Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

A new efficient multi-task applications mapping for three-dimensional network-on-chip based MPSoC

Khadidja Gaffour[1] | Mohammed Kamel Benhaoua[1,2] | Abou El Hassan Benyamina[1] | Amit Kumar Singh[3]

[1]Department of Computer Science, University of Oran1 Ahmed Ben Bella, Oran, Algeria
[2]Department of Computer Science, University of Mustapha Stambouli, Mascara, Algeria
[3]School of Computer Science and Electronic Engineering, University of Essex, Essex, UK

Correspondence
Khadidja Gaffour, Department of Computer Science, University of Oran1 Ahmed Ben Bella, Oran, Algeria.
Email: gaffour.khadidja@edu.univ-oran1.dz

Summary

Three-dimensional Network-on-Chip (3D NoC) is a promising solution for solving 2D NoC problems while optimizing the system's performance. Mapping applications in 3D NoC is a crucial step as it has a significant impact on overall system performance. Moreover, multi-task supported processing elements (PEs) are needed to run multiple applications and provide more scalability. Most of the existing 3D mapping approaches consider only the single-task platform. In this paper, we propose an efficient multi-task mapping algorithm targeting regular 3D NoC that allows an incremental mapping and parallel execution of many applications onto different partitions on the 3D NoC. The proposed mapping algorithm is composed of three main steps aiming to reduce the communications overhead, exploiting the benefits of vertical links and improving the performances. The algorithm has been evaluated with various random and realistic benchmarks and compared with existing mapping algorithms for 3D NoC. The experimental results demonstrate that the proposed mapping strategy achieves significant performance in terms of communication cost, energy consumption and execution time.

KEYWORDS:
applications mapping, dynamic multi-task mapping, heuristics, multi-processors system-on-chip, 3D network-on-chip

1 | INTRODUCTION

Technological advancement allows us nowadays to realize more and more complex applications. This evolution was possible due to the miniaturization of transistors that allows integrating several billion on a single chip[1]. Therefore, it was possible integrating a complete system on a single chip called System-on-Chip (SoC). To meet the growing demands of applications, the SoC architecture has also become more complex and has gone from a single processor to a Multiprocessors System-on-Chip (MPSoC)[2]. Such systems increase performance by integrating multiple homogeneous or heterogeneous processors. On the other hand, Network-on-Chip (NoC) has been proposed as a solution to replace conventional interconnections in SoC and MPSoC based buses ensuring flexible communication and high required bandwidth[3,4,5]. The 2D-mesh is the most studied and used NoC topology because of its regularity and simplicity. Nevertheless, when the cores number increases, resulting in a large network size, the network performance can degrade drastically[6]. The use of the 3D NoC[7,8] is the envisaged solution in order to improve the performances in terms of reduction of power and average hop-count. A 3D NoC based MPSoC is implemented by stacking

multiple tiers above each other and vertically interconnects them using through-silicon vias (TSVs)[9]. As 3D NoC based MPSoC allows integrating more processors and run more applications, finding efficient and accurate applications mapping algorithm is extremely important.

In general, applications mapping algorithms can be classified into static [10,11,12,13,14] or dynamic [15,16,17,18,19] tasks mapping. The static or offline mapping defines the placement at design-time for all tasks of the requested applications. Such mapping techniques have a holistic view of the system that helps make better decisions about the system resources use. Therefore, these techniques are not suitable for the systems in which the workload of tasks changes dynamically. For these type of systems, dynamic or online mapping techniques is needed to find the application's tasks placement at run-time. On the other hand, regarding the number of tasks mapped per PEs, the mapping approaches can be also classified as single or multi-task. In single-task mapping, only one task is allocated to each PE, while in multi-task mapping, more than one task can be allocated to each PE. In order to support an increasing number of applications and allow parallel execution of many complex applications that contain a large number of tasks, the multi-task approaches are required.

At present, there are few papers dedicated to the dynamic mapping of tasks targeting regular 3D NoC. Also, in comparison with the single-task 3D NoC mapping, very little works have been done regarding the multi-task 3D NoC mapping. On the other hand, in 3D NoC, the vertical links are shorter and faster compared to horizontal links, which provide low power and network latency, also higher bandwidth. Therefore, the 3D NoC mapping must take advantages of the vertical links by mapping as much possible the most intensive communications to vertical links.

To overcome the limitations cited above, we present a new applications mapping algorithm for regular 3D NoC based MPSoC with multi-task supported PEs taking into account the benefits of vertical links. The proposed algorithm first needs to allocate a mapping position for the required application and then for each task by exploiting the benefits of multitasking and vertical links. This feature will allow the parallel mapping and execution of many applications while reducing the communication overhead and improving the performances. The proposed algorithm is composed of three steps: (i) Dynamic 3D NoC partitioning: which finds a contiguous partition while optimizing the layout of applications on 3D NoC (i.e., the size of partitions is determined in an optimal way to minimize fragmentation). (ii) Application task graph decomposition: which collects the most intensive communication tasks into groups. (iii) Multi-task groups mapping algorithm: which maps the groups of tasks on the selected partition's tiles, aiming to minimize the network traffic, the communication energy and maximized the performance.

The rest of this paper is organized as follows. Section 2 describes related works in 3D NoC mapping. The formulation problem and the energy model are introduced in Section 3. The proposed mapping algorithm is described in Section 4. Section 5 presents the experimental setup's description and results coupled with analysis discussions. Section 6 concludes the paper.

2 | RELATED WORK AND MOTIVATION

The problem of applications mapping on 2D mesh-based NoC has been addressed in several works [15,16,20,21,22]. However, most of these proposed mapping algorithms extension to 3D NoC is not optimal, since the 3D NoC design offers the vertical communication, which has a set of characteristics that are not considered in the 2D NoC. On the other hand, there are few papers for applications mapping onto 3D mesh-based NoCs.

The mapping algorithms presented in the literature use different optimization techniques such as branch and bound (BB), genetic algorithms (GA), simulated annealing (SA), particle swarm optimization (PSO) and heuristic methods.

Wadhwani et al [13] proposed an energy-aware application mapping based on branch and bound approach on a regular 3D mesh-based NoC architecture. Moreover, the authors expose the effectiveness of 3D NoC against 2D NoC in terms of dynamic communication energy consumption. This method needs a high computational complexity making it infeasible when the number of tasks grows large.

Genetic algorithm and simulated annealing are commonly used meta-heuristics for mapping problem. In the work [14], a low-power mapping for 3D NoC based on a combined genetic algorithm and simulated annealing is presented. The authors enhance the initial population selection of genetic algorithm and integrate the simulated annealing in the operator stage of genetic algorithm to take advantage of the local search ability of the simulated annealing. The experiments show that the proposed method has a significant effect in reducing power consumption. This algorithm is population-based greedy approaches that require high execution time, especially when the problem scale is large.

Regarding the PSO-based algorithms, Bhardwaj et al [23] proposed C3Map and H-C3Map energy-aware mapping algorithms for regular 3D NoC. C3Map is a centralized 3D mapping algorithm that uses octahedral traversal approach for mapping the

application's tasks to available cores. H-C3Map is the hybridization of C3Map with attractive and repulsive particle swarm optimization (ARPSO). This algorithm is limited since a good efficiency of the energy optimization can only be maintained when the scale of tasks graph is among a certain range.

On the other hand, many works have addressed the mapping problem resolution using heuristics techniques. Unlike the previous methods, the heuristic mapping techniques do not require a long time to find the solution. Murali et al[24] proposed Nmap, a heuristic algorithm which maps the tasks of given application onto a 2D mesh. This approach tries to map the tasks with more communication needs in the central tiles and the adjacent tasks closer to reduce the communication cost. Many works have been extended Nmap to 3D NoC[25,18,26]. This algorithm has high complexities, resulting in a huge increase in computation times.

Ziaeeziabari et al[18], proposed a latency-aware task mapping algorithm targeting a 3D NoC with partially-filled TSVs. In this algorithm, the application's task graph is partitioned into high-volume, low-volume and single communication sub-graphs. Then, the mapping of the resulted sub-graphs is established according to their total intra-partition communication. The proposed algorithm tries to map all the same partition's tasks into one layer while minimizing the use of vertical links due to their 3D NoC structure's nature. Dageleh et al[27] presented a clustering-based mapping algorithm for regular 3D NoC. First, a task clustering method called FLVAMOSA is applied to create a set of clusters as the number of 3D mesh layers. Then, each of the clusters is mapped to a 2D layer using an existing 2D mapping algorithm called Castnet. Finally, the arrangement of layers is modified in order to improve the mapping quality in terms of communication cost. In both these algorithms[18,27], all applications reuse the same NoC platform in distinct time slots. However, this situation results in an overhead caused by reconfiguring the NoC and loading new applications.

Crinkle[28] is a heuristic algorithm which is based on lists of prioritized tasks. First, the priority lists are organized according to the maximum communication volume or maximum out-degree. Then, depending on the priority lists, the tasks are mapped from the corner of 2D mesh platform and ends on another corner in a zigzag manner. Many works have been extended crinkle to 3D NoC[29,30,23]. Crinkle algorithm does not perform well for all task graphs. Generally, it works better on tasks graphs with hierarchical and consecutive nodes.

Kiani et al[19] proposed mapping of multiple applications onto irregular 3D wireless NoC. First, 3D NoC architecture is partitioned into several regions. Then, based on the number of simultaneous running applications, a re-partitioning of NoC architecture is established to adjust the number of partitions as much possible to the number of applications. Finally, the task mapping is executed, taking into account the processing elements inside the dedicated partition, otherwise, outside the partition based on the minimum path load criteria. This algorithm is specified only for irregular 3D wireless NoC architecture.

Tosun[31] proposed CastNet, a heuristic algorithm for mapping tasks onto mesh-based NoCs. The tasks are selected based on their communication weights and mapped to the shorted distance of the mapped tasks. Moreover, the approach mapping is applied on more than one partition based on the symmetry properties of the mesh. The mapping with minimum total energy consumption is selected as the final solution. CastNet has been extended to 3D NoC in the work[26]. As a result of its greedy performance, this mapping algorithm has a short execution time. But, it is limited in terms of scalability and generalization.

