total number of MECs M is not larger than $12n + 9$.

Proof 3: According to Definition 6, we consider transforming the 3D DPR device into a cuboid-visibility graph $G(V, E)$, which is the union of a left-right bar-visibility graph $G_x(V, E_x)$, a front-back bar-visibility graph $G_y(V, E_y)$ and a up-down visibility graph $G_z(V, E_z)$, where E_x, E_y and E_z is the edge set of ex_i, ey_j and ez_k, respectively. Thus, the set of all MECs M on the 3D DPR device can be demonstrated by:

$$M = \left(\bigcup_{\forall ex_i \in E_x} Mex_i \right) \bigcup \left(\bigcup_{\forall ey_j \in E_y} Mey_j \right) \bigcup \left(\bigcup_{\forall ez_k \in E_z} Mez_k \right) \tag{5}$$

where Mex_i, Mey_j and Mez_k represent the MEC set that touch two surfaces connected by ex_i, ey_j and ez_k in the left-right bar-visibility graph, front-back bar-visibility graph and up-down bar-visibility graph, respectively.

According to Eq. 5, the number of MECs $|M|$ in set M satisfies the following Eq. 6.

$$\begin{aligned} |M| &\leq \sum_{\forall ex_i \in E_x} |M_{ex_i}| + \sum_{\forall ey_j \in E_y} |M_{ey_j}| + \sum_{\forall ez_k \in E_z} |M_{ez_k}| \\ &= |E_x| + \sum_{\forall ex_i \in E_x} (|M_{ex_i}| - 1) \\ &\quad + |E_y| + \sum_{\forall ey_j \in E_y} (|M_{ey_j}| - 1) \\ &\quad + |E_z| + \sum_{\forall ez_k \in E_z} (|M_{ez_k}| - 1) \end{aligned}$$

$$= \frac{1}{2} \times (|E_x| + |E_y| + |E_y| + |E_z| + |E_x| + |E_z|) + \sum_{\forall ex_i \in E_x} (|M_{ex_i}| - 1) + \sum_{\forall ey_j \in E_y} (|M_{ey_j}| - 1) + \sum_{\forall ez_k \in E_z} (|M_{ez_k}| - 1) \quad (6)$$

A *rectangle-visibility graph* mentioned in [22] is the union of two bar-visibility graphs and the total edge number of the rectangle-visibility graph is not larger than $6n+4$, which has been proved in [11]. For the 3D DPR device, any two of its three bar-visibility graphs can be considered as a rectangle-visibility graph. Thus, the number of edges $|E_x|$, $|E_y|$ and $|E_z|$ in three bar-visibility graph satisfies the following equation:

$$\begin{cases} |E_x| + |E_y| \leq 6n + 4 \\ |E_y| + |E_z| \leq 6n + 4 \\ |E_x| + |E_z| \leq 6n + 4 \end{cases} \quad (7)$$

The value of $(|M_{ex_i}| - 1)$ depends on the the number of tasks between two vertices that are connected by the edge ex_i in G_x, the same with the value of $(|M_{ey_j}| - 1)$ and $(|M_{ek_i}| - 1)$. When n tasks are placed on the 3D DPR device, the Eq. 8 is satisfied, which was already proved in [11].

$$\begin{aligned} \sum_{\forall ex_i \in E_x} (|M_{ex_i}| - 1) &= \sum_{\forall ey_j \in E_y} (|M_{ey_j}| - 1) \\ &= \sum_{\forall ez_k \in E_z} (|M_{ez_k}| - 1) \\ &\leq n + 1 \end{aligned} \quad (8)$$

Therefore, when there are n tasks placed on the 3D DPR device, where $n \geq 1$, the total number of MECs $|M|$, is not larger than $\frac{1}{2} \times (6n+4) \times 3 + 3(n+1) = 12n + 9$.

REFERENCES

[1] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, 2002.

[2] A. Rahman, S. Das, A. P. Chandrakasan, and R. Reif, "Wiring requirement and three-dimensional integration technology for field programmable gate arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 44–54, Feb. 2003.

[3] V. Pangracious, Z. Marrakchi, and H. Mehrez, "An overview of three-dimensional integration and FPGAs," in *Three-Dimensional Design Methodologies for Tree-based FPGA Architecture*. Cham, Switzerland: Springer, 2015, pp. 1–12.

[4] K. Sakuma, *3D Integration in VLSI Circuits: Implementation Technologies and Applications*. Boca Raton, FL, USA: CRC Press, 2018.

[5] T. Marconi and T. Mitra, "A novel Online hardware task scheduling and placement algorithm for 3D partially reconfigurable FPGAs," in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2011, pp. 1–6.

[6] T. Marconi, Y. Lu, K. Bertels, and G. Gaydadjiev, "A novel fast Online placement algorithm on 2D partially reconfigurable devices," in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2009, pp. 296–299.

[7] J. Tabero, J. Septién, H. Mecha, D. Mozos, and S. Roman, "A vertex-list approach to 2d hw multitasking management in rtr fpgas," in *Proc. Design Circuits Integr. Syst. (DCIS)*, 2003, pp. 545–550.

[8] M. Handa and R. Vemuri, "An efficient algorithm for finding empty space for Online FPGA placement," in *Proc. 41st Annu. Conf. Design Autom. (DA)C*, 2004, pp. 960–965.

[9] J. Cui, Z. Gu, W. Liu, and Q. Deng, "An efficient algorithm for Online soft real-time task placement on reconfigurable hardware devices," in *Proc. 10th IEEE Int. Symp. Object Compon.-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2007, pp. 321–328.