A dynamic incremental application-mapping algorithm for regular 3D NoC (we refer to this mapping technique as INC), is proposed in the work[17]. Here, the algorithm maps the tasks of a given application in a cuboid region while exploiting the vertical links as much as possible. However, the algorithm is divided into three steps: NoC region selection, which selects a convex region to map the requested application, then set matching step which allocates the most intensives communicating tasks to the vertical links and finally the mapping of tasks into the selected region while minimizing the communication power. The main drawback of this approach is that a single-task mapping is considered.

Singh et al[32] presented an efficient mapping algorithm for 3D video on 3D multicore intending to reduce the peak temperature and the consumption under throughput constraint of the application. First, the characteristics of the 3D video application and 3D architecture are extracted by using an offline analysis strategy. Then, according to this information, the tasks are mapped on cores based on the power distribution of cores and the thermal analysis. The proposed algorithm is based on application-driven methodology but without considering the mapping of multiple applications.

Synthesis.

Table 1 summarizes the reviewed works regarding the task mapping into 3D NoC. The reported works are classified according to the mapping nature (static or dynamic), the number of tasks allowed per PE, the target architecture's type and nature, NoC architecture's management and their mapping algorithm's optimization goal. We refer to the NoC architecture's management to the existence of any clustering or partitioning approach of NoC architecture for the parallel mapping and execution of multiple applications.

Table 1 highlights that most of the proposed works use static mapping algorithms [13,14,23,24,27,28,31]. Some of them [13,14,23] are based on GA, BB, PSO and SA . Although these methods are considered optimal mapping approaches, they impose very high execution time and computational complexity for large-scale graphs. Moreover, these methods are not able to manage dynamic workloads. To deal with these issues, heuristic-based dynamic mapping approaches can be used. Since, they require much lower computing time with dynamic mapping of tasks considering the run-time environments. Moreover, most of the reviewed works use a single-task mapping algorithm. But with the increasing number of processing elements and their capacities, the multi-task mapping approaches are required to handle the parallel execution of multiple applications. Also, few works have addressed the problem of mapping multiple applications in 3D NoC by investigating in the NoC architecture's management [17,19] to reduce the interference and the fragmentation between applications. On the other hand, the mapping algorithm targeting a regular 3D NoC should take advantage of the vertical links due to its high communication efficiency. Only one work [17] has devoted to the task mapping on regular 3D NoC taking into account the benefits of vertical links with the management of NoC architecture.

The present work proposes a multi-task applications mapping for regular 3D NoC with homogeneous cores. The proposed mapping algorithm allows an incremental mapping of multiple applications, using a management technique to allocate efficiently the placement of applications. Besides, a decomposition of the application's task graph is applied to create a communicating-groups set where each group's tasks will be mapped on the same PE. Our mapping algorithm tries placing the communicating groups close to each other, in order to reduce the network traffic and minimize communication power. Also, the proposed mapping explores accurately the vertical links use.

3 | SYSTEM MODEL

This section introduces some definitions for a proper understanding of the proposed mapping strategy.

3.1 | NoC architecture and application model

A three-dimensional Network-on-Chip topology structure consists of several tiles (processing elements, routers and wire links) connected in a grid-like fashion. 3D NoC with dimension $N \times M \times K$ consists of stacking multiple 2D mesh layers with dimension $N \times M$ connected by vertical links. Whereas, the length of vertical links between tiles in adjacent layers is shorter than horizontal links between neighbouring tiles in the same layer. A 3D NoC is defined by the network architecture graph.

Definition 1: A 3D NoC topology $TP(N, M, K)$ is represented by a network architecture graph $ARG = (L, E)$, where vertices L are the tiles set, with $|L| = N \times M \times K$ and the edge $e_{ij} \in E$ presents the physical channel between tile (l_i, l_j).

Application is a set of tasks where each task needs to be mapped in NoC architecture. Each application is characterized by the task graph defined below.

Definition 2: An application task graph is represented as a directed acyclic graph $ATG = (T, D)$, where T and D are the set of all tasks and edges in the application, respectively. A task t_u is defined by the parameters (t_{id}, REQ_u), where t_{id} represents the task identifier and each $req_{uv} \in REQ_u$ determines the number of cycles to be executed by the task t_u before it sends the data to its subsequent task t_v. The task t_v start its execution after it receives the complete amount of data. Each directed arc $d_{uv} \in D$ designates the existence of data communication between the two tasks t_u and t_v, and the size of data to be transferred between them is represented by the weight Q_{uv}. For example, the graph of tasks VOPD is illustrated in Figure 1.

3.2 | Energy model

The overall NoC system energy is composed of computation energy and communication energy. The computation energy is the energy consumed by the PEs, whereas, the communication energy is the energy consumed by network resources in the traffic transmission. In this study, we focus on minimizing the communication energy since it has a significant impact on the total energy [33].

$$E_{total} = E_{total}^{comp} + E_{total}^{comm} \tag{1}$$

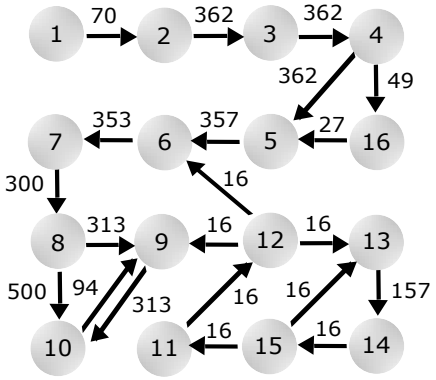

FIGURE 1 VOPD application task graph

3.2.1 | Computation energy

We have estimated the computation energy consumed by using the model proposed in the work[34].

$$E_{pe_i} = (n_i^{run} \times E_{run}) + (n_i^{idle} \times E_{idle}) \tag{2}$$

Where n_i^{run} represents the number of cycles in which the pe_i is in running mode and n_i^{idle} corresponds to the number of cycles in which the pe_i is an idle mode. Whereas, E_{run} and E_{idle} are the energy consumption of PE in running and idle mode, respectively.

The total computation energy consumption is measured as the sum of the energy consumption of all the PEs.

$$E_{total}^{comp} = \sum_{i=1}^{|L|} E_{pe_i} \tag{3}$$

3.2.2 | Communication energy

The communication energy consumption of transfer of one bit of data is given by:

$$E_{bit} = E_{bit}^R + E_{bit}^B + E_{bit}^W + E_{bit}^L \tag{4}$$

Where E_{bit}^R, E_{bit}^B, E_{bit}^W, and E_{bit}^L denote the energy consumed by the routers, buffers, internal wires, and the links, respectively. Actually, the energy consumption E_{bit}^B and E_{bit}^W are negligible compared to E_{bit}^R and E_{bit}^L, hence, Equation (4) can be reduced to:

$$E_{bit} = E_{bit}^R + E_{bit}^L \tag{5}$$

According to Equation (5), the energy consumption of transmitting a single bit from source tile (l_s) to destination tile (l_d) can be computed using the following equation:

$$E_{bit}^{l_s,l_d} = (MD_{l_s,l_d} + 1) \cdot E_{bit}^R + N_H \cdot E_{bit}^{HL} + N_V \cdot E_{bit}^{VL} \tag{6}$$

E_{bit}^{HL} and E_{bit}^{VL} are the energy consumed on horizontal and vertical links in data transmission. MD_{l_s,l_d} is the Manhattan distance from tile l_s to l_d, while N_H and N_V represent the number of horizontal and vertical links that have passed. As described in[35], E_{bit}^L can be computed as: $E_{bit}^L = dV^2C_{wire}/2$, where d is wire length, V is power supply and C_{wire} is link capacitance.

Assuming that the coordinates of source and destination tiles are (x_s,y_s,z_s) and (x_d,y_d,z_d) respectively, then MD_{l_s,l_d} can be calculated as:

$$MD_{l_s,l_d} = |x_s - x_d| + |y_s - y_d| + |z_s - z_d| \tag{7}$$

Let t_i and t_j are the tasks mapped on tile l_s and l_d, thus, the total communication energy is :

$$E_{comm}^{l_s,l_d} = Q_{ij} \cdot E_{bit}^{l_s,l_d} \tag{8}$$

Finally, the network total energy's consumption with traffic of N communication transactions is :

$$E_{total}^{comm} = \sum_{1}^{N} E_{comm}^{l_s,l_d} \tag{9}$$

3.3 | Execution time model

The execution time of given application T_{app} is computed from root to leaf task based on the critical branch of the ATG. Hence, to compute the total execution time, we have considered the configuration time and time executions of tasks including mapping time, the computation time and the communication time. The execution time for each task is calculated recursively as follows:

$$Time_{t_i} = T_{map}^{t_i} + T_s^{t_i} + T_c^{t_i} + \sum_{j=1}^{|sub_{t_i}|} \text{MAX}\ \{Time_{t_j}\} \tag{10}$$

Where $T_{map}^{t_i}$ is the time to find a placement, $T_s^{t_i}$ is the time needed to processes the task t_i and $T_c^{t_i}$ is the communication time spent of task t_i with its subsequent tasks sub_{t_i}.

The overall execution time of multiple applications corresponds to the highest execution time among all applications running in parallel. In the case of unavailability of free resources, the waiting time T_w is also considered.

$$Time_{total} = \underset{i=1 \text{ to } |applications|}{\text{MAX}} \{T_{app}\} + T_w \tag{11}$$

3.4 | Application mapping

The application mapping problem is defined as follows:

Problem: Given an application task graph $ATG = (T, D)$ and NoC architecture graph $ARG = (L, E)$ those satisfy:

$$|T| \leq |D| \tag{12}$$

Find a mapping function $map : T \rightarrow L$ with the optimization function:

$$min : \{\sum_{m=1}^{|D|} Q_{ij} \cdot E_{bit}^{map(t_i),map(t_j)}\} \tag{13}$$

Such that:

$$\forall t_i \in T, map(t_i) \in L \tag{14}$$

4 | PROPOSED MAPPING APPROACH

In this section, we describe in details our proposed multi-task 3D mapping approach that aims at reducing the network traffic and communication energy. The approach is constituted of three steps *dynamic 3D NoC partitioning*, *application task graph decomposition* and *multi-task groups mapping algorithm*. The main parameters used in the proposed approach are described in Table 2.

Step 1. *Dynamic 3D NoC partitioning*: Define a 3D contiguous partition intended for the mapping, according to the application's characteristic.