[10] X. Iturbe, K. Benkrid, T. Arslan, C. Hong, and I. Martinez, "Empty resource compaction algorithms for real-time hardware tasks placement on partially reconfigurable FPGAs subject to fault ocurrence," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Nov. 2011, pp. 27–34.

[11] T. Pan, L. Zeng, Y. Takashima, and T. Watanabe, "A fast MER enumeration algorithm for Online task placement on reconfigurable FPGAs," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E99.A, no. 12, pp. 2412–2424, 2016.

[12] S. Chiricescu, M. Leeser, and M. M. Vai, "Design and analysis of a dynamically reconfigurable three-dimensional FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 1, pp. 186–196, Feb. 2001.

[13] C. Ababei, H. Mogal, and K. Bazargan, "Three-dimensional place and route for FPGAs," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 25, no. 6, pp. 1132–1140, Jun. 2006.

[14] M. J. Alexander, J. P. Cohoon, J. L. Colflesh, J. Karro, and G. Robins, "Three-dimensional field-programmable gate arrays," in *Proc. 8th Int. Appl. Specific Integr. Circuits Conf.*, Sep. 1995, pp. 253–256.

[15] M. Leeser, W. M. Meleis, M. M. Vai, S. Chiricescu, W. Xu, and P. M. Zavracky, "Rothko: A three-dimensional FPGA," *IEEE Des. Test Comput.*, vol. 15, no. 1, pp. 16–23, Jan. 1998.

[16] K. Saban, "Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency," Xilinx, San Jose, CA, USA, White Paper wP380, 2011, vol. 1.

[17] T. Zhou, T. Pan, and T. Watanabe, "A fast Online task placement algorithm on 3D partially reconfigurable devices," in *Proc. TENCON*, Nov. 2017, pp. 427–432.

[18] R. Ayadi, B. Ouni, and A. Mtibaa, "Exploring the temporal placement for partially reconfigurable device," in *Proc. Int. Conf. Commun., Comput. Control Appl. (CCCA)*, Mar. 2011, pp. 1–4.

[19] J. Tabero, J. Septién, H. Mecha, and D. Mozos, "A low fragmentation heuristic for task placement in 2D RTR HW management," in *Proc. Int. Conf. Field Program. Logic Appl.*, 2004, pp. 241–250.

[20] D. G. Kirkpatrick and S. K. Wismath, "Weighted visibility graphs of bars and related flow problems," in *Proc. Workshop Algorithms Data Struct.* Ottawa, ON, Canada: Springer, 1989, pp. 325–334.

[21] S. K. Wismath, "Characterizing bar line-of-sight graphs," in *Proc. 1st Annu. Symp. Comput. Geometry (SCG)*, 1985, pp. 147–152.

[22] J. P. Hutchinson, T. Shermer, and A. Vince, "On representations of some thickness-two graphs," *Comput. Geometry*, vol. 13, no. 3, pp. 161–171, Sep. 1999.

TINGYU ZHOU (Student Member, IEEE) was born in Chongqing, China, in April 1995. She received the B.E. degree in software engineering from Sichuan University, in 2016, and the M.E. degree from the Graduate School of Information, Production and Systems, Waseda University, in 2017, where she is currently pursuing the Ph.D. degree. Since 2019, she has been a Research Associate with the IPS Research Center, Waseda University. Her research interests are in reconfigurable computing and run-time mapping algorithm.

TIEYUAN PAN was born in Donggang, China, in October 1988. He received the B.E. and M.E. degrees from The University of Kitakyushu, in 2012 and 2014, respectively, and the Ph.D. degree from the Graduate School of Information, Production and Systems, Waseda University, in 2017. From April 2015 to April 2017, he was a Research Associate with the IPS Research Center, Waseda University. He is currently working at DENSO Corporation. His research interest is in combinatorial algorithm for VLSI layout design and its applications.

MICHAEL CONRAD MEYER (Member, IEEE) received the B.S. degree in computer engineering and the M.A. degree in engineering management from the Rose-Hulman Institute of Technology, Terre Haute, IN, USA, in 2012 and 2013, respectively, and the Ph.D. in computer science and engineering from the University of Aizu, Fukushima, Japan, in 2017. He was a Postdoctoral Researcher with the the Data Networking Laboratory, University of Aizu. He was with Syntheon developing biomedical devices and Texas Instruments. He is currently an Assistant Professor with Waseda University. His research interests include on- and off-chip networks, reliability, and photonics.

YIPING DONG was born in Jiangsu, China, in 1983. He received the B.E. degree in electronics and engineering from Southeast University, China, in 2006, and the M.S. and Ph.D. degrees with the Graduate School of Information, Production and System, Waseda University, Japan, in 2008 and 2011, respectively. He worked with Waseda University in the fields of NoC, FPGA, and artificial neural networks. He joined the Research and Development Center, HITACHI Corporation, in 2012, where he worked in the fields of image processing and FPGA. He joined the China Key System & Integrated Circuit Company, Ltd., in August 2014, where he worked in the fields of design and application of field-programmable gate array (FPGA), and NoC design.

TAKAHIRO WATANABE (Life Member, IEEE) was born in Ube, Japan, in October 1950. He received the B.E. and M.E. degrees in electrical engineering from Yamaguchi University and the Dr. Eng. from Tohoku University. He joined the Research and Development Center, Toshiba Corporation, in 1979, where he worked in the field of LSI design automation. He joined the Department of Computer Science and Systems Engineering, Yamaguchi University, in August 1990. He moved to the Graduate School of Information, Production and Systems, Waseda University, where he is currently working, in April 2003. His current research interests are in EDA algorithm, microprocessor and MPSoC, NoC, FPGA, and their applications. He is a member of IEICE and IPSJ.