Step 2. *Application task graph decomposition*: Divide the application task graph into a set of groups while minimizing the inter-group communication.

Step 3. *Multi-task groups mapping algorithm*: Map the group of tasks into the selected partition's tiles with minimized communication energy.

4.1 | Dynamic 3D NoC partitioning

The mapping and the execution of multiple applications require a partitioning mechanism in order to reduce the interference between them. All the tasks of the same application are mapped in the same partition. Let's suppose that the NoC partitioning is not carried out, in this case, the communicating tasks of an application will be mapped in sparse tiles, which increase the cost of communication. Besides, in the case of complex applications containing a large number of tasks, the multi-task mapping is needed to improve the running of the applications. Instead of work[17], we use the dynamic 3D NoC partitioning considering a multi-task operating system.

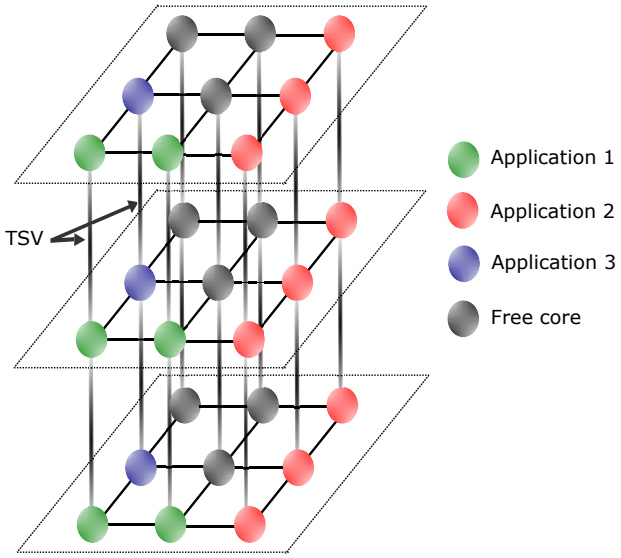

FIGURE 2 Dynamic 3D NoC partitioning step where applications 1, 2 and 3 contains 16, 28 and 8 tasks respectively

In the proposed algorithm, for each incoming application, a contiguous partition with dimension $\alpha \times \beta \times K$ should be located satisfying the following two conditions:

$$min : \{\alpha \cdot \beta \cdot K - |T|\} \tag{15}$$

$$\alpha \cdot \beta \cdot K \geq |T| \tag{16}$$

Three metrics are used to define the partition's shape: the number of tasks allowed per PE, the number of layers and the number of application's tasks. The algorithm retrieves the application's information by offline profiling. The step's details of dynamic 3D NoC partitioning are given in Algorithm 1. First, the size and the dimension of the partition $\alpha \cdot \beta$ in one layer is computed (lines 1-8). Next, a search process for free tiles is established starting by the bottom layer (lines 9-11). Then, the full partition with dimension $\alpha \cdot \beta \cdot K$ will be determined (line 12). The goal of starting the search process by the bottom layer is the creation of a partition that contains the greatest possible number of vertical links. Finally, the algorithm keeps track of free tiles by updating the unpartitioned nodes list after the creation of each partition (line 13).

Figure 2 illustrates the running of the proposed partitioning algorithm in regular 3D NoC architecture including three layers, where 3 tasks are allowed per each PE considering three applications task graphs with 16, 26 and 8 tasks. The applications 1, 2 and 3 are mapped into NoC partitions with sizes 1x2x3, 3x1x3 and 1x1x3 respectively. As we can see, all applications are mapped in contiguous partition, which decreases significantly the communication cost and enhances the system's performance.

4.2 | Application task graph decomposition

To reduce communication overheads, complex applications with a large number of tasks should be partitioned and mapped efficiently. Therefore, this step's goal is the decomposition of the application task graph into a set of groups where the most communicating tasks are in the same group and the inter-group communications are minimized.

The proposed algorithm divides the application task graph $ATG = (T, D)$ into a set of disjoint communicating multi-task groups $CG = \{\varphi_0, \varphi_1, ..., \varphi_{N_G-1}\}$, such that:

$$N_G \leq |L'| \tag{17}$$

$$1 \leq NT_G \leq \theta_{PE} \tag{18}$$

$$\forall \varphi_i, \varphi_j \in \text{CG and } i \neq j, \varphi_i \cap \varphi_j = \emptyset \tag{19}$$

Where N_G and NT_G are the number of groups and the number of tasks in each group respectively.

Algorithm 1 Dynamic 3D NoC partitioning

Input: $ATG = (T, D)$: Application task graph of the requested application
$ARG = (L, E)$: Network architecture graph
θ_{PE} : Maximum number of tasks allowed per PE
`UNP_List` : The list of unpartitioned tiles
Output: $PR = (L', E')$: Resulted partition intended for the mapping

▷ *Step1: compute the size and the dimension of the partition in one layer*
1: $\rho = |T| \,/\, (\theta_{PE} \times K)$; ▷ Compute the number of tiles needed
2: **for** $\alpha = 1$ to $\rho/2$ **do** ▷ Compute the size of the partition in one layer
3: **for** $\beta = 1$ to $\rho/2$ **do**
4: **if** $\alpha \times \beta \geq \rho$ **then**
5: Break from α loop;
6: **end if**
7: **end for**
8: **end for**

▷ *Step2: find the partition with the dimension (α, β,K)*
▷ We assume the coordinate of P_i is $(x, y, 0)$
9: **for all** $P_i \in UNP\_List$ **do**, select P_i where: ▷ Search for free tiles in the bottom layer
10: `bottom_partition` ← `select_free_partition`$(x + \alpha, y + \beta, 0)$ **or** `select_free_partition`$(y + \alpha, x + \beta, 0)$;
11: **end for**
12: $PR \leftarrow$ `get_full_partition(bottom_partition)`; ▷ Determine the full partition

▷ *Step3: Update the list of unpartitioned tiles*
13: Update(`UNP_List`);

Algorithm 2 shows the pseudo-code of the application task graph decomposition. In this algorithm, for given application task graph and a number of tasks allowed per PE represented by θ_{PE}, the communicating multi-task groups will be generated. The algorithm tries including up to θ_{PE} tasks with a minimum of one task in each group as described by the equation 18. Initially, the edges of the task graph are sorted on decreasing order of their communication weight and all tasks are labelled as unmarked (lines 1-3). Then, for each edge in sorted order (d_{uv}), a multi-task group is created including the unmarked tasks t_u and t_v (lines 7-8). In order to select the other tasks to integrate into the multi-task group created, first the neighbour's tasks are found (line 10) and they are chosen in descending order by their communication volume. Two cases must be considered: in the case of $\theta_{PE} = 2$, the selection for other tasks is not performed and the creation process for the current edge is stopped. Otherwise, the most communicating tasks will be assigned to the created group one by one according to their communication volume with tasks in the group until the number of tasks in the created group reaches the value θ_{PE} (lines 9-20). In the situation, if more than one communicating task having the same communication volume are selected, then the algorithm chooses the task having the minimum number of unmarked neighbours (lines 15-17). The aim of this decision of selection is to retain the tasks with a large number of unmarked neighbours in order to select them for the creation of other multi-task groups. Tasks are labelled as marked since they are assigned to a multi-task group (line 19). Finally, each created multi-task group is inserted into the communication group list CG (line 22).

We illustrate the functioning of our proposed application tasks graph decomposition with an example in Figure 3. Considering the VOPD application graph with 16 tasks given in Figure 1 and NoC platform with 3 tasks allowed per PE. The sorted edges list of ATG is: {(8,10), (9,10), (2,3), (3,4), (4,5), (5,6), (6,7), (8,9), (7,8), (13,14), (1,2), (4,16), (16,5), (11,12), (12, 9), (12,6), (12,13), (14,15), (15,16)}. The edge (8, 10) has the highest communication weight, thus, a multi-task group is created containing the tasks t_8 and t_{10}. Since the maximum task allowed per PE is greater than the size of the created group, the most unassigned communicating tasks with t_8 and t_{10} are added. Therefore, the first group φ_1 contains the tasks $\{t_8, t_{10}, t_9\}$ as depicted in Figure 3(1). In step 2, the edge (2, 3) is selected for the creation of the second multi-task group. We note that the edge (9, 10) is bypassed since the tasks t_9 and t_{10} are already assigned. The most communicating task with tasks t_2 and t_3 is task t_4. Thus, the second group φ_2 includes the tasks $\{t_2, t_3, t_4\}$. In step 3, the edges (3, 4) and (4, 5) are bypassed as the tasks t_3 and t_4 are assigned,

therefore, the edge (5, 6) is selected. The third multi-task group φ_3 includes the tasks $\{t_5, t_6, t_7\}$. The same strategy is followed for creating the multi-task group in step 4. In step 5, the edge (11, 12) is selected and the edges (1, 2), (4, 16), (16, 5) are bypassed. The fifth multi-task group φ_5 contains only the tasks $\{t_{11}, t_{12}\}$ because all their communicating tasks are already attributed in other groups. At this stage, only the tasks t_1 and t_{16} are not assigned. Therefore, the sixth and the seventh groups contain the task $\{t_1\}$ and $\{t_{16}\}$ respectively (steps 6 and 7).

Algorithm 2 Application task graph decomposition

Input: $ATG = (T, D)$: Application task graph of the requested application
θ_{PE} : Maximum number of tasks allowed per PE
Output: CG : List of communicating multi-task groups

1: Unmark all tasks in ATG;
2: Put each task ($\forall t_i \in T$) in separate group π_i;
3: Sort all edges of ATG in descending order according to their communication weight;
4: **for each** $d_{uv} = (t_u, t_v) \in D$ **do**
5: $\varphi \leftarrow \emptyset$ ▷ Initialization of the group φ
6: **if** (t_u and t_v are not marked) **then** ▷ Checking each edge if it is labelled as unmarked
7: $\varphi \leftarrow \pi_u \cup \pi_v$
8: mark(t_u and t_v);
9: **while** ($|\varphi| < \theta_{PE}$) **do** ▷ Assignment of tasks in the group until their number reaches the value θ_{PE}
10: `candidate_task_list` ← get the list of unmarked tasks which communicate with tasks in φ;
11: **for all** ($ct_i \in candidate\_task\_list$) **do** ▷ Selection of the dominant task among all neighbours of tasks in φ
12: $\texttt{comm\_vol}_{ct_i}$ ← communication volume of ct_i with tasks in φ;
13: Select dominant task t_i or tasks which have the maximum $\texttt{comm\_vol}_{ct_i}$;
14: **end for**
15: **if** more than one task is selected **then**
16: t_i ← choose the task which has a minimum of unmarked neighbours.
17: **end if**
18: $\varphi \leftarrow \varphi \cup \pi_i$;
19: mark(t_i); ▷ Each assigned task is labelled as marked
20: **end while**
21: **end if**
22: insert (φ to `CG`); ▷ Update the list of communicating multi-task groups
23: **end for**

4.3 | Multi-task groups mapping algorithm

After the decomposition of application tasks graph into multi-task communicating groups, the step of the mapping is carried out. Differently from the state-of-the-art algorithms, the proposed approach can handle multiple tasks simultaneously by taking mapping decisions for each group of tasks instead of each task separately. As mentioned in the second step, highly communicating tasks are collected in the same groups. Therefore, the algorithm tries to map the communicating groups close to each other while the tasks of each group are mapped in the same tile. The placement of the most communicating tasks into the same tile will reduce the communication overhead significantly due to their fast communication. Further, we consider that the length of the vertical links is shorter and faster than horizontal links. Hence, for the intensive inter-group communications the vertical links are favoured over horizontal links.

The proposed mapping described in Algorithm 3 takes the NoC partition architecture graph (PR), application task graph (ATG) and the list of decomposed groups of tasks (CG) as inputs and efficiently performs the mapping of the groups of tasks on the free tiles of the NoC partition. First, the algorithm searches and selects the initial multi-task group to be mapped (line 5).

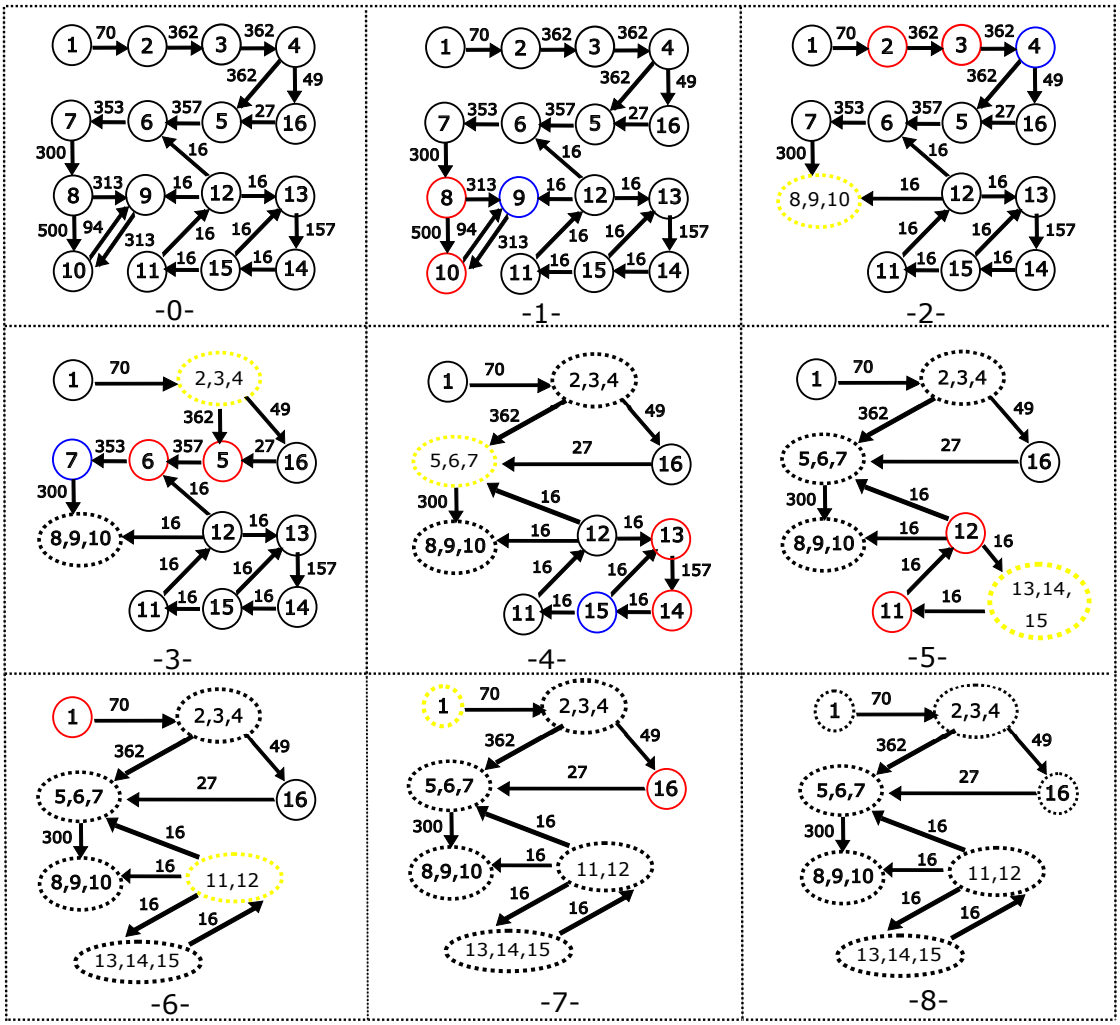

FIGURE 3 Application task graph decomposition step

Then, the multi-task groups are mapped one by one in descending order, according to their total inter-communication volume with mapped one (lines 6-43).

The multi-task groups are distinguished based on their total intra-communication and inter-communication. We assume that each group $\varphi_i \in CG$ is a connected sub-graph of application task graph, ATG = G(T,D). The following equations are used to compute the total intra-communication v_{φ_i} and inter-communication η_{φ_i} for each multi-task group.

$$v_{\varphi_i} = \sum_{\forall d_{uv} \in \varphi_i} Q_{uv}, 1 \leq i \leq N_G \tag{20}$$

$$\eta_{\varphi_i} = \sum_{\substack{\forall t_u \in \varphi_i \\ t_v \in \varphi_j \\ i \neq j \\ d_{uv} \in D}} Q_{uv}, 1 \leq i, j \leq N_G \tag{21}$$

The selection and the mapping of the first multi-task group are explained with Procedure 1. For this selection, the group with the highest total inter-communication is selected as the initial multi-task group to be mapped. If multiple groups are selected, then the algorithm chooses the group which has the maximum intra-communication (lines 1-4).

The selection of the initial tile depends on the total number of horizontal and vertical links that the initial multi-task group requires for inter-communication (lines 5-13 in Procedure 1). Hence, the algorithm computes the intra-communication volume between the initial multi-task group and its neighbour's groups (line 7 in Procedure 1). If the communication volume is greater

than or equal to the average communication, the number of vertical links is increased (lines 8-9). Otherwise, the number of horizontal links is increased (lines 10-11). The average communication volume is computed as follow:

$$\bar{A}_{comm} = \sum_{\substack{\forall d_{uv} \in D \\ u \neq v}} Q_{uv} / |D| \tag{22}$$

Then, the algorithm searches for a suitable tile inside the selected partition, with the same computed number of vertical and horizontal links (line 14). If the tile is not found, the required number of horizontal links is minimized, until a suitable tile is found. Otherwise, the required number of vertical links is minimized. For example, let $N_H = 1$ and $N_V = 2$ where N_H and N_V denote the required number of horizontal and vertical links respectively. In the search process, the algorithm evaluates the following combination until a suitable tile (L) is found: $\{L^{N_H=1}_{N_V=2}, L^{N_H=0}_{N_V=2}, L^{N_H=1}_{N_V=1}\}$. Finally, each task in the initial multi-task group will be mapped to the initial tile and resources statue are updated (lines 15-17).

After the initial mapping has been executed, the algorithm maps the remaining multi-task groups one by one. Among the unmapped multi-task groups, the algorithm selects the group with highest inter-communication volume with its mapped neighbour's groups. If more than one multi-task group is selected, the group with the maximum intra-communication v_{φ_i} is chosen to be mapped (lines 7-10).

The proposed algorithm is based on a two-search process to select the most suitable tile for the chosen multi-task group. When the multi-task group has only one communicating neighbour group already mapped, the algorithm tries to map the tasks of the chosen multi-task group on the same tile as of its communicating neighbour group. If the resource at the same tile is not able to support the tasks, the algorithm computes the communication volume between the target multi-task group and its communicating neighbour group. If the computed communication volume is greater or equal to the average communication $\bar{A}_{comm}$, a search for free vertical tile at one hop distance from already mapped communicating neighbour is carried out. Otherwise, the algorithm searches for horizontal tile at one hop distance (lines 13-25). If the multi-task group to be mapped has multiple neighbours groups already mapped, then the search process takes the location of all its communicating groups to find a suitable tile (lines 27-41). Thus, each free tile of NoC partition is examined and the tasks of the selected group will be placed onto the tile that minimizes the communication cost with mapped groups. The communication cost is computed by Equation 8. Note that, after each mapping, the status of the resources is updated (line 43). The same strategy as described above is applied until no unmapped multi-task group remain in the CG list.

For multiple applications mapping, the steps described above are applied for each application. First, a suitable NoC partition is found by Algorithm 1. Then, the application graph is decomposed into communicating multi-task groups by Algorithm 2. Finally, the tasks of each group are mapped as described by Algorithm 3.

An example of our proposed mapping algorithm is shown in Figure 4. We use the decomposed VOPD task graph resulted from the execution of step 2 as shown in 3. A 3x3x3 NoC is used with 3 tasks allowed per each PE. As the number of VOPD tasks is 16, a NoC partition with size 1x2x3 is selected for the mapping. In Figure 4, at step 1, the first multi-task group to be mapped is φ_3, since, it has the highest inter-communication. The multi-task group φ_3 needs two vertical links and one horizontal link to communicate with their neighbours as explained in Procedure 1. Thus, the tasks $\{t_5, t_6, t_7\}$ are mapped to tile L'_9. In step 2, the next multi-task group to be mapped is φ_2 as it is the most communicating group with group φ_3. As the communication volume between φ_2 and φ_3 is greater than the average communication volume, then, a vertical link is needed. So, the tasks of group φ_2 are mapped to tile L'_0. In the same manner, in step 3, the tasks of group φ_1 are mapped to tile L'_{18}. In step 4, the group φ_7 is selected to be mapped. Then we find the candidate tile location on the mesh partition. The group φ_7 has two mapped neighbours groups φ_2 and φ_3. According to the equation 8, the best tile location to map the tasks of group φ_7 is L'_1. In the same manner, we map the tasks of groups φ_6, φ_5 and φ_4 in tile L'_1, L'_{10} and L'_{19} respectively as shown in steps 5, 6 and 7.

5 | COMPLEXITY ANALYSIS

In this section, we present the complexity analysis of our proposed approach composed of three Algorithms.

In Algorithm 1, there are three main loops which must be considered in the worst-case analysis of this algorithm. The run time for the first set of nested *for loop* depends on the number of NoC tiles in the bottom layer. Hence, the complexity of lines 2 to 8 is $\mathcal{O}(|L|/|K|)^2$. The second *for loop* (Lines 9 to 11) consists of searching free tiles in the bottom layer. For that, the complexity is $\mathcal{O}(|L|/|K|)$. However, in the worst case, the complexity of the Algorithm 1 is $\mathcal{O}(max(|L|/|K|, (|L|/|K|)^2) = \mathcal{O}(|L|/|K|)^2$.

Procedure 1 Initial mapping

Input: ATG, PR, CG

Output: $map : \forall t_i \in ctg_{first} \rightarrow tile_{first} \in L'$

1: $ctg_{first} \leftarrow$ select a dominant $\varphi_i \in CG$ wich has maximum η_{φ_i}; ▷ Select a dominant group according to the total inter-communication
2: **if** more than one φ_i is selected **then** ▷ If multiple dominant groups, the selection is according to their total intra-communication
3: choose φ_i which has a maximum ν_{φ_i};
4: **end if**
5: $\psi \leftarrow$ `get_neighbours`(ctg_{first}); ▷ Get all neighbours groups of the initial multi-task group
6: **for all** $cg_i \in \psi$ **do** ▷ Compute the required number of horizontal and vertical links
7: $comm\_vol_{cg_i} \leftarrow compute\_comm(cg_i, ctg_{first})$;
8: **if** $comm\_vol_{cg_i} \geq \bar{A}_{comm}$ **then** ▷ Case of intensive communication
9: $N_V \leftarrow N_V + 1$; ▷ The number of vertical links is increased
10: **else** ▷ Case of low communication
11: $N_H \leftarrow N_H + 1$; ▷ The number of horizontal links is increased
12: **end if**
13: **end for**
14: $tile_{first} \leftarrow$ search_tile(PR, N_V, N_H); ▷ Search of the corresponding tile
15: Map each task t_i of group ctg_{first} to tile $tile_{first}$;
16: $insert(tile_{first}$ to map) and $update$(resource by map); ▷ Update the status of the resources

Algorithm 2 has partial dependency on the complexity of the sorting process Line 3 which is $\mathcal{O}(|D| \log_2 |D|)$. On the other hand, the *for loop* (Lines 4 to 23) has a nested *for loop* inside *while loop* (Lines 11 to 14 and Lines 9 to 20). The run time of *while loop* depends on θ_{PE}, whereas the inner and the outer loop depend on the number of tasks and edges, respectively. Considering the worst case, the *while loop* (Lines 9 to 23) runs $\mathcal{O}(\theta_{PE} \times |T|)$. As the value of θ_{PE} is constant, the outer *for loop* (Lines 4 to 23) has a complexity of $\mathcal{O}(|T||D|)$. Therefore, in the worst case, the complexity of Algorithm 2 is $\mathcal{O}(|T||D|)$ since the complexity of sorting process is lower.

In Algorithm 3, the step of computing the total inter and intra-communication (Lines 1 to 4) is of the order $\mathcal{O}(N_G)$. The Procedure *initial_mapping* consists of selecting the initial multi-task group and searching a suitable tile for the mapping inside the selected partition (Line 5). The selection of dominant multi-task group takes $\mathcal{O}(N_G)$, and the selection of tile depends on the number of horizontal and vertical links that the selected group requires for inter-communication. For that, the Procedure checks only the edges set of the selected group, which means that it has the worst time complexity of $\mathcal{O}(|D|)$. Moreover, there are three main loops which must be considered in the worst-case analysis of the Algorithm 3 which consist of mapping the remaining multi-task groups. Hence the complexity of the nested loops (Lines 6 to 44) is $\mathcal{O}(|N_G||L||D|)$. In the worst case, the complexity of Algorithm 3 is $\mathcal{O}(|N_G||L||D|)$.

Therefore the complexity of the proposed approach (Algorithm1 + Algorithm 2 + Algorithm 3) is $\mathcal{O}(|T||D| + |N_G||L||D|)$.

6 | EXPERIMENTAL RESULTS AND ANALYSIS

6.1 | Simulation Setup

The performance of our proposed algorithm was evaluated on a modified version of Noxim simulator. Noxim is a cycle-accurate event-based NoC simulator developed in SystemC. We have conducted two sets of experiments. In the first set, a single application is mapped to 3D NoC including random and real ones. In the second set, multiple random and real applications are incrementally mapped to a 3D NoC.

We have used the task graph for free benchmark (TGFF)[36] to generate six random applications task graphs namely G1-G6 and six real-life applications: Video Object Plane Decoder (VOPD), Multi-Window Display (MWD), Picture in Picture (PIP), 263 encoder (263 Enc), 263 decoder (263 Dec) and MP3 encoder (MP3 Enc) from[37]. In the random applications, the dependencies between tasks are varying from 200 to 400 packets with a size of 8 flits and each flit has 128-bit. The characteristics of

Algorithm 3 Application mapping algorithm

Input: $PR = (L', E')$: NoC partition intended for the mapping
$ATG = (T, D)$: Application task graph of the current application
CG : List of communicating multi-task groups
Output: $map : T \rightarrow L'$

for all $\varphi_i \in CG$ **do** ▷ Compute the total inter and intra-communication for all groups
  $\eta_{\varphi_i} \leftarrow$ compute the total inter-communication;
  $\nu_{\varphi_i} \leftarrow$ compute the total intra-communication;
end for
$intial\_mapping(ATG, PR, CG)$; ▷ Selection and mapping of the initial multi-task group
while (still exist unmapped $uctg_i \in CG$) **do** ▷ Mapping of the remaining multi-task groups
  $best\_uctg \leftarrow$ find group $uctg_i$ doing a maximum inter-communication volume with mapped one; ▷ Choosing a dominant multi-task group to be mapped
  if multiple $uctg_i$ are selected **then**
    choose $uctg_i$ with maximum ν_{uctg_i};
  end if
  $best\_tile \leftarrow -1$;
  $\psi \leftarrow$ find mapped neighbours groups of $best\_uctg$;
  if $(size(\psi) == 1)$ **then** ▷ Case of one mapped neighbour group
    $mst\_tile \leftarrow$ get the tile of the mapped neighbor group;
    if resource available at mst_tile **then** ▷ Test of PE occupation
      $best\_tile \leftarrow mst\_tile$;
    else
      $comm\_vol \leftarrow$ communication volume between groups $best\_uctg$ and ψ;
      if $comm\_vol \geq \bar{A}_{comm}$ **then** ▷ Search for horizontal or vertical tile according to the communication volume
        $best\_tile \leftarrow search\_vt\_oneHop(mst\_tile)$; ▷ Search for vertical tile at one hop distance
      else
        $best\_tile \leftarrow search\_ht\_oneHop(mst\_tile)$; ▷ Search for horizontal tile at one hop distance
      end if
    end if
  end if
  if $(size(\psi) > 1)$ OR $(best\_tile == -1)$ **then** ▷ Case of multiple mapped neighbours groups
    $min\_cost \leftarrow \infty$ ▷ Initialization of cost with highest value
    for all $pe_i \in L'$ **do**
      if resource available at pe_i **then** ▷ Test of PE occupation
        $total\_cost \leftarrow 0$;
        for all $cg_i \in \psi$ **do** ▷ Compute the communication cost
          $mst\_tile \leftarrow get\_tile(cg_i)$;
          $total\_cost \leftarrow total\_cost + compute\_comm(cg_i, best\_uctg) \times compute\_energy(pe_i, mst\_tile)$;
        end for
        if $(total\_cost < min\_cost)$ **then** ▷ Set the minimum cost and the target PE
          $min\_cost \leftarrow total\_cost$;
          $best\_tile \leftarrow pe_i$;
        end if
      end if
    end for
  end if
  Map each task t_i of group $best\_uctg$ to tile $best\_tile$;
  $insert(best\_tile$ to map) and $update$(resource by map); ▷ Update the status of the resources
end while

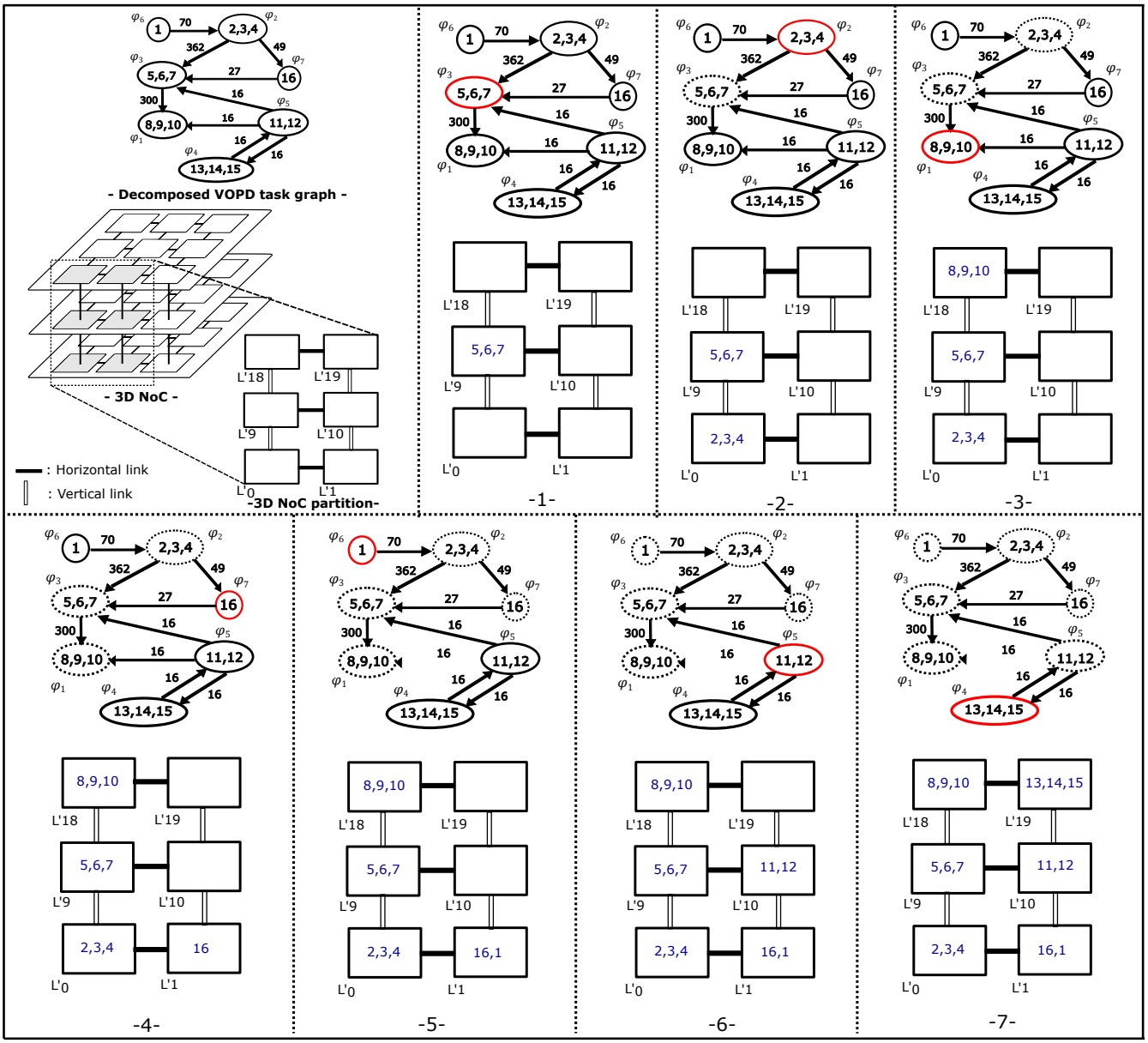

FIGURE 4 Multi-task groups mapping step

random and real applications in terms of the number of tasks and edges are shown in Table 3.

For evaluation, the platform chosen is 8x8x3 3D NoC, where full channel bandwidth has been allocated to each channel. We assume that each core is able to execute up to three tasks and to transfer a flit over a channel one cycle is needed. All experiments are run on a PC Intel i5-4310U 2.00 GHz dual-core processor. The system simulation configuration is outlined in Table 4.

6.2 | Simulation Results

The evaluated parameters are the average execution time, the communication cost, the average network latency and energy consumption.

6.2.1 | Single application mapping

In this experiment, both randomly generated graphs and real benchmarks are evaluated. The proposed algorithm is compared against INC[17], Nmap[24], Castnet[31] and Crinkle[28] algorithms. We have extended the algorithms Nmap, CastNet and Crinkle to support 3D NoC. The algorithms Nmap 3D, Castnet 3D, Crinkle 3D and INC support only the mono-tasking. As our knowledge, there are not any 3D multi-task applications mapping algorithm for regular 3D NoCs targeting our same optimization goals. In addition, to assess the efficiency of our proposed algorithm, the step of dynamic 3D NoC partitioning is not performed in Nmap 3D, Castnet 3D and Crinkle 3D algorithms. The INC algorithm uses the dynamic 3D partitioning with mono-tasking. The results will be compared to:

- Compare the results of the proposed algorithm with Nmap 3D, Castnet 3D and Crinkle 3D algorithms (algorithms without dynamic 3D NoC partitioning using different single-task mapping strategies).
- Compare the results of the proposed algorithm with INC algorithm (algorithm using dynamic 3D NoC partitioning using single-task mapping strategy).

Figure 5 and Figure 6 show the comparison result for the execution time and energy consumption using random and real applications. The total execution time is the time taken by the application as shown in Equation 10. As illustrated in Figure 5(a) and Figure 6(a), the proposed approach reduces the execution time when compared to the algorithms Nmap 3D, Castnet 3D, Crinkle 3D and INC for random and real applications. This result is explained by the fact that in our proposed approach, each processor can execute more than one task while the most communicating tasks are grouped in the same processor, resulting in a reduction of computation time and communication time. Moreover, the multi-task group mapping is carried out based on 3D NoC partitioning where the communicating groups of tasks are mapped in close proximity in order also to reduce the communication time. Our mapping strategy reduces the total execution time on average by 46.4715% and 12.40% for real and random applications respectively compared to INC algorithm. On the other hand, it reduces the execution time by (46.47%, 14.54%), (46.55%, 11.85%), (46.47%, 13.78%) for real and random applications compared to Nmap 3D, Crinkle 3D and Castnet 3D algorithms, respectively.

The total energy consumption is shown in Figure 5(b) and Figure 6(b) for random and real applications mapping, respectively. The energy consumption is greatly reduced in our proposed mapping strategy when compared to other heuristics. The total energy consumption is proportional to the average traffic in the network. Therefore, in our proposed algorithm the total traffic is well minimized as the most communicating task are in the same processor. Also, the most communicating groups of tasks are allocated to the vertical links which further reduce the energy consumption. The average reduction in energy consumption is 58.37% and 52.22% for real and random applications as compared to INC algorithm. Compared to Nmap 3D, Castnet 3D and Crinkle 3D, the average energy consumption reduction is (58.37%, 45.17%), (56.98%, 42.90%), (56.68%, 61.98%) for real and random applications, respectively.

In Table 5, we present the communication cost and the average network latency results obtained by our proposed algorithm compared with INC, Nmap 3D, Castnet 3D and Crinkle 3D algorithms. The communication cost refers to the amount of traffic in the network multiplied by the average hop count. In our proposed algorithm, the produced traffic in the network is reduced using a multi-task approach, which leads to the reduction of communication cost. Furthermore, it tries to minimise the distance between communicating group of tasks. The communication cost reduction of our proposed algorithm is (78.50% and 50.62%), (77.28%, 47.09%), (75.06%, 40.76%), (76.91%, 71.36%) for real and random applications compared to INC, Nmap 3D, Castnet 3D and Crinkle 3D algorithms, respectively. The network latency is the average time taken between packet injection in the local port and tail flit consumption in the local port of the destination node. Compared to the non-partitioning algorithms that use the total NoC space, our proposed algorithm has little increase in the average network latency in some cases, since the XYZ routing algorithm is based on shortest paths and in the partition space, the number of available network paths is in a lower number compared to NoC space. This is a small overhead since it not affect the performance of the system and the execution time, the communication cost and energy consumption are considerable better than the other algorithms.

6.2.2 | Multiples applications mapping

In this experiment, multiple random and real applications are incrementally mapped to a 3D NoC. The proposed algorithm is compared against random, Nmap 3D and INC mapping algorithms. In the random mapping algorithm, the tasks of an application are mapped to the tiles randomly. The experiments are performed for different scenarios. In each scenario, 10 applications are mapped and executed.

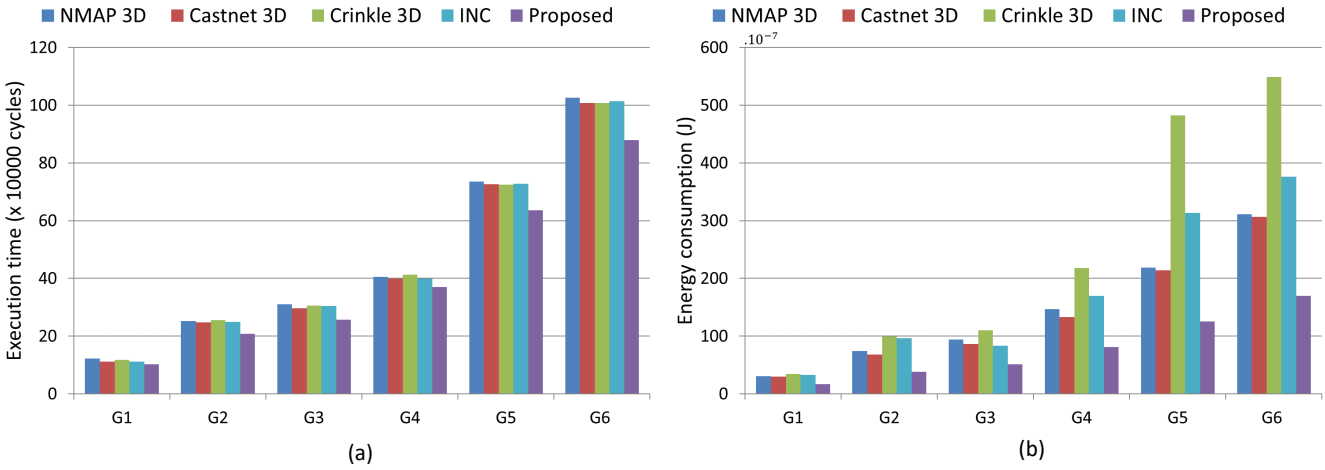

FIGURE 5 Execution time and energy consumption resulting from random single applications mapping

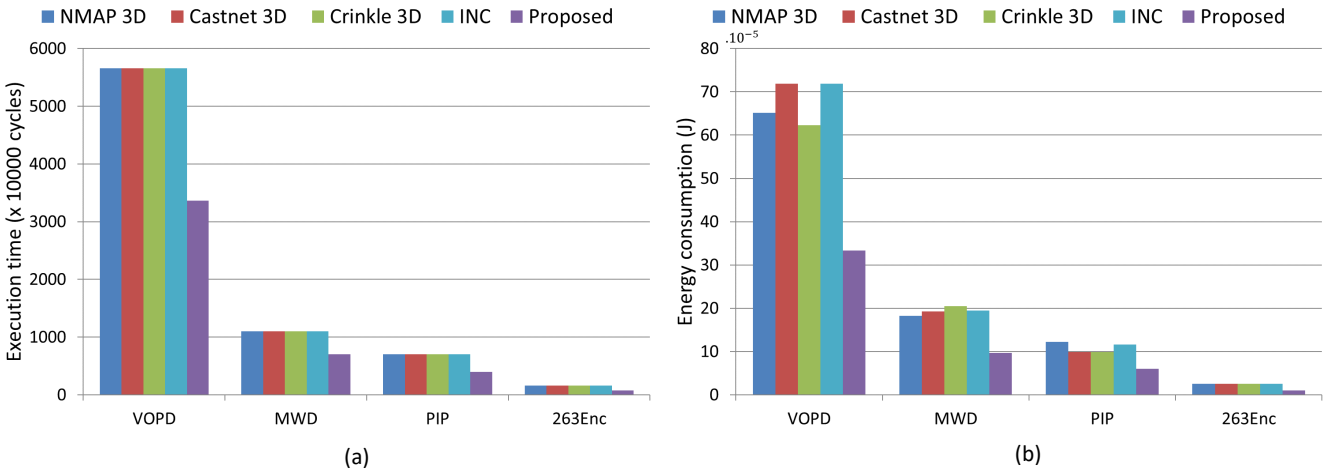

FIGURE 6 Execution time and energy consumption resulting from single multimedia applications mapping

- **Scenario 1:** Four repeats of $G2$ and three repeats of $G4$ and three repeats of $G1$.
- **Scenario 2:** Five repeats of $G3$ and two repeats of $G5$ and three repeats of $G2$.
- **Scenario 3:** VOPD, five repeats of 263 Dec and four repeats of PIP.
- **Scenario 4:** Four repeats of 263 Enc and four repeats of MP3 Enc and two repeats of MWD.

Figures 7 and 8 show the average execution time and energy consumption resulting from the execution of random applications (scenario 1 and scenario 2) and real applications (scenario 3 and scenario 4) using Random, Nmap 3D and INC mapping algorithm. In comparison with mono-task mapping algorithm using the total NoC space, our multi-task mapping algorithm using 3D NoC partitioning has less execution time and energy consumption. The reduction of execution time is (28.88%, 57.59%) and (13.93%, 44.95%) for random and real applications compared to Random and Nmap 3D algorithms, respectively. On the other hand, our proposed multi-task mapping reduces the execution time over INC algorithm by 13.43% and 44.19% for random and real applications. The reduction in the execution time against Random and Nmap 3D algorithm is due to 3D NoC partitioning which allows the execution of more applications simultaneously. Moreover, compared to INC algorithm, our proposed algorithm is based on a multi-task approach that group the most communicating task in the same processor which reduce greatly the communication time and consequently the execution time.

On the other hand, it is obvious that energy consumption is reduced in our proposed multi-task algorithm by reducing the traffic and the hop in the network. In addition, most intensive communications are allocated to vertical links that consume less energy than horizontal ones. The energy consumption reduction is (79.38%, 83.39%), (45.55%, 44.24%) and (41.61%, 40.64%) compared to Random, Nmap 3D and INC algorithms for random and real applications considered in the scenarios cited above, respectively.

Table 6 shows the communication cost and latency results for the mapping of random and real applications considered in the scenarios 1, 2, 3 and 4. The communication cost is greatly reduced in our proposed algorithm compared to other heuristic algorithms by (80.39%, 93.80%), (31.63%, 71.90%) and (42.66%, 75.58%) for random and real application compared to Random, Nmap 3D, and INC algorithms. Based on our proposed 3D NoC partitioning, the hop distance between the tasks of the same applications is minimized. In addition, as mentioned above, in our proposed algorithm, the traffic in the network is well minimized using the multi-task approach where the most communicating task are grouped together. However, the communicating tasks in each application are located in lower hop distance. Regarding the average latency, when compared to non-partitioning algorithm Nmap 3D, the latency overhead is 3.4%, 3.7% and 4.2% for scenario 1, scenario 2 and scenario 3 while it reduces the latency by 12% for scenario 4. On the other hand, the proposed algorithm reduces the latency by (57.89%, 69.4%), (12.53%, 4%) for random and real applications considered in the scenarios cited above when compared to Random and INC algorithms. We conclude that using the multi-task mapping heuristics with 3D NoC partitioning obtain better results (execution time, communication cost, energy consumption) than single mapping algorithms with small overhead in latency for some cases.

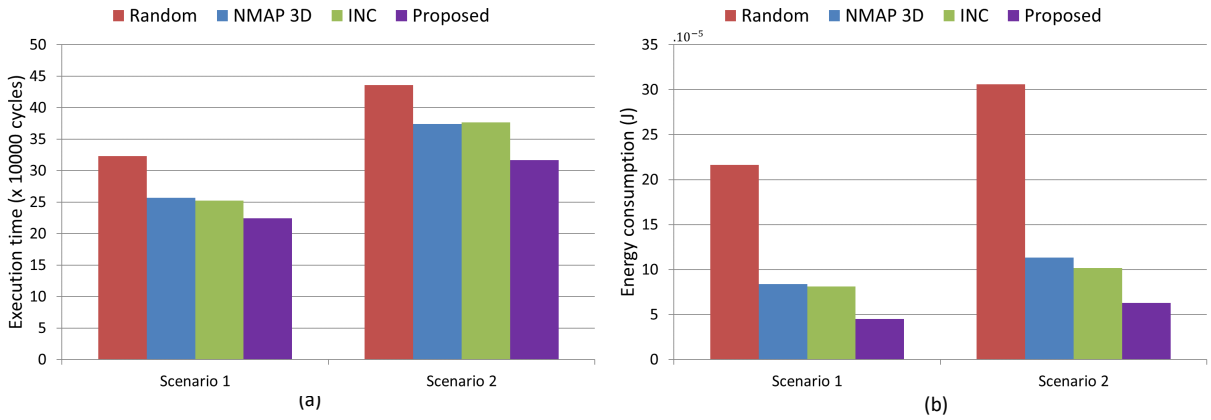

FIGURE 7 Execution time and energy consumption resulting from scenario 1 and scenario 2 mapping

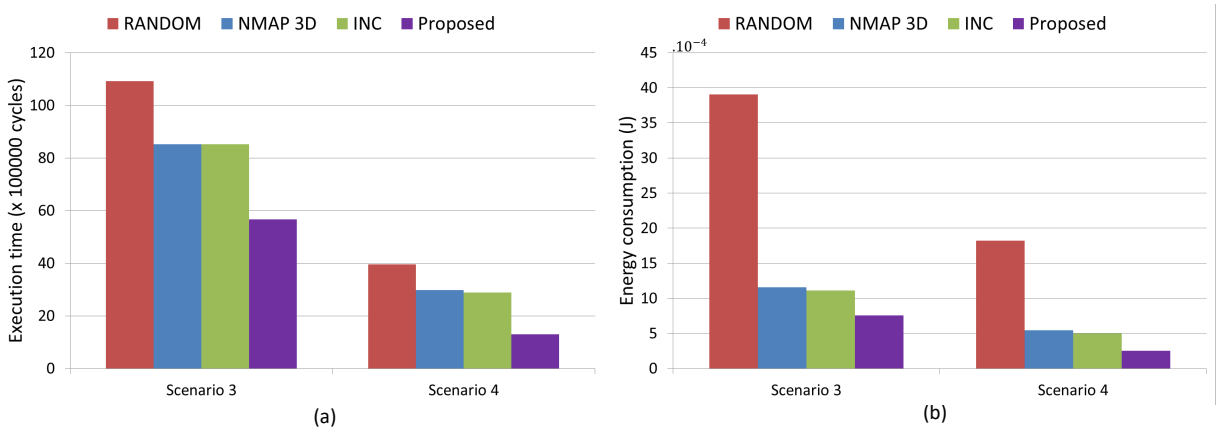

FIGURE 8 Execution time and energy consumption resulting from scenario 3 and scenario 4 mapping

7 | CONCLUSION

This paper proposes a multi-task applications mapping algorithm for regular 3D mesh-based NoCs aiming to minimize the network traffic, the communication energy and enhance the performance. The proposed approach allocates a partition for each application, to map all the tasks of the same application in the same partition aiming to minimize the communication overhead and reduce the fragmentation. The application task graph is decomposed into multiple multi-task groups while the most intensive communicating tasks are in the same group and the communication inter groups is minimized. The proposed approach tries to map the communicating multi-task groups close to each other while the tasks of each group are mapped in the same PE. For the mapping decision, our proposed approach uses the communication energy as a cost function and include the cost of already mapped multi-task groups communicating with the multi-task group being mapped instead of considering only master-slave pairs. Moreover, the proposed approach exploits accurately the use of vertical links due to their benefits over horizontal ones. The experimental results disclose that our proposed mapping exhibits better performance in term of energy consumption, communication cost and execution time compared to single-task 3D mapping algorithms.

In the future research work, we plan to integrate a routing algorithm with our proposed mapping approach, considering heterogeneous NoC structures and thermal aspect are also topics for future studies.

References

1. Moore GE. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff. *IEEE solid-state circuits society newsletter* 2006; 11(3): 33–35.

2. Jerraya A, Wolf W. *Multiprocessor systems-on-chips*. Elsevier . 2004.

3. Benini L, De Micheli G. Networks on chips: A new SoC paradigm. *computer* 2002; 35(1): 70–78.

4. Jantsch A, Tenhunen H, others . *Networks on chip*. 396. Springer . 2003.

5. De Micheli G, Benini L. Networks on chips: 15 years later. *Computer* 2017(5): 10–11.

6. Tatas K, Siozios K, Soudris D, Jantsch A. *Designing 2D and 3D network-on-chip architectures*. No. IKEEBOOK-2017-046Springer . 2014.

7. Pavlidis VF, Friedman EG. 3-D topologies for networks-on-chip. *IEEE Transactions on very large scale integration (VLSI) systems* 2007; 15(10): 1081–1090.

8. Feero BS, Pande PP. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on computers* 2008; 58(1): 32–45.

9. Davis WR, Wilson J, Mick S, et al. Demystifying 3D ICs: The pros and cons of going vertical. *IEEE Design & Test of Computers* 2005; 22(6): 498–510.

10. Benyamina AEH, Boulet P. Multi-objective Mapping for NoC Architectures.. *Journal of Digital Information Management* 2007; 5(6).

11. Ying H, Hollstein T, Hofmann K. Fast and optimized task allocation method for low vertical link density 3-dimensional networks-on-chip based many core systems. In: IEEE. ; 2013: 1777–1782.

12. Wang J, Li L, Pan H, He S, Zhang R. Latency-aware mapping for 3D NoC using rank-based multi-objective genetic algorithm. In: IEEE. ; 2011: 413–416.

13. Wadhwani P, Choudhary N, Singh D. Energy efficient mapping in 3d mesh communication architecture for NoC. *Global Journal of Computer Science and Technology* 2013.

14. He H, Fang F, Wang W. Improved simulated annealing genetic algorithm based low power mapping for 3d NoC. In: . 232. EDP Sciences. ; 2018: 02022.

15. Souza Carvalho dEL, Calazans NLV, Moraes FG. Dynamic task mapping for MPSoCs. *IEEE Design & Test of Computers* 2010; 27(5): 26–35.

16. Benhaoua MK, Singh AK, Benyamina A, Kumar A, Boulet P. Heuristic for accelerating run-time task mapping in NoC-based heterogeneous MPSoCs. *Journal of Digital Information Management* 2014; 12(5): 293.

17. Wang X, Palesi M, Yang M, Jiang Y, Huang MC, Liu P. Power-aware run-time incremental mapping for 3-D networks-on-chip. In: Springer. ; 2011: 232–247.

18. Ziaeeziabari H, Patooghy A. 3D-AMAP: a latency-aware task mapping onto 3D mesh-based NoCs with partially-filled TSVs. In: IEEE. ; 2017: 593–597.

19. Kiani V, Reshadi M. Mapping multiple applications onto 3D NoC-based MPSoCs supporting wireless links. *The Journal of Supercomputing* 2017; 73(5): 2187–2213.

20. Maqsood T, Tziritas N, Loukopoulos T, et al. Energy and communication aware task mapping for MPSoCs. *Journal of parallel and distributed computing* 2018; 121: 71–89.

21. Jiang S, Wu Q, Chen S, et al. Optimizing dynamic mapping techniques for on-line NoC test. In: IEEE. ; 2018: 227–232.

22. Sharma PK, Biswas S, Mitra P. Energy efficient heuristic application mapping for 2-D mesh-based network-on-chip. *Microprocessors and Microsystems* 2019; 64: 88–100.

23. Bhardwaj K, Mane PS. C3Map and ARPSO based mapping algorithms for energy-efficient regular 3-D NoC architectures. In: IEEE. ; 2014: 1–4.

24. Murali S, De Micheli G. Bandwidth-constrained mapping of cores onto NoC architectures. In: . 2. IEEE. ; 2004: 896–901.

25. Manna K, Chattopadhyay S, Sengupta I. Through silicon via placement and mapping strategy for 3d mesh based network-on-chip. In: IEEE. ; 2014: 1–6.

26. Agyeman MO, Ahmadinia A, Bagherzadeh N. Energy and performance-aware application mapping for inhomogeneous 3D networks-on-chip. *Journal of Systems Architecture* 2018; 89: 103–117.

27. Dageleh MZ, Jamali MAJ. V-CastNet3D: A novel clustering-based mapping in 3-D Network on chip. *Nano communication networks* 2018; 18: 51–61.

28. Saeidi S, Khademzadeh A, Vardi F. Crinkle: A heuristic mapping algorithm for network on chip. *IEICE Electronics Express* 2009; 6(24): 1737–1744.

29. Jha V, Deol S, Jha M, Sharma G. Energy and latency aware application mapping algorithm & optimization for homogeneous 3d network on chip. *arXiv preprint arXiv:1404.2512* 2014.

30. Jha V, Jha M, Sharma G. Estimation of optimized energy and latency constraints for task allocation in 3d network on chip. *arXiv preprint arXiv:1405.0109* 2014.

31. Tosun S. New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs. *Journal of Systems Architecture* 2011; 57(1): 69–78.

32. Singh AK, Shafique M, Kumar A, Henkel J. Analysis and mapping for thermal and energy efficiency of 3-D video processing on 3-D multicore processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2016; 24(8): 2745–2758.

33. Huang J, Buckl C, Raabe A, Knoll A. Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems. In: IEEE. ; 2011: 447–454.

34. Atitallah RB, Niar S, Greiner A, Meftali S, Dekeyser JL. Estimating energy consumption for an MPSoC architectural exploration. In: Springer. ; 2006: 298–310.

35. Matsutani H, Koibuchi M, Amano H. Tightly-coupled multi-layer topologies for 3-D NoCs. In: IEEE. ; 2007: 75–75.

36. Dick RP, Rhodes DL, Wolf W. TGFF: task graphs for free. In: IEEE. ; 1998: 97–101.

37. Sahu PK, Shah T, Manna K, Chattopadhyay S. Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2013; 22(2): 300–312.

TABLE 1 Reviewed works on task mapping into 3D NoC

References	Mapping nature	Mono/Multi task	Target architecture	Architecture model	Management of NoC architecture	Optimization Goal
Wadhwani et al[13]	Static	Mono-task	Regular	Homogeneous	No	Energy consumption
Kiani et al[19]	Dynamic	Multi-task	Irregular wireless	Heterogeneous	Yes	Improve system performances
Ziaeeziabari et al[18]	Dynamic	Mono-task	Irregular partially TSVs	Homogeneous	No	Network latency
Dageleh et al[27]	Static	Mono-task	Regular	Homogeneous	No	Execution time
Saeidi et al[28]	Static	Mono-task	Regular	Homogeneous	No	Energy consumption, execution time
He et al[14]	Static	Mono-task	Regular	Homogeneous	No	Energy consumption
Singh et al[32]	Hybrid	Multi-task	Regular	Homogeneous	No	Peak temperature, energy consumption
Wang et al[17]	Dynamic	Mono-task	Regular	Homogeneous	Yes	Energy consumption
Bhardwaj et al[23]	Static	Mono-task	Regular	Homogeneous	No	Energy consumption, network latency
Murali et al[24]	Static	Mono-task	Regular	Homogeneous	No	Communication delay
Tosun[31]	Static	Mono-task	Regular	Homogeneous	No	Energy consumption
This work	**Dynamic**	**Multi-task**	**Regular**	**Homogeneous**	**Yes**	**Communication volume, energy consumption and improve system performances**

TABLE 2 List of parameters

Parameter	Definition
P_i	The tile
t_i	The task
φ_i	The multi-task group
pe_i	The processing element
$uctg_i$	The unmapped multi-task group
θ_{PE}	Maximum number of tasks allowed per PE
$\|T\|$	Number of tasks
N_G	Number of multi-task groups
NT_G	Number of tasks that each group can contain
ν_{φ_i}	Total intra-communication of group φ_i
η_{φ_i}	Total inter-communication of group φ_i
$\bar{A}_{comm}$	Average communication volume of task graph
N_H	Required number of horizontal links
N_V	Required number of vertical links
`UNP_List`	The list of unpartitioned tiles
PR	Partition graph
CG	The list of communicating multi-task groups

TABLE 3 Specification of applications graphs

Application	Number of tasks	Number of edges
VOPD	16	21
MWD	12	12
PIP	8	8
263 Enc	12	12
263 Dec	14	15
MP3 Enc	13	13
$G1$	20	38
$G2$	48	87
$G3$	63	114
$G4$	84	168
$G5$	132	263
$G6$	190	391

TABLE 4 Simulation settings

Parameters	Values
NoC size	8 x 8 x 3
Length of vertical links	60 um
Length of horizontal links	1 mm
Maximum tasks per PE	3 tasks
Buffer depth	4 flits
Packet size	8 flits
Flit size	128 bit
Routing	XYZ

TABLE 5 Communication cost and average latency for random and real applications under different 3D mapping algorithms

Task Graph	Communication cost (Mb/s)					Average latency (cycles)				
	NMAP 3D	CastNet 3D	Crinkle 3D	INC	Proposed	NMAP 3D	CastNet 3D	Crinkle 3D	INC	Proposed
VOPD	4135	4119	4157	4723	911	5.29	5.39	5.30	6.07	6.35
MWD	1248	1184	1472	1376	416	5.23	5.11	5.63	5.46	5.77
PIP	896	640	640	832	256	10.89	9.55	9.55	12.11	12.66
263 Enc	162.024	161.998	166.370	176.938	28.683	10.01	10.01	10.06	8.17	5
263 Dec	19.822	19.822	20.046	21.500	1.138	5.02	5.02	9.59	5.19	5.27
MP3 Enc	8.596	8.596	9.782	9.138	2.487	5.0074	5.0074	5.28	5.1339	5
G1	22.810	20.231	29.825	26.979	12.054	12.10	7.38	11.58	10.57	11.22
G2	56.797	46.334	88.042	59.184	27.947	11.94	11.36	17.93	14.28	13.08
G3	71.532	60.767	127.254	76.058	37.644	12.23	10.88	19.158	15.99	15.82
G4	74.962	65.623	136.723	79.326	38.580	13.18	11.41	20.21	17.39	16.88
G5	160.724	154.903	416.163	165.985	95.07	12.52	12.29	23.80	22.65	16.28
G6	232.994	227.261	637.297	248.181	121.487	12.79	12.15	21.94	21.22	16.66

TABLE 6 Performance parameters evaluation resulting from the mapping of scenarios 1, 2, 3 and 4

Task Graph	Scenario 1				Scenario 2			
	NMAP 3D	INC	Random	Proposed	NMAP 3D	INC	Random	Proposed
Avg. execution time (cycles)	256769	252157	322745	224508	373940	376624	435795	316757
Comm. cost (Mb/s)	643.28	710.82	2130	324.9	842.63	1053.55	3030.53	726.58
Avg. latency (cycles)	14.98	17.54	38.56	15.50	14.85	17.80	35	15.41
Energy consumption ($\times 10^2$ pJ)	840123	813546	2164680	449047	1130890	1018340	3060220	627144
	Scenario 3				**Scenario 4**			
	NMAP 3D	INC	Random	Proposed	NMAP 3D	INC	Random	Proposed
Avg. execution time (cycles)	8525098	8525103	10927212	5669929	2984573	2884616	3950491	1301282
Comm. cost (Mb/s)	7118.47	8928.8	32124.6	1944.7	3310.62	3534.24	15077.31	956.4
Avg. latency (cycles)	7.83	8.33	23.74	8.16	7.3	6.05	23.89	6.41
Energy consumption ($\times 10^2$ pJ)	11593300	11135100	39038500	7546920	5451070	4967290	18225500	2530070